**MUSIC MATCHES MOOD**

A

Mini Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

PASULA PAVAN KALYAN 1602-20-737-089

BALAJI MAHENDERKAR 1602-20-737-079

IEPPALA SAI KIRAN 1602-20-737-099



Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Ibrahimbagh, Hyderabad-31

2022

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University and Approved by AICTE)

Hyderabad-500 031

Department of Information Technology



DECLARATION BY THE CANDIDATE

We, Pasula Pavan Kalyan, Mahenderkar Balaji, and Ieppala Sai Kiran, bearing hall ticket numbers, 1602-20-737-089, 1602-20-737-079 aand1602-20-737-099, hereby declare that the project report entitled **“MUSIC MATCHES MOOD”** is submitted in partial fulfilment of the requirement for the award of the degree of Bachelor of Engineering in Information Technology

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

PASULA PAVAN KUMAR

1602-20-737-089

BALAJI MAHENDERKAR

1602-20-737-079

IEPPALA SAI KIRAN

1602-20-737-099

(Faculty In-Charge) (Head,Dept of IT)

ACKNOWLEDGMENT

We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering for his encouragement.

We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi College of Engineering, for introducing the Mini-Project module in our curriculum, and also for his suggestions, motivation, and co-operation for the successful completion of our Mini Project.

We also want to thank and convey our gratitude towards our mini project coordinators **DRL PRASANNA** madam and **RADHA** madam, for guiding us in understanding the process of project development & giving us timely suggestions at every phase.

We would also like to sincerely thank the project reviewers for their valuable inputs and suggestions.

**ABSTRACT**

Our Mini Project Music matches mood player is a simple console based GUI application using pygame module. Music matches mood is a music player that recommends user songs based on their mood. There are songs for moods like happy, sad and party. User can enjoy the songs and can add the songs he/she likes to the favorites list. It isn't mandatory for the user to create an account. He/she can just use guest login feature but cannot use different features of the music player.User is also allowed to add and delete songs from the playlist

This program is made by using tkinter, pyqt5, os and pygame modules

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| S.No | Topic | Page No. |
| 1. | Introduction | 6 |
| 2. | Technology | 7 |
| 3. | Design | 8 |
| 4. | Implementation | 14 |
| 5. | Testing | 30 |
| 6. | Additional Knowledge Acquired | 42 |
| 7. | Conclusion and Future work | 43 |
| 8. | References | 44 |

**INTRODUCTION**

**Overview:**

Music matches mood is a simple music player with a really good extra feature which is it suggests the user a playlist based on his mood. It suggests him a playlist after asking them about their mood.

First of all, the user must register/ sign up first. Then, they can login with their credentials and use different features like adding songs to the favorites, uploading favorites, adding songs to the playlist and deleting songs from the playlist.

After entering the current username and password the user will be asked to select his mood from the options given based on the option selected music player with a particular playlist is showed.

A user can also login as a guest but he cannot use the same features given for the registered users. They can just select a mood and listen to the songs. Menu option will be disabled for the guest users.

In this project we use pygame it is a 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. Many of the functions of the pygame are used in every game it also has many prebuilt libraries which are used to directly construct the screen unlike the normal package in this module we also have mixer with screen display with it we can able to add music to game and action music where we want music at particular event

We have also used tkinter, qt designer and pyqt5 to build the GUI.

**PURPOSE:**

To give user the feature of listening to the songs based on their mood.

Explore different modules and libraries in python

To improve critical thinking skills using python

**OBJECTIVE:**

The objective of this application is to make users listen to the songs based on their mood and be able to enjoy their time while using our music player.

## SYSTEM REQUIREMENTS

**Hardware Requirements:**

Minimum Ram required: 512 mb

Minimum Disk Space required: 50 mb

Processor: core i3

Input devices: Mouse

Output devices: Monitor

**Software Requirements:**

IDLE Python (3.8 32 bit)

Python Launcher

Windows 8.1 or above

**Modules used:**

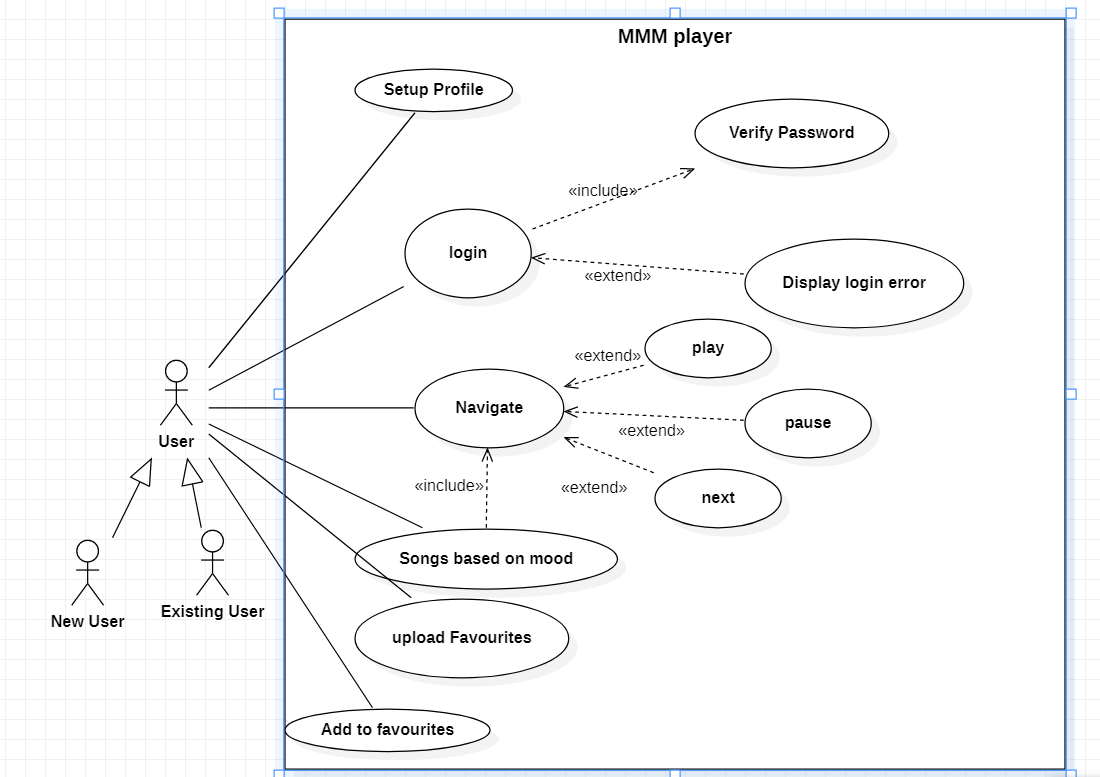
pygame module tkinter

pyqt5

**PROPOSED WORK**

**DESIGN**

USE CASE DIAGRAM -



**USE CASES :**

* Profile setup
* Login
* Guest login
* Navigation
* Mood based music
* Add to favourites
* Uploading songs

**USE CASE DESCRIPTIONS**

Use Case ID: UC01

Name: Setup profile

Actors: User

Description: Allows the user to create an account

Pre-conditions: None

Post-conditions: User Count is created

Flow:

|  |  |
| --- | --- |
| **User** | **System** |
| 1) Chooses the signup option  3) Enters the data prompted by the system. | 2) Prompts for data required for registration.  4)  4.1) If the information is valid, create a new account.  4.2) If the information is invalid, display error message and prompt for new values. |

User case ID: UC02

Name: Login

Actors: User

Description: Allows user to login to the account.

Pre-condition: User must be register first.

Post-condition: User logs in and the home page is displayed.

Flow:

|  |  |
| --- | --- |
| **User** | **System** |
| **1)** Enters username and password and chooses to sign in | **2)** Validates username and password  **2.1)** If username is invalid, display error and prompt for valid username.  **2.2)** If the password is incorrect, display error and prompt for valid password.  **2.3)** If the username and password and valid, then login and display home page. |

User case ID: UC03

Name: Skip/guest login

Actors: User

Description: Allows user to open music player without actually creating an account

Pre-condition: None.

Post-condition: User logs in and the home page is displayed.

Main flow:

|  |  |
| --- | --- |
| **User** | **System** |
| **1)** User chooses to skip/guest login | **2)** System performs the requested action. |

User case ID: UC04

Name: Navigation

Actors: User

Description: Allows user to play, pause and change the tracks

Pre-condition: None.

Post-condition: Appropriate action is performed.

Main flow:

|  |  |
| --- | --- |
| **User** | **System** |
| **1)** User chooses to play ,pause or resume the song. | **2)** System performs the requested action. |

User case ID: UC05

Name: Mood Based music

Actors: User

Description: Allows user to select mood and recommends music based on his/her mood.

Pre-condition: None.

Post-condition: Appropriate mood based songs are recommended.

Flow:

|  |  |
| --- | --- |
| **User** | **System** |
| **1)** User chooses to select his/her mood from the given ones | **2)** System recommends songs based on the mood chosen by the user. |

User case ID: UC06

Name: Add to favorites

Actors: User

Description: Adds the user’s favorite songs into a playlist

Pre-condition: User must be logged in

Post-condition: Songs are added into fav playlist.

Flow:

|  |  |
| --- | --- |
| **User** | **System** |
| **1)** User chooses to add songs to a new fav playlist. | **2)** System creates a new playlist and adds the songs |

User case ID: UC07

Name: Uploading of Songs

Actors : User

Description: user adds songs to the music player .

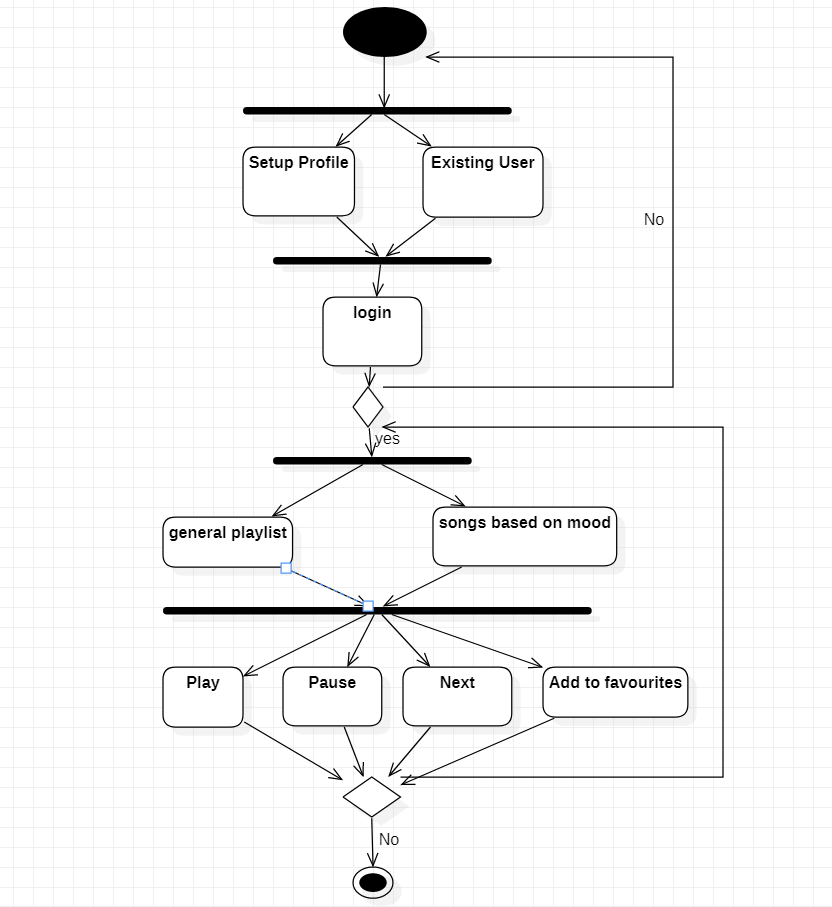
Pre-condition: None

Post-condition: Songs are added to the music player.

Flow:

|  |  |
| --- | --- |
| **User** | **System** |
| 1.User selects a song from the given options. | 2) System adds the song to the music player. |

**ACTIVITY DIAGRAM**



**IMPLEMENTATION**

**Source code:**

**Login Page code:**

from os import listdir

from time import sleep

from sign\_up import Ui\_Sign\_up

from PyQt5 import QtCore, QtGui, QtWidgets

from mood import Ui\_mood

import mmm

class Ui\_LoginWindow(object):

def setupUi(self, LoginWindow):

LoginWindow.setObjectName("LoginWindow")

LoginWindow.resize(568, 524)

LoginWindow.setStyleSheet("background-color: rgb(255, 49, 218);\n"

"font: 75 12pt \"MS Shell Dlg 2\";\n"

"background-color: rgb(0, 255, 255);\n"

"")

self.centralwidget = QtWidgets.QWidget(LoginWindow)

self.centralwidget.setObjectName("centralwidget")

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setGeometry(QtCore.QRect(70, 50, 171, 16))

self.label.setStyleSheet("font: 75 10pt \"MS Shell Dlg 2\";\n"

"font: 12pt \"MS Shell Dlg 2\";")

self.label.setObjectName("label")

self.label\_2 = QtWidgets.QLabel(self.centralwidget)

self.label\_2.setGeometry(QtCore.QRect(80, 130, 151, 41))

self.label\_2.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";")

self.label\_2.setObjectName("label\_2")

self.login\_btn = QtWidgets.QPushButton(self.centralwidget)

self.login\_btn.setGeometry(QtCore.QRect(210, 230, 101, 31))

self.login\_btn.setStyleSheet("background-color: rgb(170, 255, 255);\n"

"font: 75 12pt \"MS Shell Dlg 2\";\n"

"")

self.login\_btn.setObjectName("login\_btn")

self.Usernam\_text = QtWidgets.QLineEdit(self.centralwidget)

self.Usernam\_text.setGeometry(QtCore.QRect(240, 40, 221, 41))

self.Usernam\_text.setStyleSheet("background-color: rgb(255, 255, 255);\n"

"font: 75 14pt \"MS Shell Dlg 2\";")

self.Usernam\_text.setText("")

self.Usernam\_text.setObjectName("Usernam\_text")

self.passwrd\_text = QtWidgets.QLineEdit(self.centralwidget)

self.passwrd\_text.setGeometry(QtCore.QRect(240, 130, 221, 41))

self.passwrd\_text.setStyleSheet("background-color: rgb(255, 255, 255);\n"

"font: 75 14pt \"MS Shell Dlg 2\";")

self.passwrd\_text.setObjectName("passwrd\_text")

self.New\_user\_btn = QtWidgets.QPushButton(self.centralwidget)

self.New\_user\_btn.setGeometry(QtCore.QRect(370, 270, 171, 41))

self.New\_user\_btn.setStyleSheet("font: 75 14pt \"MS Shell Dlg 2\";\n"

"background-color: rgb(255, 255, 255);")

self.New\_user\_btn.setObjectName("New\_user\_btn")

self.guest\_login\_btn = QtWidgets.QPushButton(self.centralwidget)

self.guest\_login\_btn.setGeometry(QtCore.QRect(370, 330, 171, 41))

self.guest\_login\_btn.setStyleSheet("font: 75 14pt \"MS Shell Dlg 2\";\n"

"background-color: rgb(255, 255, 255);")

self.guest\_login\_btn.setObjectName("guest\_login\_btn")

self.label\_3 = QtWidgets.QLabel(self.centralwidget)

self.label\_3.setGeometry(QtCore.QRect(160, 390, 201, 41))

self.label\_3.setText("")

self.label\_3.setObjectName("label\_3")

LoginWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(LoginWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 568, 30))

self.menubar.setObjectName("menubar")

LoginWindow.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(LoginWindow)

self.statusbar.setObjectName("statusbar")

LoginWindow.setStatusBar(self.statusbar)

self.retranslateUi(LoginWindow)

QtCore.QMetaObject.connectSlotsByName(LoginWindow)

self.login\_btn.clicked.connect(self.login)

self.guest\_login\_btn.clicked.connect(self.guestLogin)

self.New\_user\_btn.clicked.connect(self.newUser)

def retranslateUi(self, LoginWindow):

\_translate = QtCore.QCoreApplication.translate

LoginWindow.setWindowTitle(\_translate("LoginWindow", "MainWindow"))

self.label.setText(\_translate("LoginWindow", "Enter User Name:"))

self.label\_2.setText(\_translate("LoginWindow", "Enter Password:"))

self.login\_btn.setText(\_translate("LoginWindow", "LOGIN"))

self.New\_user\_btn.setText(\_translate("LoginWindow", "->New User"))

self.guest\_login\_btn.setText(\_translate("LoginWindow", "->Guest Login"))

def login(self):

userid = self.Usernam\_text.text()

pssw = self.passwrd\_text.text()

if userid in listdir("Users"):

file = open(f"Users\\{userid}\\credentials", "r")

verify = file.read().splitlines()

if pssw in verify:

self.label\_3.setText("Success")

self.loginSuccess()

else:

self.label\_3.setText("Wrong Password ")

print("Wrong Password")

else:

self.label\_3.setText("UnRegistered User")

print("User Not registered")

def loginSuccess(self):

print("Login Successful")

self.label\_3.setText("Login Successful")

sleep(1)

LoginWindow.close()

self.newWindow = QtWidgets.QMainWindow()

self.guest = False

self.ui = Ui\_mood(self.guest,self.Usernam\_text.text())

self.ui.setupUi(self.newWindow)

self.newWindow.show()

def newUser(self):

self.newWindow = QtWidgets.QMainWindow()

self.ui = Ui\_Sign\_up()

self.ui.setupUi(self.newWindow)

self.newWindow.show()

def guestLogin(self):

print("Guest Login")

self.label\_3.setText("Guest Login")

LoginWindow.close()

self.newWindow = QtWidgets.QMainWindow()

self.guest = "False"

self.ui = Ui\_mood(self.guest,"")

self.ui.setupUi(self.newWindow)

self.newWindow.show()

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

LoginWindow = QtWidgets.QMainWindow()

ui = Ui\_LoginWindow()

ui.setupUi(LoginWindow)

LoginWindow.show()

sys.exit(app.exec\_())

**Signup window code:**

import os

from os import listdir

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_Sign\_up(object):

def setupUi(self, Sign\_up):

Sign\_up.setObjectName("Sign\_up")

Sign\_up.resize(628, 430)

Sign\_up.setStyleSheet("background-color: rgb(85, 255, 255);")

self.centralwidget = QtWidgets.QWidget(Sign\_up)

self.centralwidget.setObjectName("centralwidget")

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setGeometry(QtCore.QRect(70, 80, 151, 41))

self.label.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";")

self.label.setObjectName("label")

self.label\_2 = QtWidgets.QLabel(self.centralwidget)

self.label\_2.setGeometry(QtCore.QRect(70, 160, 141, 41))

self.label\_2.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";")

self.label\_2.setObjectName("label\_2")

self.User\_name = QtWidgets.QLineEdit(self.centralwidget)

self.User\_name.setGeometry(QtCore.QRect(230, 80, 231, 41))

self.User\_name.setStyleSheet("font: 75 13pt \"MS Shell Dlg 2\";\n"

"background-color: rgb(255, 255, 255);")

self.User\_name.setText("")

self.User\_name.setObjectName("User\_name")

self.password = QtWidgets.QLineEdit(self.centralwidget)

self.password.setGeometry(QtCore.QRect(230, 160, 231, 41))

self.password.setStyleSheet("font: 75 13pt \"MS Shell Dlg 2\";\n"

"background-color: rgb(255, 255, 255);")

self.password.setText("")

self.password.setObjectName("password")

self.label\_3 = QtWidgets.QLabel(self.centralwidget)

self.label\_3.setGeometry(QtCore.QRect(170, 320, 231, 31))

self.label\_3.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";\n"

"background-color: rgb(0, 255, 255);\n"

"")

self.label\_3.setText("")

self.label\_3.setObjectName("label\_3")

self.pushButton = QtWidgets.QPushButton(self.centralwidget)

self.pushButton.setGeometry(QtCore.QRect(230, 240, 121, 41))

self.pushButton.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";")

self.pushButton.setObjectName("pushButton")

Sign\_up.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(Sign\_up)

self.menubar.setGeometry(QtCore.QRect(0, 0, 628, 26))

self.menubar.setObjectName("menubar")

Sign\_up.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(Sign\_up)

self.statusbar.setObjectName("statusbar")

Sign\_up.setStatusBar(self.statusbar)

self.retranslateUi(Sign\_up)

QtCore.QMetaObject.connectSlotsByName(Sign\_up)

self.pushButton.clicked.connect(self.register)

def retranslateUi(self, Sign\_up):

\_translate = QtCore.QCoreApplication.translate

Sign\_up.setWindowTitle(\_translate("Sign\_up", "MainWindow"))

self.label.setText(\_translate("Sign\_up", "Enter Username"))

self.label\_2.setText(\_translate("Sign\_up", "Enter Password"))

self.pushButton.setText(\_translate("Sign\_up", "Sign up"))

def register(self):

userid = self.User\_name.text()

pssw = self.password.text()

if(userid=="" or pssw==""):

self.label\_3.setText("Wrong Input")

if userid in listdir("Users"):

self.label\_3.setText("Username already exists!")

else:

os.chdir('C:\\Users\\pa1ka\\Desktop\\Mini Project\\Users')

os.mkdir(userid)

file = open(f"C:\\Users\\pa1ka\\Desktop\\Mini Project\\Users\\{userid}\\"+"credentials","w")

file.write(userid+"\n"+pssw)

file.close()

self.label\_3.setText("Registration Success")

print("Registration Successful")

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

Sign\_up = QtWidgets.QMainWindow()

ui = Ui\_Sign\_up()

ui.setupUi(Sign\_up)

Sign\_up.show()

sys.exit(app.exec\_())

**Select anyone Mood from options Window code:**

from time import sleep

from PyQt5 import QtCore, QtGui, QtWidgets

import mmm

class Ui\_mood(object):

def \_\_init\_\_(self,guest,user\_id):

self.guest = guest

self.user\_id = user\_id

def setupUi(self, mood):

mood.setObjectName("mood")

mood.resize(622, 522)

self.centralwidget = QtWidgets.QWidget(mood)

self.centralwidget.setObjectName("centralwidget")

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setGeometry(QtCore.QRect(170, 90, 241, 71))

self.label.setStyleSheet("font: 75 16pt \"MS Shell Dlg 2\";\n"

"background-color: rgb(255, 255, 255);")

self.label.setObjectName("label")

self.comboBox = QtWidgets.QComboBox(self.centralwidget)

self.comboBox.setGeometry(QtCore.QRect(210, 210, 141, 41))

self.comboBox.setStyleSheet("font: 75 13pt \"MS Shell Dlg 2\";\n"

"background-color: rgb(255, 255, 255);")

self.comboBox.setObjectName("comboBox")

self.comboBox.addItem("")

self.comboBox.addItem("")

self.comboBox.addItem("")

self.submitButton = QtWidgets.QPushButton(self.centralwidget)

self.submitButton.setGeometry(QtCore.QRect(220, 330, 131, 51))

self.submitButton.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";")

self.submitButton.setObjectName("submitButton")

mood.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(mood)

self.menubar.setGeometry(QtCore.QRect(0, 0, 622, 26))

self.menubar.setObjectName("menubar")

mood.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(mood)

self.statusbar.setObjectName("statusbar")

mood.setStatusBar(self.statusbar)

self.retranslateUi(mood)

QtCore.QMetaObject.connectSlotsByName(mood)

self.submitButton.clicked.connect(self.pressed)

def pressed(self):

mmm.main(self.comboBox.currentText(), self.guest, self.user\_id)

mood.close()

def retranslateUi(self, mood):

\_translate = QtCore.QCoreApplication.translate

mood.setWindowTitle(\_translate("mood", "MainWindow"))

self.label.setText(\_translate("mood", " How\'s your Mood?"))

self.comboBox.setItemText(0, \_translate("mood", "Happy"))

self.comboBox.setItemText(1, \_translate("mood", "Sad"))

self.comboBox.setItemText(2, \_translate("mood", "Party"))

self.submitButton.setText(\_translate("mood", "SUBMIT"))

if \_\_name\_\_ == "\_\_main\_\_":

import sys

app = QtWidgets.QApplication(sys.argv)

mood = QtWidgets.QMainWindow()

ui = Ui\_mood()

ui.setupUi(mood)

mood.show()

sys.exit(app.exec\_())

**Music Player code:**

# importing libraries

import os

import tkinter

from tkinter import filedialog

from PIL import ImageTk

from pygame import mixer

from tkinter import \*

import tkinter.font as font

def main(mood,guest,user\_id):

global temp\_song

root = Tk()

root.title('Music Matches Mood Player')

mixer.init()

songs\_list = Listbox(root, selectmode=SINGLE, bg="navyblue", fg="white", font=('arial', 15), height=12, width=47,

selectbackground="lightblue", selectforeground="black")

songs\_list.grid(columnspan=9)

# inserting songs in the playlist based on mood

temp\_song = os.listdir(f"C:\\Users\\pa1ka\\Desktop\\Mini Project\\{mood}")

for s in temp\_song:

songs\_list.insert(END, s)

def addfav():

song = songs\_list.get(ACTIVE)

file = open(f"C:\\Users\\pa1ka\\Desktop\\Mini Project\\Users\\{user\_id}\\" + "fav", "a+")

file.write(song + '\n')

file.close()

def upload\_fav():

songs\_list.delete(0, END)

file = open(f"C:\\Users\\pa1ka\\Desktop\\Mini Project\\Users\\{user\_id}\\fav")

lst = file.read()

lst = lst.split("\n")

for s in lst:

songs\_list.insert(END, s)

def deletesong():

curr\_song = songs\_list.curselection()

songs\_list.delete(curr\_song[0])

def add\_songs():

temp\_song = tkinter.filedialog.askopenfilenames(initialdir="all songs/", title="Choose a song",

filetypes=(("mp3 Files", "\*.mp3"),))

##loop through every item in the list to insert in the listbox

for s in temp\_song:

s = s.replace("all songs/", "")

s = s.split("Mini Project/")

songs\_list.insert(END, s[1])

def Play():

song = songs\_list.get(ACTIVE)

song = f"C:\\Users\\pa1ka\\Desktop\\Mini Project\\all songs\\{song}"

mixer.music.load(song)

mixer.music.play()

# to pause the song

def Pause():

mixer.music.pause()

# to stop the song

def Stop():

mixer.music.stop()

songs\_list.selection\_clear(ACTIVE)

# to resume the song

def Resume():

mixer.music.unpause()

# Function to navigate from the current song

def Previous():

# to get the selected song index

previous\_one = songs\_list.curselection()

# to get the previous song index

previous\_one = previous\_one[0] - 1

# to get the previous song

temp2 = songs\_list.get(previous\_one)

temp2 = f'C:\\Users\\pa1ka\\Desktop\\Mini Project\\{mood}\\{temp2}'

mixer.music.load(temp2)

mixer.music.play()

songs\_list.selection\_clear(0, END)

# activate new song

songs\_list.activate(previous\_one)

# set the next song

songs\_list.selection\_set(previous\_one)

def Next():

# to get the selected song index

next\_one = songs\_list.curselection()

# to get the next song index

next\_one = next\_one[0] + 1

# to get the next song

temp = songs\_list.get(next\_one)

temp = f'C:\\Users\\pa1ka\\Desktop\\Mini Project\\{mood}\\{temp}'

mixer.music.load(temp)

mixer.music.play()

songs\_list.selection\_clear(0, END)

# activate newsong

songs\_list.activate(next\_one)

# set the next song

songs\_list.selection\_set(next\_one)

play\_button\_img = PhotoImage(file="C:\\Users\\pa1ka\\Desktop\\Mini Project\\icons\\play.png")

pause\_button\_img = PhotoImage(file="C:\\Users\\pa1ka\\Desktop\\Mini Project\\icons\\pause.png")

stop\_button\_img = PhotoImage(file="C:\\Users\\pa1ka\\Desktop\\Mini Project\\icons\\stop.png")

prev\_button\_img = PhotoImage(file="C:\\Users\\pa1ka\\Desktop\\Mini Project\\icons\\previous.png")

next\_button\_img = PhotoImage(file="C:\\Users\\pa1ka\\Desktop\\Mini Project\\icons\\next.png")

resume\_button\_img = PhotoImage(file="C:\\Users\\pa1ka\\Desktop\\Mini Project\\icons\\resume\_1..png")

# font is defined which is to be used for the button font

defined\_font = font.Font(family='Helvetica')

# play button

play\_button = Button(root, image=play\_button\_img, borderwidth=0, command=Play)

play\_button['font'] = defined\_font

play\_button.grid(row=1, column=2)

# pause button

pause\_button = Button(root, image=pause\_button\_img, borderwidth=0, command=Pause)

pause\_button['font'] = defined\_font

pause\_button.grid(row=1, column=1)

# stop button

stop\_button = Button(root, image=stop\_button\_img, borderwidth=0, command=Stop)

stop\_button['font'] = defined\_font

stop\_button.grid(row=1, column=4)

# resume button

Resume\_button = Button(root, image=resume\_button\_img, borderwidth=0, command=Resume)

Resume\_button['font'] = defined\_font

Resume\_button.grid(row=1, column=3)

# previous button

previous\_button = Button(root, image=prev\_button\_img, borderwidth=0, command=Previous)

previous\_button['font'] = defined\_font

previous\_button.grid(row=1, column=0)

# nextbutton

next\_button = Button(root, image=next\_button\_img, borderwidth=0, command=Next)

next\_button['font'] = defined\_font

next\_button.grid(row=1, column=5)

# menu

my\_menu = Menu(root)

root.config(menu = my\_menu)

add\_fav\_menu = Menu(my\_menu)

my\_menu.add\_cascade(label="Menu", menu=add\_fav\_menu)

add\_fav\_menu.add\_command(label="Add fav", command=addfav)

add\_fav\_menu.add\_command(label="Upload fav", command=upload\_fav)

add\_fav\_menu.add\_command(label="Add Songs", command=add\_songs)

add\_fav\_menu.add\_command(label="Delete Song", command=deletesong)

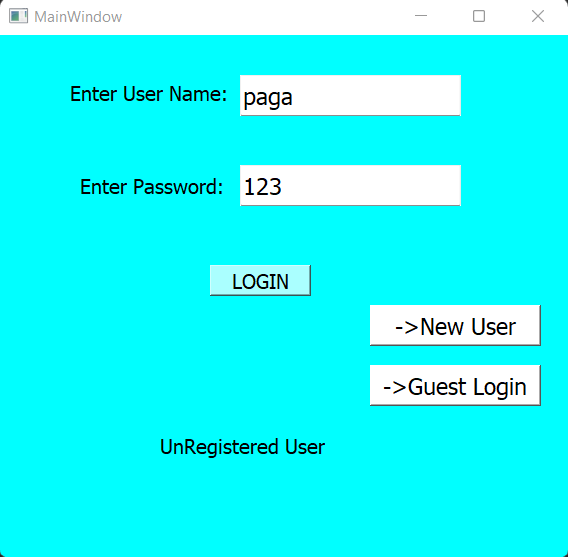
if(guest=="False"):

my\_menu.entryconfig("Menu",state = DISABLED)

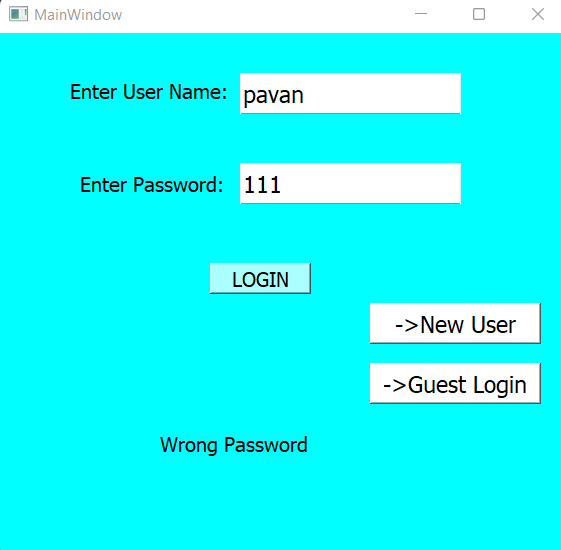
mainloop()

**SCREEN SHOTS OF TEST CASES**

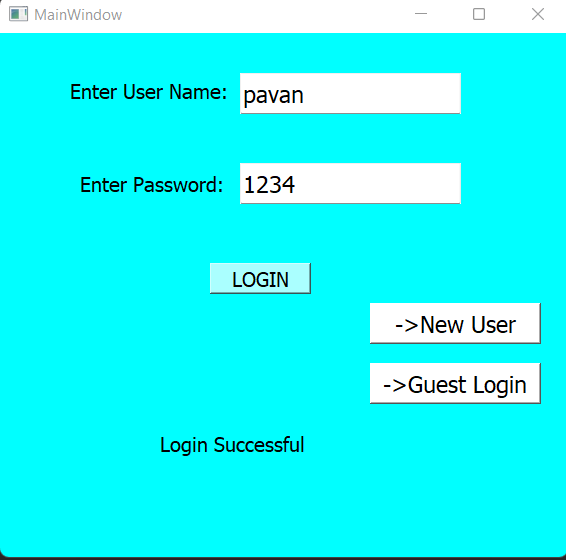
* **Login**



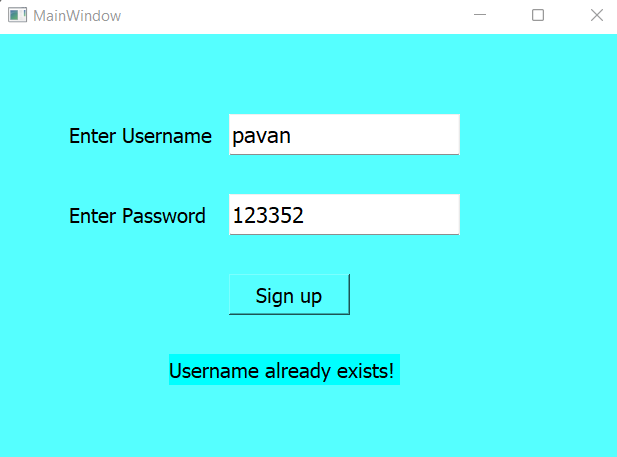
* Unregistered user test case.



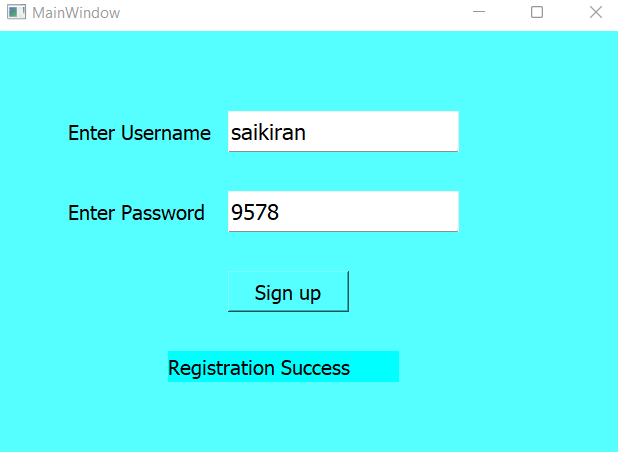
* Wrong Password test case



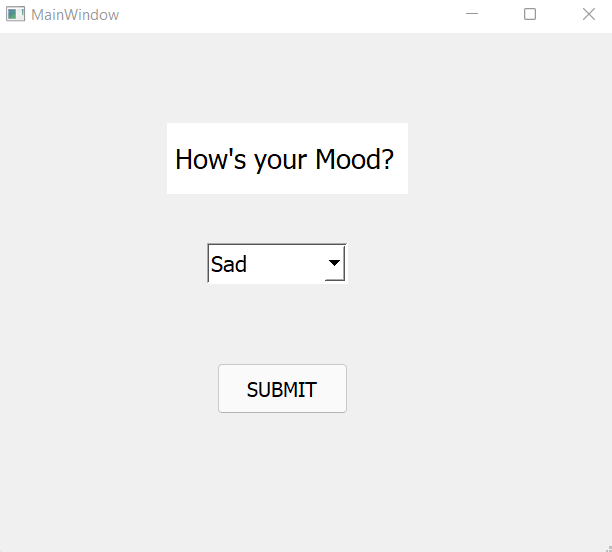
* Successful login test case
* **Signup**



* Username already exists testcase



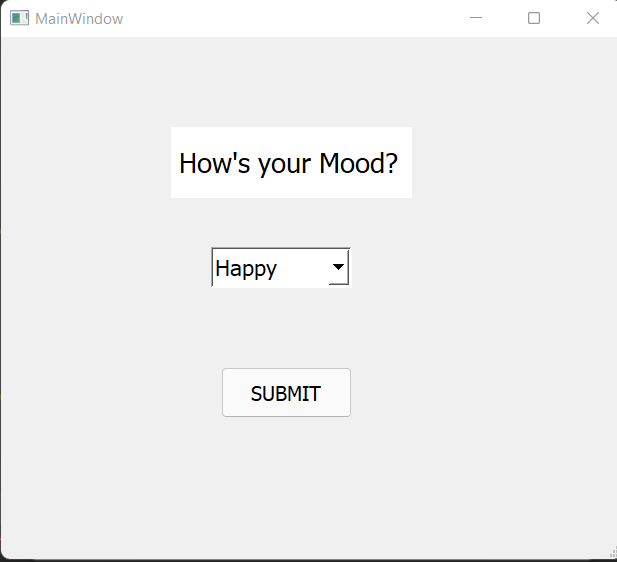
* Successful registration test case.
* **Mood selecting window and music player with playlist based on mood**



* Mood sad is selected



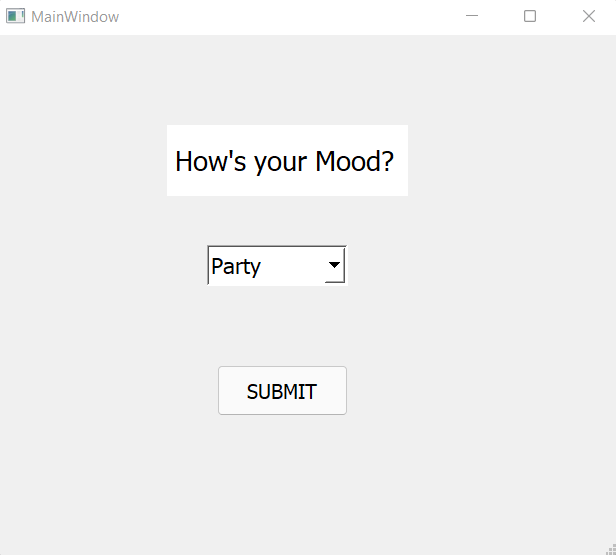
* Music player with sad playlist test case.



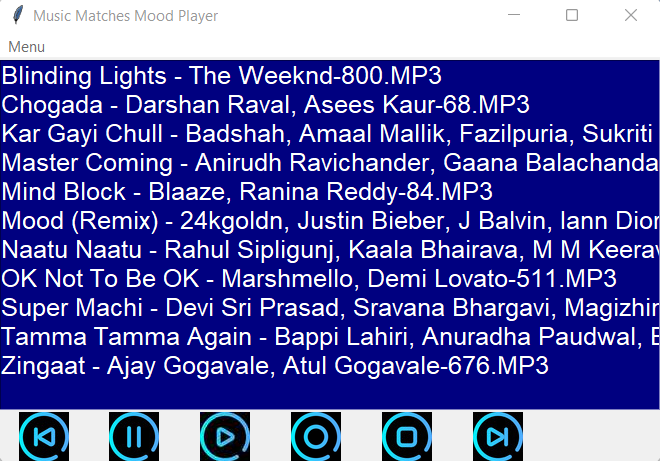
* Mood Happy is selected



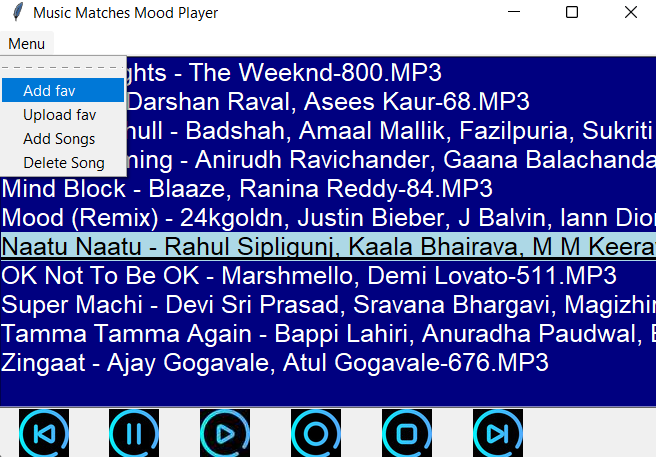
* Music player with happy playlist testcase.

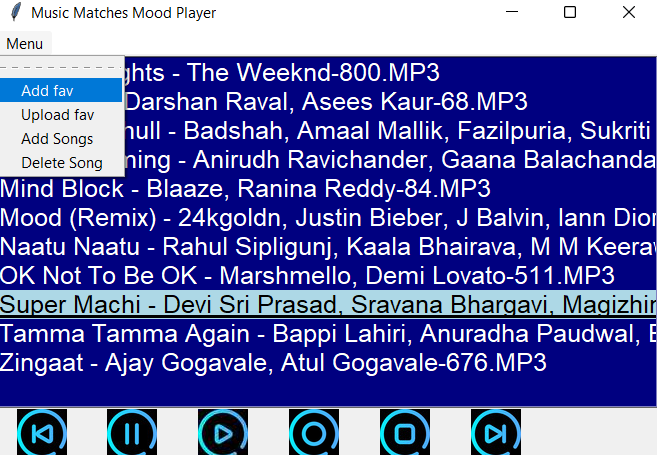


* Party mood is selected.

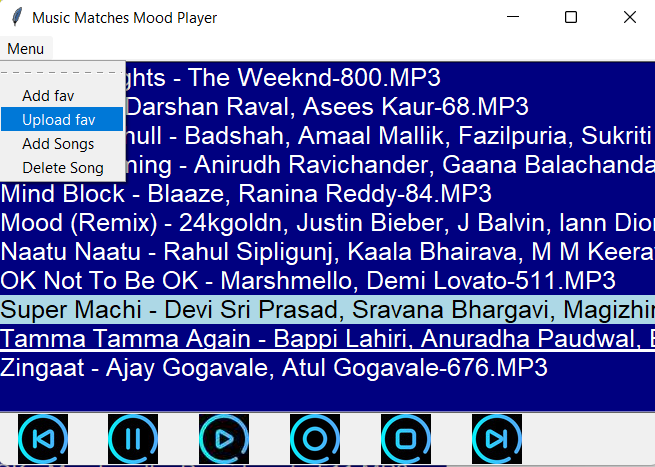


* Music player with party mood test case.
* **Music Player features:**

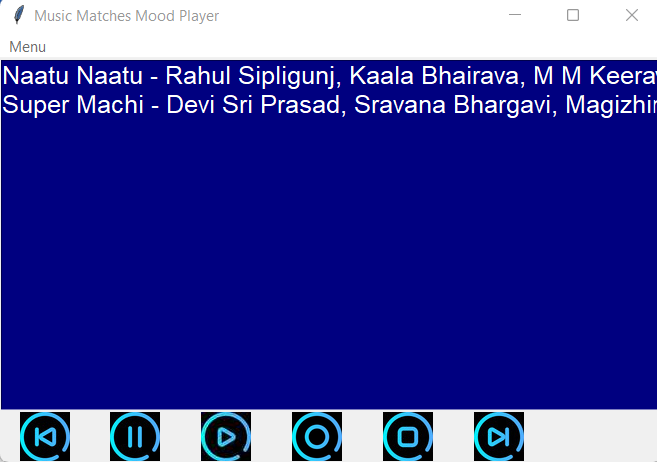




* Adding songs to favorites.



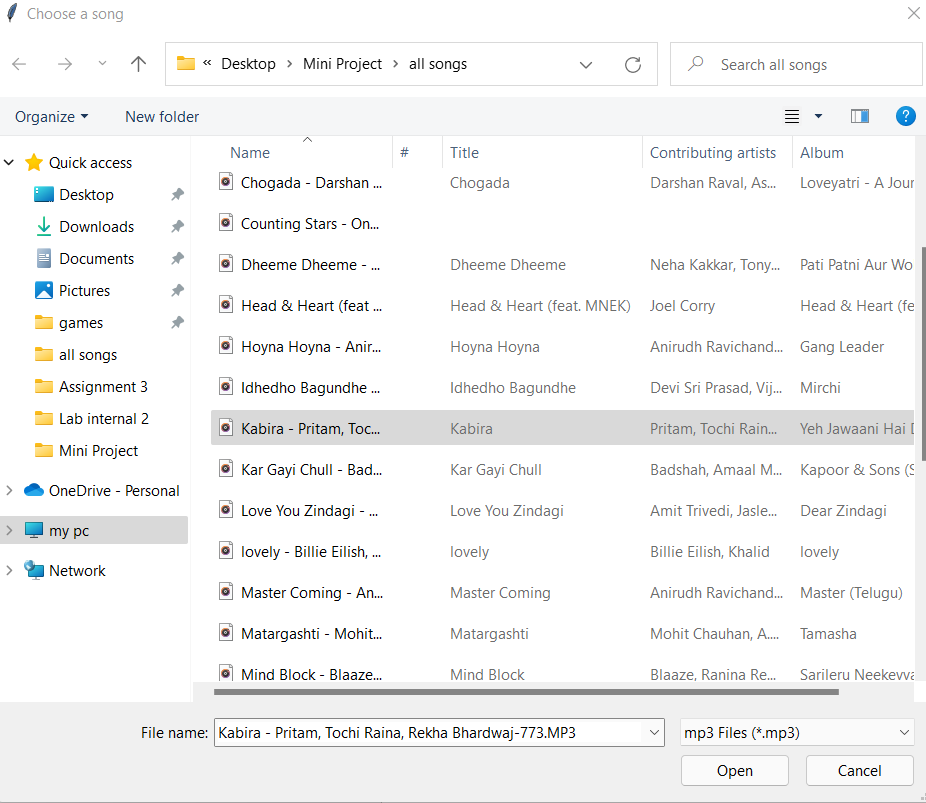
* Choosing upload favorites option.



* Favorites playlist uploaded test case.



* Selecting add songs option from menu.



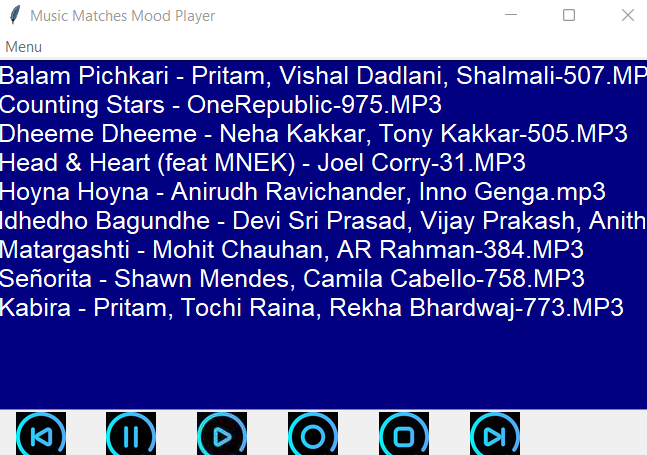
* Selecting the song which has to be added to the playlist.



* Selected song is added to the playlist.

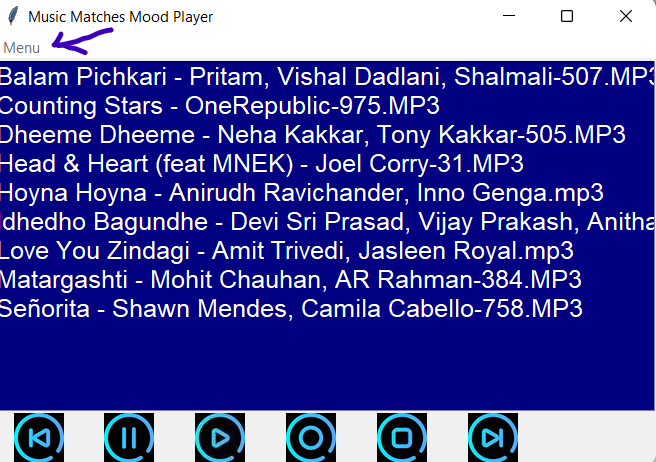


* Delete song option is selected.



* Selected song is deleted.

**Guest login**



* Menu option is disabled for the guest user.

**ADDITIONAL KNOWLEDGE ACQUIRED**

* Python GUI programming using tkinter
* Python GUI programming using pyqt5
* Using pygame module to play music
* User case diagram
* Activity diagram
* Learnt how to use qt designer

## CONCLUSION

Music matches mood player is a general music player with the additional feature of giving users the playlist based on their mood. It also has general features like adding songs to favorites, uploading favorites ,play, pause and many more.

This project also gives us an idea of how we can use the pygame module and the tkinter module to create GUI application with less effort and encourage us to create many more interesting projects.

## FUTURE WORK

Our project will implement in future after making some changes and modifications that can be done in our project are:

* Sql for storing data
* Machine learning for recommending songs
* Equalizer to amplify music
* Adding Download songs feature
* Lyrics(speech to text).

## REFERENCES

**Basic Python**

1. Course covered during 2nd semester by Dr. Ramesh Vassapanavara

Sir.

1. Ppts and handouts provided by the sir.
2. Python Programming - Using Problem Solving Approach First Edition by Reema Thareja.

**Pygame tutorials :** [**https://realpython.com/pygame-a-primer/**](https://realpython.com/pygame-a-primer/) [**https://www.youtube.com/watch?v=FfWpgLFMI7w**](https://www.youtube.com/watch?v=FfWpgLFMI7w)

[**Pygame in 90 Minutes - For Beginners**](https://www.youtube.com/watch?v=jO6qQDNa2UY)

**Tkinter tutorials:** [**Create Graphical User Interfaces With Python And TKinter**](https://youtu.be/yQSEXcf6s2I)

**Pyqt5 tutorials: https://youtu.be/Vde5SH8e1OQ**

Use cases and activity diagram

1. Hand-outs provided by DRL PRASANNA ma’am.
2. <https://www.youtube.com/watch?v=zid-MVo7M-E>
3. <https://www.youtube.com/watch?v=knM8BGY9yVI&t=161s>