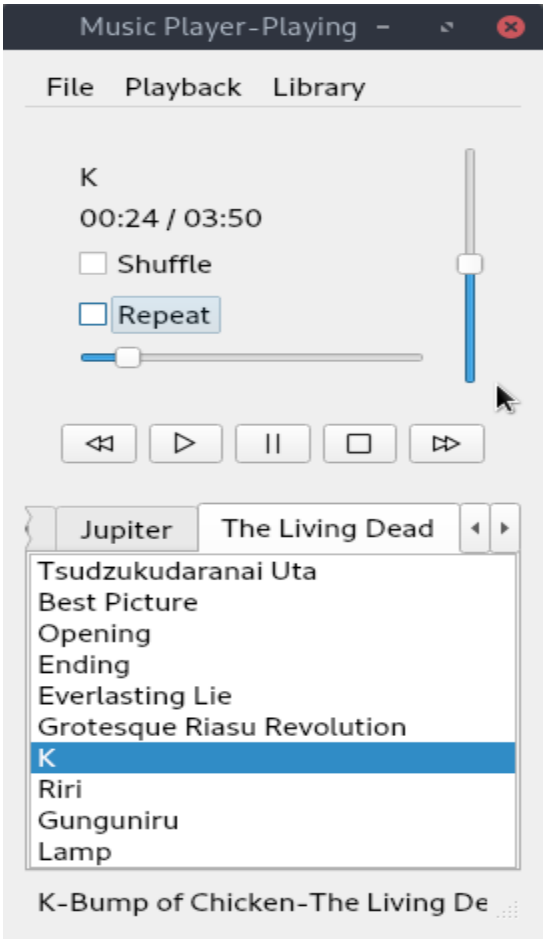
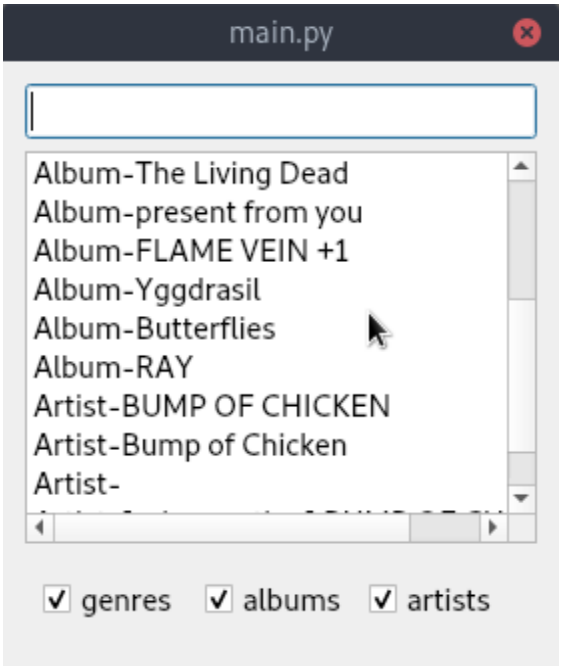
Music player





دو عکس نمونه از پنجره برنامه و همچنین قابلیت جستجو و مرتب سازی بر حسب ژانر و ....

پیش نیاز ها:

برای اجرای برنامه ابتدا نیاز هست که کتابخانه های PySide2 (برای qt که قسمت گرافیکی و بخش صدا را بر عهده دارد) و TinyTag(که برای گرفتن metadata آهنگ ها استفاده می شود) توسط pip نصب شده باشند سپس می توان با اجرای فایل main.py توسط مفسر بایتون برنامه را باز کرد.

همچنین برای مدیریت دیتابیس از sqlite3 استفاده می شود

قابلیت ها:

داخل برنامه ها آهنگ ها داخل چندین playlist مرتب شده اند که می توان آن ها را توسط قابلیت های داخل menubar برنامه یا shortcut های متناظرشان کنترل کرد که به این شرح اند:

Open file (Ctrl + O):

یک دیالوگ برای انتخاب یک آهنگ باز شده و بعد از انتخاب آن آهنگ به playlist فعلی که در برنامه انتخاب شده است اضافه می شود

Add playlist (Ctrl + T):

یک دیالوگ برای انتخاب اسم playlist ایجاد شده و بعد از انتخاب یک playlist خالی به آن اسم به برنامه اضافه می شود (در صورتی که playlist دیگری به همان اسم وجود نداشته باشد زیرا playlist های مختلف با توجه به نامشان تعیین می شود)

Playlist From a folder:

یک دیالوگ برای انتخاب یک پوشه باز می شود و بعد انتخاب یک بوشه تمام آهنگ هایی که درون آن پوشه یا درون یکی از بوشه های درون آن بوشه هستند به صورت بازگشتی درون یک playlist جدید هم نام با اسم آن پوشه قرار می گیرند

همچنان playlist درون یک دیتابیس ذخیره می شوند به همین دلیل بعد از خروج از برنامه و شروع دوباره آن هنوز باقی می مانند هم چنین دو playlist یlibrary و now playing خاص هستند به صورت اینکه در اولی(library) تمام آهنگ هایی که تا به حال به یک playlist اضافه شده اند در آن هستند (مجموعه ای از کل آهنگ ها هست) و now playing نیز یک playlist موقت است که اطلاعات آن در دیتابیس ذخیره نمی شود

علاوه بر مدیریت پلی لیست ها می توان سرعت بخش آهنگ را نیز با استفاده از دو قابلیت زیر سریع تر یا آرام تر کرد

Faster Playback (Alt + Right):

سرت بخش آهنگ را بیشتر می کند(با سقف دو برابر سرعت معمول)

Slower Playback (Alt + Left):

سرعت بخش آهنگ را کمتر می کند(با سقف نصف سرعت معمول)

همچنین با قابلیت Search Library(Ctrl + f)d آهنگ ها را با توجه به خواننده ژانر و یا آلبوم آن ها دسته بندی کرد به این صورت که یک پنجره جدید باز می شود که خواننده ژانر و آلبوم های مختلف لیست شده اند (و می شود آنها را با استفاده از سه checkbox قرارداده شده فیلتر کرد) و بعد از دابل کلیک کردن روی یکی از آیتم ها تمام آهنگ های آن خواننده ژانر و یا آلبوم داخل playlist nowplaying قرار می گیرند و پنجره بسته می شود

علاوه بر اینها کاربر می تواند با استفاده از دو slider قرار داده شده شدت صدار را کم و یا زیاد کند و همچنین آهنگ را جلو یا عقب ببرد (اگر focus روی آن اسلایدر باشد می توان از کلید های جهت دار و یا چرخ موس نیز استفاده کرد)

همچنین می توان با دو checkbox اه shuffle و repeat نیز حرکت در playlist را نیز کنترل کرد به این صورت که اگر repeat چک شده باشد آهنگ فعلی تکرار می شود و اگر shuffle چک شده باشد آهنگ بعدی به صورت رندوم انتخاب می شود (می توان با استفاده از دکمه و یا اتمام یک آهنگ به آهنگ بعدی رفت)

همچنین اطلاعات مربوط به آهنگ و بخش آن هم در بالای check box ها و هم در status bar پایین نمایش داده می شود

ساختار پروژه:

بروژه ای از ۴ فایل تشکیل شده که به شکل زیرند

Main.py:

تنها کاری که می کند فقط شروع برنامه و شی های مورد نیاز آن است که در فایل های دیگر قرار دارند همچنین بوشه فعلی را بوشه خود فایل می کند(نه بوشه ای که از آن برنامه شروع شده) که آدرس های نسبی درست کار کنند

Model.py:

شامل کد هایی می شود که بخش آهنگ مدیریت دیتابیس و مدیریت بلی لیست ها را به عهده دارد شامل دو کلاس AudioMetaData است که برای نگهداری metadata آهنگ ها استفاده می شود و class Model که وظایف گفته شده را به عهده دارد.

هنگام ساخت شی main_model از Model تابع __init__ کاری انجام نمی دهد و شروع تنظیم شی در تابع init قرار دارد که در main.py صدا زده می شود به این علت که بعضی از تنظیمات باید بعد از شروع qt انجام شود در ابتدا init تابع setup_datavase صدا زده می شود که در صورت وجود دیتابیس از قبل (برنامه قبلا شروع شده باشد) فقط به دیتابیس متصل شده و برمیگردد وگرنه جدول های مورد نیاز را می سازد بعد از آن لیست ها و دیکشنری های مورد نیاز برای مدیریت playlist ها درست می شود

بعد از وصل شدن به دیتابیس برنامه اطلاعات مربوط به پلی لیست ها را دریافت کرده و playlist ها را می سازد و با استفاده از سیگنال های playlistAdded و playlistUpdated بخش گرافیکی را از این تغییرات آگاه می کند (در مورد این قسمت در قسمت فایل Gui.py صحبت می شود)

شی Model برای بخش آهنگ از شی QMediaPlayer داخل Qt استفاده می کند و از QMediaPlaylist برای مدیریت playlist ها کمک می گیرد

شی Model. درخواست های مربوط به کار برنامه (مانند اضافه کردن یک بلی لیست و یا قطع و بخش آهنگ) را از قسمت دریافت می کند و به آنها عمل می کند

Gui.py:

کد بخش گرافیکی در این قسمت قرار که به صورت دستی شی ها(مثل دکمه ها menubar و ...) ساخته می شوند

و همچنین این بخش با استفاده از درخواست ها به شی Model (مانند صدا زدن تابع setVolume وقتی که اسلایدر صدا حرکت کند یا تابع add_file هنگامی که آهنگی به یک پلی لیست اضافه می شود) تغییرات را ایجاد می کند و با وصل کردن اسلات ها درست به سیگنال های QmediaPlayer مسئول بخش آهنگ و دو سیگنال داخل شی Model از تغییرات درونی ایجاد شده با خبر می شود (مانند تغییر status bar هر وقت که آهنگ فعلی عوض می شود و یا اضافه کردن یک tab هنگامی که یک playlist اضافه می شود)[1]

همچنین یک کلاس GuiHelper است که شامل چند static method هست که و دیالوگ ها مورد نیاز برای قابلیت های برنامه را باز کرده و ورودی را از کاربر می گیرند

LibSearch.py:

شامل کد مربوط به بنچره ایست که هنگام صدا زدن تابع search_library در کلاس GuiHelper و یا انتخاب Search library باز می شود که اطلاعات مربوط به آرتیست آلبوم ها و ژانر ها را از Model دریافت می کند و هنگام دابل کلیک روی یکی از آیتم ها با صدا زدن تابع add_files از Model آهنگ ها مربوط به آن آیتم را به بلی لیست Now Playing اضافه می کند (قسمت گرافیکی خود به خود توسط سیگنال PlaylistUpdated اه Model از این تغییر آگاه می شود)

[1] اسلات های و سیگنال ها از قابلیت ها ی qt هستند