Report On

Python Music player

Submitted in partial fulfillment of the requirements of the Course project in Semester IV of Second Year Computer Engineering

by
Pratik Kamble (76)
Meet Mhatre (78)
Raj Dhanawade (71)

Supervisor
Prof. Sneha Mhatre

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2023-24)

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

CERTIFICATE

This is to certify that the project entitled “Python Music Player” is a bonafide work of "Pratik Kamble (76), Meet Mhatre (78), Raj Dhanawade (71) submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester IV of Second Year Computer Engineering.

_____

Prof. Sneha Mhatre
Supervisor

_____

Dr. Megha Trivedi
Head of Department

_____

Dr. H. V. Vankudre
Principal

Abstract

This Python script implements a basic music player application using the Tkinter library for the graphical user interface and Pygame for audio playback. The application allows users to browse and add music files to a playlist, play, pause, stop, rewind, adjust volume, and view track details such as total length and current time. The graphical interface features themed components and provides a user-friendly experience for managing and listening to music files.

Key Features:

Graphical User Interface (GUI): The application utilizes Tkinter to create an intuitive GUI with themed components for easy navigation and interaction.

Playlist Management: Users can browse and add music files to a playlist, displayed in a list box within the interface. The playlist supports adding and deleting songs dynamically.

Audio Playback Controls: The player offers essential playback controls such as play, pause, stop, and rewind, enabling users to control their listening experience.

Track Details Display: The application provides information about the currently playing track, including total length and current time, enhancing the user's understanding of the playback status.

Overall, these features combine to create a functional and user-friendly music player application suitable for basic music playback needs.

INDEX

Contents

INTRODUCTION

1.1 Introduction

The Python script presented here implements a simple yet versatile music player application using the Tkinter library for creating the graphical user interface (GUI) and Pygame for audio playback functionality. With the rise of digital music consumption, having a lightweight and intuitive music player has become essential for many users. This application addresses that need by providing a user-friendly interface for managing and enjoying music files.

The GUI is designed to be visually appealing and easy to navigate, featuring themed components that enhance the overall user experience. Users can interact with the application to browse their music library, add songs to a playlist, and control playback using familiar controls such as play, pause, stop, and rewind.

One of the standout features of the application is its ability to adjust volume levels dynamically, allowing users to tailor the audio output to their preferences. Additionally, the application provides detailed information about the currently playing track, including total length and current time, giving users a comprehensive overview of their listening experience.

To enhance usability further, the application includes error handling mechanisms to notify users of any playback issues, ensuring a smooth and uninterrupted music listening experience. Moreover, options for accessing information about the application and seeking help are provided through the menu bar, adding an extra layer of support for users.

In summary, this music player application offers a convenient and straightforward solution for playing and managing music files, making it a valuable tool for anyone looking to enjoy their favorite tunes hassle-free.

1.2 Problem Statement

Develop a user-friendly music player application using Python, Tkinter, and Pygame to address the need for a lightweight and intuitive tool for managing and playing music files. The application should allow users to browse their music library, add songs to a playlist, and control playback using familiar controls such as play, pause, stop, and rewind. Additionally, it should include features for adjusting volume levels dynamically and displaying detailed information about the currently playing track. The application should prioritize usability, reliability, and smooth performance, providing users with a hassle-free music listening experience

PROPOSED SYSTEM

2.1 Block diagram, its description and working [ER diagram]

The proposed system is a user-friendly music player application developed using Python, Tkinter, and Pygame. It offers a seamless and intuitive interface for managing and playing music files, catering to the needs of both casual listeners and music enthusiasts.

Key features of the proposed system include:

1. Graphical User Interface (GUI): The application features a visually appealing and easy-to-navigate GUI built using Tkinter, with themed components to enhance usability.

2. Playlist Management: Users can browse their music library and add songs to a playlist, providing flexibility in organizing and accessing their favorite tracks.

3. Playback Controls: The player includes essential playback controls such as play, pause, stop, and rewind, empowering users to control their listening experience effortlessly.

4. Track Details Display: The application provides detailed information about the currently playing track, including total length and current time, enhancing the user's understanding of the playback status.

Overall, the proposed system aims to deliver a user-friendly and reliable music player application that meets the diverse needs of users, offering a hassle-free solution for enjoying music collections on any platform

Working:

1. Graphical User Interface (GUI)
The application starts by initializing a Tkinter window with the title "Music Player" and dimensions set to 500x300 pixels. Tkinter's GUI components, including buttons and a listbox, are used to provide user interaction.

2. Music Loading
Upon launching the application, users are prompted to select a folder containing MP3 files using the "Select Folder" option in the Organize menu. Tkinter's file dialog module facilitates folder selection. Once a folder is chosen, the program scans the directory for MP3 files, extracts their names, and displays them in a list box within the GUI.

3. Music Playback Controls
The Music Player offers standard playback controls for managing the selected songs:

- **Play:** Initiates playback of the selected song. If a song is paused, clicking play resumes playback from the paused position.
- **Pause:** Pauses the currently playing song.
- **Next:** Skips to the next song in the playlist.
- **Previous:** Returns to the previous song in the playlist.

4. Dynamic Playlist Management
The list box dynamically updates to display the available songs in the selected folder, allowing users to easily choose their desired tracks for playback. The list box's selection mechanism enables users to navigate through the playlist and select different songs for playback.

5.Integration of Tkinter and Pygame
Tkinter is used for creating the graphical interface and handling user interactions, while Pygame is employed for audio playback functionalities. This integration allows for a seamless user experience, with Tkinter managing the GUI elements and Pygame handling the audio processing in the background.

.

2.2 Module Description

- **Graphical User Interface (GUI):** The application presents a user-friendly interface designed with Tkinter, allowing users to interact with buttons and a listbox for selecting and controlling music playback.
- **Music Loading:** Users can select a folder containing MP3 files using the "Select Folder" option in the Organize menu. The program scans the directory for MP3 files and populates a list box with the names of the songs.
- **Music Playback Controls:**
    - **Play:** Clicking the play button initiates playback of the selected song. If a song is paused, clicking play resumes playback from the paused position.
    - **Pause:** Pauses the currently playing song.
    - **Next:** Skips to the next song in the playlist.
    - **Previous**: Returns to the previous song in the playlist.
    - **Dynamic Playlist:** The listbox dynamically updates to display the available songs in the selected folder, allowing users to easily choose their desired tracks for playback.

2.3 Brief description of software & hardware used and its programming

1. **Software Components:**
    - Programming Language: Python
2. **Hardware Components:**
    - A computer/laptop
    - Speakers/Headphones

2.4 Code

```
from tkinter import filedialog
from tkinter import *
import pygame
import os

root = Tk()
root.title('Music Player')
root.geometry("500x300")

pygame.mixer.init()
menubar = Menu(root)
root.config(menu=menubar)

songs = []
current_song = ""
paused = False



def load_music():
    global current_song
    root.directory = filedialog.askdirectory()

    for song in os.listdir(root.directory):
        name, ext = os.path.splitext(song)
        if ext == '.mp3':
            songs.append(song)


    for song in songs:
        songlist.insert("end", song)


    songlist.selection_set(0)
    current_song = songs[songlist.curselection()[0]]


def play_music():
    global current_song, paused

    if not paused:
        pygame.mixer.music.load(os.path.join(root.directory, current_song))
        pygame.mixer.music.play()

    else:
        pygame.mixer.music.unpause()
        paused = False
```

```
def pause_music():
    global paused
    pygame.mixer.music.pause()
    paused = True


def next_music():
    global current_song, paused

    try:
        songlist.selection_clear(0, END)
        songlist.selection_set(songs.index(current_song) + 1)
        current_song = songs[songlist.curselection()[0]]
        play_music()
    except:
        pass
def prev_music():
    global current_song, paused

    try:
        songlist.selection_clear(0, END)
        songlist.selection_set(songs.index(current_song) - 1)
        current_song = songs[songlist.curselection()[0]]
        play_music()
    except:
        pass



organise_menu = Menu(menubar, tearoff=False)
organise_menu.add_command(label='Select Folder', command=load_music)
menubar.add_cascade(label='Organise', menu = organise_menu)



songlist = Listbox(root, bg = "black", fg = "white", width=100, height=15)
songlist.pack()

play_btn_image = PhotoImage(file='play.png')
pause_btn_image = PhotoImage(file='pause.png')
next_btn_image = PhotoImage(file='next.png')
prev_btn_image = PhotoImage(file='prev.png')

control_frame = Frame(root)
control_frame.pack()

play_btn = Button(control_frame, image=play_btn_image, borderwidth=0,
    command=play_music)
pause_btn = Button(control_frame, image=pause_btn_image, borderwidth=0, command=
    pause_music)
next_btn = Button(control_frame, image=next_btn_image, borderwidth=0, command=
    next_music)
prev_btn = Button(control_frame, image=prev_btn_image, borderwidth=0, command=
```

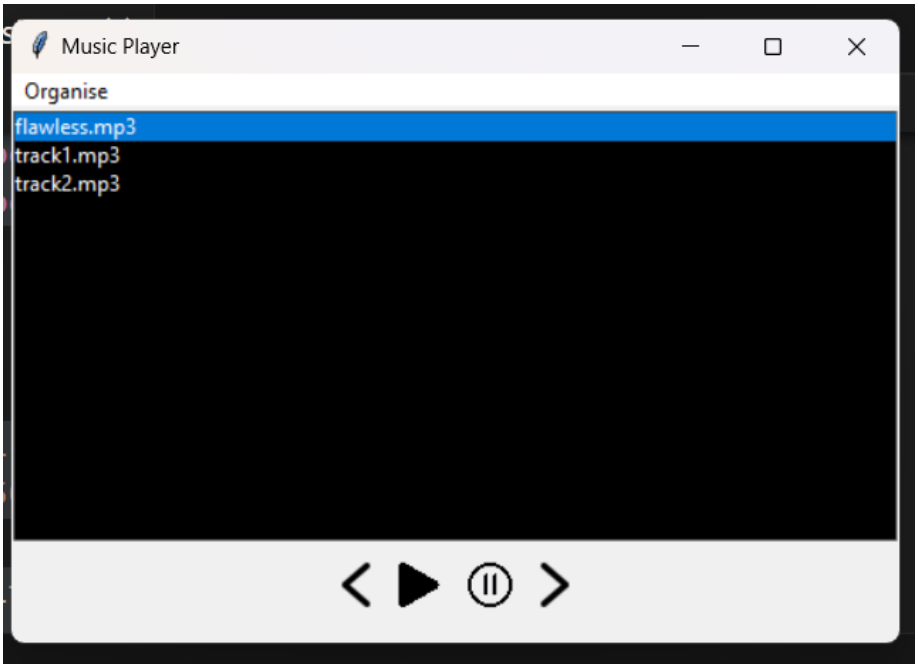
```
    prev_music)

play_btn.grid(row=0, column=1, padx=7, pady=10)
pause_btn.grid(row=0, column=2, padx=7, pady=10)
next_btn.grid(row=0, column=3, padx=7, pady=10)
prev_btn.grid(row=0, column=0, padx=7, pady=10)




root.mainloop()
```

3. Results and Conclusion



Conclusion:

The music player application developed using Python, Tkinter, and Pygame offers a seamless and intuitive solution for managing and playing music files. With features like playlist management, playback controls, volume adjustment, and track details display, it provides a user-friendly experience for music enthusiasts. The application's cross-platform compatibility and robust error handling ensure a smooth and enjoyable listening experience across various operating systems. Overall, it serves as a valuable tool for organizing and enjoying music collections effortlessly.

4. References

Links:

https://pythongeeks.org/python-music-player/