SAVEETHA ENGINEERING COLLEGE AUTONOMOUS

Affiliated to Anna University | Approved by AICTE

nirf Ranked 124 | NAAC A GRADE | Accredited by NBA NATIONAL BOARD OF ACCREDITATION 5 UG Programs

LABORATORY RECORD NOTEBOOK

Name of the Student : ALEN WENISH J

Register Number : 212219040009

Department : COMPUTER SCIENCE AND ENGINEERING

Semester : 4TH SEMESTER

Subject : SOFTWARE ENGINEERING

Thandalam, Chennai - 602 105, Tamil Nadu, India. Phone : +91 44 6672 6672 / 6672 6677
Website : www.saveetha.ac.in, Email : principal@saveetha.ac.in

SAVEETHA AUTONOMOUS ENGINEERING COLLEGE

Affiliated to Anna University | Approved by AICTE

Department of COMPUTER SCIENCE AND ENGINEERING.

LABORATORY RECORD NOTE BOOK

2021- 2022

It is to certify that this is a bonafide record work done by

Mr. / Ms. ..... ALEN WENISH J ..... Register Number 212219040009

of I / ✔ II / III / IV year ✔ B.E / B.Tech. / M.E / M.B.A Department

of COMPUTER SCIENCE AND ENGINEERING. in the SOFTWARE ENGINEERING

Laboratory in the ..... IV ..... Semester.

Staff in – Charge **Head of the Department**

University Examination held on .....

Internal Examiner **External Examiner**

INDEX

Ex.no	Date	Title of Experiment	Component		Marks	SIGN
1	18/2/2021	PROBLEM ANALYSIS	1.1	Overview of the project		
			1.2	Identification of the project scope		
	25/2/2021		1.3	Objectives		
			1.4	Infrastructure		
2	4/3/2021	SOFTWARE REQUIREMENT ANALYSIS AND PLANNING	2.1	Description of individual phase/module		
			2.1.1	User Characteristics		
	11/3/2021		2.1.2	General Constraints		
			2.1.3	Assumption		
			2.1.4	Functional Requirements		
			2.2	Indentify Individual Module deliverable		
3	18/3/2021	DATA MODELING	3.1	System Architecture Design		
			3.2	Use case diagram		
			3.3	Class diagram		
	25/3/2021		3.4	Activity diagram		
			3.5	Sequence Diagram		
4	1/4/2021	DEVELOPMENT AND DEBUGGING	4.1	System Design / Data Structures		
5	8/4/2021	SOFTWARE TESTING	5.1	Test plan & Code		
	15/4/2021		5.2	Test results & Debugging		
6	22/4/2021	DEMO AND SCREEN SHOTS	6.1	Output screenshots		
			6.2	CONCLUSION		

1. PROBLEM ANALYSIS

1.1. Introduction

Listening to music is kind of a hobby for almost every person we meet around the globe daily. For playing any kind of music we need to install a music player app in our device, each and every operating system, whether it is Windows, Linux, MAC, or even Android or Apple IOS.

In order to listen to music, we should have a music player application which will let to manage all the music in the device, at one place. One such kind of music player is this one, which lets us to manage all the music files quickly and easily.

This stylish and innovative music player application is created by using Python Programming language, which is known for its very rich library support. This music player application lets you to access any kind of tracks and it supports various formats of audio files like, MP3, wav, midi, AAC files and other audio formats.

By using this application and without any installation of external app, we can just run our music player and hear the songs which are downloaded in our system. This application is a GUI based app, which is very versatile and lets you to play, pause, unpause and stop the music playing.

The main and primary aim of this “Music Player Application” is that to provide a great user experience and to properly manage all the music files in the system. This application brings us a improved experience and it also have all the basic functionalities that every person expect in a music player application, that is, Play, Pause, Unpause, and stop a song. It also provides a separate space for the playlist, so that user can see what the songs present and to choose the next song.

1.2. Identification of the Scope of the Project

In our everyday life, we see every person listen to music either knowingly or unknowingly. So, in order to listen to music, we need a music player, either in the form of hardware or in software, where we can be able to play the songs we want.

With the help of this music player app, we can listen to the music and audio files that are downloaded and present in our system. This would enable the users to listen to music and songs without requiring any internet connection.

The songs can be easily added to the folder, and by which it is automatically added to the music player application playlist.

1.3. Objectives

The first and main objective of the project is to design an audio MP3 player that can support different audio files and can be suitable for different level of users.

The main goal of this project is to provide a platform to play audio files, support various playlists and provide playlist management.

Another main objective is that, it should be user friendly such that users can easily access their music and audio files through this application. This application also includes functions like Play, Pause, Unpause, and Stop for the ease of access for the user.

1.4. Infrastructure

Here is some of the various software, module and functions are used in this project:

1. **Python IDE**:

   We would require any python editor, like PyCharm, Python IDLE, etc. to add our python codes and run it.

2. **PIP**:

   PIP is a package manager or python package which is used to install different python modules..

3. **Modules**:

   The Python modules used in this project are Tkinter, OS, and PyGame. Through the use of PIP, we can install these modules in out system.

2. SOFTWARE REQUIREMENT ANALYSIS AND PLANNING

2. Requirement Analysis:

i). Modules Used:

PyGame, OS, and Tkinter modules are used in Python.

ii). Hardware Requirement:

Desktop or Laptop with a minimum requirement of:

- Processor: Intel Pentium III or later
- Main Memory: 500 MB
- Cache Memory: 512 KB
- Monitor: 14 – Inch Color Monitor
- Keyboard: 108 Keys (Standard Keyboard)
- Mouse: Optical Mouse
- Hard Disk: 160 GB of Disk Space
- Operating System: Windows or MAC

iii). Software Requirements:

Any Operating System – Windows, Linux or MAC OS, Python IDE and some audio files.

iv). Python:

Python is a popular programming language which is used for web development, software development, etc. Python is an object-oriented language and can be used to handle Big Data.

2.1. Description of Individual Phase/Module

2.1.1. User Characteristics:

- The songs are arranged in a order under the playlist column.
- The songs track displays the name of the current song playing.
- The Play button is used to play any song which is listened in the sing playlist column.
- The Pause button is used to pause the current playing song.

- The Unpause button is used to start or unpause the currently playing song again from where it is paused.
- The stop button is used to stop the song from being played.

2.1.2. General Constraints:

- General Constraints are the requirements that are not functional in nature. They are often called as the qualities of the system.
- Other Terms for non-functional requirements are “constraints”, “quality attributes”, “quality goals”, “quality of service requirements”, and “non-behavioral requirements”.

i). Scope:

The scope of this project is the use of the music player without any requiring of hardware or software installation.

ii). Functionality:

One user at a time can use this app by playing a single song of his/her’s choice.

iii). Usability:

The desktop User interface can be Windows, Linux, or MAC OS.

iv). Reliability:

The music player app must be able to store many sons and work properly without any delay or any error.

2.1.3. Assumption:

- The assumption is the directory in which it is mentioned in the python code is used to access the songs.
- In case we have mentioned the wrong path of the directory, where the songs are stored, it will not be able to access those songs.
- It is impossible to play any song which is not present in the specified directory, even though it is downloaded in the system.

2.1.4. Functional Requirements:

Description:

__init__ Constructor:

- With the help of this constructor, we will set the title for the windows and geometry for the window. We will initiate PyGame and PyGame mixer and then declare track variable and status variable.
- We will then create the track frames for the song label and status label and then after insert the song track label and status label.
- After that, we will create the button frame and insert Play, Pause, Unpause and Stop buttons into it.
- Then we will create the playlist frame and add the scrollbar to it and we will add songs into the playlist.

def playsong(self):

- It loads the active song from the list and plays the required song. It gets executed when the user clicks on “play”.

def stopsong(self):

- It stops the required song. It gets executed when the user clicks on “stop” button.

def pausesong(self):

- It pauses the required song. It gets executed when the user clicks on “pause”.

def unpause(self):

- It resumes the required song. It gets executed when the user clicks on “unpause”.

Input:

- The input to this project is mainly the songs and the audio files which are selected by the user.

Output:

- The output is the GUI music player window, which is displayed when we run the python code.

2.2. Identify Individual Module Delivery

The modules used in this project are Tkinter, Pygame and OS.

- **Tkinter:**
  - Tkinter library, which is used is a standard library for GUI creation. The Tkinter library is the most popular and very easy to use and it comes with many widgets , where these widgets ghelps in the creation of nice looking GUI application.
  - Also Tkinter is a very light-weight module and it is helpful in creating cross-platform application, so the same code can be easily worked on windows, MAC OD and Linux.
  - To use all the functions of Tkinter, we need to import it in your code and the command for the same is:

    From tkinter import *

- **Pygame Module:**
  - Pygame is a python module that works with computer graphics and sound libraries and designed with the power of playing with different multimedia formats like audio, video, etc. While creating the music player application, we will be using Pygame's mixer. Music module for providing different functionalities to our music player application that is usually related to the manipulation of the sing tracks.
  - Command used to install pygame is:

    PIP install pygame
  - To use this module into your code, you need to write this:

    Import pygame

- **OS Module:**
  - There is no need to install this module explicitly, as it comes with the standard library of Python. This module provides different function for the interactions with the operating system.
  - We use this module for fetching the playlist of songs from the specified directory and make it available to the music player application.
  - To use this module, you need to import it:

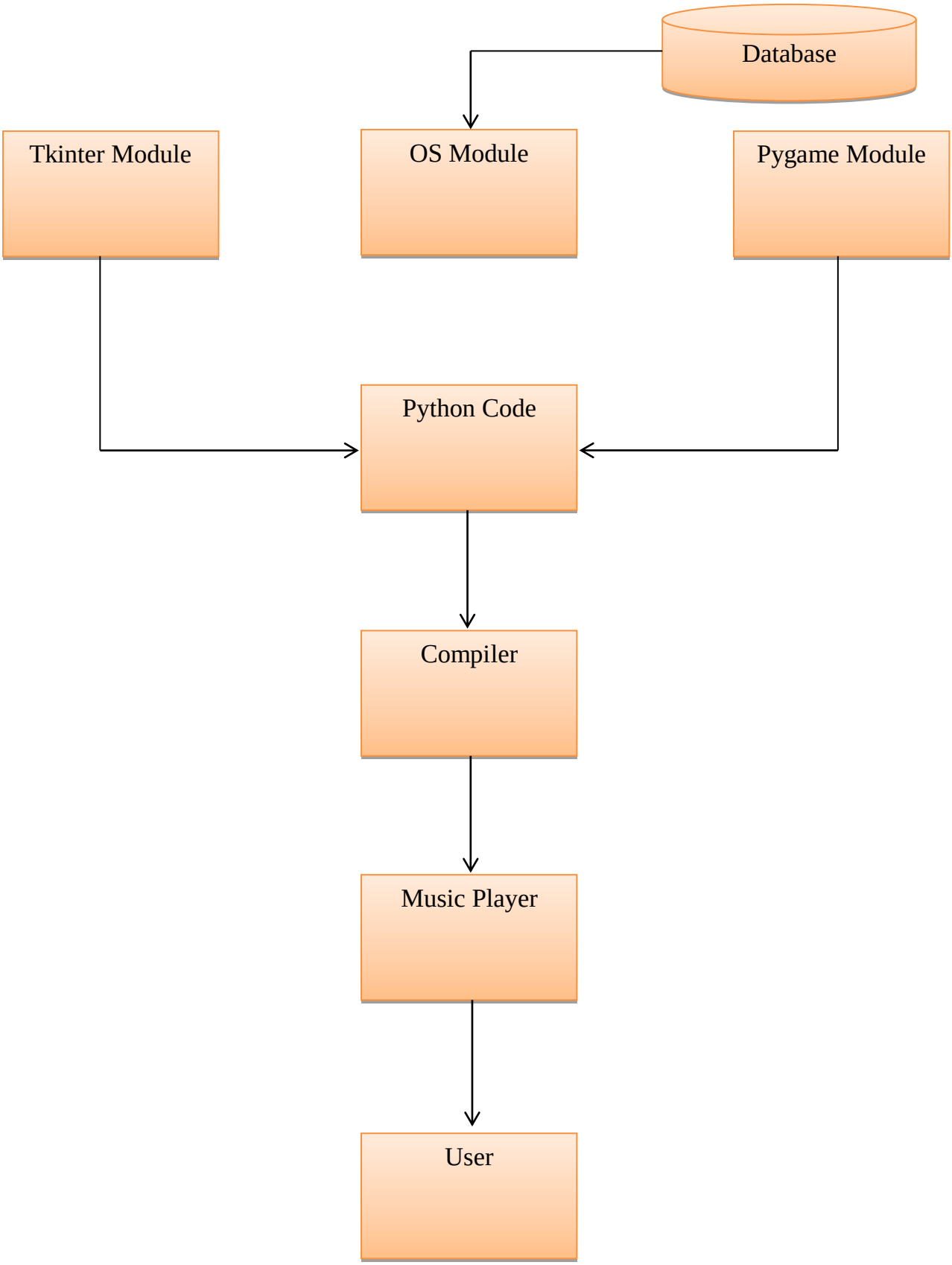
    Import OS

3. DATA MODELING

3.1. System Architecture design:

- The architecture of a system describes its major components, their relationship (structure) and how they interact with each other
- Architecture Serves as a blueprint for a system. It provides an abstraction to manage the system complexity and establish a communication and coordination mechanism among components.
- Architecture design of the music player app combines the structure of the model as used and database from the songs are fetched.
- Combination of this makes the Python code and it is compiled to get the required output (ie) music player app.
- Then the music player app is used by the user and its functionalities are tested.

System Architecture design for Music Player:



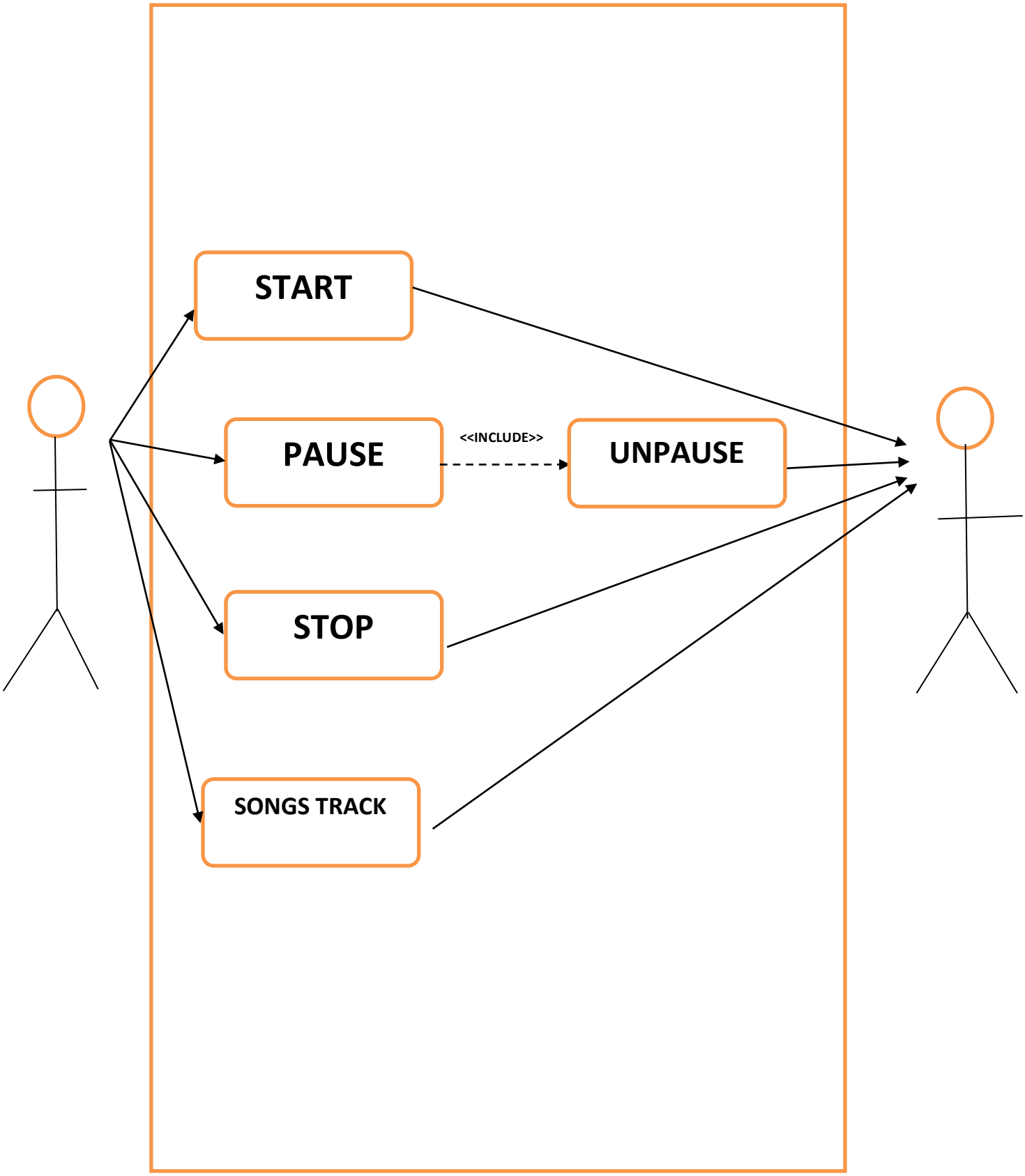
3.2. Use Case Diagram

- A use case diagram is the primary form of system /software requirements for a new software program underdeveloped.
- Use case specify the expected behavior (what)work unknot the exact methods of making it happen. (how)
- It only summarizes some of the relationship between use cases and actors and systems.
- The internal and external agents are known as actors. A single use diagram captures a particular functionality of a system.
- Use case diagrams are used to gather the requirements of a system, including internal and external influences. These requirements are mostly design requirements.

Use Case Diagram for Music Player:

- The actors present in the use case diagram for music player is user and music player Interface.
- The start, pause, stop songs, tracks are the use case present in the use case diagram.
- The unpause use case is the include relationship to the pause use case without the unpause use case the function of pause is not meaningful.
- In this use case diagram, one actor is associated with multiple use cases.
- The stop use case is the extended functionality of pause use case. The stop use case is optional, and it can also be used as an alternate to the pause use case.

Diagram:



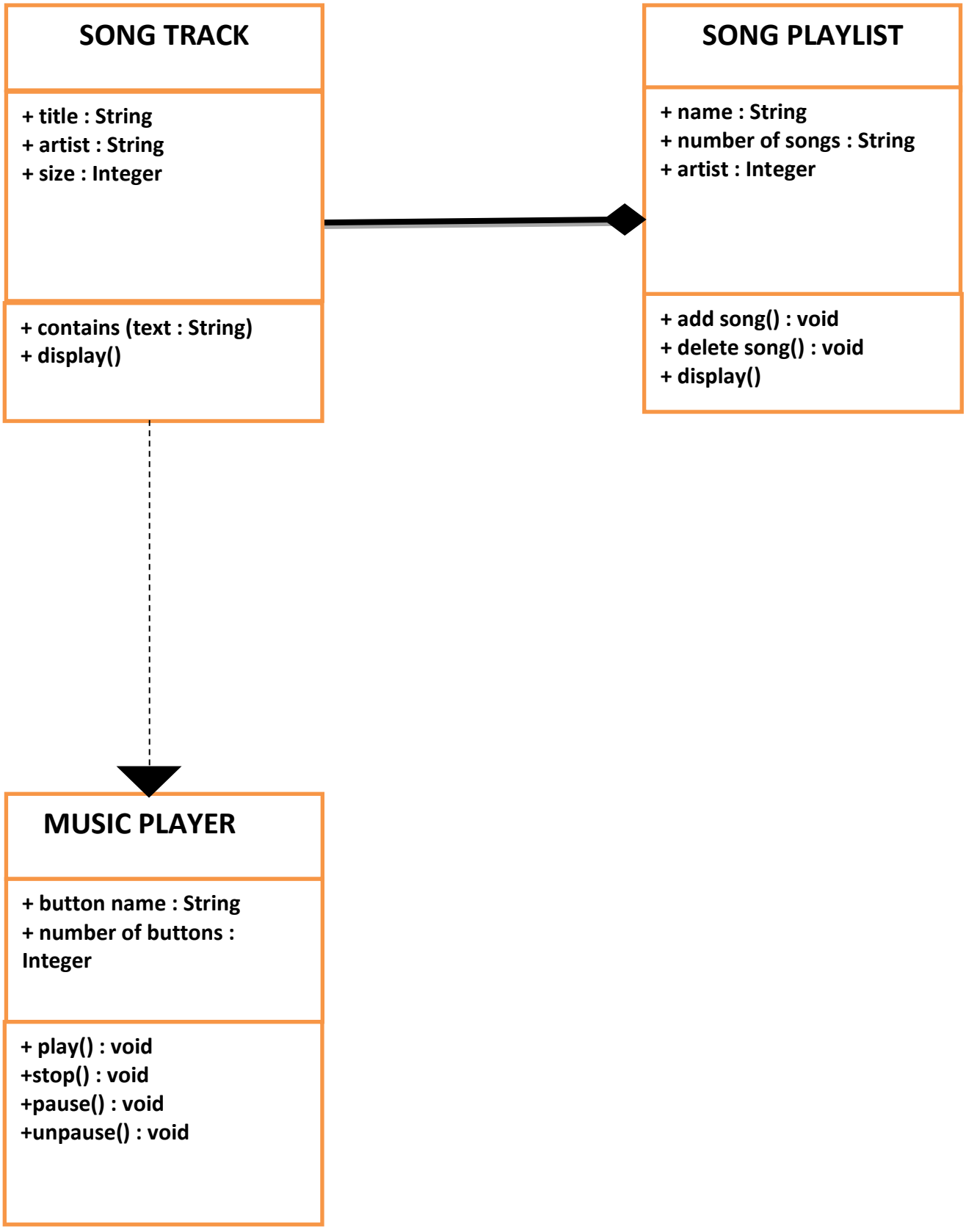
3.3. Class Diagram

- The class diagram depicts static view of an application. A class diagram is basically used to visualize, describe, document the various aspects of the system and also construct executable software code.
- Classes in the class diagram are represented by boxes that are partitioned into three.
- It constitutes class names, attributes, and functions in separate compartment that helps in software development.
- Since it is a collection of classes, interfaces, associations, collaborations and constraints, it is termed as a structural diagram.
- The main purpose of class diagrams is to buid a static view of an application.
- The top partition contains the name of the class.
- The middle part contains the class's attributes.
- The bottom partition shows the possible operations that are associated with the class
- Relationships in class diagrams include different types of logical connections.

Class Diagram for Music Player:

- The class diagram for the Music player application consists of three classes, namely, Song Track, Song Playlist, and the Music Player itself.
- Respective attributes and methods are added to each class and they are connected to provide a relationship among them.
- The song track is dependent on the music player, as if one song plays on the app, music player has the functions to control it, so there is a dependency among these two classes
- Composition relationship lies between song playlists (all songs) are deleted. We cannot be able to choose any song from it. So, there is a composition relationship between these two classes.

Diagram:


SONG TRACK
+ title : String
+ artist : String
+ size : Integer
+ contains (text : String)
+ display()
SONG PLAYLIST
+ name : String
+ number of songs : String
+ artist : Integer
+ add song() : void
+ delete song() : void
+ display()
MUSIC PLAYER
+ button name : String
+ number of buttons : Integer
+ play() : void
+stop() : void
+pause() : void
+unpause() : void

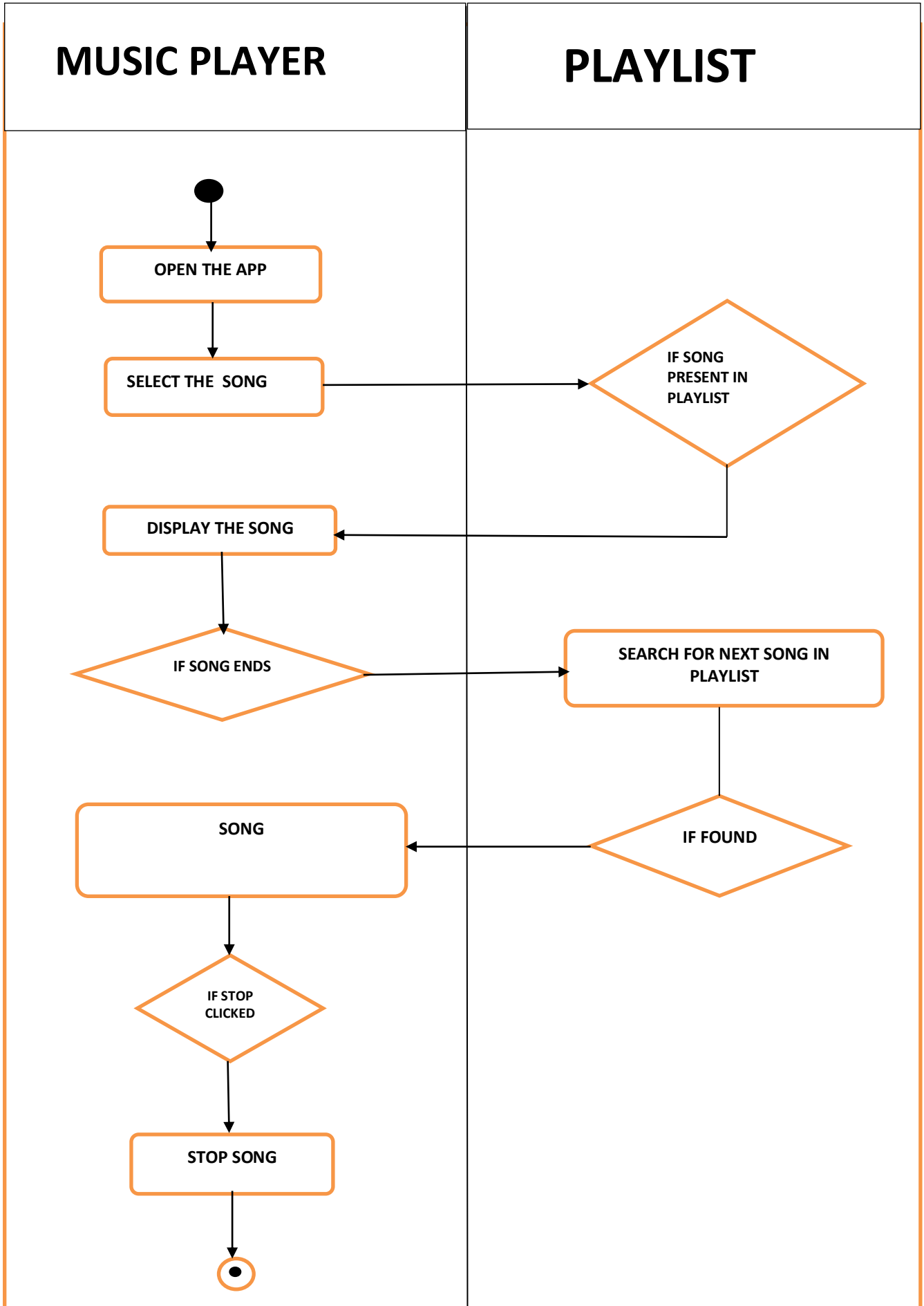
3.4. Activity Diagram

- The Activity diagram is used to demonstrate the flow of control within the system rather than the implementation.
- It is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.
- The control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.
- The Activity Diagram captures the dynamic behavior of the system. It is also use dot construct the executable system by using forward and reverse engineering techniques.
- Before drawing an activity diagram, we should identify the following elements: activities, association, conditions, and constraints.

Class Diagram for Music Player:

- The activity diagram consists of two swim lanes namely Music Player and playlist for the music player app.
- The activity diagram represents the music player app workflow in a graphical way.
- First, open the app, select the song, display the song, search for the next song in playlist, stop the song are the activity state present in the activity diagram.
- The decision nodes like if song ends, if found are the notations which are used to take decisions.
- The decision nodes are usually in the shape of diamond. The arrows are used to denote the control flow of the activity diagram.
- From start to end, how the activity tool place in the music player app is mentioned.

Diagram:


MUSIC PLAYER
PLAYLIST
OPEN THE APP
SELECT THE SONG
IF SONG PRESENT IN PLAYLIST
DISPLAY THE SONG
IF SONG ENDS
SEARCH FOR NEXT SONG IN PLAYLIST
SONG
IF FOUND
IF STOP CLICKED
STOP SONG

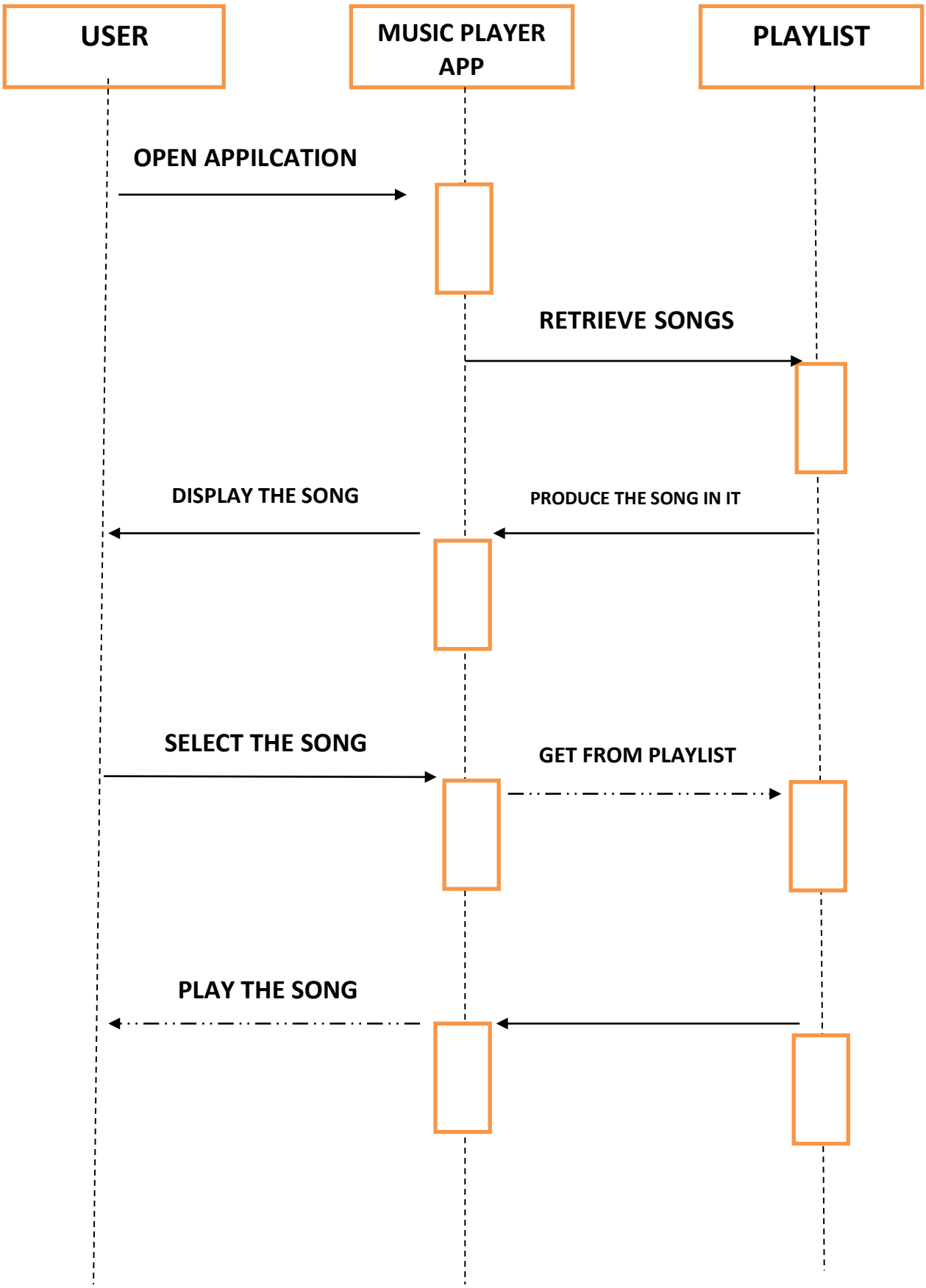
3.5. Sequence Diagram

- The sequence diagram represents the flow of messages in the system and is also termed as an event diagram.
- A sequence diagram simply depicts interaction between objects in a sequential order (i.e) the order in which these interaction take place.
- It portrays the communication between any two lifelines as a time-ordered sequence of events, each that these lifelines took part at the run time.
- These diagrams are widely used by businessman and understand requirements for new and existing systems.
- The purpose of a sequence diagram is to model high-level interaction between active objects in a system.
- It explores the real time application and implement both forward and reverse engineering.

Sequence Diagram for Music Player:

- The sequence diagram for the music player app consists of three lifelines namely, User, Music Player app, and the playlist.
- The three lifelines in the sequence diagram interact with each other through messages.
- At first, user sends a synchronous message (i.e) open application.
- In reply to that, Music player retrieve the songs from the playlist and retrieve the songs from the playlist and display it to the user, which is the asynchronous message in this case.
- Then, the user selects the song from list displayed in the music player.
- As a reply message, the music player plays the song which is chosen by the user.
- It is very important in sequence diagram to note that they show the interactions for a particular scenario. The processes are represented vertically and interactions are shown as arrows.

Diagram:


USER
MUSIC PLAYER APP
PLAYLIST
OPEN APPILCATION
RETRIEVE SONGS
DISPLAY THE SONG
PRODUCE THE SONG IN IT
SELECT THE SONG
GET FROM PLAYLIST
PLAY THE SONG

4. DEVELOPMENT AND DEBUGGING

4. Development and Debugging

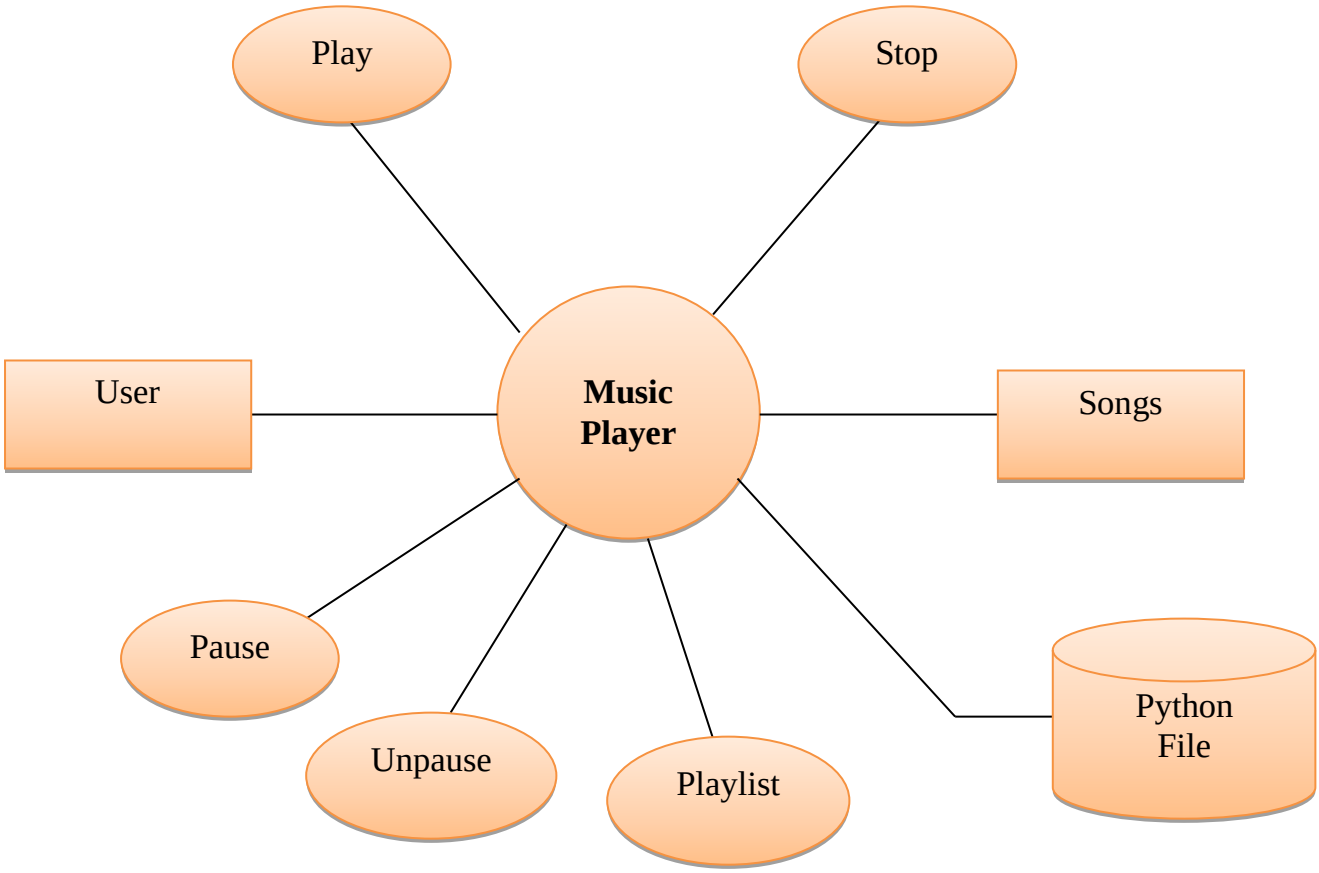
- Development and Debugging is a very important phase for any project. Development is the process of designing, programming, documenting and testing, which then involves in creating and maintaining applications frameworks or any other software components.
- Debugging is the process of detecting and removing of existing and potential errors, which is otherwise called as ‘bugs’, in a software code that can cause it to behave unexpectedly or crash.
- To prevent incorrect operation of a software or system, debugging is used to find and resolve the bug.

4.1 System Design/Database Structure

- The database structure is defined as the collection of record type and field type definitions that comprise the database.
- The development stage of this project begins with the step of GUI window for the users.
- After that, different buttons like play, pause, unpause, and stop are inserted and two columns namely song track and song playlist is created.
- Then we have to create one folder (i.e named songs) to store the music which are downloaded in the system.
- Once the folder is created, we have to mention the path of the folder in OS directory.
- This will help us to give access to that folder using the python code. Then we have to add function to the button which we are created.

- Separate functions are created for each button and there functionalities are added to it.
- After completion of the music player app, the code is saved and it is run using the editor.
- The database structure helps the music player app to manage different songs and audio files.
- This gives a basic structure to the app and creates the app more versatile.

Diagram:



Debugging:

- According to this project, mostly error will occur in the directory, which is the path of the music file.
- After mentioning the correct path, the music player app is started and it is used for some time for testing.
- The database structure of the music player app is simple. In this project, mostly the data is only the songs which are stored in the folder.
- The modules used in this project can also considered as a data to this music player app. The main source code is stored in the python file (.py file) and the songs are stored in the same folder where the code is stored.

5. SOFTWARE TESTING

5. Software Testing

- Software testing is a method or the process to check whether the actual software product matches expected requirements and to ensure that software product is defect free.
- It is basically the process of verifying a system with the purpose of identifying any errors, gaps or missing requirement. Software testing process should be done during the development process.
- Testing involves the execution of a software components or system component to evaluate one or more properties of interest.
- The purpose of the software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.
- The music player app is given to multiple users for testing the application. Various improvements given by the users is noted.
- In music player application, we test for its usability, the basic functionalities, errors encountered, bugs and crashing, and vague error messages.

5.1. Test Plan and Coding:

- The software test plan includes the various stages of testing. It is a detailed document which describes software testing areas and activities.
- The test plan for this project is made in different levels and it is made for testing.
- For the music player application, we first analyze the product and test for its accessibility and installation. We introduce the app to various platforms and environment to test its adaptability.
- Then, check for its basic features like, play, pause, unpause, and stop and ensure it works perfectly fine under any given situation.

- Testing foe the playlist or playlist finding is very important as the entire application depends upon the playlists and ensuring it to work properly is an important stage in testing.
- More number of songs is stored in the folder and its efficiency is checked whether it works without any time delay or error.
- More concentration is made on interface side, to make it user friendly.
- Many songs are switched between in a short amount of time to check its accuracy.

Coding (Python Source Code):

Project.py:

```
from tkinter import *
import pygame
import os


class MusicPlayer:
    def __init__(self,root):
        self.root = root
        # Title of the window
        self.root.title("MusicPlayer")
        # Window Geometry
        self.root.geometry("1000x200+200+200")


        # Initiating Pygame
        pygame.init()
        # Initiating Pygame Mixer
        pygame.mixer.init()
```

```
        # Declaring track Variable

        self.track = StringVar()


        # Declaring Status Variable

        self.status = StringVar()

        # Creating the Track Frames for Song label & status label

        trackframe = LabelFrame(self.root,text="Song Track",font=("times new
roman",15,"bold"),bg="grey",fg="white",bd=5,relief=GROOVE)

        trackframe.place(x=0,y=0,width=600,height=100)


        # Inserting Song Track Label

        songtrack = Label(trackframe,textvariable=self.track,width=20,font=("times new
roman",24,"bold"),bg="grey",fg="gold").grid(row=0,column=0,padx=10,pady=5)


        # Inserting Status Label

        trackstatus = Label(trackframe,textvariable=self.status,font=("times new
roman",24,"bold"),bg="grey",fg="gold").grid(row=0,column=1,padx=10,pady=5)


        # Creating Button Frame

        buttonframe = LabelFrame(self.root,text="Control Panel",font=("times new
roman",15,"bold"),bg="grey",fg="white",bd=5,relief=GROOVE)

        buttonframe.place(x=0,y=100,width=600,height=100)


        # Inserting Play Button

        playbtn =
Button(buttonframe,text="PLAY",command=self.playsong,width=10,height=1,font=(
"times new
roman",16,"bold"),fg="navyblue",bg="gold").grid(row=0,column=0,padx=10,pady=5
)
```

```
        # Inserting Pause Button

        playbtn =
Button(buttonframe,text="PAUSE",command=self.pausesong,width=8,height=1,font=
("times new
roman",16,"bold"),fg="navyblue",bg="gold").grid(row=0,column=1,padx=10,pady=5
)



        # Inserting Unpause Button

        playbtn =
Button(buttonframe,text="UNPAUSE",command=self.unpausesong,width=10,height=
1,font=("times new
roman",16,"bold"),fg="navyblue",bg="gold").grid(row=0,column=2,padx=10,pady=5
)



        # Inserting Stop Button

        playbtn =
Button(buttonframe,text="STOP",command=self.stopsong,width=10,height=1,font=("
times new
roman",16,"bold"),fg="navyblue",bg="gold").grid(row=0,column=3,padx=10,pady=5
)



        # Creating Playlist Frame

        songsframe = LabelFrame(self.root,text="Song Playlist",font=("times new
roman",15,"bold"),bg="grey",fg="white",bd=5,relief=GROOVE)

        songsframe.place(x=600,y=0,width=400,height=200)



        # Inserting scrollbar

        scrol_y = Scrollbar(songsframe,orient=VERTICAL)



        # Inserting Playlist listbox
```

```
        self.playlist =
Listbox(songsframe,yscrollcommand=scrol_y.set,selectbackground="gold",selectmod
e=SINGLE,font=("times new
roman",12,"bold"),bg="silver",fg="navyblue",bd=5,relief=GROOVE)


        # Applying Scrollbar to listbox
        scrol_y.pack(side=RIGHT,fill=Y)
        scrol_y.config(command=self.playlist.yview)
        self.playlist.pack(fill=BOTH)


        # Changing Directory for fetching Songs
        os.chdir(r"E:\Python Project\songs")
        # Fetching Songs
        songtracks = os.listdir()
        # Inserting Songs into Playlist
        for track in songtracks:
            self.playlist.insert(END,track)


    def playsong(self):
        # Displaying Selected Song title
        self.track.set(self.playlist.get(ACTIVE))
        # Displaying Status
        self.status.set("-Playing")
        # Loading Selected Song
        pygame.mixer.music.load(self.playlist.get(ACTIVE)
        # Playing Selected Song
        pygame.mixer.music.play()
```

```
    def stopsong(self):
        # Displaying Status
        self.status.set("-Stopped")
        # Stopped Song
        pygame.mixer.music.stop()

    def pausesong(self):
        # Displaying Status
        self.status.set("-Paused")
        # Paused Song
        pygame.mixer.music.pause()

    def unpausesong(self):
        # It will Display the  Status
        self.status.set("-Playing")
        # Playing back Song
        pygame.mixer.music.unpause()

root = Tk()
MusicPlayer(root)
root.mainloop
```

5.2. Test Results and Debugging:

- A test result is a kind of confirmation that gradually leads us to the right track towards the pursuit of a robust software application.
- A test result concludes the software life cycle, thus by allowing us to look into the facts and figures and progressively plan the subsequent activities.
- After the test result from the user, debugging is more based on the additional requirement to the project.
- After different levels of testing, the project is deployed to the end users. Any suggestions or improvements from their side are implemented and project is ready for use.
- The music player application must pass all the tests mentioned in the test plan. A successful test result will give a green signal to deploy the project and to rollout to the public.

6. PROJECT DEMO AND SCREENSHOTS

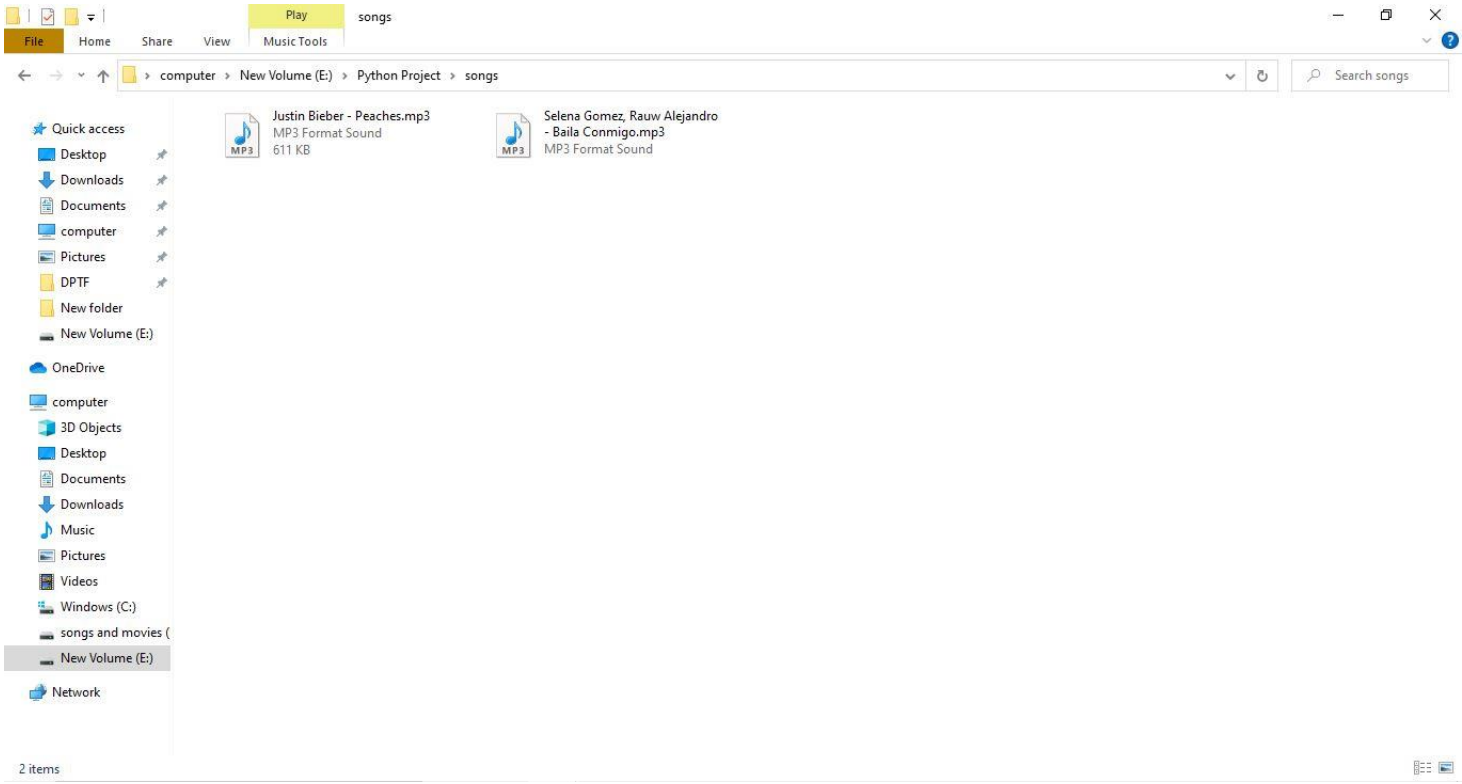
6. Project Demo:

- A project demo or project demonstration is a means of promoting innovations and capturing and disseminating best practices through the development and analysis of a live project.
- Project demo measures the performance improvements using key performance indicators.

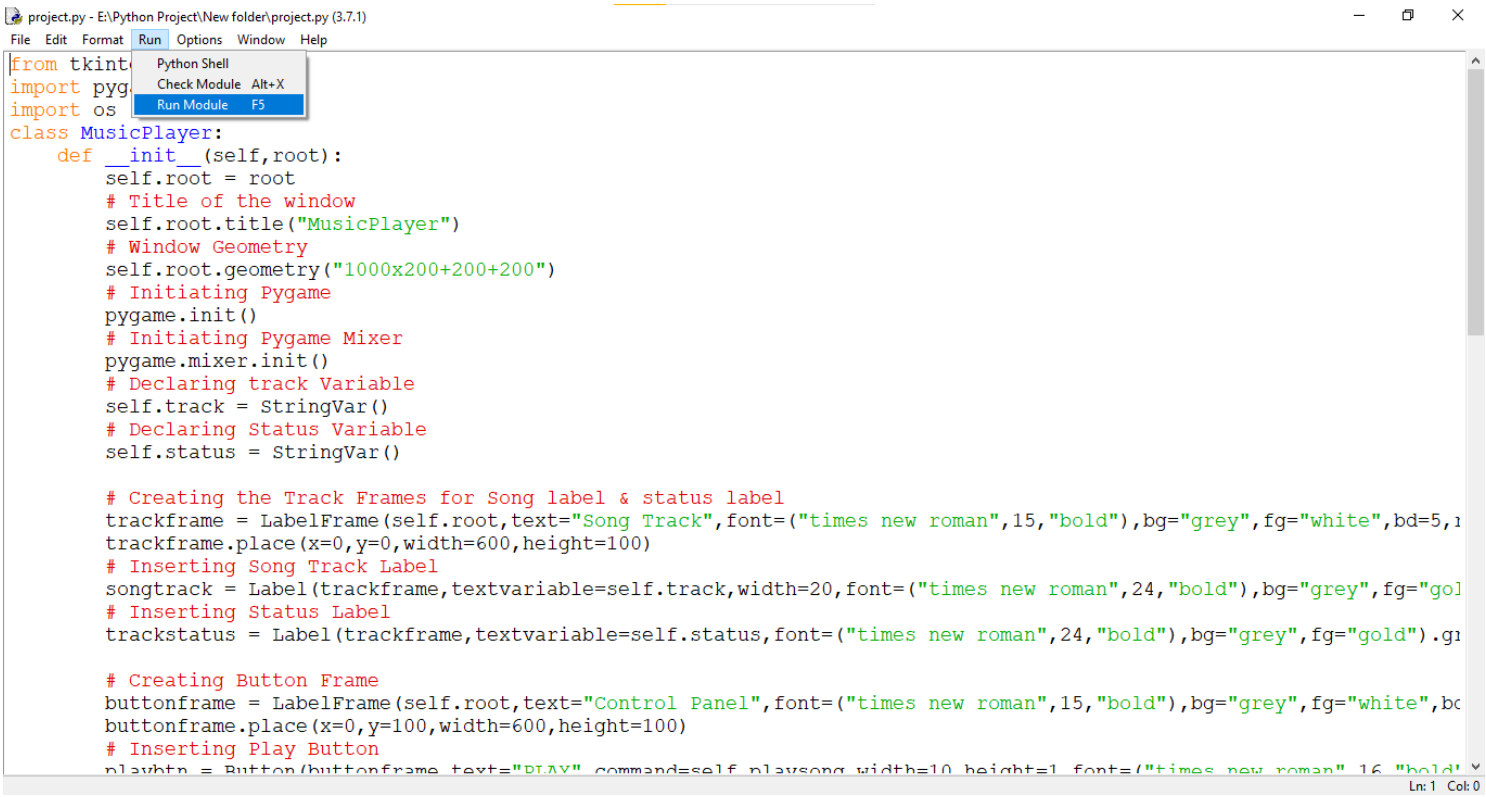
- When we run the Music player application, we get a starting GUI window with the song track, control panel and the song playlist in it with a beautiful User Interface (UI).
- A song will be chosen from the song playlist and the song or the music begins playing
- All the controls required for a music player are present here in the control panel.
- We can play, pause, unpause, or stop the song/music currently playing from the control panel.
- When the song/music reaches its end of duration, it automatically plays the next song/music in the playlist.
- If no song/music is present, then it automatically stops playing.

6.1. Output Screenshots

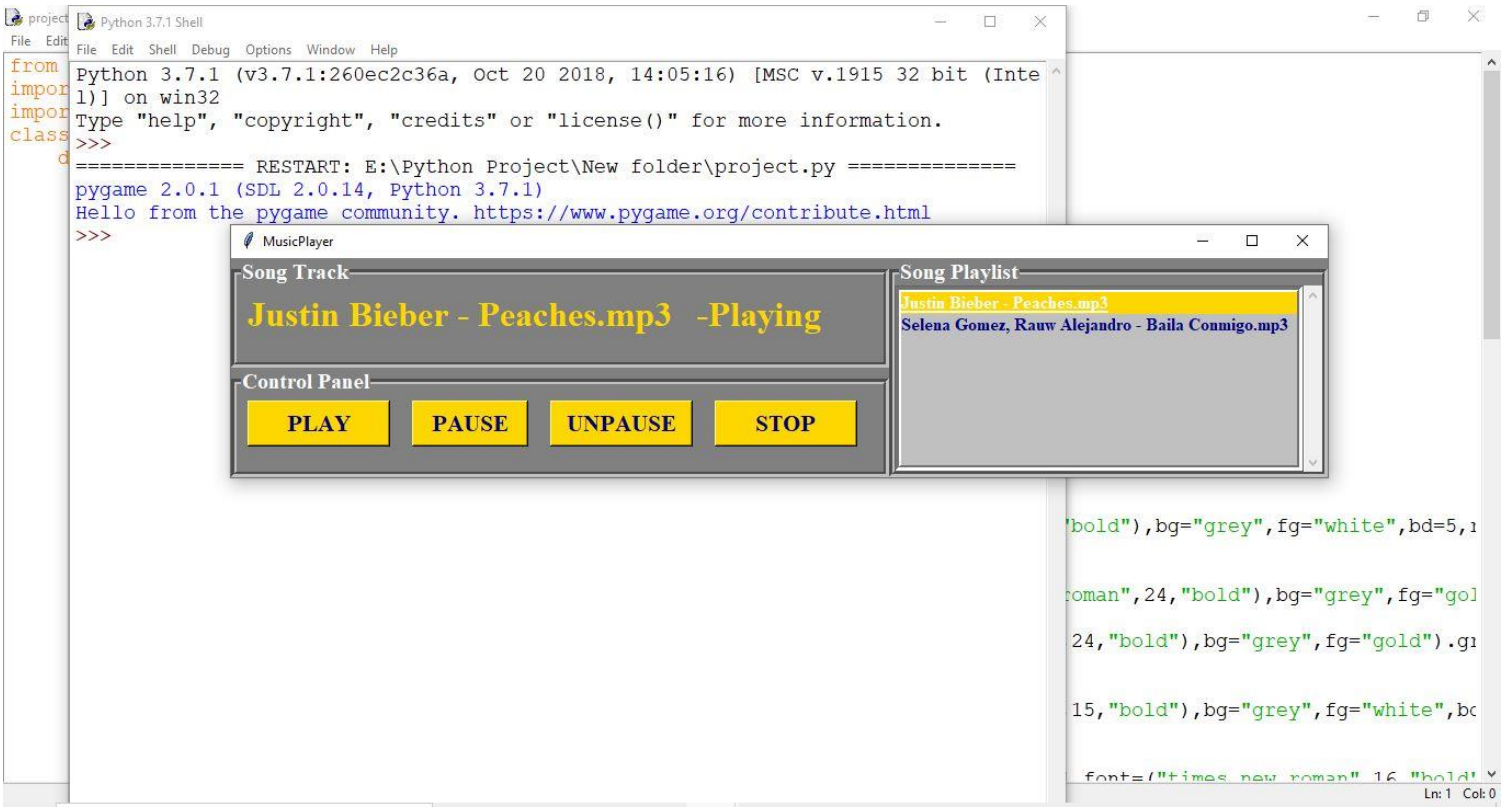
1. Save the required songs/music in the correct directory mentioned.



2. Open the source code using Python editor, like Python IDLE, and execute the program.



3. You will get a GUI window (as shown below), and we can play the songs shown from the playlist.



4. Final Output:

6.2. Conclusion

- The Project “Music Player Application” is developed to listen to the songs/music and to manage the playlists.
- This music player application lets us to access any kind of tracks and supports various formats of audio files.
- By using this application and without any installation of external app, we can just run our music player and hear the songs which are downloaded in our system.
- This application is a GUI based app, which is very versatile and lets you to play, pause, unpause and stop the music playing.
- The main and primary aim of this “Music Player Application” is that to provide a great user experience and to properly manage all the music files in the system.
- This application brings us a improved experience and it also have all the basic functionalities that every person expect in a music player application, that is, Play, Pause, Unpause, and stop a song. It also provides a separate space for the playlist, so that user can see what the songs present and to choose the next song.

Thus, the Music Player Application using Python is developed and executed successfully.