Specification - Audio Streaming Platform

The purpose of this application is to provide users with a music-playing platform that allows them to listen to their favorite tracks seamlessly. All music content available in the app will be licensed under public domain, ensuring that users can enjoy the music without any copyright concerns. The application will support on-demand music playback, user-generated playlists, and basic media controls. It aims to offer a smooth and intuitive experience for users who want to access, organize, and play public domain music effortlessly.

Roles

The application distinguishes three roles:

User Role

The User role represents anyone who has created an account and is signed in. Guest users are those who haven't signed in, and they will have limited functionality until they sign up and log in.

- **Guest Users (Not Signed In)**: Users who are not signed in are considered guest users. They can still browse music, play tracks, and control playback (e.g., play, pause, skip, volume). However, they cannot create or save playlists, upload music, or access account settings. Their actions are temporary, and they don’t have an account to save any data.
- **Signed-in Users**: Once a guest user signs up and logs in, they become a signed-in user and gain access to additional features, such as saving playlists, uploading music, and managing their account preferences.

Admin Role

The Admin will manage the app’s content, monitor user activity, and ensure the proper functioning of features.

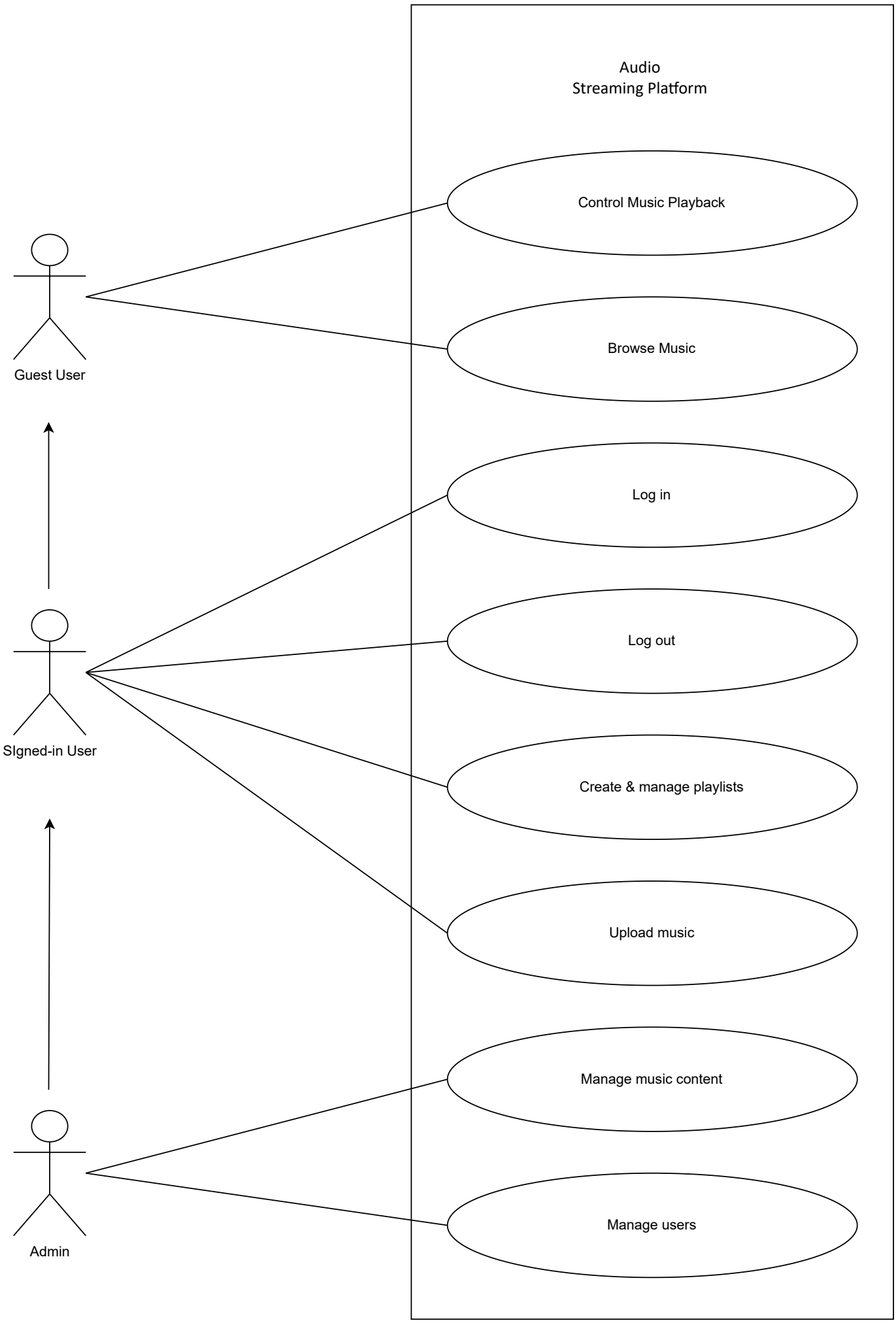
Use Case Diagram

Notes:

- **Control Music Playback**: allows users to interact with the media player by performing actions such as playing, pausing, skipping tracks, adjusting volume, and modifying playback preferences.

1. **Play Music** – Starts playing the selected track. If no track is selected, the first available track in the list will play.
2. **Pause Music** – Pauses the currently playing track. Playback can be resumed from the same position.
3. **Skip Track** – Skips to the next track in the queue or playlist.
4. **Previous Track** – Goes back to the previous track (if applicable).
5. **Seek Within a Track** – Allows the user to jump to a specific timestamp within the currently playing track.
6. **Adjust Volume** – Increases or decreases the volume level.
7. **Mute/Unmute** – Temporarily mutes or restores sound.
8. **Shuffle Mode** – Enables or disables random track playback.
9. **Repeat Mode** – Sets repeat options (repeat current track, repeat playlist, or no repeat).

- **Manage music content (add/remove tracks.**
- **Manage users (suspend/ban users).**


Audio
Streaming Platform
Control Music Playback
Browse Music
Log in
Log out
Create & manage playlists
Upload music
Manage music content
Manage users
Guest User
SIgned-in User
Admin

Data Model

The following conceptual data model contains the entities, attributes and relations.

Entities and attributes

Account

- An account used for authentication to the application. The login is unique.
- Attributes:
    - password: Stored as a hash value
    - is_admin: True in case of admin account, false in case of non-admin account
- Constraints & rules:
    - Each account belongs to exactly one user.

User

- Each user must have a unique email address.
- Attributes:
    - releases: Represents collections of tracks uploaded by users. A release can be an album, single, or EP. Each release consists of one or more tracks and is associated with a user who uploaded it.
    - playlists: Users can create and manage custom playlists by adding different tracks. Playlists do not contain releases but instead reference individual tracks. Playlists are saved and linked to the user's account.
- Constraints & rules:
    - A user can upload multiple releases (albums, singles, or EPs).
    - A release can have multiple authors (collaborators who contributed to the release).
    - A user can create multiple playlists.
    - Each playlist has only one creator (playlists are personal).

Release

- Each release (album, single, or EP) has a unique identifier.
- Constraints & rules:
    - A playlist can contain multiple tracks.
    - A playlist can belong to multiple categories (e.g., "Rock", "Instrumental", "Electronic").
    - A playlist or release must have at least one category to ensure proper classification.

Playlist

- Each playlist has a unique identifier.
- Constraints & rules:
    - A playlist can contain multiple tracks.
    - A release can belong to multiple categories (e.g., "Rock", "Instrumental", "Electronic").
    - A playlist or release must have at least one category to ensure proper classification

Track

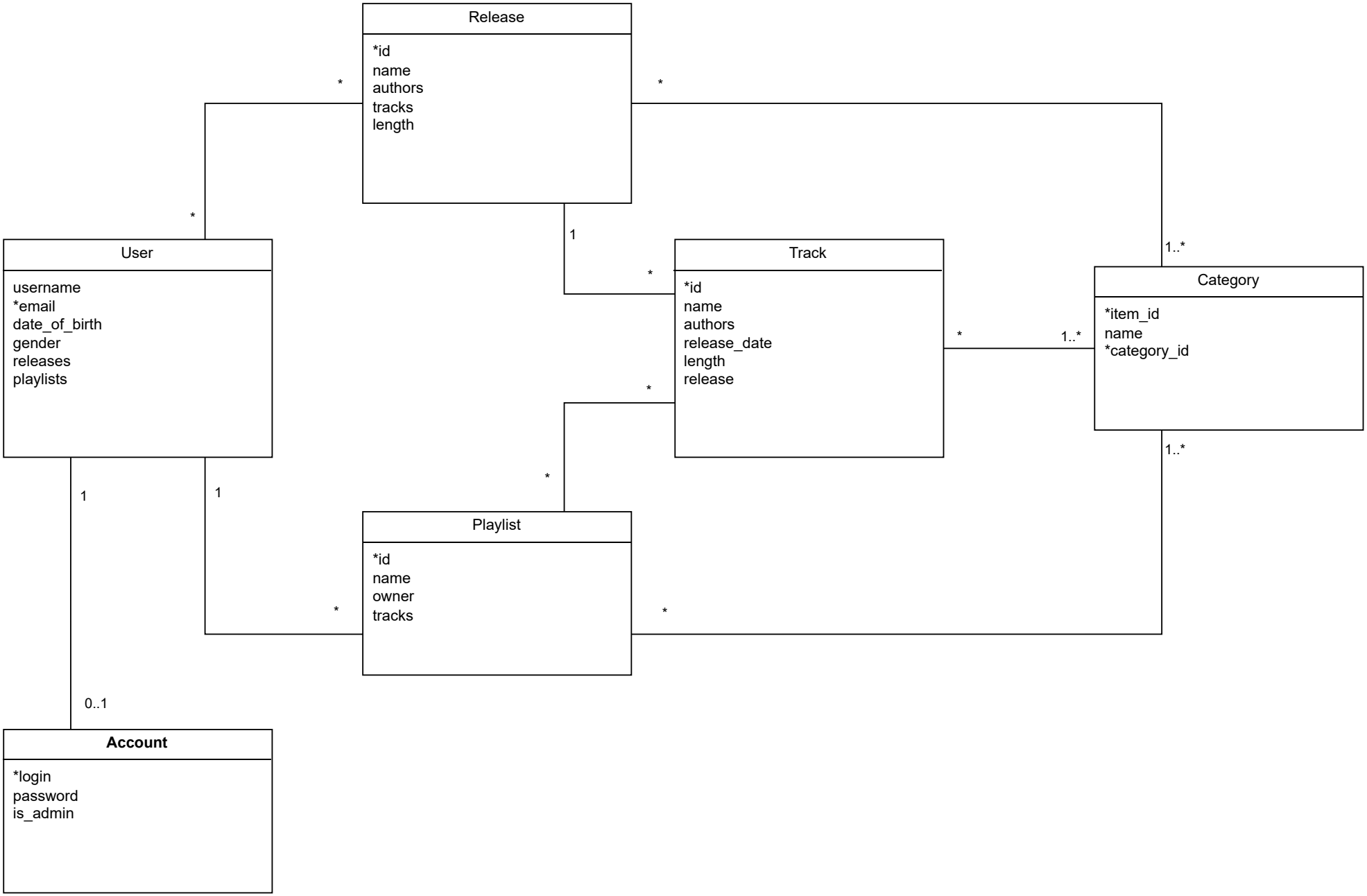
- Each playlist has a unique identifier.

- Constraints & rules:
    - Each track belongs to one release (album, single, or EP).
    - A release can contain multiple tracks.
    - A track to multiple categories (e.g., "Rock", "Instrumental", "Electronic").
    - A track must have at least one category to ensure proper classification

Category

- Each category has a unique identifier
- A category can be associated with different types of items:
    - Tracks
    - Playlists
    - Releases

Each categorized item must have its own unique item identifier, ensuring that every record in the category system is correctly linked.


Release
*id
name
authors
tracks
length
User
username
*email
date_of_birth
gender
releases
playlists
Track
*id
name
authors
release_date
length
release
Category
*item_id
name
*category_id
Playlist
*id
name
owner
tracks
Account
*login
password
is_admin
*
*
*
*
1
*
1..*
*
1..*
1
1
*
*
*
*
0..1

Architecture

- The application will be based on the client-server architecture and it will use the SPA (Single Page Application) approach.

Technological requirements

- Client-side: React 18, JavaScript, HTML5, CSS3
- Server-side: node.js 23, express.js 4.21.2, JavaScript
- Database: PostgreSQL 16
- Interface client - server: Rest API
- Hosting: render.com
- Supported browsers: Chrome, Firefox

Future work

- In future iterations of the application, we plan to introduce a shared playlists feature. This functionality will allow users to collaborate on playlists, making them more interactive and social.
- To improve user experience, we plan to implement an intelligent music recommendation system that suggests tracks based on user preferences and listening history.
- In future versions of the application, we plan to introduce a **rating system** that allows users to rate songs and releases (albums, singles, etc.), providing valuable feedback for other users and enhancing the overall user experience. This feature will also inform the recommendation algorithm, improving the quality of music suggestions.

Time schedule

Week 4

- Info k špecifikácii

Week 5

- Setup vývojového prostredia, Mock frontend

Week 6

- API, backend, DB

Week 7

- Autentifikácia

Week 8

- Deployment

Week 9

- Complete missing parts from beta version

Week 10

- Complete missing parts from beta version

Week 11

- Complete missing parts from beta version