

Diagrams Spotiflop

Chitam Harone
Leyx-valade Hugo

Table of contents :

- Introduction
- ER Diagram
- Table Structure Diagram
- Use Case Diagram Authentication
- Use Case Diagram Playlist
- Gantt Diagram
- Activity Diagram (adding a song to playlist)
- Activity Diagram Login
- Sequence Diagram Login
- Sequence Diagram (Add Song To Playlist)
- Wireframe Diagram Login + Register
- Wireframe Diagram (Songs + edit + show song)
- Component Diagram Authentication
- Component Diagram Author
- Class Diagram

Introduction :

Music is a subject that can be created by many artists with their own style. Furthermore, since music is made and listened to worldwide, it was necessary to categorize them into different categories and genres to make it easier for listeners to search for them. Our project aims for a user to create multiple playlists and add 1 or many songs to them. Below you will find multiple diagrams to explain all the features our website could do.

ER Diagram :

The Entity-Relationship (ER) Diagram models the relationships between various entities in the system. In this diagram:

Users can have one or more playlists.

Each **Playlist** contains multiple songs.

Songs are created by an **Author** and belong to a specific **Genre**.

This diagram provides a conceptual understanding of how data entities (e.g., users, playlists, songs) are linked.

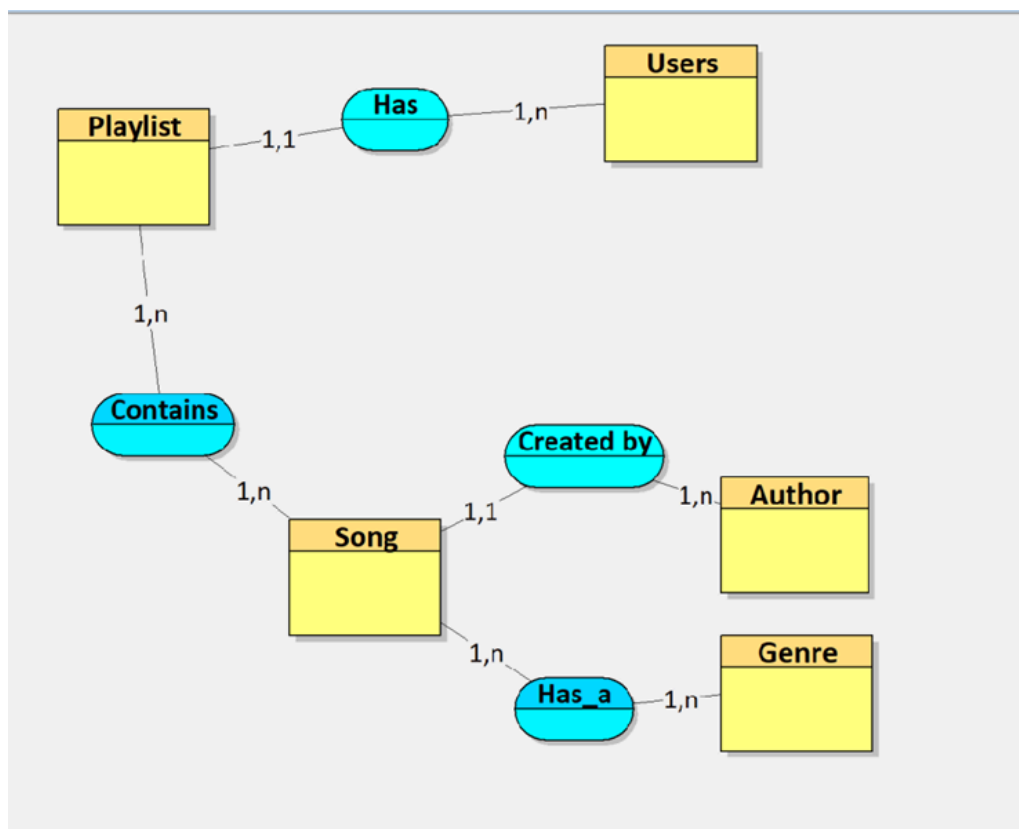
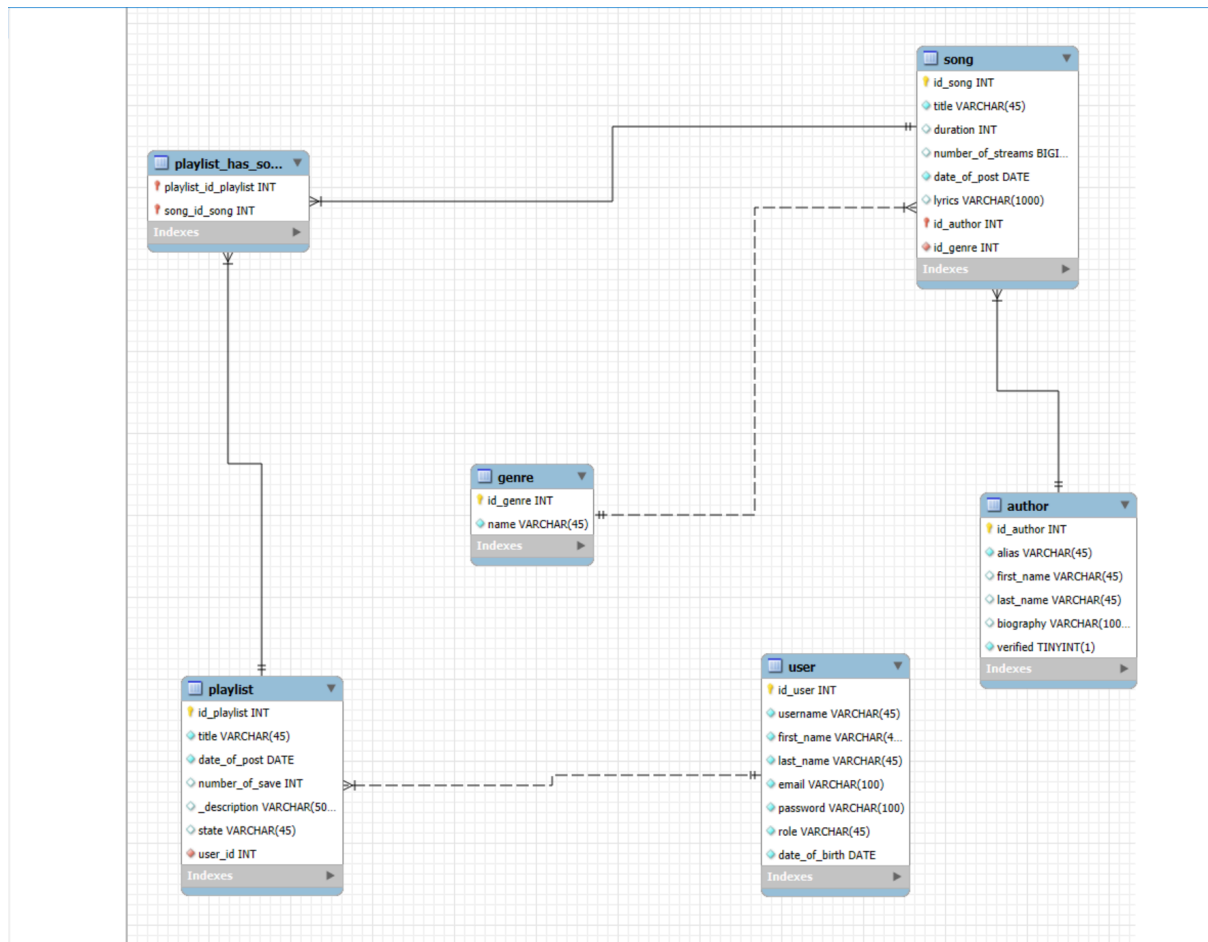


Table Structure Diagram :

This ER diagram represents the structure of the playlist management system, showing the relationships between key entities. Users can create multiple playlists, with each playlist belonging to one user and containing multiple songs through a many-to-many relationship managed by a junction table. Songs are created by a single author, belong to a specific genre, and can appear in multiple playlists. Authors can create multiple songs, while genres group multiple songs into categories. The PlaylistHasSong table links playlists and songs, facilitating the many-to-many relationship and ensuring modular data management.

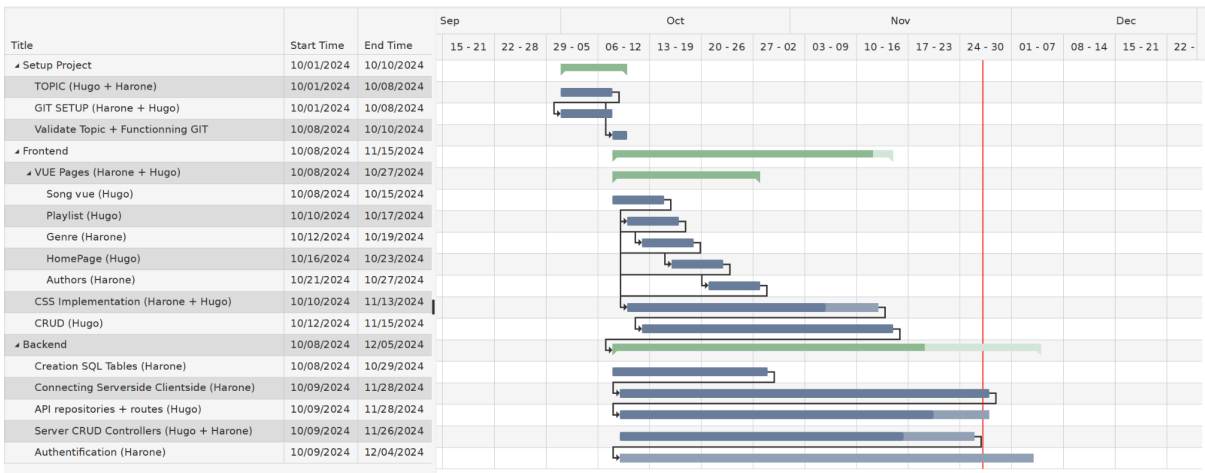


Gantt Diagram :

The Gantt Diagram provides a timeline view of the project's tasks and milestones. It includes:

- Tasks:** Each task is represented as a horizontal bar with a start and end date (e.g., GIT Setup, Frontend development).
- Dependencies:** Some tasks are linked to indicate that one task must finish before another begins.
- Team Members:** Tasks are assigned to specific team members (e.g., Hugo, Harone).

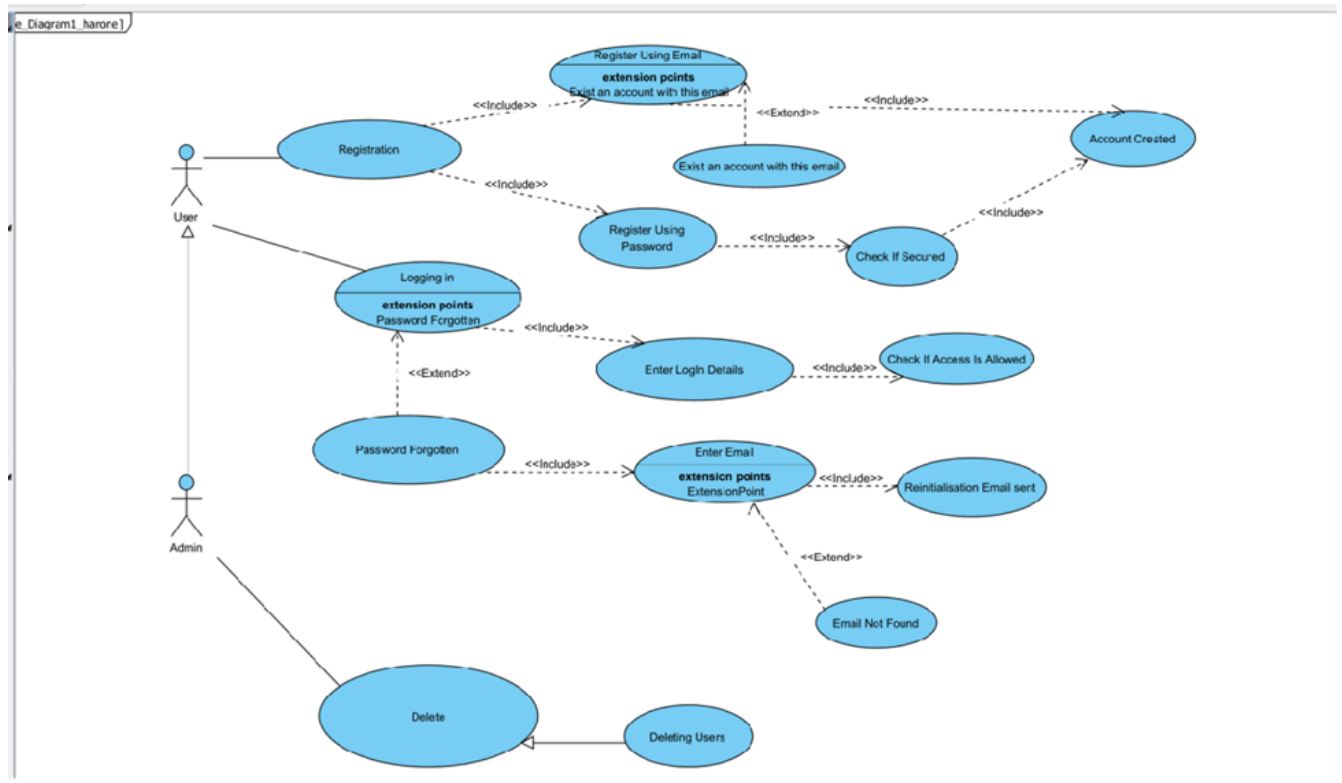
This chart is useful for tracking progress, managing workloads, and ensuring the project stays on schedule. It visually communicates deadlines and overlaps between tasks, helping coordinate the team's efforts effectively.



Use Case Diagram (Authentification) :

This diagram shows the different interactions users and administrators can have with the authentication system. Users can register, log in, and recover forgotten passwords. Admins can delete users or manage system configurations.

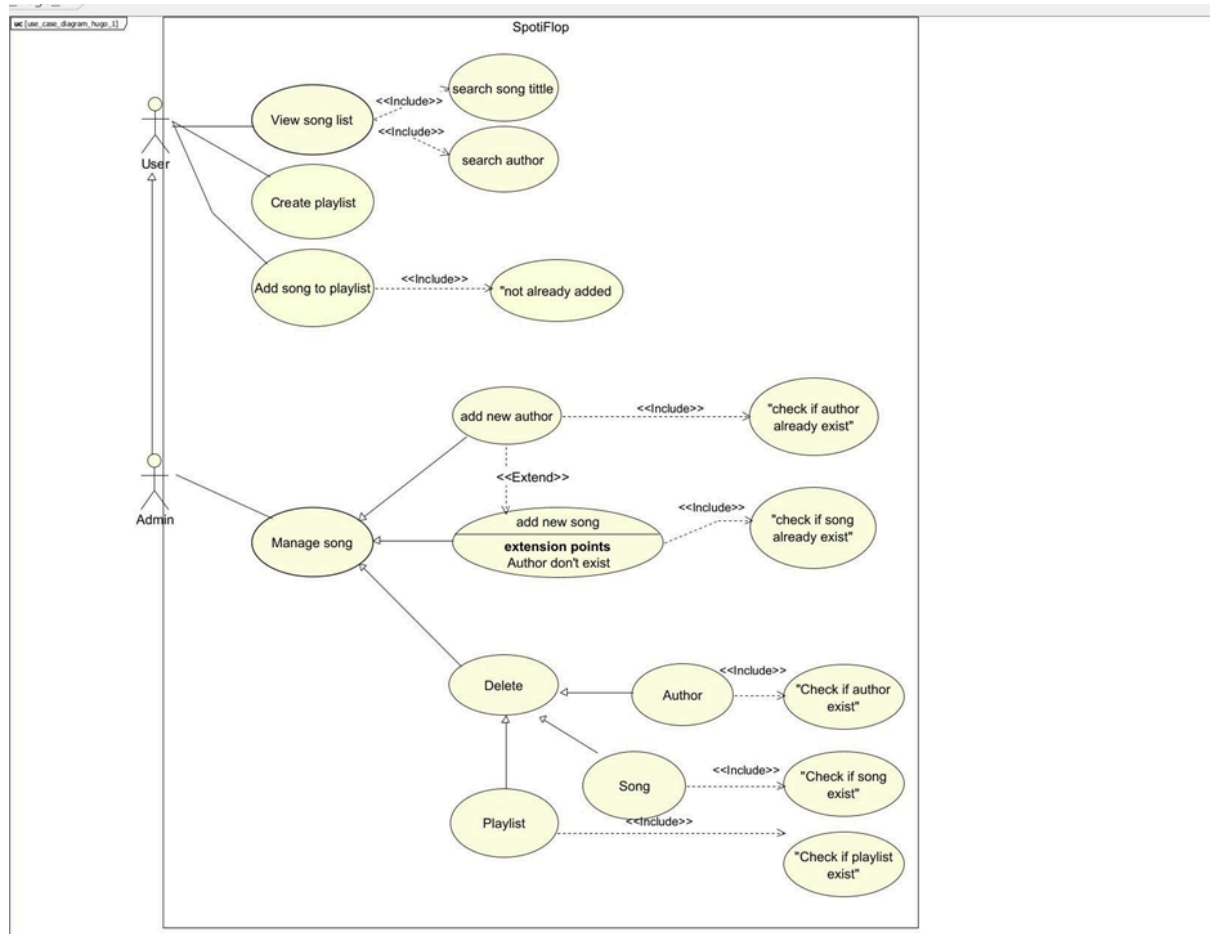
The diagram uses relationships like <<include>>: it shows something that will happen everytime and <<extend>> to show optional and mandatory interactions.



Use Case Diagram (Interaction User/Admin with Songs) :

The diagram represents how a user and an admin can manage a song. Both have distinct managing of IT:

- users can view a list of song, search one by the title and search an author, create a playlist and add a song to a playlist
- admin manages by adding one, adding an author, delete an author a song or a playlist.

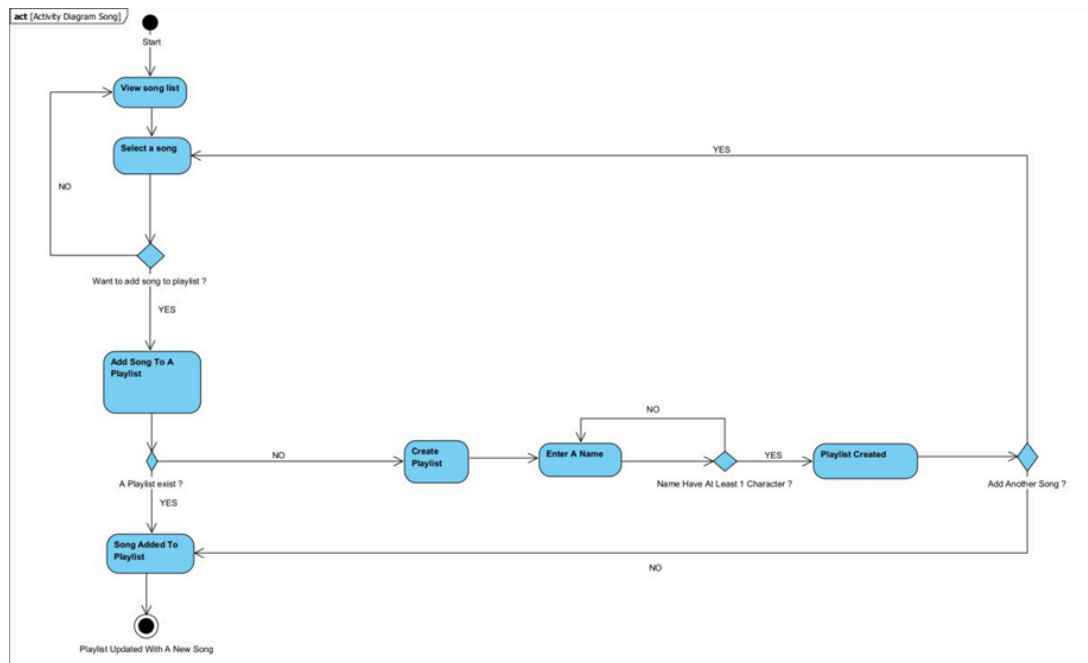


Activity Diagram (Adding Song To Playlist) :

This activity diagram illustrates the process of adding a song to a playlist. The flow begins when a user selects a song:

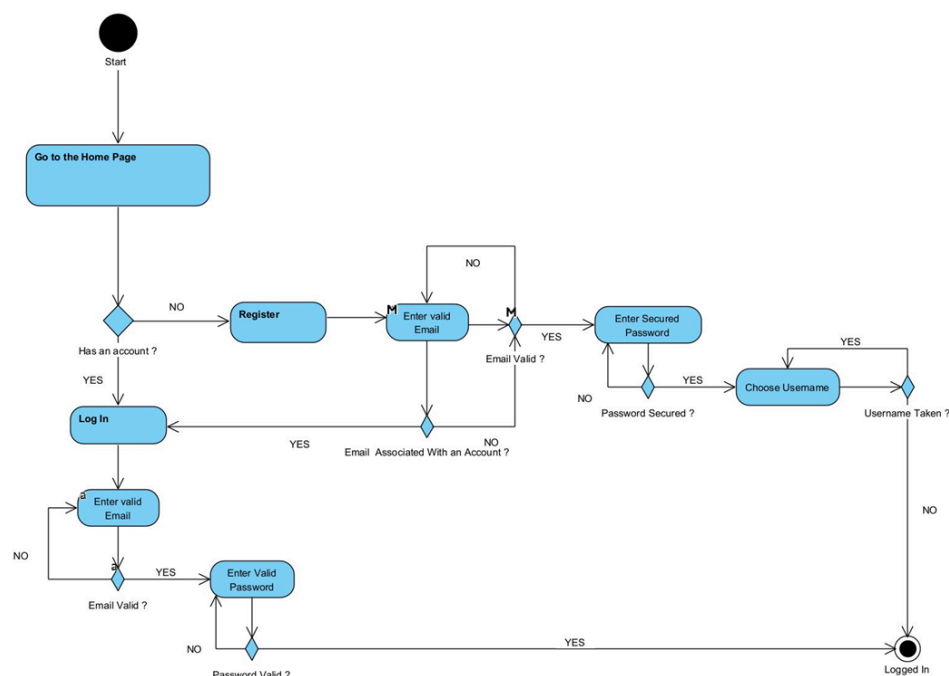
The system checks if the user wants to add it to a playlist. If no playlist exists, the user can create one. The song is then added to the playlist.

This diagram highlights user decisions and the system's responses step-by-step.



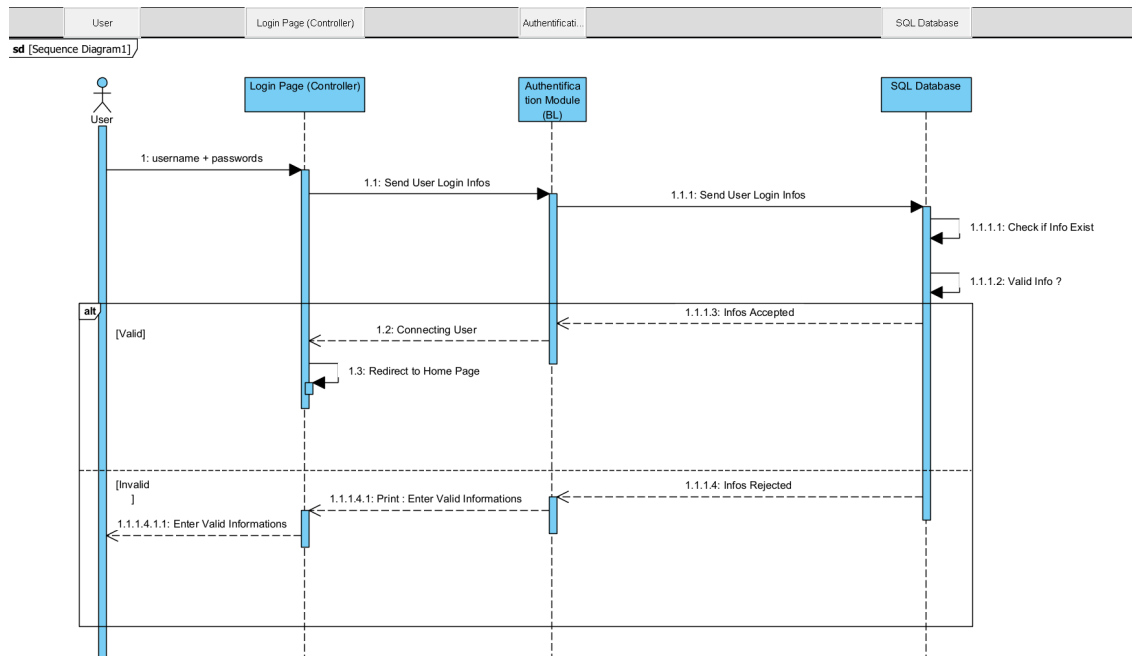
Activity Diagram (Login) :

This diagram shows step-by-step the process of user login. It starts with the user choosing whether to login or to register. For login, the system validates the email and password. For registration : the system checks if the email is unique (not already used) and if the password meets the necessary security.



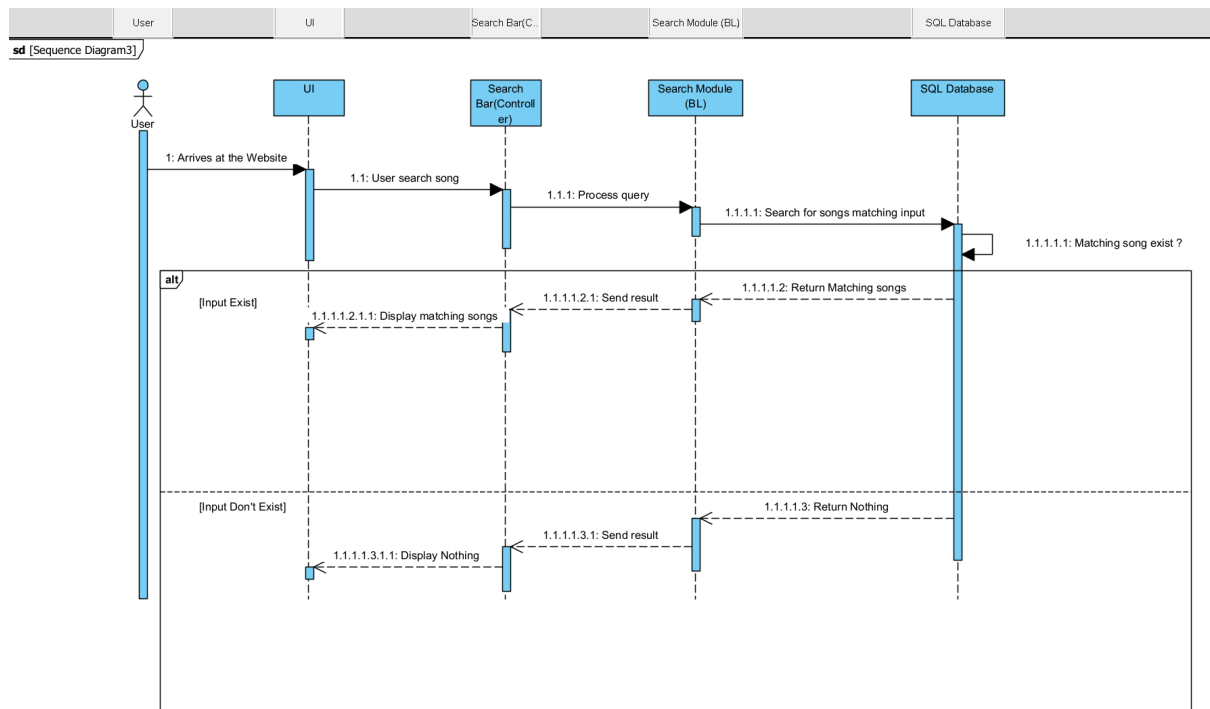
Sequence Diagram (Login) :

This diagram details the sequence of interactions between the user, login page, authentication module, and database: The user provides login credentials. The authentication module validates the credentials using the database. The system either grants access or rejects the login attempt. It showcases the system's logic flow during user login.



Sequence Diagram (Search Song) :

This sequence diagram illustrates how a song is being searched from the UI to the backend. If a matching song exists in the database it will display the song in the UI, else it will display nothing.



Wireframe Diagram (Login + Register) :

The wireframe shows the user interface for login and registration: Users can enter credentials to log in or fill in details (name, email, password) to register. It provides a visual layout of the forms and options available on the authentication pages

The wireframe diagram illustrates a web browser interface for authentication. The browser window has a title bar labeled "Browser" and a URL bar showing "http://localhost:8080/#/authentication/option".

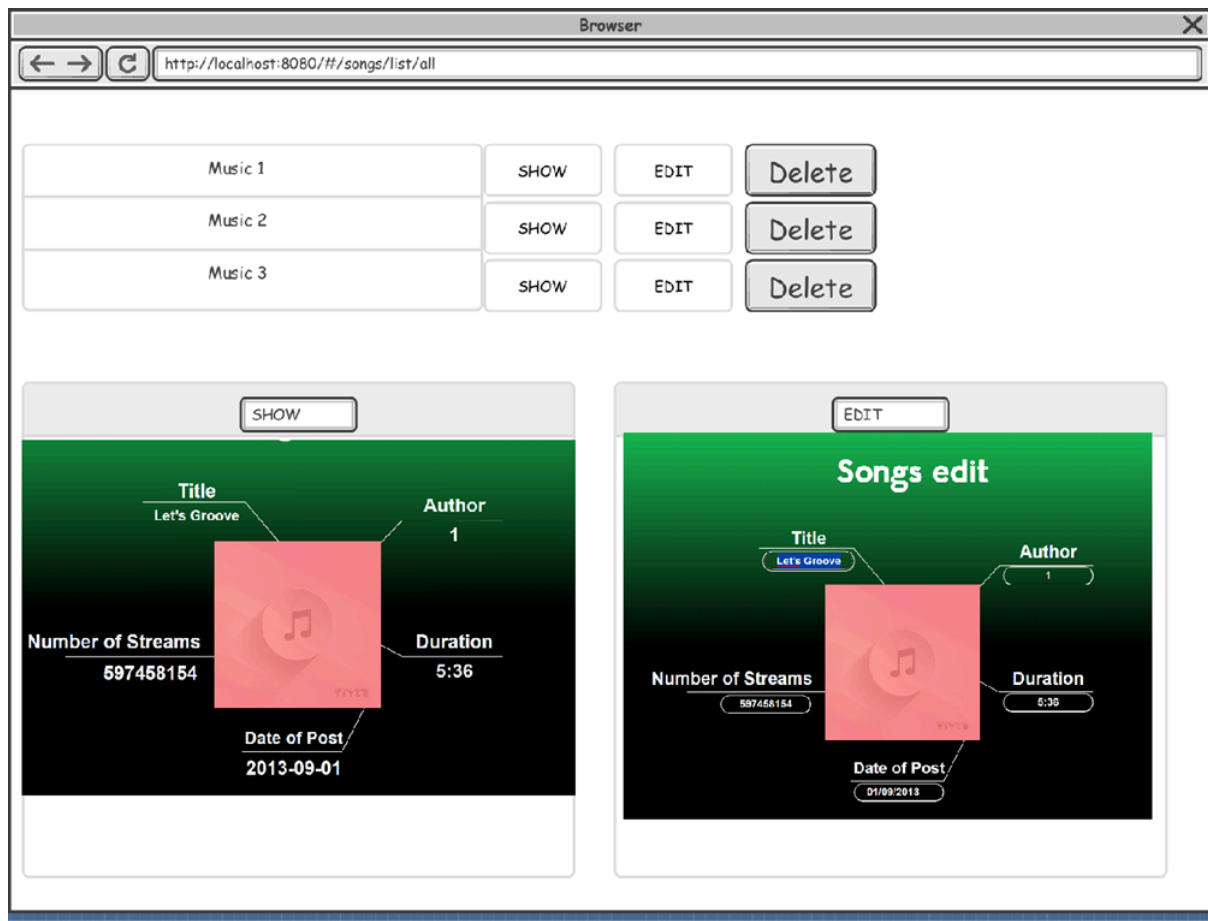
At the top center, there is a green circular logo with black horizontal lines and a dripping effect. Below the logo are two buttons: "Login" and "Register".

Below the buttons, there are two side-by-side form panels. The left panel is titled "http://localhost:8080/#/authentication/login" and contains three input fields: "Username", "Password", and a "Login" button. The right panel is titled "http://localhost:8080/#/authentication/register" and contains five input fields: "Username", "Firstname", "Lastname", "Email", and "Date of birth", followed by a "Password" field and a "Register" button.

Wireframe Diagram (Songs) :

This wireframe depicts the interface for managing songs: Users can view a list of songs, edit details, or delete songs. Detailed views for individual songs include information like title, author, and duration.

This design focuses on usability and efficient song management.

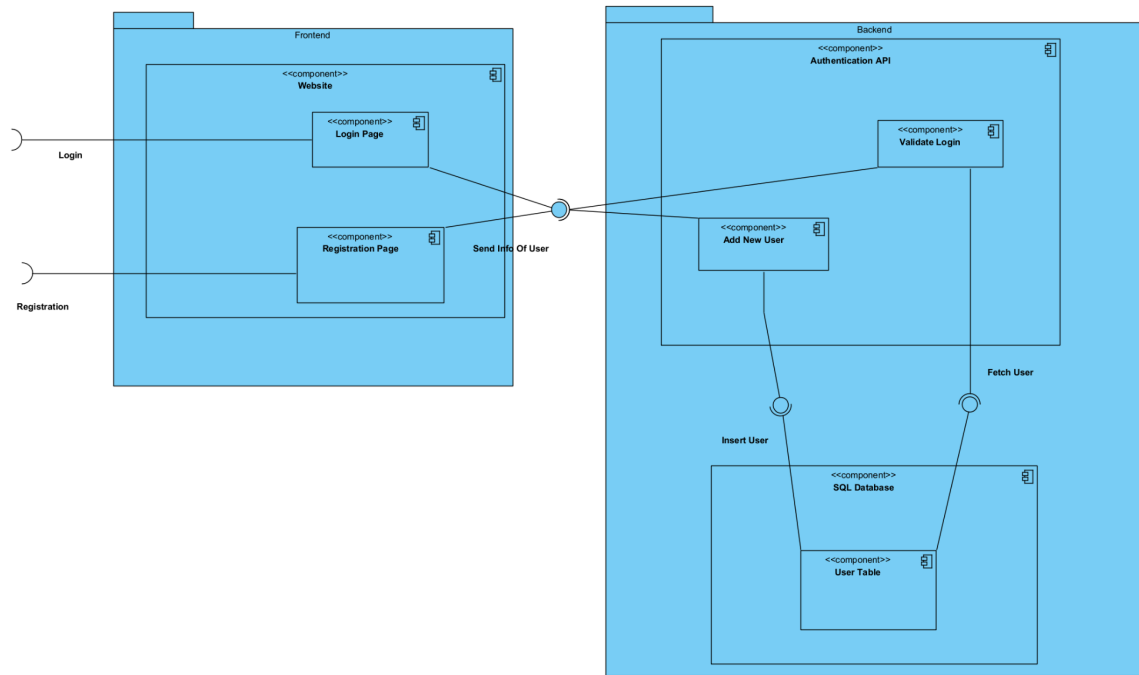


Component Diagram Authentication:

This component diagram illustrates the authentication process within the system. The Frontend Module includes two main components: the Login Page, where users enter their credentials to access the system, and the Registration Page, where new users can register by providing their details. These frontend components interact with the Authentication API, which contains two key services: Validate Login, responsible for verifying user credentials, and Add New User, which handles the registration process.

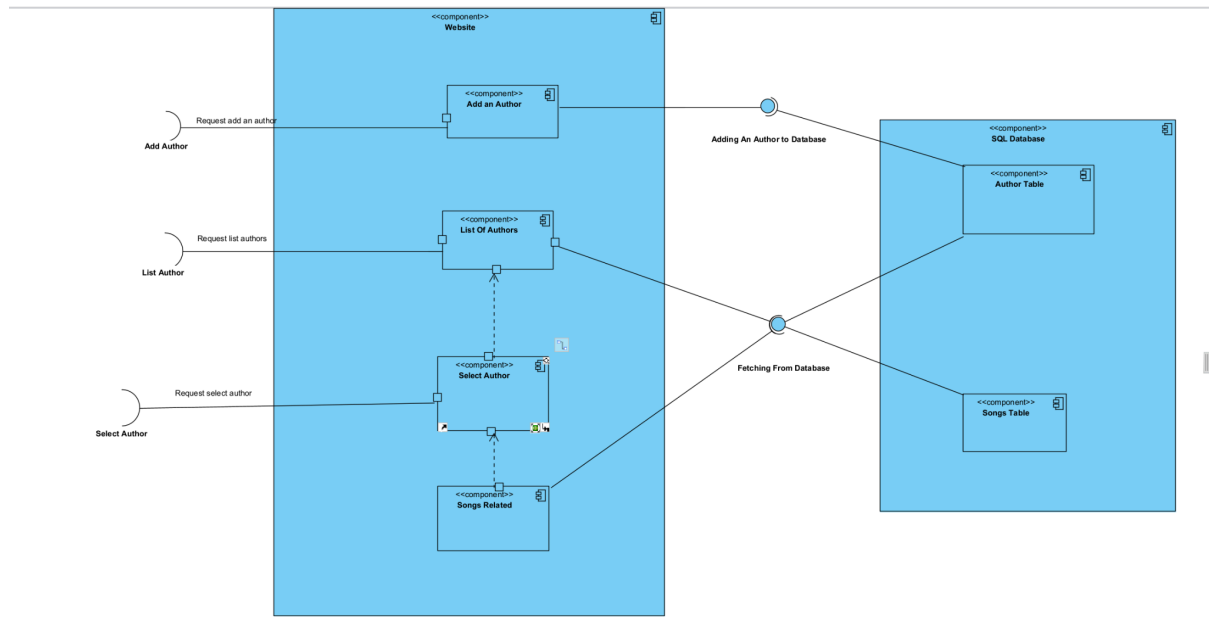
The Authentication API interacts with the Database Module, specifically the User Table, to perform operations such as inserting new user data during registration or fetching user records to validate login credentials. Connections like "Insert User" and "Fetch User" clearly represent these operations.

This diagram demonstrates a well-organized and modular authentication flow, highlighting how the frontend, backend, and database collaborate to handle user authentication securely and efficiently.



Component Diagram Author :

This component diagram illustrates how the system manages authors and songs. The frontend module includes several components: "Website," which acts as the main interface for users; "Add an Author," allowing users to add new authors to the database; "List of Authors," which fetches and displays all authors from the database; "Select Author," enabling users to choose a specific author to view or manage their details; and "Songs Related," which retrieves songs associated with a selected author. The backend module includes the "SQL Database" that stores all data, with specific components like "Author Table," which handles operations such as adding or fetching author details, and "Songs Table," which manages song data and retrieves songs related to authors. Interactions between the frontend and backend components are well-defined, ensuring smooth communication for actions like adding authors, fetching lists, and retrieving songs.



Class Diagram :

This class diagram represents the architecture of the playlist management system. Users own playlists, which contain songs. Songs are associated with authors and genres. The administrator can manage entities through the AdminPanel interface. Repositories and routes connect entities to the database, ensuring persistence and linking the backend to the frontend. Pages like SongsPage and AuthorsPage provide user interfaces to access and manipulate entities. It is showing all of the pages, repositories, classes and entities that makes the website working from server side to client side.

