Spotify Clone – Project Specification Document

Spring MVC Final Project – Exercise 4

Submission Date (Part 1): 12/6/2025

Final Submission Date: 5/7/2025

# General Explanation

Our project is a simplified Spotify clone web application built with Spring Boot MVC, Thymeleaf, and MySQL. It enables users to register, browse and upload songs in MP3 format, organize them into playlists, and mark favorites. The entire logic resides on the server side, with only minimal JavaScript used where necessary.

# Main Features

* MP3 song upload (by users)
* User authentication (register/login/logout)
* Browse songs and albums
* Create and manage playlists
* Mark songs as favorites
* View and play songs from favorites
* Upload and manage albums (title, cover, artist, date)
* Optional SPA-style page: basic audio player using REST API

# Main Pages and Their Goals

* Home Page (/home): Welcome page, featured albums/songs. Navigation bar for songs, playlists, favorites.
* Songs Page (/songs): Browse/search songs. Upload new song form with metadata and MP3 file.
* Albums Page (/albums): Browse albums and their songs. Upload album form with cover image.
* Playlist Page (/playlists): User-specific playlists. Create playlist, add/remove songs.
* Favorite Songs Page (/favorites): Songs marked as favorites by the current user.
* Login / Register Page: Authentication handled via Spring Security.
* Optional Player Page (/player): SPA-like page to preview and play MP3s via REST.

# Database Beans & Relationships

Entity Classes:

* User: id, username, email, password. One-to-many with Playlist, many-to-many with Song (favorites).
* Song: id, title, artist, genre, filePath, album\_id, uploadedBy\_id. Many-to-one with Album, many-to-many with Playlist and User.
* Album: id, title, artist, releaseDate, coverImagePath. One-to-many with Song.
* Playlist: id, name, user\_id. Many-to-many with Song.

# MP3 Upload Details

MP3 files are uploaded via a form using Spring's MultipartFile, stored on disk (or cloud), with metadata saved in the Song entity. Only authenticated users can upload songs. Security filters ensure only .mp3 files are accepted.

# Technology Requirements Mapping

|  |  |
| --- | --- |
| Requirement | Implementation Plan |
| Full Spring Boot MVC project | Uses Thymeleaf, controllers, service beans, and Spring MVC |
| 5+ major pages | Home, Songs, Albums, Playlists, Favorites, Player (optional SPA) |
| Use of session | Store logged-in user, active playlist etc. |
| Beans & DI | @Autowired services and repositories |
| JPA + MySQL | “ex4” schema with all entities and relations |
| Spring Security | Handles login and authorization |
| Minimal JavaScript | Used only for audio preview and basic validation |

# Optional Features (Stretch Goals)

* Users can edit/delete their uploaded songs
* Display total play counts
* RESTful mini-player (SPA page)
* File storage optimization and cloud upload