🎵 Custom MP3 Player – Project Overview

🔧 Built With:

- **Language**: Python 3.13
- **UI Framework**: CustomTkinter (CTk)
- **Audio Backend**: Pygame (mixer module)
- **Metadata**: mutagen for MP3 duration parsing
- **File Management**: JSON for folders and settings

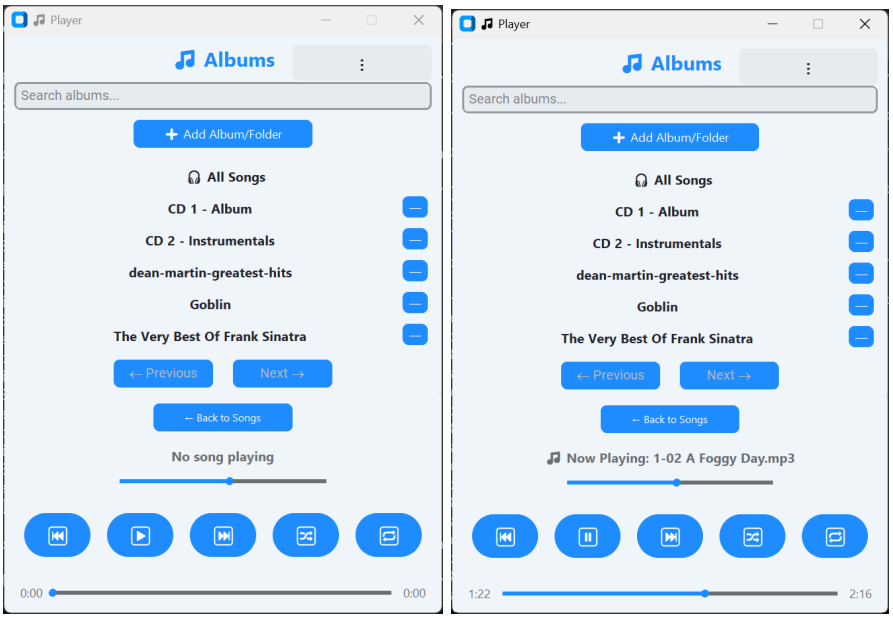
🎯 Purpose:

A lightweight, desktop MP3 player with album browsing, global search, theme customization, and intuitive playback controls.📦 **Core Features**🎨 **Interface**

- Album view, song view, and player view
- Light/Dark theme support with dynamic color accents
- Screens managed cleanly through centralized main_ui.py

🎵 Playback System

- Play, pause, seek, skip, volume, loop
- Loop mode to repeat the current song
- Progress bar and time indicators



🗂️ Album Management

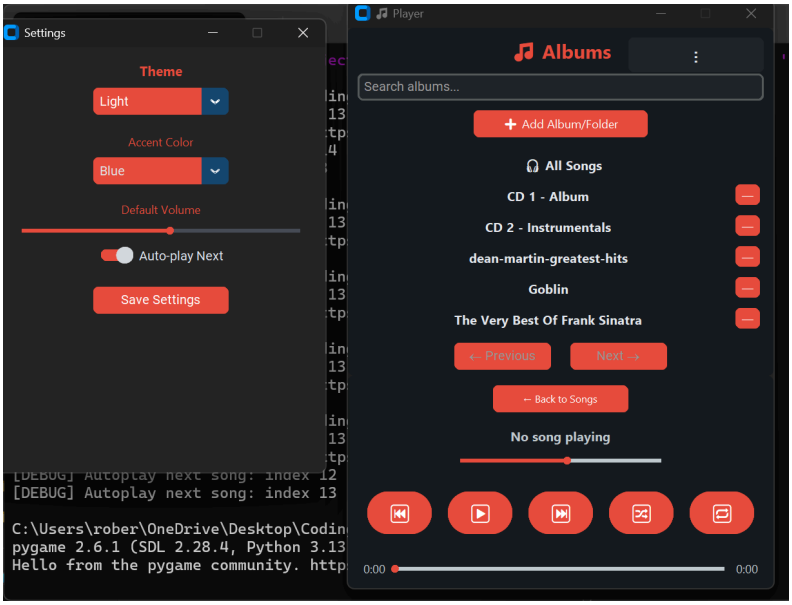
- User can add/remove local folders containing .mp3 files
- Albums are displayed in a grid and sorted alphabetically
- All folder data is stored in music_folders.json

🔍 Global Song Search

- **All Songs** virtual album aggregates songs from all folders
- Allows instant access and searching across the entire music library

⚙️ Settings

- Save/load theme, default volume, shuffle/loop preferences
- settings.json manages persistent user preferences
- Settings UI with live theme switching and accent selection



🗃️ Data Files

- music_folders.json: Stores paths to user-selected music folders
- settings.json: Stores theme, volume, shuffle, loop, and color preferences

📁 Project Directory Overview

Root

- `main.py`: Entry point of the application
- `main_ui.py`: Main UI controller managing all screens

`/core` – Core Playback Logic

- `player.py`: Handles all audio playback using `pygame`
- `constants.py`: Shared constants

`/screens` – User Interface Screens

- `album_screen.py`: Displays albums (music folders)
- `song_screen.py`: Lists songs within a selected album
- `player_screen.py`: Shows playback controls and song info
- `settings_window.py`: UI for app settings
- `base_screen.py`: Abstract base for all screens

`/managers` – Data Management

- `settings_manager.py`: Loads and saves `settings.json`
- `folder_manager.py`: Manages `music_folders.json` (user-added folders)

`/data` – Saved User Data

- `settings.json`: Stores UI theme, volume, shuffle, and loop settings
- `music_folders.json`: Stores user-added music folders

✅ Completed Enhancements

- Album sorting (alphabetical)
- All Songs virtual album
- Global search
- Refactored screen management
- Dynamic playback UI
- Theme + accent customization

Github: https://github.com/roberthoust/pythonmusicplayer