Software Project Report

Satvik Pasalapudi EE22BTECH11212

AIM:To develop a music player application that allows users to play their music files in a shuffled order.

Libraries Used:

1) pygame
2) random
3) os
4) Tkinter

Flow of Code:

1) First, we import required libraries and define the required functions.
2) Then the folder which contains the mp3 files is selected and then those files are retrieved.
3) The files are then shuffled into a playlist.
4) Then the GUI window is created and the ”next” and ”close” widgets are added.
5) The songs in the playlist are then played.
6) Once all songs are played, a new random list of songs is generated and played.
7) This process is repeated until the user closes the program.

Conclusion:

We have made the a music player application using libraries like Pygame and Tkinter which allows the users to select a music folder, retrieve the music files, and provides controls for playing, skipping songs, and quitting the player. The GUI provides a user-friendly interface, and the playlist shuffling functionality adds variety and randomness to the music playback experience.

```python
import pygame
import random
import os
import numpy as np
import soundfile as sf
from tkinter import Tk, filedialog
import tkinter as tk

os.environ['SDL_VIDEODRIVER'] = 'x11'

def folder_select():
    Tk().withdraw()
    path = filedialog.askdirectory()
    return path

def get_music_files(path):
    mp3_files = []
    for file in os.listdir(path):
        if file.endswith(".mp3"):
            mp3_files.append(os.path.join(path, file))
    return mp3_files

def update_current_song_label():
    current_song_label.config(text=f"Current Song: {os.path.basename(playlist[current_song_index])}")

def play_next():
    global current_song_index
    current_song_index = (current_song_index + 1) % len(playlist)
    pygame.mixer.music.load(playlist[current_song_index])
    pygame.mixer.music.play()

def next_song():
    play_next()
    update_current_song_label()

def shuffle_playlist():
    global playlist, shuffled_playlist, current_song_index
    shuffled_playlist = playlist.copy()
    np.random.shuffle(shuffled_playlist)
    current_song_index = 0
    pygame.mixer.music.load(shuffled_playlist[current_song_index])
    pygame.mixer.music.play()

def quit_music_player():
    pygame.mixer.music.stop()
    window.quit()

# Initialize Pygame
pygame.init()

# Choose the music folder
folder = folder_select()

# Get the music files from the chosen folder
playlist = get_music_files(folder)

# Shuffle the playlist initially
current_song_index = 0
shuffle_playlist()

# Create the GUI window
window = tk.Tk()
window.title("Songs")
window.geometry("500x250")

# Current Song Label
current_song_label = tk.Label(window, text="Now playing: ")
current_song_label.pack()

# Next Song Button
next_song_button = tk.Button(window, text="Next", command=next_song)
next_song_button.pack()

# Quit Button
quit_button = tk.Button(window, text="Close", command=quit_music_player)
quit_button.pack()

# Update the initial current song label
update_current_song_label()
# Initialize mixer module
pygame.mixer.init()
pygame.mixer.music.load(playlist[current_song_index])
pygame.mixer.music.play()

# Run the GUI main loop
window.mainloop()

# Quit Pygame
pygame.quit()
```

Fig. 7. Code

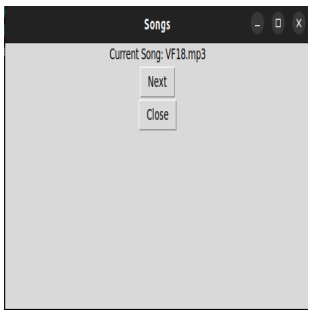

Fig. 7. Output Popup