Project 2 Report (Bonus)

Bonus Feature: Even better library management

Implementation Details

We further edited the `Playlist` class to include another parameter `isCollaborative` taking a `Boolean` value to implement this feature. And again, we had to make changes in the SQL files to change the table structure. And finally, changes were made in the front-end to add the functionality of making public playlists collaborative and read-only.

Base Code (Backend)

- Changes were made to `PlaylistDto.java`, `PlaylistCriteria.java` and `Playlist.java` to include the parameter `isCollaborative` in the class structure.
- `PlaylistMapper.java` was edited to map the `iscollaborative` parameter from the SQL query result to `isCollaborative` parameter of the `PlaylistDto` class object parameter.
- Changes were made to `PlaylistDao.java` to change the SQL `SELECT` query, `CREATE` query and the `UPDATE` query to support the table with the `ISCOLLABORATIVE` column.
- In `PlaylistResource.java`, new API calls were defined to return public collaborative playlists, toggle the collaboration status and integrate `isPublic` parameter in playlists.

Database

We added the following statement to `dbupdate-001-1.sql` to update the structure of the table storing the playlists to add a new boolean column that tells whether the playlist is public.

```
alter table
  T_PLAYLIST
add
  ISCOLLABORATIVE NUMBER(1);
```

Frontend

- In `NamedPlaylist.js` we added another array to the root scope named `publicCollaborativePlaylists` to store the data of all the public collaborative playlists.
- In `playlist.html` all of our changes are reflected.
  - We add a button to make public playlists collaborative and read-only.
  - We re-enabled buttons to delete tracks and move tracks when one user accesses another user's public collaborative playlist.
- In `playlistdropdown.directive.html` we display all the public collaborative playlists not belonging to the user below the user's playlist.