USE-CASE 1

INTRODUCTION TO MUSIC PLAYER

What is music player?

- They are portable digital music players that play music as audio files, such as MP3.
- In addition, most of these devices allow to store video, pictures, and to receive radio and TV programs (podcasting).
- Earphones and external speakers are the typical output devices delivering sound to the listener.

What are the basic operation present in music player?

- Start playing music
- Skip this song
- Stop the music
- Pause the music
- Add song
- Remove the music

MUSIC PLAYER APPLICATION

Here are the key features and functionalities of a music player application, explained in a simple, point-wise manner:

User Interface:

- A visually appealing and intuitive interface for easy navigation.
- Display of album art, song information (title, artist, album), and progress bar.
- Options to create playlists and manage existing ones.

Audio Playback:

- Support for various audio file formats, such as MP3, WAV, FLAC, AAC, etc.
- Play, pause, stop, and skip functionalities for controlling the playback.
- Shuffle and repeat options to enhance the listening experience.
- Volume control to adjust the audio output.

Library Management:

- Scan and import music files from the user's device or specific folders.
- Organize the music library by albums, artists, genres, or playlists.
- Search functionality to quickly find desired songs.

Playlist Management:

- Create custom playlists by adding songs from the library.
- Edit, reorder, and delete songs within playlists.
- Smart playlists based on criteria like recently added, most played, etc.

Cross-platform Availability:

- Support for multiple operating systems and devices, such as iOS, Android, Windows, macOS, etc.

These points outline the main features you would typically find in a music player application. Developers may add additional functionalities or customize these features based on the specific requirements and target audience.

Benefits:

This application will helps the users to listen the songs. So in this case I am helping him with providing my music player application which provide the user to perform several task,task such as:

- User Interface: Display of album art, song information (title, artist, album), and progress bar
- Library Management: Scan and import music files from the user's device or specific folders.
- It is very fast and easy to operate

- Cross-platform Availability: Support for multiple operating systems and devices, such as iOS, Android, Windows, macOS, etc
- Playlist Management: Edit, reorder, and delete songs within playlists

PROBLEM STATEMENT

You are tasked with building a simple music player application that can play different types of music files. Implement a Python class called MusicPlayer that represents a music player. The MusicPlayer class should have the following attributes and methods:

Attributes:

- Playlist (list of strings): A list of music files in the playlist.
- current_song (string): The name of the currently playing song.

Methods:

- _init_(self, playlist): Initializes a new music player with the given playlist.
- play(self, song): Plays the specified song from the playlist and updates the current_song attribute accordingly.
- pause(self): Pauses the currently playing song.
- resume(self): Resumes playing the currently paused song.
- stop(self): Stops playing the current song and resets the current_song attribute to an empty string.
- add_song(self, song): Adds the specified song to the playlist.
- remove_song(self, song): Removes the specified song from the playlist.
- Write the MusicPlayer class implementation and provide a sample code snippet that demonstrates the usage of the class by creating an instance of MusicPlayer and performing various operations on it.
- Feel free to add any additional helper methods or attributes to enhance the functionality of the MusicPlayer class if you wish.

AI Implementation:

In the context of a music player, AI could be incorporated to enhance various aspects, such as:

In this program we create a playlist where the user can add the number of songs in the playlist. We can also run the song, change the song and also run another song

Which user can add the song in the playlist. Also add the different song in the playlist and user can remove the different in the playlists.

The running song user can stop and play another song in the playlists and user change the song by clicking next.

These are just a few examples of how AI can be utilized to enhance the functionality of a music player.

PROGRAM FOR MUSIC PLAYER APPLICATION

```
class MusicPlayer:
    def _init_(self, playlists):
        self.playlists = playlists
        self.current_song = ''

    def play(self, song):
        if song in self.playlists:
            self.current_song = song
            print(f"Playing {song}")
        else:
            print(f"{song} is not in the playlist.")

    def pause(self, current_song):
        if self.current_song:
            print(f"Pausing {self.current_song}")
        else:
            print("No song is currently playing.")

    def resume(self,current_song):
        if self.current_song:
            print(f"Resuming {self.current_song}")
        else:
            print("No song is currently playing.")

    def stop(self,current_song):
        if self.current_song:
            print(f"Stopping {self.current_song}")
            self.current_song = ''
        else:
            print("No song is currently playing.")

    def add_song(self, song):
        if song not in self.playlists:
            self.playlist.append(song)
            print(f"Added {song} to the playlist.")
        else:
            print(f"{song} is already in the playlist.")


    def remove_song(self, song):
        if song in self.playlists:
            self.playlists.remove(song)
            print(f"Removed {song} from the playlist.")
        else:
            print(f"{song} is not in the playlist.")
```

```
playlists = ["song1.mp3", "song2.mp3", "song3.mp3"]

for playlist in playlists:
    print(playlist)

player = MusicPlayer(playlists)
crntSong=0
while True:

    print("1.play")
    print("2.pause")
    print("3.resume")
    print("4.stop")
    print("5.add_song")
    print("6.remove_song")

    n=int(input("Enter option"))

    if n==1:
        player.play(player.playlists[crntSong])
    if n==2:
        player.pause(player.playlists[crntSong])
    if n==3:
        player.resume(player.playlists[crntSong])
    if n==4:
        player.stop(player.playlists[crntSong])
    if n==5:
        player.add_song(player.playlists[crntSong])
    if n==6:
        player.remove_song(player.playlists[crntSong])

player.play("song1.mp3")
player.pause()
player.resume()
player.stop()
player.add_song("song4.mp3")
player.remove_song("song2.mp3")
```

Explanation of the code

- The provided code defines a class called MusicPlayer that represents a basic music player. It has methods to control the playback of songs, manage playlists, and add or remove songs from the playlists. Here's a brief explanation of the code:

- The MusicPlayer class is defined with an _init_ method that initializes the player with a list of playlists and sets the current_song to an empty string.

- The play method takes a song as input and checks if the song is present in the playlists. If it is, it sets the current_song to the given song and prints a message indicating that the song is being played. If the song is not in the playlist, it prints a message indicating that the song is not in the playlist.

- The pause method checks if there is a song currently playing (current_song is not empty). If there is, it prints a message indicating that the current song is being paused. If there is no song playing, it prints a message indicating that no song is currently playing.

- The resume method checks if there is a song currently playing. If there is, it prints a message indicating that the current song is being resumed. If there is no song playing, it prints a message indicating that no song is currently playing.

- The stop method checks if there is a song currently playing. If there is, it prints a message indicating that the current song is being stopped, sets the current_song to an empty string, and stops the playback. If there is no song playing, it prints a message indicating that no song is currently playing.

- The add_song method takes a song as input and checks if it is already present in the playlists. If it is not, it adds the song to the playlists and prints a message indicating that the song has been added. If the song is already in the playlist, it prints a message indicating that the song is already present.

- The remove_song method takes a song as input and checks if it is present in the playlists. If it is, it removes the song from the playlists and prints a message indicating that the song has been removed. If the song is not in the playlist, it prints a message indicating that the song is not present.

- The code then creates an instance of the MusicPlayer class, passing the playlists list as an argument.

- It enters a loop where it continuously prompts the user to enter an option (1-6) for controlling the music player.

- Based on the user's input, it calls the corresponding method of the player object (play, pause, resume, stop, add_song, remove_song) by passing the current song (player.playlists[crntSong]) as an argument.

- Lastly, it calls some methods of the player object directly to demonstrate their usage, such as playing a specific song, pausing, resuming, stopping, adding a new song, and removing an existing song

OUTPUT:

```
song1.mp3
song2.mp3
song3.mp3
1.play
2.pause
3.resume
4.stop
5.add_song
6.remove_song
Enter option1
Playing song1.mp3
1.play
2.pause
3.resume
4.stop
5.add_song
6.remove_song
Enter option2
Pausing song1.mp3
1.play
2.pause
3.resume
4.stop
5.add_song
6.remove_song
Enter option3
Resuming song1.mp3
1.play
2.pause
3.resume
4.stop
5.add_song
6.remove_song
Enter option4
Stopping song1.mp3
1.play
2.pause
3.resume
4.stop
5.add_song
6.remove_song
Enter option5
song1.mp3 is already in the playlist.
1.play
2.pause
3.resume
4.stop
5.add_song
6.remove_song
Enter option6
```

Removed song1.mp3 from the playlist.
1.play
2.pause
3.resume
4.stop
5.add_song
6.remove_song
Enter option