**PyMusic**

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

D. SREE SURYA 1602-20-737-043

Y. VISHNU TEJA 1602-20-737-059

R. SUMANTH ABHINAY 1602-20-737-50



**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, D. SREE SURYA, Y. VISHNU TEJA and R. SUMANTH ABHINAY, bearing hall ticket numbers, 1602-20-737-043, 1602-20-737-059 and 1602-20-737-050, hereby declare that the project report entitled “PYMUSIC” 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.

D.SREE SURYA

1602-20-737-043

Y. VISHNU TEJA

1602-20-737-059

R. SUMANTH ABHINAY

1602-20-737-050

(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 Mrs.L.Divya and, 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**

In the current generation, everyone loves listening music.MP3 players have made it possible to take your whole music collection anywhere. So, in our python project **PyMusic,** we are going to create an MP3 player with the help of Python and its libraries.

Many of us don’t like any kind of disturbance or ads when we are listening to music. Hence, we have designed our MP3 player where you can listen to music uninterruptedly without ads. The user can add any MP3 songs, make a playlist or add songs to favorites.

We have used the libraries Tkinter, Pyqt5 (GUI creation), Pygame (computer graphics and sound libraries) and OS (for interaction with the Operating System).

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| TITLE | PAGE NUMBER |
| Abstract | 4 |
| Introduction | 6 |
| Technology | 7 |
| Proposed work | 8 |
| Results | 47 |
| Additional Learnings | 53 |
| Conclusion and Future work | 54 |
| References | 55 |

**INTRODUCTION**

Entertainment is anything that gives you enjoyment. Almost all of us need some sort of entertainment in life. Music is a source of entertainment. Music is said to be a universal language. Music is everywhere.

Our project “PyMusic” is all about entertainment. It has a variety of options for free which are otherwise paid features in other music players. PyMusic is also **Ad Free,** meaning interruption free music. In this player, a user can add songs, remove them, create a playlist, add songs to favorites, download songs and sort songs based on mood or actors. A user can also login as a guest, but he cannot use the same features given for the registered users.

We have used Pyqt5, pytube, mutagen, PIL modules for executing the project. For storing user details, we used files while an SQL database has been used to store playlists and favorites of the user.

**TECHNOLOGY**

To implement any project successfully, there will be technological specifications which can either be software or hardware.

**A) Software Requirements**

The project is based on Python programming language. Hence knowledge of this language is required along with an Integrated Development Environment (IDE) to write and run the programs.

IDE: Some of the manty IDEs for Python include-

1)Visual Studio Code

2)Replit

3)PyCharm

4)NetBeans

Among the many available IDEs, we have installed and used **PyCharm** to write and execute the code for “PyMusic”.

The following Python modules were used to implement various features of our Project:

1. **Pyqt5**: This is the module which was used to design the User Interface (U.I.) and navigate through the songs
2. **Os**: This module was used for accessing various directories
3. **Sqlite3**: This module is an API for SQLite Database to store URL of songs
4. **Pytube**: It is a lightweight module for downloading YouTube videos

**B) Hardware Requirements**

A personal computer or laptop with 4GB ram and an audio output (speakers or headphone) is sufficient.

This project was executed with a laptop having 8GB ram and intel i5 processor having audio output

**PROPOSED WORK**

**DESIGN**

1. USE CASE DIAGRAM

Graphical user interface

Description automatically generated

**Description:**

Use Case ID: UC01

Name: Register

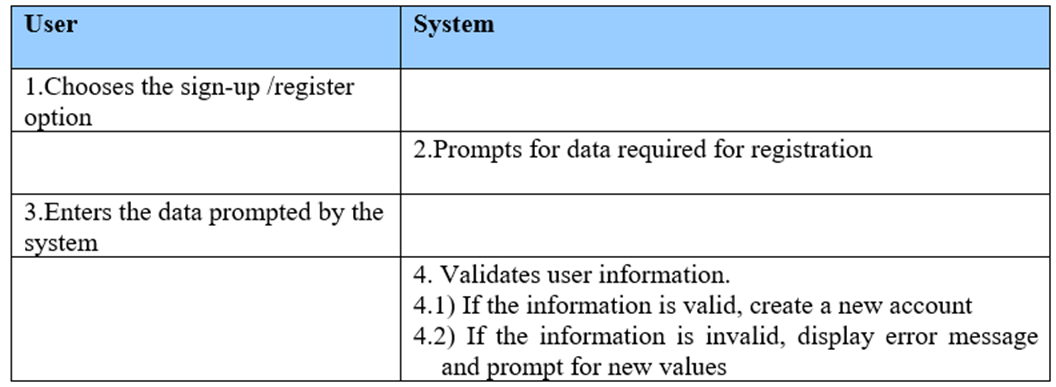
Actors: User

Description: Allows registered users to Login

Pre-conditions: user should be registered with the system

Post-conditions: User logs in and all the options are displayed on the screen

Main Flow:



Use Case ID: UC02

Name: Login

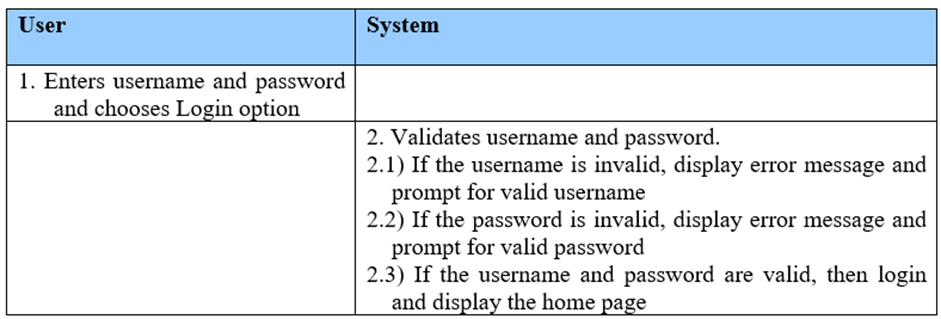
Actors: User

Description: Allows registered users to Login

Pre-conditions: user should be registered with the system

Post-conditions: User logs in and all the options are displayed on the screen

Main Flow:



Use Case ID: UC03

Name: Navigation

Actors: User

Description: Allows registered users to play, pause and can change the tracks.

Pre-conditions: User should be successfully signed-in.

Post-conditions: Tracks are played, paused and can be changed.

Main Flow:

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

Use Case ID: UC04

Name: Creating Playlists

Actors: User

Description: User creates the Playlists based on their listening choice.

Pre-conditions: User should be signed in successfully.

Post-conditions:  Playlist is created.

Main Flow:

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

Use Case ID: UC05

Name: Add to Favourites

Actors: User

Description: User adds his favourite song to favourites list.

Pre-conditions: User should be signed in successfully.

Post-conditions: Favourite Songs are added.

Main Flow:

|  |  |
| --- | --- |
| **USER** | **SYSTEM** |
| 1. User adds his favourite song to favourites list. |  |
|  | 2. System adds the favourite songs. |

Use Case ID: UC06

Name: Uploading of Songs

Actors: User, System\_admin.

Description: User or system\_admin uploads songs to the existing song library .

Pre-conditions: User should be signed in successfully.

Post-conditions: Songs are uploaded in the Songs library.

Main Flow:

|  |  |
| --- | --- |
| **USER** | **SYSTEM** |
| 1. User or System gives the path of a song to add to existing library |  |
|  | 2. System adds the song to the existing songs library |

**Use Case ID**: UC07

Name: View Lyrics

Actors: User, System Admin.

Description: System Displays the Lyrics of the Song

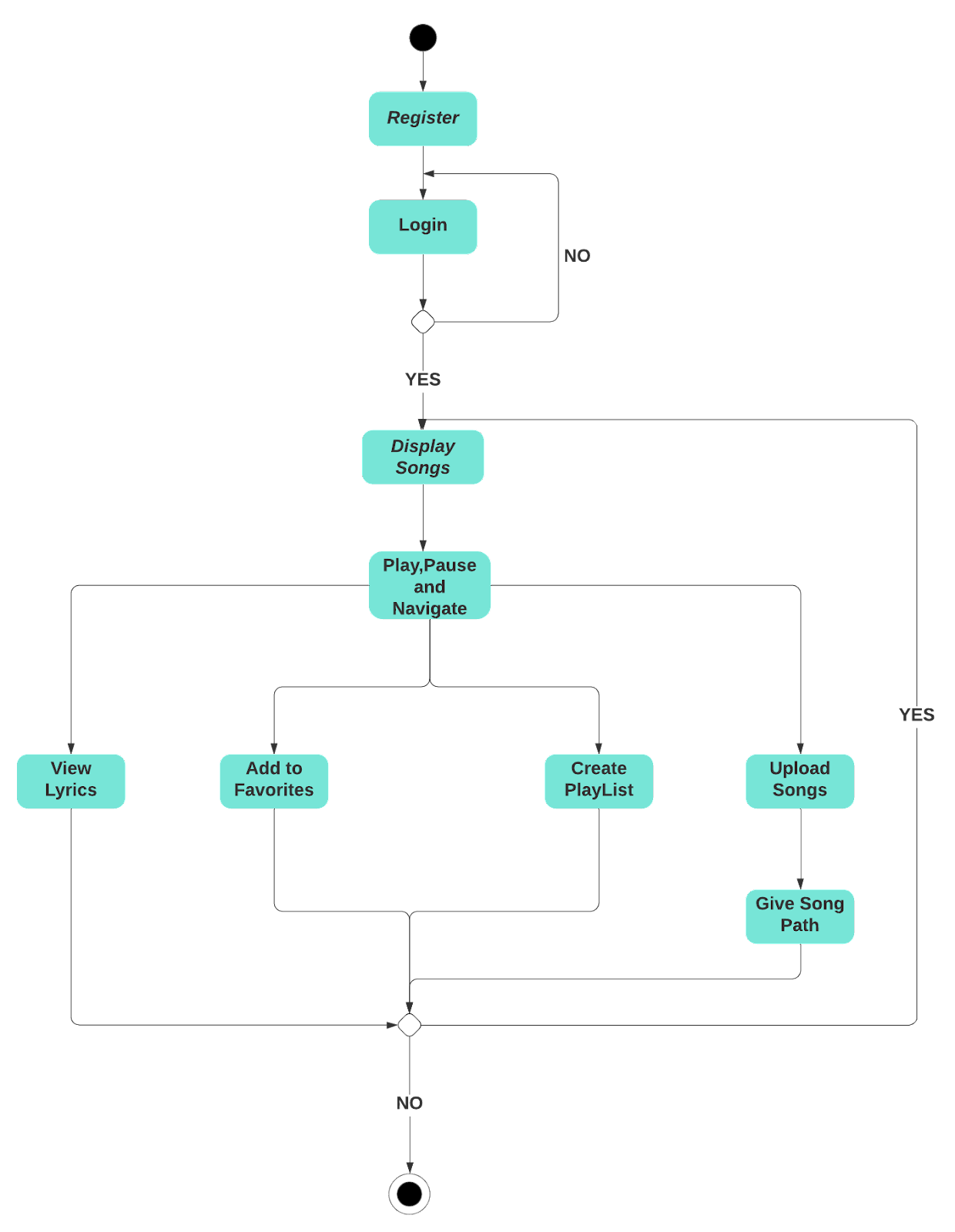
Pre-conditions: User should be signed in successfully.

Post-conditions: Lyrics are displayed.

Main Flow:

|  |  |
| --- | --- |
| **USER** | **SYSTEM** |
| 1. User chooses to view the Lyrics of the current song |  |
|  | 2. System displays the lyrics. |

2) ACTIVITY DIAGRAM



**Implementation**

1) MODULE WISE CODE

**MUSIC PLAYER U.I.**

from PyQt5 import QtCore, QtGui, QtWidgets,QtMultimedia

import ntpath

import YTSongDownload

from DownloadUI import Ui\_Dialog

import os

import sqlite3

from mutagen.id3 import ID3

from PIL import Image

class Ui\_MainWindow(object):

playing = False

paused = False

stopped = False

guestLogin = False

userid = "Guest"

def check(self):

if(self.guestLogin==True):

self.addToFavoritesButton.setDisabled(True)

self.downloadSongsMenu.setDisabled(True)

self.sortSongsMenu.setDisabled(True)

self.createPlaylistMenu.setDisabled(True)

self.importPlaylist.setDisabled(True)

self.loadFav.setDisabled(True)

def initial(self):

songs = os.listdir("Songs/")

self.songsList.addItems(songs)

for i in songs:

self.playlist.addMedia(QtMultimedia.QMediaContent(QtCore.QUrl.fromLocalFile("Songs/"+i)))

def setupUi(self, MainWindow):

self.Window=MainWindow

MainWindow.setObjectName("MainWindow")

MainWindow.resize(1122, 980)

font = QtGui.QFont()

font.setPointSize(14)

MainWindow.setFont(font)

MainWindow.setStyleSheet("background-color: rgb(0, 0, 0);\n""")

MainWindow.setTabShape(QtWidgets.QTabWidget.Rounded)

MainWindow.setWindowIcon(QtGui.QIcon("icon.ico"))

MainWindow.setFixedSize(1122,980)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

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

self.songImageLabel.setGeometry(QtCore.QRect(130, 250, 350, 350))

self.songImageLabel.setStyleSheet("border-color: rgb(255, 255, 255);\n"

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

self.songImageLabel.setText("")

self.songImageLabel.setObjectName("songImageLabel")

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

self.playButton.setEnabled(True)

self.playButton.setGeometry(QtCore.QRect(140, 780, 101, 40))

font = QtGui.QFont()

font.setFamily("Goudy Old Style")

font.setPointSize(12)

self.playButton.setFont(font)

self.playButton.setStyleSheet("background-color: rgb(85, 255, 127);")

self.playButton.setAutoDefault(False)

self.playButton.setDefault(False)

self.playButton.setFlat(False)

self.playButton.setObjectName("playButton")

#self.playButton.setIcon(QtGui.QIcon("images/play50.png"))

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

self.pauseButton.setGeometry(QtCore.QRect(340, 780, 100, 40))

font = QtGui.QFont()

font.setFamily("Goudy Old Style")

font.setPointSize(12)

self.pauseButton.setFont(font)

self.pauseButton.setStyleSheet("background-color: rgb(255, 255, 127);")

self.pauseButton.setObjectName("pauseButton")

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

self.nextButton.setGeometry(QtCore.QRect(540, 780, 100, 40))

font = QtGui.QFont()

font.setFamily("Goudy Old Style")

font.setPointSize(12)

self.nextButton.setFont(font)

self.nextButton.setStyleSheet("background-color: rgb(221, 221, 221);")

self.nextButton.setObjectName("nextButton")

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

self.previousButton.setGeometry(QtCore.QRect(740, 780, 100, 40))

font = QtGui.QFont()

font.setFamily("Goudy Old Style")

font.setPointSize(12)

self.previousButton.setFont(font)

self.previousButton.setStyleSheet("background-color: rgb(221, 221, 221);")

self.previousButton.setObjectName("previousButton")

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

self.stopButton.setGeometry(QtCore.QRect(940, 780, 100, 40))

font = QtGui.QFont()

font.setFamily("Goudy Old Style")

font.setPointSize(12)

self.stopButton.setFont(font)

self.stopButton.setStyleSheet("background-color: rgb(255, 51, 24);")

self.stopButton.setObjectName("stopButton")

self.songProgressBar = QtWidgets.QSlider(self.centralwidget)

self.songProgressBar.setGeometry(QtCore.QRect(190, 700, 700, 22))

sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Fixed)

sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.songProgressBar.sizePolicy().hasHeightForWidth())

self.songProgressBar.setSizePolicy(sizePolicy)

self.songProgressBar.setStyleSheet("background-color: rgb(0, 0, 0);")

self.songProgressBar.setOrientation(QtCore.Qt.Horizontal)

self.songProgressBar.setObjectName("songProgressBar")

self.volumeDial = QtWidgets.QDial(self.centralwidget)

self.volumeDial.setGeometry(QtCore.QRect(1000, 665, 81, 81))

self.volumeDial.setObjectName("volumeDial")

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

self.addToFavoritesButton.setGeometry(QtCore.QRect(480, 870, 171, 41))

font = QtGui.QFont()

font.setFamily("Goudy Old Style")

font.setPointSize(12)

self.addToFavoritesButton.setFont(font)

self.addToFavoritesButton.setStyleSheet("background-color: rgb(234, 255, 0);\n"

"background-color: rgb(85, 170, 255);")

self.addToFavoritesButton.setObjectName("addToFavoritesButton")

self.songsList = QtWidgets.QListWidget(self.centralwidget)

self.songsList.setGeometry(QtCore.QRect(600, 100, 500, 521))

self.songsList.setStyleSheet("background-color: rgb(118, 118, 118);\n"

"font: 14pt \"NSimSun\";\n"

"background-color: rgb(153, 153, 153);")

self.songsList.setObjectName("songsList")

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

self.addedLabel.setGeometry(QtCore.QRect(730, 870, 141, 41))

font = QtGui.QFont()

font.setFamily("MV Boli")

font.setPointSize(20)

self.addedLabel.setFont(font)

self.addedLabel.setStyleSheet("color: rgb(255, 255, 0);")

self.addedLabel.setText("ADDED !!")

self.addedLabel.setObjectName("addedLabel")

self.addedLabel.setHidden(True)

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

self.playlistLabel.setGeometry(QtCore.QRect(780, 50, 161, 41))

font = QtGui.QFont()

font.setFamily("Gabriola")

font.setPointSize(35)

self.playlistLabel.setFont(font)

self.playlistLabel.setStyleSheet("color:rgb(151, 6, 255);")

self.playlistLabel.setText("PLAYLIST")

self.playlistLabel.setObjectName("plalistLabel")

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

self.iconLabel.setGeometry(QtCore.QRect(190, 10, 240, 220))

self.iconLabel.setText("")

self.iconLabel.setPixmap(QtGui.QPixmap("images/logo3.jpeg"))

self.iconLabel.setScaledContents(True)

self.iconLabel.setObjectName("iconLabel")

MainWindow.setCentralWidget(self.centralwidget)

self.menuBar = QtWidgets.QMenuBar(MainWindow)

self.menuBar.setGeometry(QtCore.QRect(0, 0, 1122, 21))

self.menuBar.setStyleSheet("background-color: rgb(231, 231, 231);")

self.menuBar.setObjectName("menuBar")

self.songsMenu = QtWidgets.QMenu(self.menuBar)

self.songsMenu.setObjectName("songsMenu")

self.createPlaylistMenu = QtWidgets.QMenu(self.songsMenu)

self.createPlaylistMenu.setObjectName("createPlaylistMenu")

self.removeSong = QtWidgets.QAction(MainWindow)

self.removeSong.setObjectName("removeSong")

self.removeSong.setText("Remove Song")

self.downloadSongsMenu = QtWidgets.QMenu(self.menuBar)

self.downloadSongsMenu.setObjectName("downloadSongsMenu")

self.sortSongsMenu = QtWidgets.QMenu(self.menuBar)

self.sortSongsMenu.setObjectName("sortSongsMenu")

MainWindow.setMenuBar(self.menuBar)

self.statusBar = QtWidgets.QStatusBar(MainWindow)

self.statusBar.setEnabled(True)

font = QtGui.QFont()

font.setFamily("Microsoft YaHei UI")

font.setPointSize(12)

self.statusBar.setFont(font)

self.statusBar.setAutoFillBackground(False)

self.statusBar.setStyleSheet("background-color: rgb(255, 255, 255);")

self.statusBar.setObjectName("statusBar")

self.statusBar.showMessage("Welcome "+self.userid+" !!")

MainWindow.setStatusBar(self.statusBar)

self.youtubeDownload = QtWidgets.QAction(MainWindow)

self.youtubeDownload.setObjectName("youtubeDownload")

self.sortActors = QtWidgets.QAction(MainWindow)

self.sortActors.setObjectName("sortActors")

self.loadFav = QtWidgets.QAction(MainWindow)

self.loadFav.setObjectName("loadFavorites")

self.sortMood = QtWidgets.QAction(MainWindow)

self.sortMood.setObjectName("sortMood")

self.addSongs = QtWidgets.QAction(MainWindow)

self.addSongs.setObjectName("addSongs")

self.currentSongsSelect = QtWidgets.QAction(MainWindow)

self.currentSongsSelect.setObjectName("currentSongsSelect")

self.newSongsSelect = QtWidgets.QAction(MainWindow)

self.newSongsSelect.setObjectName("newSongsSelect")

self.importPlaylist = QtWidgets.QAction(MainWindow)

self.importPlaylist.setObjectName("importPlaylist")

self.clearPlaylist = QtWidgets.QAction(MainWindow)

self.clearPlaylist.setObjectName("clearPlaylist")

self.createPlaylistMenu.addAction(self.currentSongsSelect)

self.createPlaylistMenu.addAction(self.newSongsSelect)

self.songsMenu.addAction(self.addSongs)

self.songsMenu.addAction(self.createPlaylistMenu.menuAction())

self.songsMenu.addAction(self.importPlaylist)

self.songsMenu.addAction(self.loadFav)

self.songsMenu.addAction(self.clearPlaylist)

self.downloadSongsMenu.addAction(self.youtubeDownload)

self.sortSongsMenu.addAction(self.sortActors)

self.sortSongsMenu.addAction(self.sortMood)

self.menuBar.addAction(self.songsMenu.menuAction())

self.menuBar.addAction(self.removeSong)

self.menuBar.addAction(self.downloadSongsMenu.menuAction())

self.menuBar.addAction(self.sortSongsMenu.menuAction())

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

self.playTimeLabel.setGeometry(QtCore.QRect(120, 700, 51, 21))

font = QtGui.QFont()

font.setPointSize(14)

self.playTimeLabel.setFont(font)

self.playTimeLabel.setStyleSheet("color: rgb(255, 222, 57);")

self.playTimeLabel.setText("00:00")

self.playTimeLabel.setObjectName("playTimeLabel")

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

self.songDurationLabel.setGeometry(QtCore.QRect(910, 700, 51, 21))

font = QtGui.QFont()

font.setPointSize(14)

self.songDurationLabel.setFont(font)

self.songDurationLabel.setStyleSheet("color: rgb(255, 222, 57);")

self.songDurationLabel.setText("00:00")

self.songDurationLabel.setObjectName("songDurationLabel")

self.player = QtMultimedia.QMediaPlayer()

self.playlist = QtMultimedia.QMediaPlaylist()

self.player.setPlaylist(self.playlist)

self.check()

self.initial()

self.retranslateUi(MainWindow)

QtCore.QMetaObject.connectSlotsByName(MainWindow)

def retranslateUi(self, MainWindow):

\_translate = QtCore.QCoreApplication.translate

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

self.playButton.setText(\_translate("MainWindow", "PLAY"))

self.pauseButton.setText(\_translate("MainWindow", "PAUSE"))

self.nextButton.setText(\_translate("MainWindow", "NEXT"))

self.previousButton.setText(\_translate("MainWindow", "PREVIOUS"))

self.stopButton.setText(\_translate("MainWindow", "STOP"))

self.addToFavoritesButton.setText(\_translate("MainWindow", "ADD TO FAVORITES"))

self.songsMenu.setTitle(\_translate("MainWindow", "Songs"))

self.createPlaylistMenu.setTitle(\_translate("MainWindow", "Create PlayList"))

#self.removeSong.setTitle(\_translate("MainWindow", "Remove Song"))

self.downloadSongsMenu.setTitle(\_translate("MainWindow", "Download Songs"))

self.sortSongsMenu.setTitle(\_translate("MainWindow", "Sort Songs"))

self.youtubeDownload.setText(\_translate("MainWindow", "Youtube"))

self.sortActors.setText(\_translate("MainWindow", "Actors"))

self.sortMood.setText(\_translate("MainWindow", "Mood"))

self.addSongs.setText(\_translate("MainWindow", "Add Songs"))

self.currentSongsSelect.setText(\_translate("MainWindow", "Current Songs"))

self.newSongsSelect.setText(\_translate("MainWindow", "Select Songs"))

self.importPlaylist.setText(\_translate("MainWindow", "Import PlayList"))

self.clearPlaylist.setText(\_translate("MainWindow", "Clear PlayList"))

self.loadFav.setText(\_translate("MainWindow","Load Favorites"))

def showDetails(self,song):

with open('temp.jpg', 'wb') as img:

a = ID3(song)

img.write(a.getall('APIC')[0].data)

self.resizeImage()

self.songImageLabel.setPixmap(QtGui.QPixmap("temp.jpg"))

def resizeImage(self):

image = Image.open("temp.jpg")

image = image.resize((350, 350), Image.ANTIALIAS)

image.save("temp.jpg")

def insertSongs(self):

songsPath = QtWidgets.QFileDialog.getOpenFileNames(QtWidgets.QMainWindow(),"Select Songs","Songs\\","MP3 Files(\*mp3)")

print(songsPath)

if (songsPath):

for \_ in songsPath[0]:

self.songsList.addItem(ntpath.basename(\_))

self.playlist.addMedia(QtMultimedia.QMediaContent(QtCore.QUrl.fromLocalFile(\_)))

def playSong(self):

if (self.playlist.isEmpty()): pass

else:

if(self.songsList.currentRow()==-1): pass

else:

try:

if(self.paused==False):

self.player.setVolume(50)

self.volumeDial.setValue(50)

self.playlist.setCurrentIndex(self.songsList.currentRow())

self.player.play()

self.showDetails(self.playlist.currentMedia().canonicalUrl().toLocalFile())

print(self.songsList.currentRow()," ",self.playlist.currentIndex())

self.statusBar.showMessage("Now Playing : "+str(self.songsList.currentItem().text()))

except Exception:

print("Error")

def pauseSong(self):

self.player.pause()

self.paused=True

if((self.player.NoMedia) or self.player.PausedState):

self.statusBar.showMessage("Music Paused")

def stopSong(self):

self.songDurationLabel.setText("00:00")

self.player.stop()

self.songsList.clearSelection()

self.songsList.setCurrentRow(-1)

self.volumeDial.setValue(0)

self.paused=False

self.stopped=True

self.statusBar.showMessage("Music Stopped")

self.songImageLabel.clear()

def nextSong(self):

currentSong = self.songsList.currentRow()

if(currentSong==-1): pass

else:

nextSong = currentSong +1

if(nextSong==self.songsList.count()):

nextSong=0

self.songsList.clearSelection()

self.songsList.setCurrentRow(nextSong)

self.paused=False

self.playSong()

def previousSong(self):

currentSong = self.songsList.currentRow()

if (currentSong == -1):

pass

else:

previousSong = currentSong - 1

if (previousSong == -1):

previousSong = self.songsList.count()-1

self.songsList.clearSelection()

self.songsList.setCurrentRow(previousSong)

self.paused = False

self.playSong()

def addFavorite(self):

if(self.songsList.currentRow()==-1):pass

else:

connection = sqlite3.connect("Users/" + self.userid + "/Favorites.sqlite3")

cur = connection.cursor()

cur.execute("create table if not exists Favorites (Songs varchar)")

media = self.playlist.media(self.songsList.currentRow())

filename = media.canonicalUrl().toLocalFile()

cur.execute(' INSERT INTO Favorites (Songs) VALUES (?) ',(filename,))

connection.commit()

self.statusBar.showMessage("Added "+self.songsList.currentItem().text()+" to Favorites !")

def loadFavorites(self):

self.clear()

connection = sqlite3.connect("Users/" + self.userid + "/Favorites.sqlite3")

cur = connection.cursor()

cur.execute("SELECT \* FROM FAVORITES")

songsPath = cur.fetchall()

songs=[]

for i in songsPath:

songs.append(i[0])

for \_ in songs:

self.playlist.addMedia(QtMultimedia.QMediaContent(QtCore.QUrl.fromLocalFile(\_)))

self.songsList.addItem(ntpath.basename(\_))

def setVolume(self):

self.player.setVolume(self.volumeDial.value())

def seekSong(self):

self.songProgressBar.setFocus()

if self.player.state() == QtMultimedia.QMediaPlayer.PlayingState or self.player.state() == QtMultimedia.QMediaPlayer.PausedState:

seek = self.songProgressBar.value()

self.player.setPosition(seek)

def formatTime(self,dur):

hour, r = divmod(dur, 3600000)

minute, r = divmod(r, 60000)

second, \_ = divmod(r, 1000)

return ("%d:%02d:%02d" % (hour, minute, second)) if hour else ("%d:%02d" % (minute, second))

def sliderMove(self, position):

self.songProgressBar.setValue(position)

self.playTimeLabel.setText(self.formatTime(position))

self.songDurationLabel.setText(self.formatTime(self.player.duration()))

# self.infolabel.setText(

#"{now} / {total}".format(now=self.formatTime(position), total=self.formatTime(self.player.duration())))

def sliderMax(self, duration):

self.songProgressBar.setRange(0, duration)

def downloadUI(self):

self.ui = Ui\_Dialog()

self.dialog=QtWidgets.QDialog()

self.ui.setupUi(self.dialog)

self.ui.downloadButton.clicked.connect(self.downloadYTSongs)

self.dialog.show()

def downloadYTSongs(self):

url = self.ui.urlTextField.text()

#https://www.youtube.com/watch?v=BuW1SqwH\_g0

try:

YTSongDownload.run(url,self.userid)

self.ui.downloadedIconLabel.setHidden(False)

self.ui.downloadSuccessLabel.setHidden(False)

QtCore.QTimer.singleShot(2000, self.dialog.close)

except Exception:

error = QtWidgets.QMessageBox()

error.setIcon(QtWidgets.QMessageBox.Critical)

error.setText("URL not found !!")

error.setWindowTitle("ERROR")

error.show()

error.exec\_()

def clear(self):

self.player.stop()

self.statusBar.showMessage("Playlist Cleared !")

self.playlist.clear()

self.songsList.clear()

self.songImageLabel.clear()

def remove(self):

#print("remove")

if(self.songsList.currentRow()== -1):pass

else:

self.playlist.removeMedia(self.songsList.currentRow())

x=self.songsList.takeItem(self.songsList.currentRow())

self.stopSong()

self.statusBar.showMessage("Removed: "+x.text())

self.songDurationLabel.setText("00:00")

def createPlaylistCurrent(self):

totalSongs = self.playlist.mediaCount()

currSongs=[]

if (len(currSongs) > 0):

currSongs.clear()

if(totalSongs==0): pass

else:

playListName,status = QtWidgets.QInputDialog.getText(QtWidgets.QMainWindow(),"Playlist Name", "Enter Playlist Name: ")

if status:

connection = sqlite3.connect("Users/"+self.userid+"/Playlists/"+playListName+".sqlite3")

cursor = connection.cursor()

cursor.executescript("CREATE TABLE "+playListName+" (Songs varchar);")

for i in range(totalSongs):

media = self.playlist.media(i)

filename = media.canonicalUrl().toLocalFile()

currSongs.append(filename)

for i in currSongs:

cursor.execute(''' INSERT INTO {} (Songs) VALUES (?) '''.format(playListName), (i,))

connection.commit()

self.statusBar.showMessage("Playlist '"+playListName+"' created successfully !")

def importSongs(self):

songPlaylist = QtWidgets.QFileDialog.getOpenFileName(QtWidgets.QMainWindow(),"Select Playlist","Users/"+self.userid+"/Playlists","Database Files(\*.sqlite3)")

print(songPlaylist)

if(songPlaylist[0] != ""):

database = songPlaylist[0]

databaseName = os.path.split(database)[1]

tableName = os.path.splitext(databaseName)[0]

songs=[]

if (len(songs)>0):

songs.clear()

connection = sqlite3.connect(database)

cursor = connection.cursor()

cursor.execute(''' SELECT \* FROM {} '''.format(tableName))

playlist = cursor.fetchall()

for i in playlist:

songs.append(i[0])

for \_ in songs:

self.playlist.addMedia(QtMultimedia.QMediaContent(QtCore.QUrl.fromLocalFile(\_)))

self.songsList.addItem(ntpath.basename(\_))

self.statusBar.showMessage("Playlist Imported !")

def selectSongs(self):

currSongs = []

if (len(currSongs) > 0):

currSongs.clear()

songsPath = QtWidgets.QFileDialog.getOpenFileNames(QtWidgets.QMainWindow(), "Select Songs",

"C:\\Users\\SURYA DEVARAKONDA\\Desktop\\Mini Project\\Songs",

"MP3 Files(\*mp3)")

if(songsPath):

playListName, status = QtWidgets.QInputDialog.getText(QtWidgets.QMainWindow(), "Playlist Name",

"Enter Playlist Name: ")

if(status):

connection = sqlite3.connect("Users/" + self.userid+ "/Playlists/" + playListName + ".sqlite3")

cursor = connection.cursor()

cursor.executescript("CREATE TABLE " + playListName + " (Songs varchar);")

for i in songsPath[0]:

cursor.execute(''' INSERT INTO {} (Songs) VALUES (?) '''.format(playListName), (i,))

connection.commit()

self.statusBar.showMessage("Playlist '" + playListName + "' created successfully !")

def actorSongs(self):

def selected(self):

self.stopSong()

self.clear()

self.Actors.close()

op = self.actorUi.actorsList.currentIndex()

path="Actor songs/"

if(op==0):

path+="MB/"

elif(op==1):

path+="VD/"

elif(op==2):

path+="PK/"

elif(op==3):

path+="AA/"

elif(op==4):

path+="Prabhas/"

file = os.listdir(path)

self.songsList.addItems(file)

for i in file:

self.playlist.addMedia(QtMultimedia.QMediaContent(QtCore.QUrl.fromLocalFile(os.path.join(path)+i)))

from ActorSortUI import Ui\_Actors

self.actorUi = Ui\_Actors()

self.Actors = QtWidgets.QDialog()

self.actorUi.setupUi(self.Actors)

self.actorUi.submitButton.clicked.connect(lambda : selected(self))

self.Actors.show()

def events(self):

self.playButton.clicked.connect(self.playSong)

self.pauseButton.clicked.connect(self.pauseSong)

self.stopButton.clicked.connect(self.stopSong)

self.nextButton.clicked.connect(self.nextSong)

self.previousButton.clicked.connect(self.previousSong)

self.addSongs.triggered.connect(self.insertSongs)

self.songProgressBar.sliderReleased.connect(self.seekSong)

self.volumeDial.valueChanged.connect(self.setVolume)

self.youtubeDownload.triggered.connect(self.downloadUI)

self.clearPlaylist.triggered.connect(self.clear)

self.removeSong.triggered.connect(self.remove)

self.player.positionChanged.connect(self.sliderMove)

self.player.durationChanged.connect(self.sliderMax)

self.addToFavoritesButton.clicked.connect(self.addFavorite)

self.currentSongsSelect.triggered.connect(self.createPlaylistCurrent)

self.importPlaylist.triggered.connect(self.importSongs)

self.loadFav.triggered.connect(self.loadFavorites)

self.newSongsSelect.triggered.connect(self.selectSongs)

self.sortActors.triggered.connect(self.actorSongs)

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

import sys

app = QtWidgets.QApplication(sys.argv)

MainWindow = QtWidgets.QMainWindow()

ui = Ui\_MainWindow()

ui.setupUi(MainWindow)

ui.events()

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

**LOGIN U.I.**

from PyQt5 import QtCore, QtGui, QtWidgets

from SignupUI import Ui\_SignupUI

from os import listdir

import MusicPlayer

class Ui\_LoginUI(object):

def setupUi(self, LoginUI):

LoginUI.setObjectName("LoginUI")

LoginUI.resize(641, 544)

LoginUI.setWindowIcon(QtGui.QIcon("images/login.jpg"))

self.loginLabel = QtWidgets.QLabel(LoginUI)

self.loginLabel.setGeometry(QtCore.QRect(230, 40, 81, 41))

font = QtGui.QFont()

font.setPointSize(16)

self.loginLabel.setFont(font)

self.loginLabel.setObjectName("loginLabel")

self.usernameLabel = QtWidgets.QLabel(LoginUI)

self.usernameLabel.setGeometry(QtCore.QRect(90, 150, 131, 31))

font = QtGui.QFont()

font.setPointSize(14)

self.usernameLabel.setFont(font)

self.usernameLabel.setObjectName("usernameLabel")

self.passwordLabel = QtWidgets.QLabel(LoginUI)

self.passwordLabel.setGeometry(QtCore.QRect(90, 220, 131, 31))

font = QtGui.QFont()

font.setPointSize(14)

self.passwordLabel.setFont(font)

self.passwordLabel.setStyleSheet("border-color: rgb(255, 255, 127);")

self.passwordLabel.setObjectName("passwordLabel")

self.loginButton = QtWidgets.QPushButton(LoginUI)

self.loginButton.setGeometry(QtCore.QRect(200, 300, 111, 41))

font = QtGui.QFont()

font.setPointSize(10)

self.loginButton.setFont(font)

self.loginButton.setObjectName("loginButton")

self.usernameText = QtWidgets.QLineEdit(LoginUI)

self.usernameText.setGeometry(QtCore.QRect(270, 150, 181, 31))

font = QtGui.QFont()

font.setPointSize(12)

self.usernameText.setFont(font)

self.usernameText.setObjectName("usernameText")

self.passwordText = QtWidgets.QLineEdit(LoginUI)

self.passwordText.setGeometry(QtCore.QRect(270, 220, 181, 31))

font = QtGui.QFont()

font.setPointSize(12)

self.passwordText.setFont(font)

self.passwordText.setObjectName("passwordText")

self.passwordText.setEchoMode(QtWidgets.QLineEdit.Password)

self.newuserLink = QtWidgets.QCommandLinkButton(LoginUI)

self.newuserLink.setGeometry(QtCore.QRect(380, 350, 161, 48))

self.newuserLink.setObjectName("newuserLink")

self.loginGuestLink = QtWidgets.QCommandLinkButton(LoginUI)

self.loginGuestLink.setGeometry(QtCore.QRect(380, 420, 222, 48))

self.loginGuestLink.setObjectName("loginGuestLink")

self.displayStatus = QtWidgets.QLabel(LoginUI)

self.displayStatus.setGeometry(QtCore.QRect(220, 480, 200, 40))

font.setPointSize(14)

self.displayStatus.setFont(font)

self.loginButton.clicked.connect(self.login)

self.newuserLink.clicked.connect(self.newUser)

self.loginGuestLink.clicked.connect(self.guestLogin)

self.retranslateUi(LoginUI)

QtCore.QMetaObject.connectSlotsByName(LoginUI)

def retranslateUi(self, LoginUI):

\_translate = QtCore.QCoreApplication.translate

LoginUI.setWindowTitle(\_translate("LoginUI", "LOGIN PAGE"))

self.loginLabel.setText(\_translate("LoginUI", "LOGIN"))

self.usernameLabel.setText(\_translate("LoginUI", "USERNAME"))

self.passwordLabel.setText(\_translate("LoginUI", "PASSWORD"))

self.loginButton.setText(\_translate("LoginUI", "LOGIN"))

self.newuserLink.setText(\_translate("LoginUI", "NEW USER?"))

self.loginGuestLink.setText(\_translate("LoginUI", "LOGIN AS GUEST"))

def login(self):

self.userid = self.usernameText.text()

pssw = self.passwordText.text()

if self.userid in listdir("Users"):

file = open("Users/"+self.userid+"/login", "r")

verify = file.read().splitlines()

if pssw in verify:

self.displayStatus.setText("Success")

self.loginSuccess(0)

else:

self.displayStatus.setText("Wrong Password ")

print("Wrong Password")

else:

self.displayStatus.setText("UnRegistered User")

print("User Not registered")

def loginSuccess(self,flag):

self.musicWindow = QtWidgets.QMainWindow()

self.musicUi = MusicPlayer.Ui\_MainWindow()

if(flag==0):

print("Login Successful")

self.displayStatus.setText("Login Successful")

self.musicUi.userid = self.userid

else:

print("Guest Login")

self.displayStatus.setText("Guest Login")

self.musicUi.guestLogin = True

QtCore.QTimer.singleShot(1000, LoginUI.close)

self.musicUi.setupUi(self.musicWindow)

self.musicUi.events()

self.musicWindow.show()

def newUser(self):

self.newWindow = QtWidgets.QDialog()

self.ui = Ui\_SignupUI()

self.ui.setupUi(self.newWindow)

self.newWindow.show()

def guestLogin(self):

self.loginSuccess(1)

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

import sys

app = QtWidgets.QApplication(sys.argv)

LoginUI = QtWidgets.QDialog()

ui = Ui\_LoginUI()

ui.setupUi(LoginUI)

LoginUI.show()

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

**SIGN UP U.I**

import os

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_SignupUI(object):

def setupUi(self, SignupUI):

SignupUI.setObjectName("SignupUI")

SignupUI.resize(563, 441)

SignupUI.setWindowTitle("REGISTER")

SignupUI.setWindowIcon(QtGui.QIcon("images/newuser.png"))

self.signupLabel = QtWidgets.QLabel(SignupUI)

self.signupLabel.setGeometry(QtCore.QRect(210, 40, 111, 41))

font = QtGui.QFont()

font.setPointSize(16)

self.signupLabel.setFont(font)

self.signupLabel.setObjectName("signupLabel")

self.usernameLabel = QtWidgets.QLabel(SignupUI)

self.usernameLabel.setGeometry(QtCore.QRect(40, 150, 171, 31))

font = QtGui.QFont()

font.setPointSize(14)

self.usernameLabel.setFont(font)

self.usernameLabel.setObjectName("usernameLabel")

self.passwordLabel = QtWidgets.QLabel(SignupUI)

self.passwordLabel.setGeometry(QtCore.QRect(40, 220, 171, 31))

font = QtGui.QFont()

font.setPointSize(14)

self.passwordLabel.setFont(font)

self.passwordLabel.setStyleSheet("border-color: rgb(255, 255, 127);")

self.passwordLabel.setObjectName("passwordLabel")

self.signupButton = QtWidgets.QPushButton(SignupUI)

self.signupButton.setGeometry(QtCore.QRect(220, 310, 101, 31))

font = QtGui.QFont()

font.setPointSize(10)

self.signupButton.setFont(font)

self.signupButton.setObjectName("signupButton")

font = QtGui.QFont()

font.setPointSize(12)

self.usernameText = QtWidgets.QLineEdit(SignupUI)

self.usernameText.setGeometry(QtCore.QRect(280, 150, 171, 31))

self.usernameText.setObjectName("usernameText")

self.usernameText.setFont(font)

self.passwordText = QtWidgets.QLineEdit(SignupUI)

self.passwordText.setGeometry(QtCore.QRect(280, 220, 171, 31))

self.passwordText.setObjectName("passwordText")

self.passwordText.setFont(font)

self.signupButton.clicked.connect(lambda : self.register(SignupUI))

self.displayStatus = QtWidgets.QLabel(SignupUI)

self.displayStatus.setGeometry(QtCore.QRect(190,370,220,30))

font.setPointSize(12)

self.displayStatus.setFont(font)

self.displayStatus.setText("Registration Success !")

self.displayStatus.hide()

self.retranslateUi(SignupUI)

QtCore.QMetaObject.connectSlotsByName(SignupUI)

def retranslateUi(self, SignupUI):

\_translate = QtCore.QCoreApplication.translate

SignupUI.setWindowTitle(\_translate("SignupUI", "REGISTER"))

self.signupLabel.setText(\_translate("SignupUI", "SIGN UP"))

self.usernameLabel.setText(\_translate("SignupUI", "Enter UserName"))

self.passwordLabel.setText(\_translate("SignupUI", "Enter Password"))

self.signupButton.setText(\_translate("SignupUI", "Sign Up !"))

def register(self,win):

userid = self.usernameText.text()

pssw = self.passwordText.text()

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

self.displayStatus.setText(" Wrong Input")

else:

try:

os.mkdir("Users/"+userid)

os.mkdir("Users/"+userid+"/Downloads")

os.mkdir("Users/" + userid + "/Playlists")

file = open("Users/"+userid+"/login","w")

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

file.close()

self.displayStatus.setHidden(False)

print("Registration Successful")

QtCore.QTimer.singleShot(750, win.close)

except Exception:

self.displayStatus.setText(" UserName already exists !")

**DOWNLOAD U.I**

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_Dialog(object):

def setupUi(self, Dialog):

Dialog.setObjectName("Dialog")

Dialog.resize(500, 400)

self.urlLabel = QtWidgets.QLabel(Dialog)

self.urlLabel.setGeometry(QtCore.QRect(40, 40, 211, 16))

font = QtGui.QFont()

font.setFamily("Mongolian Baiti")

font.setPointSize(14)

font.setBold(True)

font.setWeight(75)

self.urlLabel.setFont(font)

self.urlLabel.setObjectName("urlLabel")

self.urlTextField = QtWidgets.QLineEdit(Dialog)

self.urlTextField.setGeometry(QtCore.QRect(40, 90, 400, 28))

font = QtGui.QFont()

font.setPointSize(12)

self.urlTextField.setFont(font)

self.urlTextField.setText("")

self.urlTextField.setObjectName("urlTextField")

self.downloadButton = QtWidgets.QPushButton(Dialog)

self.downloadButton.setGeometry(QtCore.QRect(160, 130, 121, 41))

font = QtGui.QFont()

font.setPointSize(12)

self.downloadButton.setFont(font)

self.downloadButton.setObjectName("downloadButton")

self.exampleLabel = QtWidgets.QLabel(Dialog)

self.exampleLabel.setEnabled(False)

self.exampleLabel.setGeometry(QtCore.QRect(40, 200, 430, 21))

self.exampleLabel.setStyleSheet("")

self.exampleLabel.setText("Example: https://www.youtube.com/watch?v=ZYzbalQ6Lg8")

self.exampleLabel.setObjectName("exampleLabel")

self.downloadSuccessLabel = QtWidgets.QLabel(Dialog)

self.downloadSuccessLabel.setGeometry(QtCore.QRect(160, 280, 171, 41))

font = QtGui.QFont()

font.setPointSize(14)

self.downloadSuccessLabel.setFont(font)

self.downloadSuccessLabel.setObjectName("downloadSuccessLabel")

self.downloadedIconLabel = QtWidgets.QLabel(Dialog)

self.downloadedIconLabel.setEnabled(True)

self.downloadedIconLabel.setGeometry(QtCore.QRect(80, 280, 51, 41))

self.downloadedIconLabel.setStyleSheet("background-color: rgb(235, 235, 235);")

self.downloadedIconLabel.setText("")

self.downloadedIconLabel.setPixmap(QtGui.QPixmap("images/downloadsuccess.png"))

self.downloadedIconLabel.setScaledContents(True)

self.downloadedIconLabel.setObjectName("downloadedIconLabel")

self.label = QtWidgets.QLabel(Dialog)

self.label.setGeometry(QtCore.QRect(170, 20, 120, 51))

self.label.setText("")

self.label.setPixmap(QtGui.QPixmap("images/youtubelogo.jpg"))

self.label.setScaledContents(True)

self.label.setObjectName("label")

self.downloadedIconLabel.hide()

self.downloadSuccessLabel.hide()

self.retranslateUi(Dialog)

QtCore.QMetaObject.connectSlotsByName(Dialog)

def retranslateUi(self, Dialog):

\_translate = QtCore.QCoreApplication.translate

Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))

self.urlLabel.setText(\_translate("Dialog", "Enter URL:"))

self.downloadButton.setText(\_translate("Dialog", "Download !!"))

self.downloadSuccessLabel.setText(\_translate("Dialog", "Download Success !"))

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

import sys

app = QtWidgets.QApplication(sys.argv)

Dialog = QtWidgets.QDialog()

ui = Ui\_Dialog()

ui.setupUi(Dialog)

Dialog.show()

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

**ACTOR SORT U.I.**

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui\_Actors(object):

def setupUi(self, Actors):

Actors.setObjectName("Actors")

Actors.resize(435, 217)

self.actorsList = QtWidgets.QComboBox(Actors)

self.actorsList.setGeometry(QtCore.QRect(110, 100, 171, 31))

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

self.actorsList.setObjectName("actorsList")

self.actorsList.addItem("")

self.actorsList.addItem("")

self.actorsList.addItem("")

self.actorsList.addItem("")

self.actorsList.addItem("")

self.submitButton = QtWidgets.QPushButton(Actors)

self.submitButton.setGeometry(QtCore.QRect(150, 150, 101, 41))

font = QtGui.QFont()

font.setFamily("Microsoft YaHei UI")

font.setPointSize(12)

font.setBold(False)

font.setItalic(False)

font.setWeight(50)

self.submitButton.setFont(font)

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

"font: 12pt \"Microsoft YaHei UI\";")

self.submitButton.setObjectName("submitButton")

self.actorLabel = QtWidgets.QLabel(Actors)

self.actorLabel.setGeometry(QtCore.QRect(30, 50, 171, 31))

self.actorLabel.setStyleSheet("font: 14pt \"MS Reference Sans Serif\";")

self.actorLabel.setObjectName("actorLabel")

self.retranslateUi(Actors)

QtCore.QMetaObject.connectSlotsByName(Actors)

def retranslateUi(self, Actors):

\_translate = QtCore.QCoreApplication.translate

Actors.setWindowTitle(\_translate("Actors", "Sort by Actors"))

self.actorsList.setItemText(0, \_translate("Actors", "Mahesh Babu"))

self.actorsList.setItemText(1, \_translate("Actors", "Vijay Devarkonda"))

self.actorsList.setItemText(2, \_translate("Actors", "Pawan Kalyan"))

self.actorsList.setItemText(3, \_translate("Actors", "Allu Arjun"))

self.actorsList.setItemText(4, \_translate("Actors", "Prabhas"))

self.submitButton.setText(\_translate("Actors", "Select"))

self.actorLabel.setText(\_translate("Actors", "Choose an Actor: "))

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

import sys

app = QtWidgets.QApplication(sys.argv)

Actors = QtWidgets.QDialog()

ui = Ui\_Actors()

ui.setupUi(Actors)

Actors.show()

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

**YT SONG DOWNLOAD**

from pytube import YouTube

from moviepy.editor import \*

import os

def run(url,id):

yt = YouTube(url)

t = yt.streams.filter(file\_extension="mp4").first()

t.download("Users/"+id+"/Downloads")

videoFile = "Users/"+id+"/Downloads/"+yt.title+".mp4"

audioFile="Users/"+id+"/Downloads/"+yt.title+".mp3"

convert(videoFile,audioFile)

def convert(mp4file, mp3file):

video = VideoFileClip(mp4file)

audio = video.audio

audio.write\_audiofile(mp3file)

audio.close()

video.close()

os.remove(mp4file)

**Git Hub Link:** [**https://github.com/thedevarakonda/PyMusic**](https://github.com/thedevarakonda/PyMusic)

**Testing and Results**

Graphical user interface

Description automatically generated

1. This is the first window that shows up after running the code. It is the login page where a registered user can Sign in, a new user can Sign up or login as a guest

Graphical user interface

Description automatically generated

1. This is the Sign-up window for a new user. After entering the details, they will be stored in the system using files

Graphical user interface, application

Description automatically generated

Graphical user interface

Description automatically generated

1. After successful login, the login window closes, and the music player window pops up with an initial playlist.

Graphical user interface

Description automatically generated

1. A song from the playlist is selected and played. Here there are several options:

* Pause the song (pause button)
* Unpause the song (play button)
* Stop the song (stop button)
* Go to next or previous song (next, previous buttons)
* Add the song to favorites (add to favorites button)
* Move the song forward or backward (using the slider)
* Adjust the volume (using the volume dial)
* The song cover art is also shown

Graphical user interface

Description automatically generated Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

1. These are the features on the menu bar:

* Add songs from any directory in the system
* Create playlists from existing songs or select songs from any directory in the system.
* Import the playlist that has been created
* Load the songs that has been added as favorite in the playlist
* Clear the playlist
* Remove a song from the playlist
* Download songs from YouTube
* Sorting songs based on Actors or Mood

A screenshot of a computer

Description automatically generated Graphical user interface, text

Description automatically generated

1. Downloading music from YouTube.

After providing the correct URL, the video will be downloaded in mp3 format

Graphical user interface, application

Description automatically generated

1. After selecting an actor, their song will be shown on the playlist

Graphical user interface, application

Description automatically generated

1. Selecting songs and adding them to playlist

A screenshot of a computer

Description automatically generated

1. Importing created playlist

Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

1. If “Login as Guest” is chosen, then most of the features are disabled except Add Songs, Clear Playlist and Remove Song

**Additional Learnings**

This project has been a great learning curve for all the team members. Instead of taking this Mini Project as a subject in our curriculum, we have taken it as a learning opportunity to explore Python and its features in depth.

We have learnt various new modules such as pyqt5, mutagen, PIL, pytube, moviepy etc. It has given a lot of insight into Python. Apart from technical knowledge, we have learnt soft skills like patience, teamwork, planning and management. At every stage, we encountered lot of errors, but we were patient to deal with them. We also had to self-learn and search for a lot of modules which helped in our researching skills.

Apart from this, interactions with our seniors and mentors Mrs. L. Divya and Ms. S. Rajya Laxmi has provided invaluable inputs for our project.

**Conclusion and Future work**

PyMusic is a general music player which has many features like creating and importing playlists, marking favorite songs, downloading songs, sorting songs based on actors or mood etc. Overall PyMusic is a hybrid music player which allows to play songs present in the system as well as download songs from the internet (YouTube).

PyMusic eliminates the main problem in music players which are available in the internet today i.e. ads and premium subscriptions. Without premium subscription, one can avail most of the features available in paid apps for **free**.

PyMusic has a lot of scope for future development. Some possibilities are:

Remixing Music

Displaying Song Lyrics

Social Media Connectivity

Personalization

Developing into an App.

**References**

1. <https://doc.qt.io/qt-5/>
2. <https://intellipaat.com/community/51499/how-to-play-sound-with-pyqt5-qtmultimedia>
3. <https://likegeeks.com/python-sqlite3-tutorial/>
4. <https://www.geeksforgeeks.org/download-video-in-mp3-format-using-pytube/>
5. Python Programming using Problem Solving approach by Reema Thereja.