Music Database Application (GitHub)

Caleb Huang, Dylan Hernandez, Moe Aung

Summary

This is an application similar to Spotify where one can look for songs based on genres and insert/remove songs essentially. The application will allow a variety of features. Firstly, you can *search a song* from keywords in the song's album title, song title, or artist name, further being able to *filter* down *by genre.* The produced list of songs can be *sorted* by song title, duration with ascending and descending order options. The database itself can be manipulated in several ways. '*Albums' can be added* to insert new songs and artists, and *songs can be deleted* individually. We also added the function of creating groups of existing songs in the database into *playlists.* If the user gets lost in the changes they've made, there is a *reset button* to restore default settings.

Key Features:

- **Searching**: Users can filter songs by album, song title, or artist using SQL WHERE conditions, with an additional genre filter.
- **Sorting**: The song list is ordered using SQL ORDER BY with options for ascending or descending sorting by title or duration.
- **Adding Albums**: Inserts new songs and artists into the database using SQL INSERT INTO.
- **Creating Playlists**: Playlists store song associations in the Playlist_Song table, allowing users to group songs together.
- **Deleting Songs/Playlists**: Deletes songs and playlists using SQL DELETE FROM WHERE, ensuring related entries are deleted as needed.
- **Resetting**: Clears applied filters and restores the default song list through a button.

Design

User Interface

Our application is built on a JavaFX AnchorPane and features a scrollable ListView of songs sorted alphabetically by default. Users have the option to change the sorting criteria to song length and can reverse the sorting order using a checkbox. The Search Bar allows users to

search by song name, artist, or album name; pressing the reset button clears any applied filters and returns the view to display all songs in the database. Additionally, users can filter songs by genre. Another function is to create playlists, which prompts the user for a playlist name. A new tab opens where the user can add existing songs to the playlist.

Functions

The application utilizes different SQL functions, including Data Definition Language, Data Query Language, and Data Manipulation Language.

DDL

The database consists of five tables: Artist, Song, Performs, Playlist, and Playlist_Song. The Artist table has an id and name attribute. The Song table has an id, name, album, and genre attributes. The Performs table connects the Artist and Song tables through foreign keys that reference Artist id and Song id; it also has a “on delete cascade” constraint, ensuring that if a song is deleted, the corresponding tuples in Performs are also removed. The Playlist table is a table with an id and name, and the Playlist_Song table connects playlists and songs together using foreign keys pointing to the playlist id and song id. It also has an “on delete cascade” constraint so that its tuples can be deleted if a playlist is deleted. Plus, Playlist_Song has an integer attribute that maintains the order in which songs were added to a playlist, allowing them to be displayed in the correct sequence.

DQL

We used many data queries to fetch the songs inside the database. The primary query selects all songs while joining them with the Performs table and Artist table, enabling the UI to display song attributes with the artist name. The filtering and sorting used data queries on the attributes of the Song table and Artist table. For instance, the dropdown that allows users to sort by “song name” executes a query with an “order” clause to arrange results in either ascending or descending alphabetical order.

DML

Users can add albums, which results in an album with one artist and multiple songs. In SQL, this involves inserting multiple records into the Song table, all with the same album name,

while also adding entries in the Performs table to link songs to their artists. If the artist exists, it will reuse the existing artist id. Otherwise, it will create a new artist. To delete individual songs, the program will delete by song name and artist name.

A user can create playlists in the Playlist table by pressing a button and entering song details. This triggers a query that joins Song and Artist together to fetch their ids, which are inserted into the Playlist_Song table with the Playlist id. To delete a playlist, the program removes the Playlist tuple along with its corresponding tuples in the Playlist_Song table.

How it was built

1. Setting it up:
    - Making Github repository and set rules for committing and managing
        - Assigning issues to group members
        - Making branches for different issues
        - Merging branch into main when finished with issue
        - Creating Readme with instructions to run project
    - Downloading JavaFX sdk and link to project
    - Downloading Java Scenebuilder and create fxml files for different pages
    - Connecting our SQL database with our Java
2. Learning
    - Learning how to use the JavaFX library and connect it with our fxml files
    - Learning how to use the SQL database in Java
3. Actual creation/coding
    - Laying out our database schemas
    - Creating our database tables
    - Inserting data into our database
    - Creating the UI through Scene Builder and the fxml files
    - Linking button functionalities through JavaFX
    - Creating SQL functionality
    - Linking buttons to certain SQL functionality
    - Displaying the results in a listview