1. Introduction

The music player project is a simple console-based application built using Python, designed to allow users to play, pause, stop, and navigate between audio tracks. It also includes functionality such as displaying song information, controlling playback speed, and managing playlists.

This documentation provides an overview of the project, details about its features, usage instructions, and the technical aspects involved in developing the application.

2. Objectives

- **To provide a user-friendly console interface for controlling music playback.**
- **To allow users to load, play, pause, and stop audio files.**
- **To navigate between different audio tracks in a playlist.**
- **To manage and organize songs in the playlist.**
- **To enable basic music controls like volume adjustment and playback speed.**
- **To provide an educational experience in working with file handling, audio processing, and user interface design in Python.**

3. Features

1. **Audio Playback Control:**
    - **Play, Pause, Stop audio.**
    - **Resume playback after pause.**
    - **Navigate to the next and previous songs in the playlist.**
    - **Shuffle the playlist for random playback order.**
    - **Adjust playback speed (slow down or speed up).**
2. **Playlist Management:**
    - **Add songs to a playlist.**
    - **Display the list of songs in the current playlist.**
    - **Remove a song from the playlist.**
3. **Volume Control:**
    - **Adjust the volume using command-line options.**
4. **Song Information:**
    - **Display song metadata like title, artist, and album.**
5. **Command-line Interface:**
    - **Easy-to-use commands for all functionalities.**
    - **Clear text prompts and feedback to guide the user.**

4. Technologies Used

- **Programming Language: Python 3.x**
- **Libraries/Modules:**
    - **pygame (for audio playback and control).**
    - **os (for file handling and directory management).**
    - **time (for time-based actions like pause duration).**
    - **glob (for finding all audio files in a directory).**

5. System Requirements

- **Operating System: Any OS (Windows, Linux, macOS) with Python 3 installed.**
- **Python Version: Python 3.7 or higher.**
- **Required Libraries:**
    - **pygame – Install via pip install pygame.**
    - **Other Python standard libraries (os, time, glob, etc.) are used for file handling and basic operations.**

6. Project Structure

The project is organized into the following directory structure:

bash

Copy code

music_player_project/

│

├── main.py # Main file to run the music player

├── music_files/ # Directory where audio files are stored

├── playlist.txt # Text file for saving the playlist

├── README.md # Project documentation (this file)

└── requirements.txt # Required Python libraries for the project

7. Functional Flow and Description of Components

main.py

This is the entry point for the music player. The file contains functions for controlling the flow of the program and the core features.

Key Functions:

1. **load_playlist(): Loads a playlist from the text file.**

2. **add_to_playlist(song): Adds a new song to the playlist.**
3. **remove_from_playlist(song): Removes a song from the playlist.**
4. **play_song(song): Plays the selected song using the pygame.mixer.**
5. **pause_song(): Pauses the currently playing song.**
6. **stop_song(): Stops the playback.**
7. **next_song(): Moves to the next song in the playlist.**
8. **prev_song(): Moves to the previous song in the playlist.**
9. **shuffle_playlist(): Shuffles the playlist for random song order.**
10. **adjust_volume(level): Changes the volume based on user input.**
11. **display_song_info(song): Displays metadata like title, artist, and album.**

The script begins by loading a playlist and presenting the user with a menu to choose from various actions like playing a song, adjusting the volume, and viewing song info.

playlist.txt

A text file that stores the paths or names of audio files. The format is simple, with each line containing one song file path. The playlist is saved when modifications are made.

music_files/

This folder contains all audio files in supported formats (e.g., MP3, WAV). These audio files can be played using the music player.

8. Code Example

python

Copy code

```
import pygame
import os
import random


# Initialize the mixer module from pygame for audio playback
pygame.mixer.init()


# Global variables
playlist = []
current_song = None
```

```
def load_playlist():
    global playlist
    if os.path.exists("playlist.txt"):
        with open("playlist.txt", "r") as f:
            playlist = [line.strip() for line in f.readlines()]

def add_to_playlist(song):
    playlist.append(song)
    with open("playlist.txt", "a") as f:
        f.write(song + "\n")

def play_song(song):
    global current_song
    if song != current_song:
        pygame.mixer.music.load(song)
        pygame.mixer.music.play()
        current_song = song
    print(f"Now playing: {song}")

def pause_song():
    pygame.mixer.music.pause()

def stop_song():
    pygame.mixer.music.stop()

def next_song():
    global current_song
    index = playlist.index(current_song)
    next_index = (index + 1) % len(playlist)
    play_song(playlist[next_index])
```

```
def prev_song():
    global current_song
    index = playlist.index(current_song)
    prev_index = (index - 1) % len(playlist)
    play_song(playlist[prev_index])

def shuffle_playlist():
    random.shuffle(playlist)
    print("Playlist shuffled.")

def display_song_info(song):
    # Placeholder for metadata (can be extended with libraries like mutagen for real data)
    print(f"Song: {song} | Artist: Unknown | Album: Unknown")

def main():
    load_playlist()

    while True:
        print("\nMusic Player Menu:")
        print("1. Play Song")
        print("2. Pause Song")
        print("3. Stop Song")
        print("4. Next Song")
        print("5. Previous Song")
        print("6. Shuffle Playlist")
        print("7. Show Song Info")
        print("8. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
```

```
            song = input("Enter song filename to play: ")
            play_song(song)
        elif choice == "2":
            pause_song()
        elif choice == "3":
            stop_song()
        elif choice == "4":
            next_song()
        elif choice == "5":
            prev_song()
        elif choice == "6":
            shuffle_playlist()
        elif choice == "7":
            song = input("Enter song filename to view info: ")
            display_song_info(song)
        elif choice == "8":
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

9. Usage

1. **Running the Application:**
    - **To start the music player, run the following command in the terminal:**

```
bash
Copy code
python main.py
```

2. **Basic Commands:**
    - **To play a song: 1 then enter the song filename.**
    - **To pause a song: 2.**

- **To stop a song: 3.**
- **To skip to the next song: 4.**
- **To go to the previous song: 5.**
- **To shuffle the playlist: 6.**
- **To view song info: 7 then enter the song filename.**

10. Challenges & Solutions

Challenge: Handling different audio file formats.

- **Solution: The pygame.mixer module supports a variety of audio file formats like MP3 and WAV. By ensuring the files are in compatible formats, we can avoid issues during playback.**

Challenge: User experience on the console.

- **Solution: To enhance the user experience, clear and detailed instructions are provided, and user inputs are validated.**

11. Future Improvements

- **Add support for more advanced audio file formats using other libraries like pydub.**
- **Implement metadata extraction (e.g., artist, album) from audio files using libraries like mutagen.**
- **Implement error handling for unsupported file formats or broken paths.**
- **Integrate a search feature to find songs in large playlists.**
- **Add graphical user interface (GUI) using tkinter or another Python GUI framework.**

12. Conclusion

This console-based music player offers a simple yet functional application for managing and playing music. By utilizing Python's libraries, the project achieves an interactive, easy-to-use solution for music control and playlist management. It serves as a great foundation for expanding the features and enhancing the user experience.