**Use Case: Music Player Application**

1. User launches the music player application.
2. The application initializes with a pre-defined playlist containing some songs.
3. The user is presented with a menu of options to interact with the music player.
4. The user can perform the following actions:

**a. Play a Song:**

* The user selects the "Play a song" option.
* The application prompts the user to enter the number of the song they want to play.
* The application checks if the entered song number is valid and exists in the playlist.
* If the song is found, it starts playing the song and displays a message with the currently playing song.
* If the song is not found, it displays a message indicating that the song is not in the playlist.

**b. Pause the Current Song:**

* The user selects the "Pause the current song" option.
* If a song is currently playing, the application pauses the song and displays a message with the paused song's name.
* If no song is currently playing, it displays a message indicating that no song is currently playing.

**c. Resume the Current Song:**

* The user selects the "Resume the current song" option.
* If a song is currently paused, the application resumes playing the paused song and displays a message with the resumed song's name.
* If no song is currently paused, it displays a message indicating that no song is currently paused.

**d. Stop the Current Song:**

* The user selects the "Stop the current song" option.
* If a song is currently playing, the application stops playing the song, clears the current song, and displays a message with the stopped song's name.
* If no song is currently playing, it displays a message indicating that no song is currently playing.

**e. Add a Song to the Playlist:**

* The user selects the "Add a song to the playlist" option.
* The application prompts the user to enter the name of the song they want to add.
* The application checks if the entered song is already in the playlist.
* If the song is not in the playlist, it adds the song to the playlist and displays a message indicating the successful addition.
* If the song is already in the playlist, it displays a message indicating that the song is already in the playlist.

**f. Exit the Application:**

* The user selects the "Exit" option.
* The application terminates the loop and exits the program.

1. The menu is displayed repeatedly until the user chooses to exit the application.

This use case represents a basic music player application where users can play, pause, resume, stop songs, and also add new songs to the playlist. It provides an interactive interface for users to control their music playback experience.

**Problem Statement: Music Player Application**

Design and implement a Music Player application that allows users to manage and play songs from a predefined playlist. The application should provide a menu-driven interface for users to interact with the player and perform various actions such as playing a song, pausing/resuming the current song, stopping the current song, and adding new songs to the playlist.

Requirements:

1. The application should initialize with a pre-defined playlist containing a set of songs.
2. The user should be able to play a song by selecting the "Play a song" option from the menu. They should be prompted to enter the number of the song they want to play. The application should validate the input and play the selected song if it exists in the playlist. If the song is not found, an appropriate error message should be displayed.
3. The user should be able to pause the currently playing song by selecting the "Pause the current song" option from the menu. If a song is playing, it should be paused and a message displaying the paused song's name should be shown. If no song is currently playing, an appropriate message should be displayed.
4. The user should be able to resume the currently paused song by selecting the "Resume the current song" option from the menu. If a song is paused, it should be resumed and a message displaying the resumed song's name should be shown. If no song is currently paused, an appropriate message should be displayed.
5. The user should be able to stop the currently playing song by selecting the "Stop the current song" option from the menu. If a song is playing, it should be stopped, the current song should be cleared, and a message displaying the stopped song's name should be shown. If no song is currently playing, an appropriate message should be displayed.
6. The user should be able to add new songs to the playlist by selecting the "Add a song to the playlist" option from the menu. They should be prompted to enter the name of the song they want to add. The application should check if the song is already in the playlist. If it is not, the song should be added to the playlist and a message indicating the successful addition should be displayed. If the song is already in the playlist, an appropriate message should be shown.
7. The user should be able to exit the application by selecting the "Exit" option from the menu.

**Constraints:**

* The playlist should be initialized with a predefined set of songs.
* The menu should be displayed repeatedly until the user chooses to exit the application.
* The application should handle invalid inputs gracefully and display appropriate error messages.
* The application should maintain the state of the currently playing and paused songs.
* The application should provide clear and informative messages to the user for each action performed.

**Code for Music Player Application :**

class MusicPlayer:

def \_\_init\_\_(self, playlist):

self.playlist = playlist

self.current\_song = ''

def play(self, song):

if song in self.playlist:

self.current\_song = song

print()

print("\* Now playing: " + song)

print()

else:

print(song + " is not in the playlist.")

def pause(self):

if self.current\_song:

print()

print("Paused: " + self.current\_song)

print()

else:

print("No song is currently playing.")

def resume(self):

if self.current\_song:

print()

print("Resuming: " + self.current\_song)

print()

else:

print("No song is currently paused.")

def stop(self):

if self.current\_song:

print()

print("Stopped: " + self.current\_song)

print()

self.current\_song = ''

else:

print("No song is currently playing.")

def add\_song(self, song):

if song not in self.playlist:

self.playlist.append(song)

print()

print("Added " + song + " to the playlist.")

print()

else:

print(song + " is already in the playlist.")

# Create an instance of MusicPlayer

playlist = ['song1.mp3', 'song2.mp3', 'song3.mp3']

player = MusicPlayer(playlist)

while True:

print("\*\*Choose an action\*\*")

print("1. Play a song")

print("2. Pause the current song")

print("3. Resume the current song")

print("4. Stop the current song")

print("5. Add a song to the playlist")

print("0. Exit")

choice = input("Enter your choice: ")

if choice == "1":

song\_index = int(input("Enter the song number to play: ")) - 1

if song\_index >= 0 and song\_index < len(playlist):

player.play(playlist[song\_index])

else:

print("Invalid song number.")

elif choice == "2":

player.pause()

elif choice == "3":

player.resume()

elif choice == "4":

player.stop()

elif choice == "5":

new\_song = input("Enter the name of the song to add: ")

player.add\_song(new\_song)

elif choice == "0":

break

else:

print("Invalid choice. Please try again.\n")

**Explanation of code :**

1. The MusicPlayer class has several methods and an initializer (\_\_init\_\_):

* The \_\_init\_\_ method takes a playlist parameter and initializes the playlist and current\_song attributes of the class instance.
* The play method takes a song parameter and checks if the song is in the playlist. If it is, the current\_song attribute is updated, and the song is printed as the current song. If the song is not in the playlist, a message is printed indicating that it's not in the playlist.
* The pause method checks if there is a current song playing. If there is, it prints a message indicating that the current song is paused. If no song is playing, it prints a message indicating that there is no song currently playing.
* The resume method checks if there is a current song paused. If there is, it prints a message indicating that the current song is resumed. If no song is paused, it prints a message indicating that there is no song currently paused.
* The stop method checks if there is a current song playing. If there is, it prints a message indicating that the current song is stopped and sets the current\_song attribute to an empty string. If no song is playing, it prints a message indicating that there is no song currently playing.
* The add\_song method takes a song parameter and adds it to the playlist if it's not already present. It prints a message indicating whether the song was added or if it's already in the playlist.

1. An instance of the MusicPlayer class is created with an initial playlist containing three songs: 'song1.mp3', 'song2.mp3', and 'song3.mp3'.
2. The program enters a while loop that allows the user to choose an action:

* The available actions are displayed, and the user is prompted to enter their choice.
* Depending on the choice, the corresponding method of the player instance is called to perform the selected action.
* If the choice is '1', the user is prompted to enter a song number to play. The play method is called with the selected song from the playlist.
* If the choice is '2', '3', or '4', the corresponding method (pause, resume, or stop) is called.
* If the choice is '5', the user is prompted to enter the name of a song to add to the playlist. The add\_song method is called with the entered song name.
* If the choice is '0', the while loop is exited, and the program terminates.
* If the choice is invalid, an error message is displayed.

Overall, this code provides a basic music player functionality allowing you to play, pause, resume, stop, and add songs to a playlist.

**Output of code Music Player Application**

\*\*Choose an action\*\*

1. Play a song

2. Pause the current song

3. Resume the current song

4. Stop the current song

5. Add a song to the playlist

0. Exit

Enter your choice: 1

Enter the song number to play: 3

\* Now playing: song3.mp3

\*\*Choose an action\*\*

1. Play a song

2. Pause the current song

3. Resume the current song

4. Stop the current song

5. Add a song to the playlist

0. Exit

Enter your choice: 2

\* Paused: song3.mp3

\*\*Choose an action\*\*

1. Play a song

2. Pause the current song

3. Resume the current song

4. Stop the current song

5. Add a song to the playlist

0. Exit

Enter your choice: 39

\* Resuming: song3.mp3

\*\*Choose an action\*\*

1. Play a song

2. Pause the current song

3. Resume the current song

4. Stop the current song

5. Add a song to the playlist

0. Exit

Enter your choice: 4

\* Stopped: song3.mp3

\*\*Choose an action\*\*

1. Play a song

2. Pause the current song

3. Resume the current song

4. Stop the current song

5. Add a song to the playlist

0. Exit

Enter your choice: 5

Enter the name of the song to add: song4.mp3

\* Added song4.mp3 to the playlist.

\*\*Choose an action\*\*

1. Play a song

2. Pause the current song

3. Resume the current song

4. Stop the current song

5. Add a song to the playlist

0. Exit

Enter your choice: 0

PS D:\6thsem\_Internship2023\Python>