Spotify Playlist Application

Regular Exam [Spring Fundamentals]

The **Spotify Playlist Application** is here to help us keep in mind our shopping needs. The application is here for your music taste and aims to make music a more social experience. In Spotify Playlist, you can find hundreds of new songs that people had discovered and then you can add them to your customized playlist or maybe add your favorite songs so other people can listen to them. You can find new musical experiences by searching for styles of music you prefer.

There are several requirements you must follow in the implementation:

1. Database Requirements

The **Database** of the **Spotify Playlist** application needs to support **3 entities**:

User

- **Id** – Accepts **UUID-String** or **Long** values
- **Username**
  - The **length** of the **values** should be **between 3** and **20** characters long (both numbers are **INCLUSIVE**)
  - The values should be **unique** in the database
- **Password**
  - The **length** of the **values** should be **between 3** and **20** characters long (**INCLUSIVE**)
- **Email**
  - The values should contain a **'@'** symbol)
  - The values should be **unique** in the database
- **Playlist**
  - The playlist contains songs. One **user** may have many **songs** and one **song** can be **saved** by many **users to their playlist**.

Song

- **Id** – Accepts **UUID-String** or **Long** values
- **Performer**
  - The **length** of the **values** should be **between 3** and **20** characters long (both numbers are **INCLUSIVE**)
- **Title**
  - The **length** of the **values** should be **between 2** and **20** characters long (both numbers are **INCLUSIVE**)
- **Duration**
  - Duration in seconds must be **positive integer number**
- **Release date**
  - The Date cannot be in the future
- **Style**
  - One song **has one** style and one style **can have many** songs

Style

- **Id** – Accepts **UUID-String** or **Long** values
- **Style name**
  - The values should be **unique** in the database
  - an option between (**EXCELLENT**, **GOOD** and **ACCEPTABLE**)
- **Description**
  - Fell free to add some description to every classification

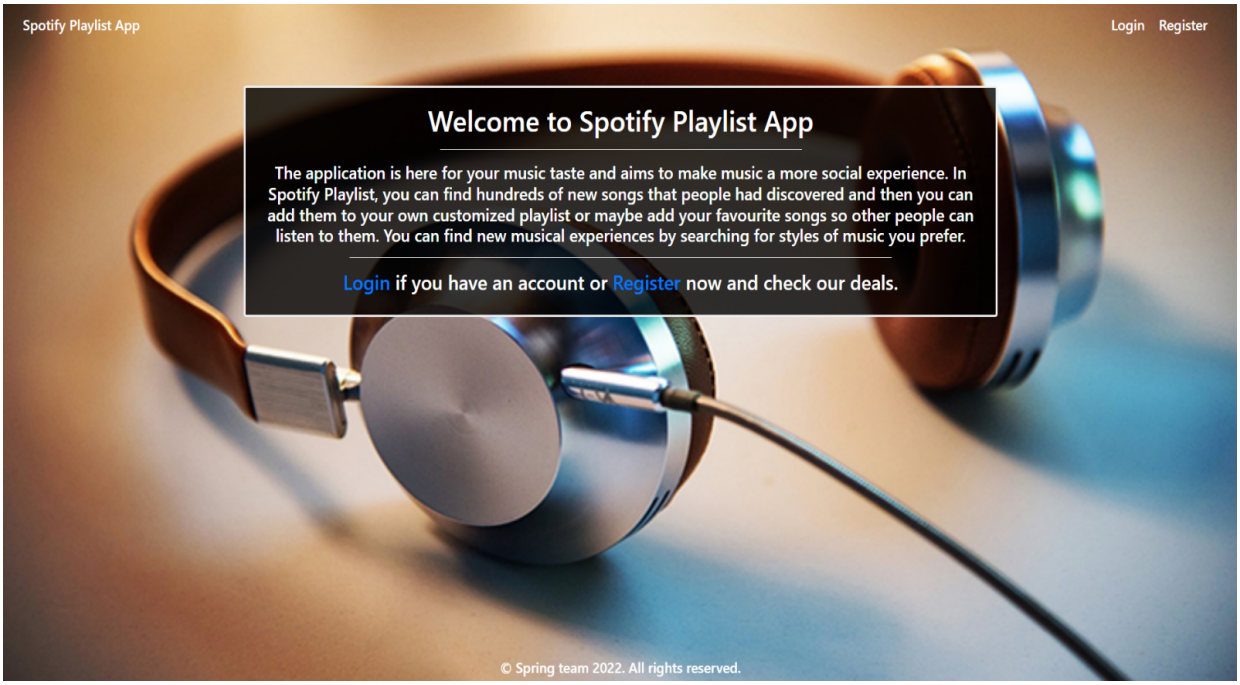
Nullable/Empty/Blank values are not allowed unless explicitly mentioned. Implement the entities with the **correct data types** and implement the **repositories** for them.

2. Initialize categories

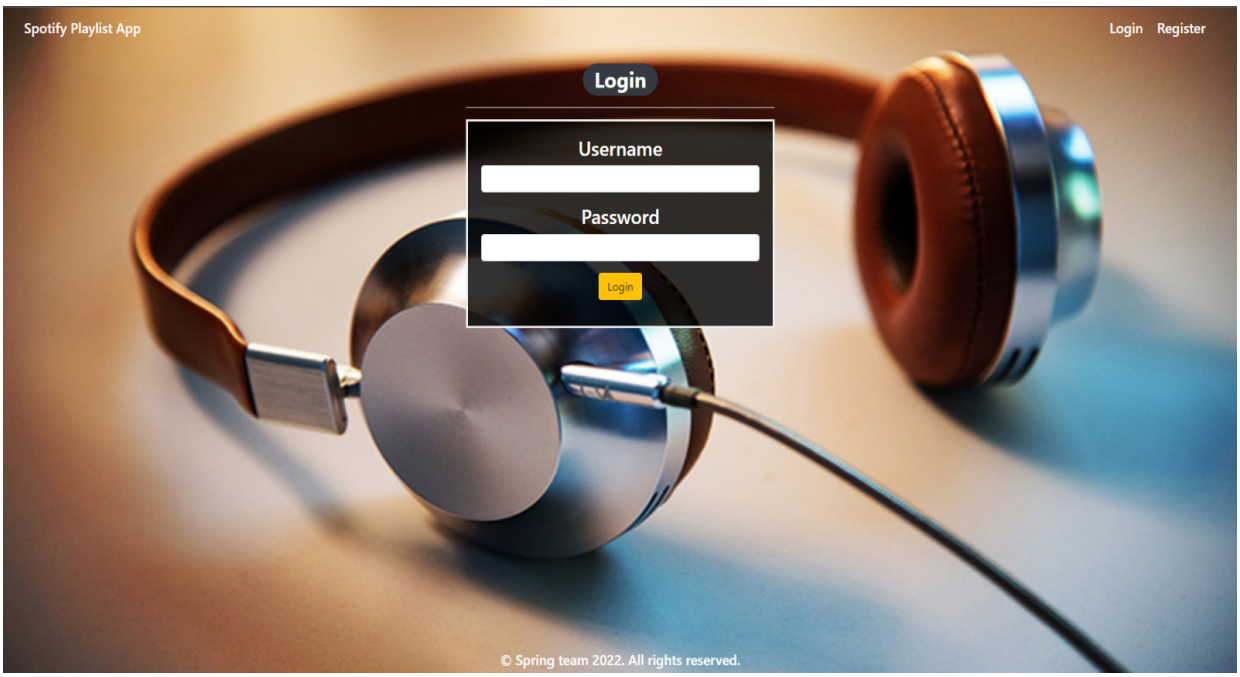
- Implement a method that checks (when app started) if the database does not have any category and initialize them
  - You are free to do this in some different ways
  - You can skip the description if you want

3. Page Requirements

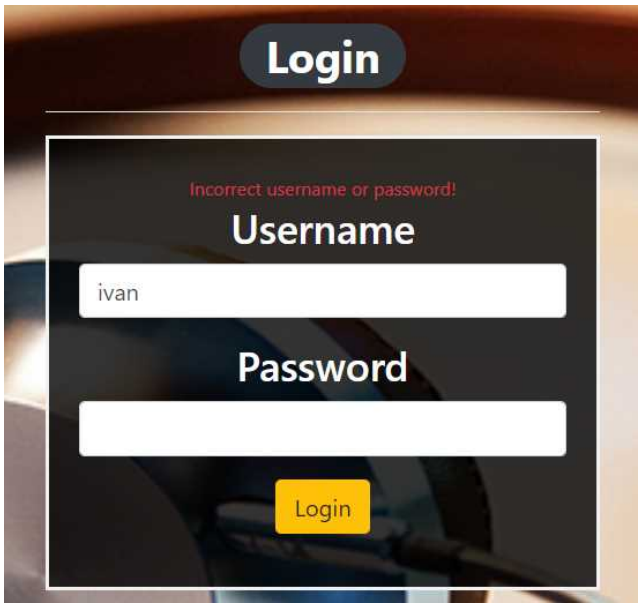
Index Page (logged out user)

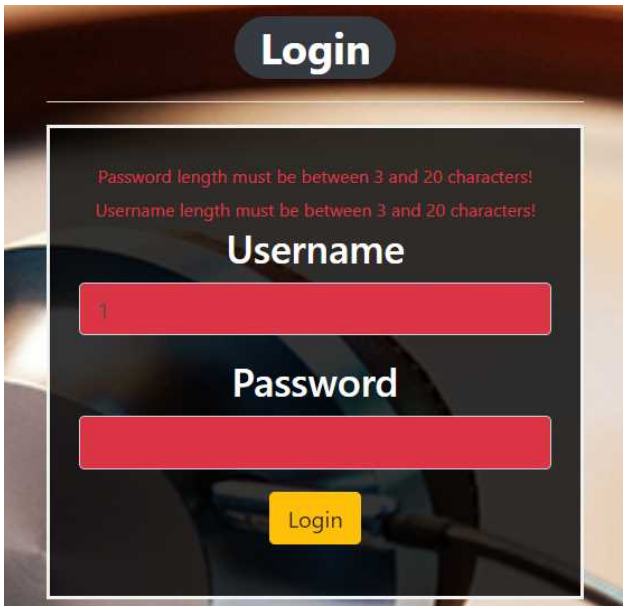


Login Page (logged out user)

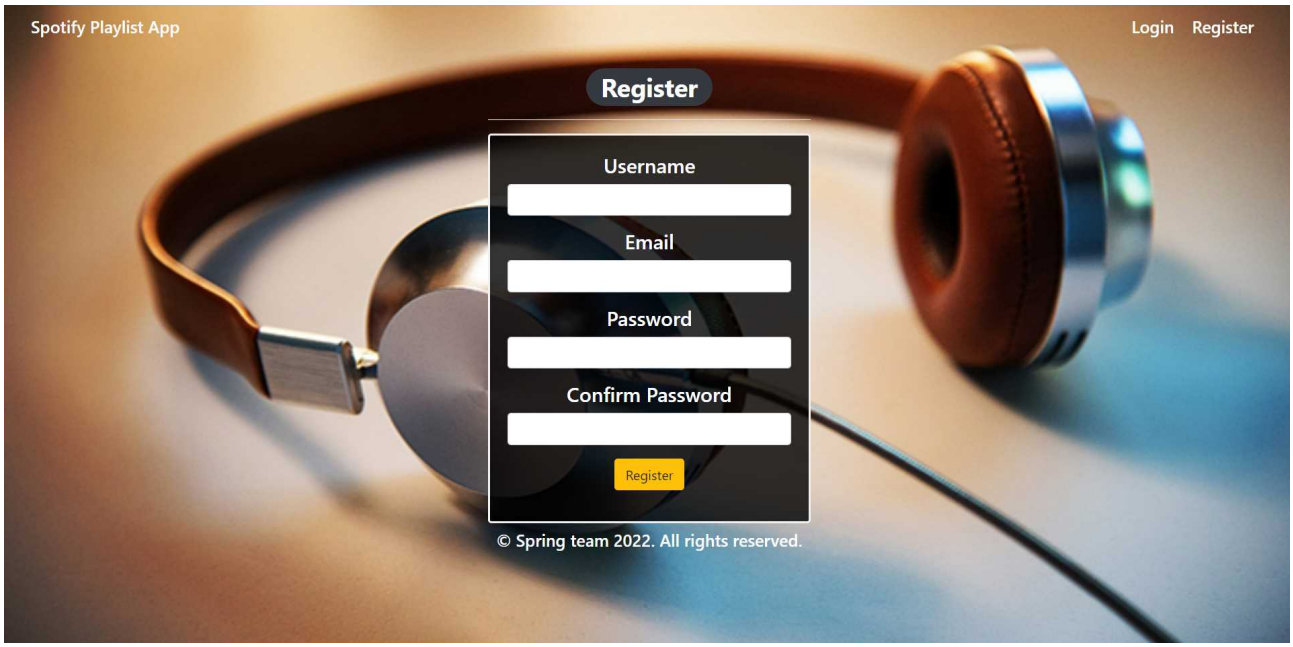


Login Page validations

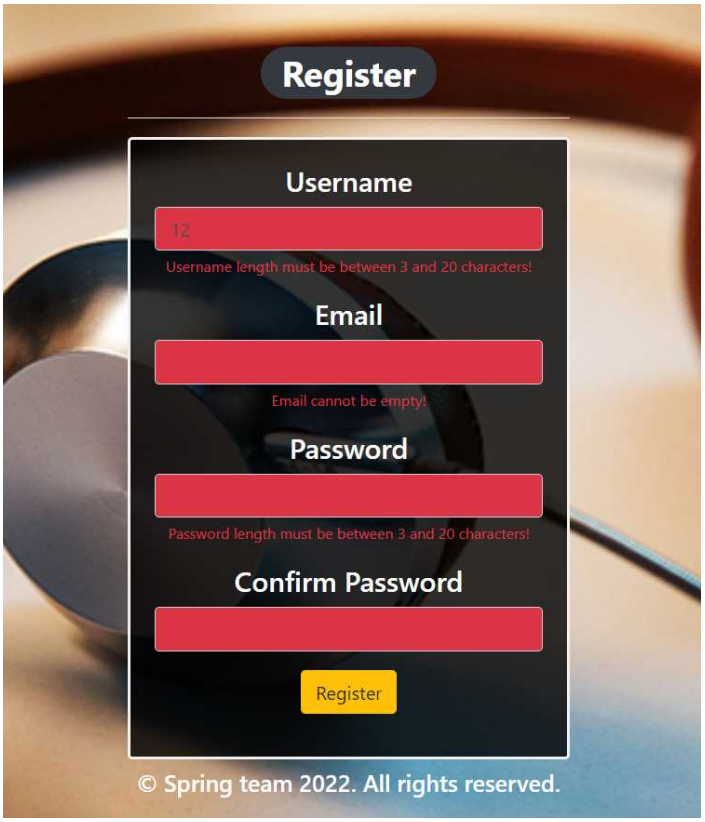




Registration Page (logged out user)

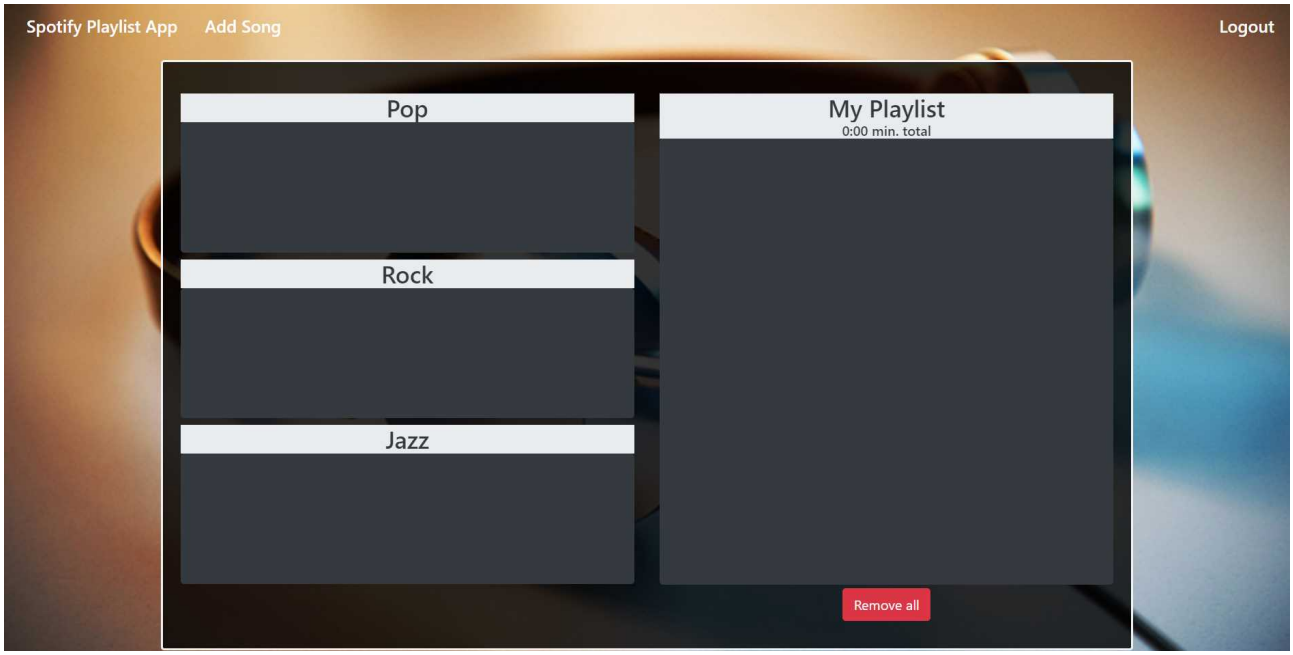


Registration Page validations

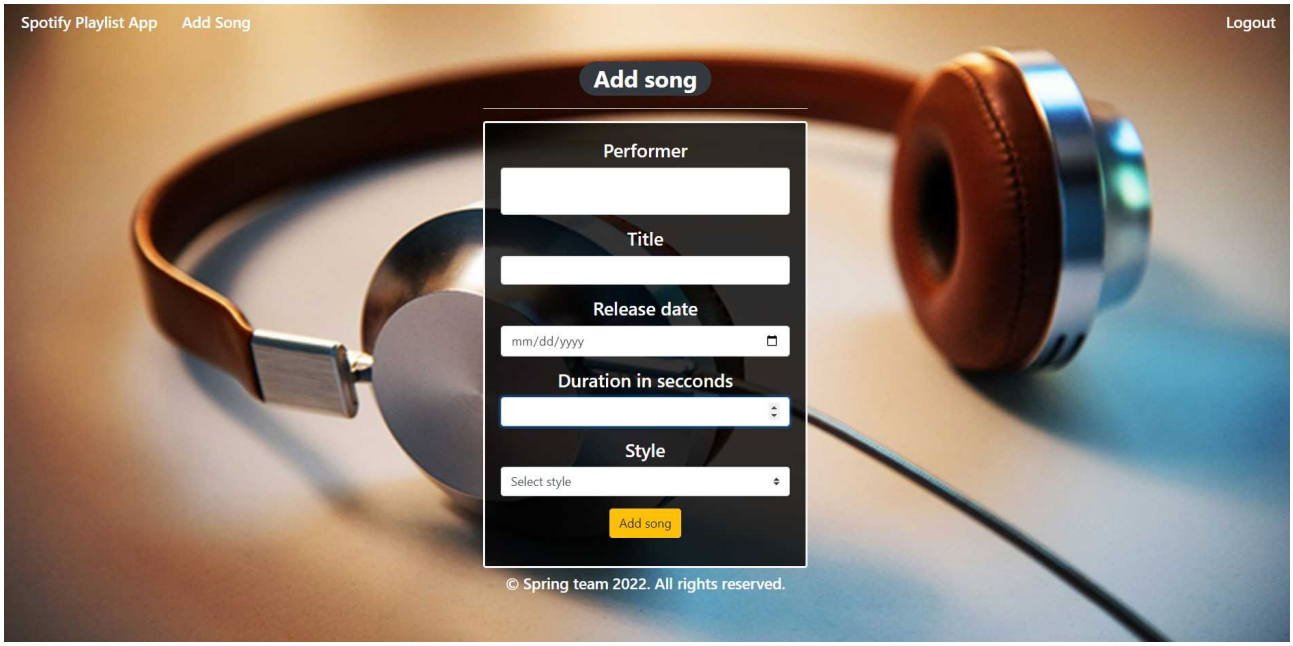


Home Page (without having any songs)

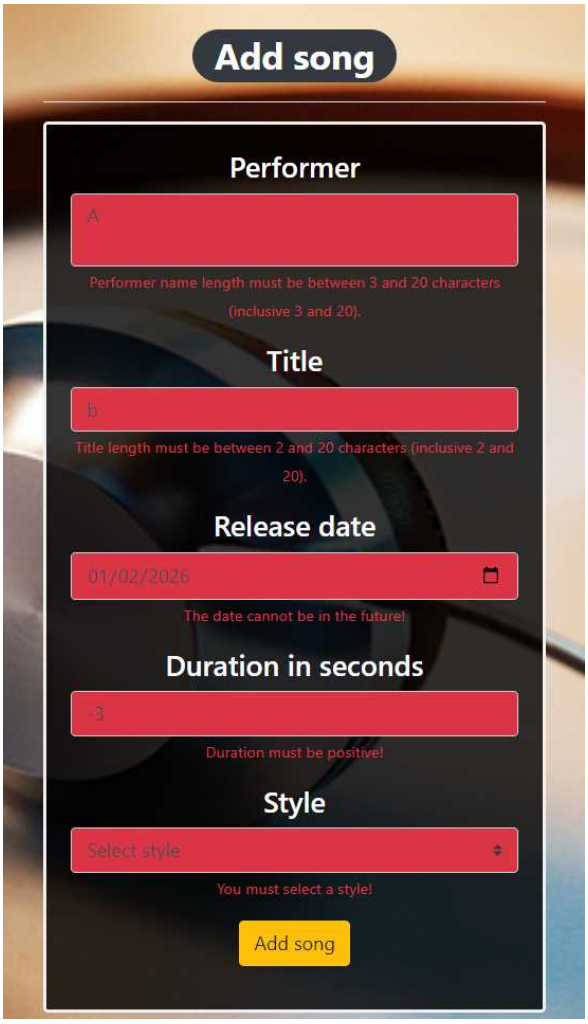
- The left section of the page should visualize **all of the songs** from the database.
- The right section (My Playlist) of the page should visualize **only songs** from the playlist of the **current logged user**.



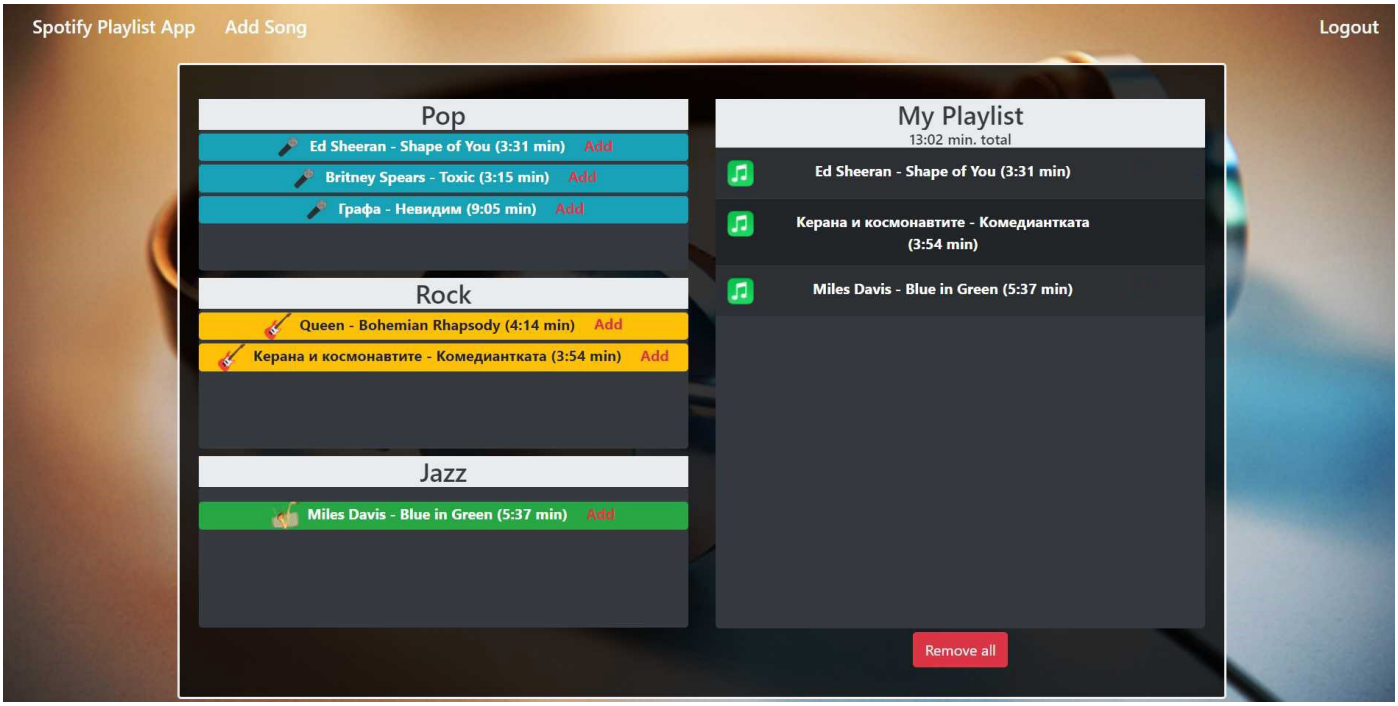
Add Song



Add Offer Validation



Home Page (with songs)



The templates have been given to you in the application skeleton, so make sure you implement the pages correctly.

NOTES:

- The templates should look **EXACTLY** as shown above.
- The templates do **NOT require additional CSS** for you to write. Only **bootstrap** and the **given CSS** are enough.

4. Functional Requirements

The Functionality Requirements describe the functionality that the application must support. The application should provide **Guest** (not logged in) users with the functionality to log in, register and view the Index page.

The application should provide Users (logged in) with the functionality to **log out, add a new song (Add song page), view all songs (Home page) and add the songs to My Playlist or Remove All songs from My Playlist.**

Spotify Playlist App in navbar should **redirect** to the appropriate URL depending on that if the **user is logged in**. The application should provide functionality for **adding songs** with styles of **Pop, Rock, or Jazz**.
The songs should be separated into different sections according to their styles. The image also depends on the item's category.
When the user clicks on the **Add button of some song**, he saves the song to **his playlist**. You should not **delete** this song from DB. When he clicks on **Remove all songs**, just delete all songs in the **user's playlist** in DB and again redirect to the home page.

The Remove all songs button should remove all songs only from the current user's playlist.

Bellow the My Playlist banner is located an info bar that shows the **sum of the duration** of all added songs in My Playlist.

The application should store its data in a **MySQL** database.

5. Security Requirements

The Security Requirements are mainly access requirements. Configurations about which users can access specific functionalities and pages.

- **Guest** (not logged in) users can access:
  - **Index** page;
  - **Login** page;
  - **Register** page.
- **Users** (logged in) can access:
  - **Home** page;
  - **Add Song** page;
  - **Logout** functionality.

6. Scoring

Database – 10 points
Pages – 25 points
Functionality – 35 points
Security – 5 points
Validations – 15 points
Code Quality – 10 points