
Final Design Document

for

Music Streaming System

Version 1.0 approved

Prepared by Group 5

2324II INT2208E 23, VNU-UET

April 9, 2024

Revision History

Date	Version	Description	Author
May 7th 2024	0.1	First draft with section titles	Nguyễn Đức Khánh
May 14th 2024	1.0	Complete detailed with subsections	Trần Thế Mạnh Ngô Lê Hoàng

Table of Contents

1. Introduction	1
1.1. Purpose	1
1.2. Scope	1
2. System Architecture	1
2.1. Component diagram (overview)	1
2.2. Component diagram (Detail)	2
3. User Interface	3
3.1. Create an account	3
3.2. Login	4
3.3. Search	4
3.4. Play track	4
3.5. Create playlist	5
3.6. Manage playlist	5
3.7. Download track	6
4. Database	6
5. Others	7
5.1. Sequence Diagrams	7
5.1.1. Create an account	7
5.1.2. Log in	7
5.1.3. Play the track	8
5.1.4. Search the tracks, artists and genres	8
5.1.5. Create playlists	9
5.1.6. Manage playlists	10
5.1.7. Upload the track	11
5.1.8. Manage track	12
5.1.9. Create album	13
5.1.10. Manage album	14
5.2. Class Diagrams	15
5.2.1. Sign up	15
5.2.2. Log in	16
5.2.3. Play the tracks	17
5.2.4. Search tracks, artists and genres	18
5.2.5. Download tracks	19
5.2.6. Create playlists	20
5.2.7. Upload tracks	21
5.2.8. Create album	22
5.2.9. Upgrade account	23

u

1. Introduction

1.1. Purpose

The purpose of this document is to present the final design for a comprehensive Music Streaming System. This system is envisioned as a robust, user-friendly platform that will enable music enthusiasts to stream their favorite tracks, discover new music, and curate personalized playlists. The document aims to provide a detailed revision for the development team, outlining the system's architecture, components, and their interactions. It will serve as a blueprint during the development phase and a reference point for future system enhancements and maintenance.

1.2. Scope

The Music Streaming System is a web-based application that will allow users to listen to music from a vast library of tracks spanning various genres, moods, and eras. Users will be able to search for songs, albums, or artists, create and manage their own playlists, and explore music based on their listening history and preferences. The system will also include features for user authentication, music recommendation, and social interaction among users.

2. System Architecture

2.1. Component diagram (overview)

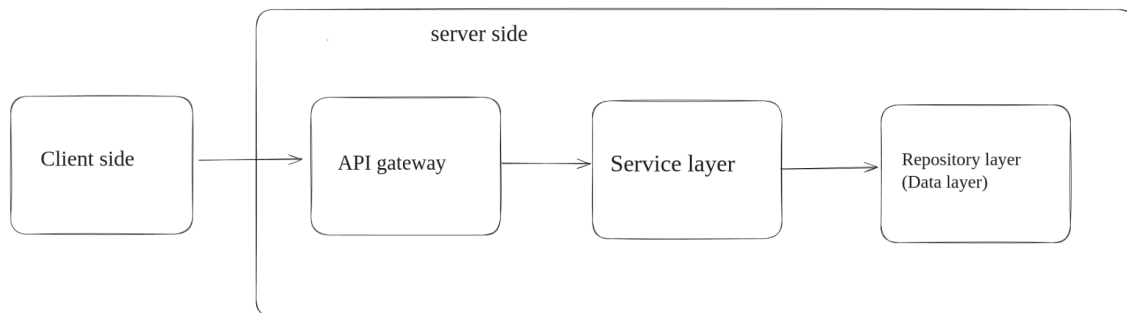


Figure 1: Overview component diagram

A client-server architecture made up of clients, servers, and resources, with requests managed through REST API

The client sends requests with methods like GET/POST/PUT/DELETE, etc., to the server-side, and the server-side responds with JSON format. On the client side, you can reference the Swagger API document and directly interact with it for testing.

Client-side manages UI rendering, user interactions, and executes scripts like JavaScript for dynamic behaviors. Data is fetched from the server side.

The API gateway is responsible for routing requests to the correct service for processing, then aggregating and organizing the results (including exceptions and errors) returned from the service, and sending the response back to the client side. Additionally, the API gateway manages rate limiting and authentication.

Service layer handles the business logic for the application:

- Data validation, such as checking if the requested song ID is valid or if the size of the uploaded track file exceeds 50MB

- Process handling, for example, when adding a new song: first, checking if the album and artist of the track exist, then verifying the size of the uploaded music file, followed by storing the information in the database and saving the mp3 file to static storage.

- Interaction with other system components or external APIs, such as sending requests to the data layer to store track information.

Repository layer (Data layer) is responsible for directly interacting with the database to manage data for the application. Unlike the service layer, which only calls abstract APIs, the repository layer implements details such as creating sessions, transactions, etc.

2.2. Component diagram (Detail)

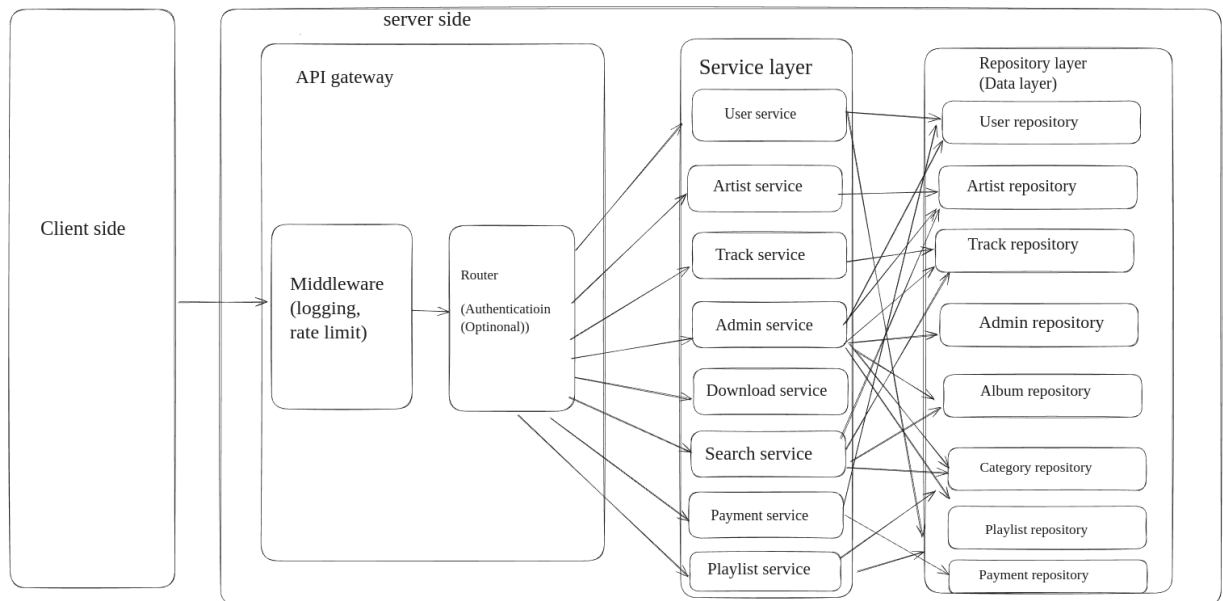


Figure 2: Detail component diagram

Functions like logging and rate limiting are embedded within middleware, meaning that these middleware can be applied before, during, or after processing each request sent.

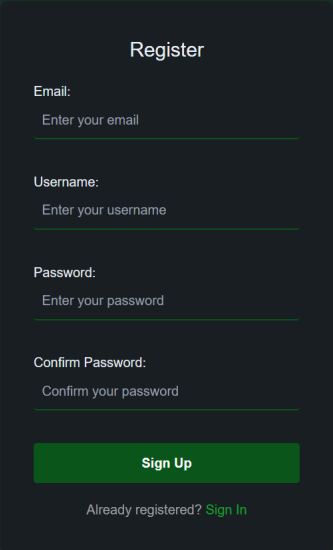
Authentication will be applied to critical APIs to ensure the integrity and security of the application's data. Authentication in the application is implemented using JWT tokens with a refresh mechanism.

The router is responsible for directing requests to the corresponding service and aggregating responses to send to the client side.

Each service can utilize multiple repositories from the repository layer to handle requests. This approach increases the logical coherence of the application and enhances the separation of concerns. (For example, the user service can perform functions related to creating an account, deleting an account, or editing account information by calling APIs from the user repository. Additionally, the user service can also call APIs from the playlist repository to perform functions like creating a playlist for that user's account)

3. User Interface

3.1. Create an account



The image shows a registration form titled "Register" centered on a dark background. The form contains four input fields with labels and placeholder text: "Email:" with "Enter your email", "Username:" with "Enter your username", "Password:" with "Enter your password", and "Confirm Password:" with "Confirm your password". Below these fields is a green "Sign Up" button. At the bottom of the form, there is a link that says "Already registered? Sign In".

Figure 23: Demo registration form

3.2. Login

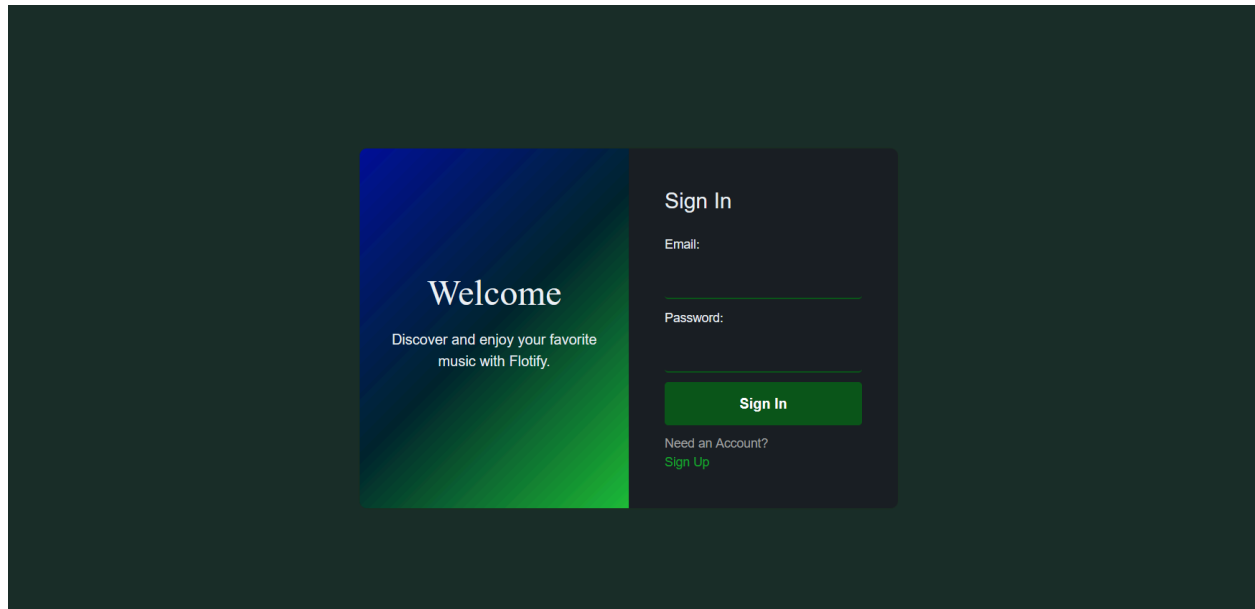


Figure 24: Demo login view

3.3. Search

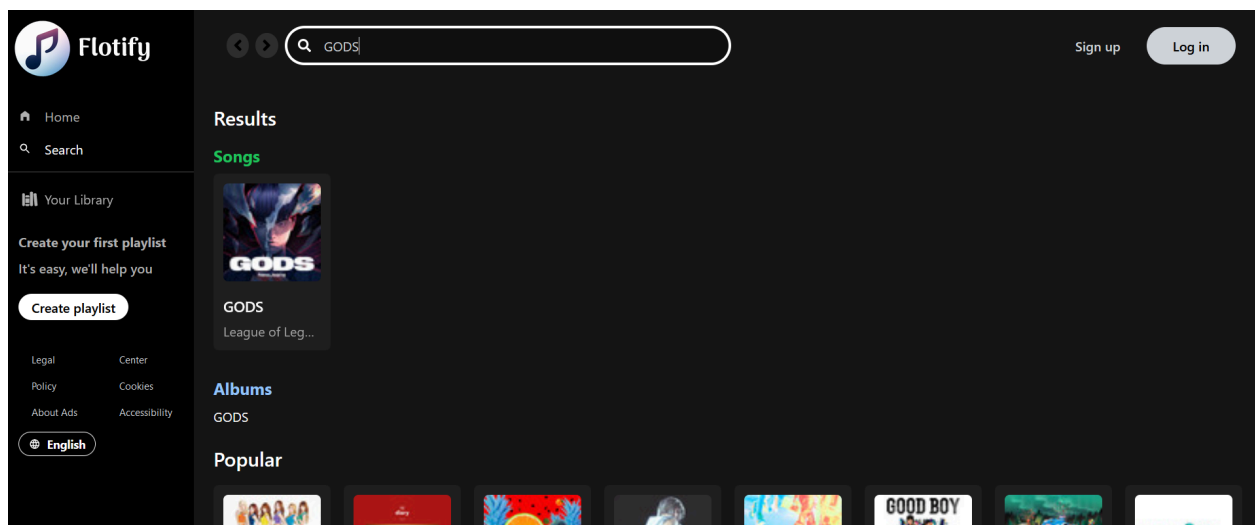


Figure 25: Demo search area

3.4. Play track

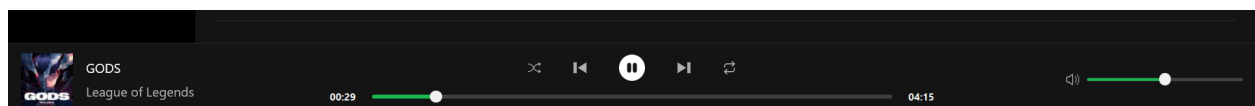


Figure 26: Demo track player

3.5. Create playlist

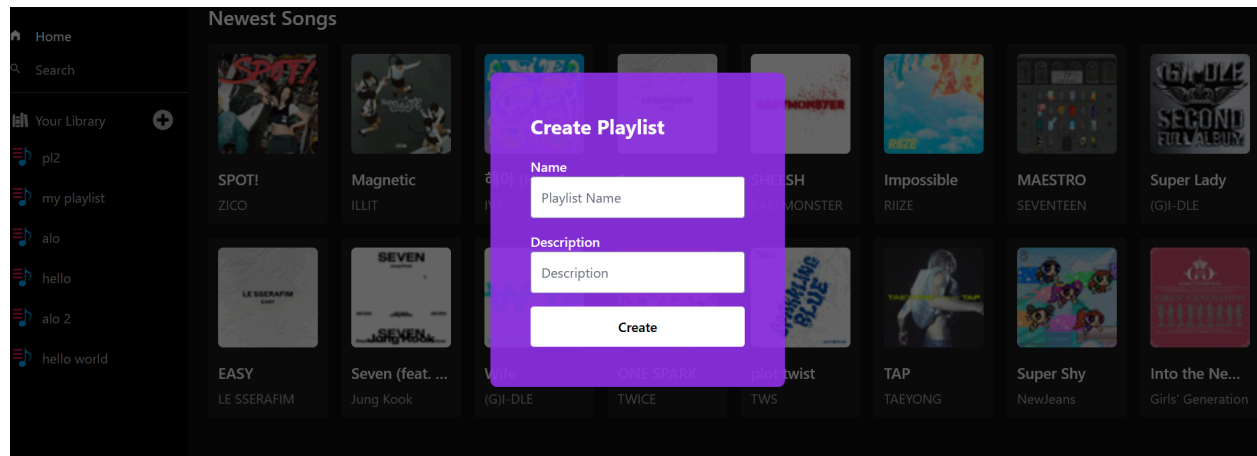


Figure 27: Demo create playlist

3.6. Manage playlist

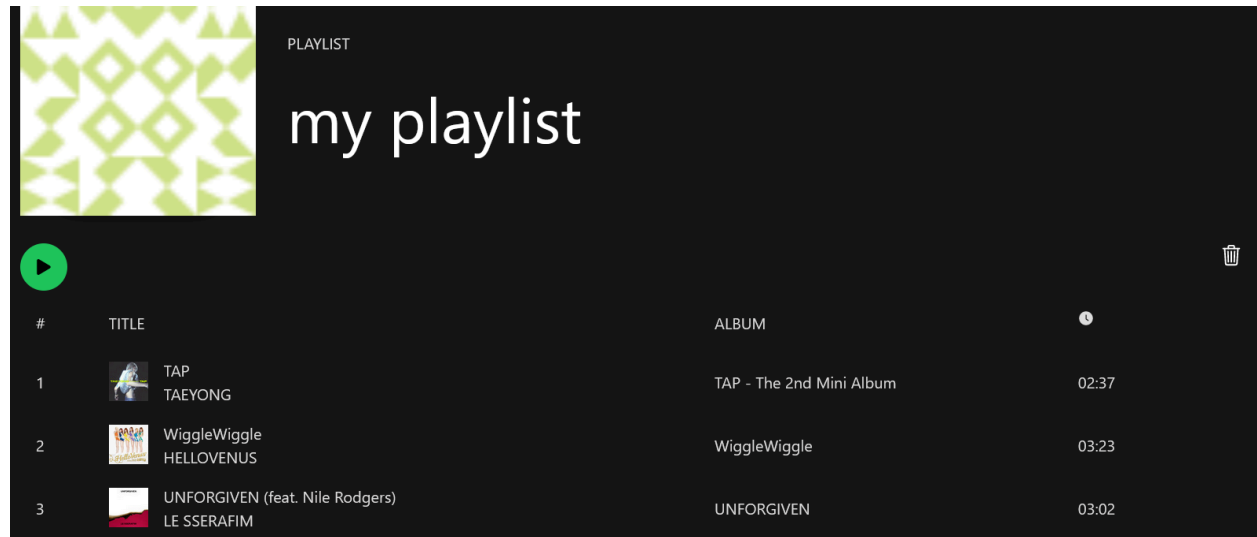


Figure 28: Demo playlist management

3.7. Download track

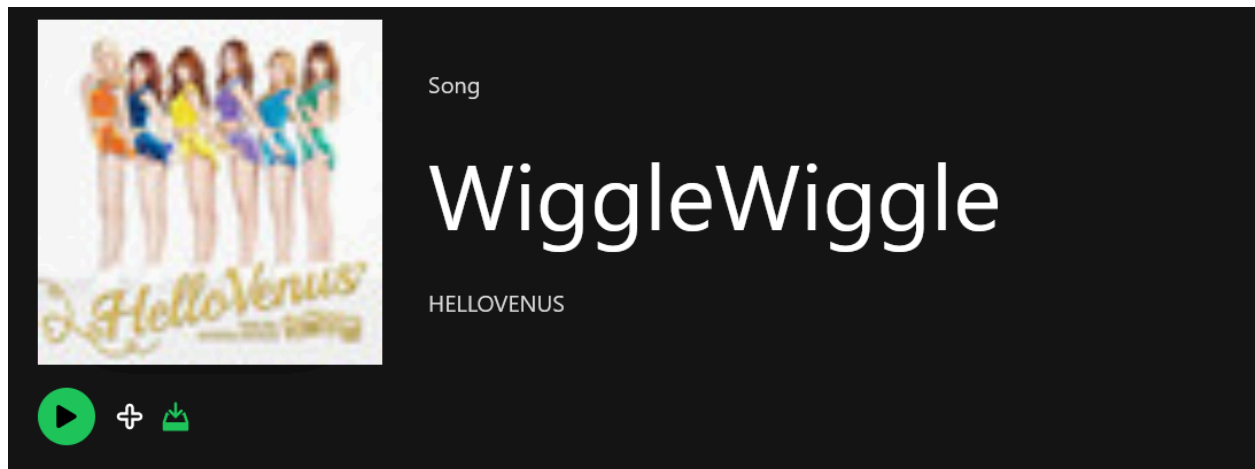
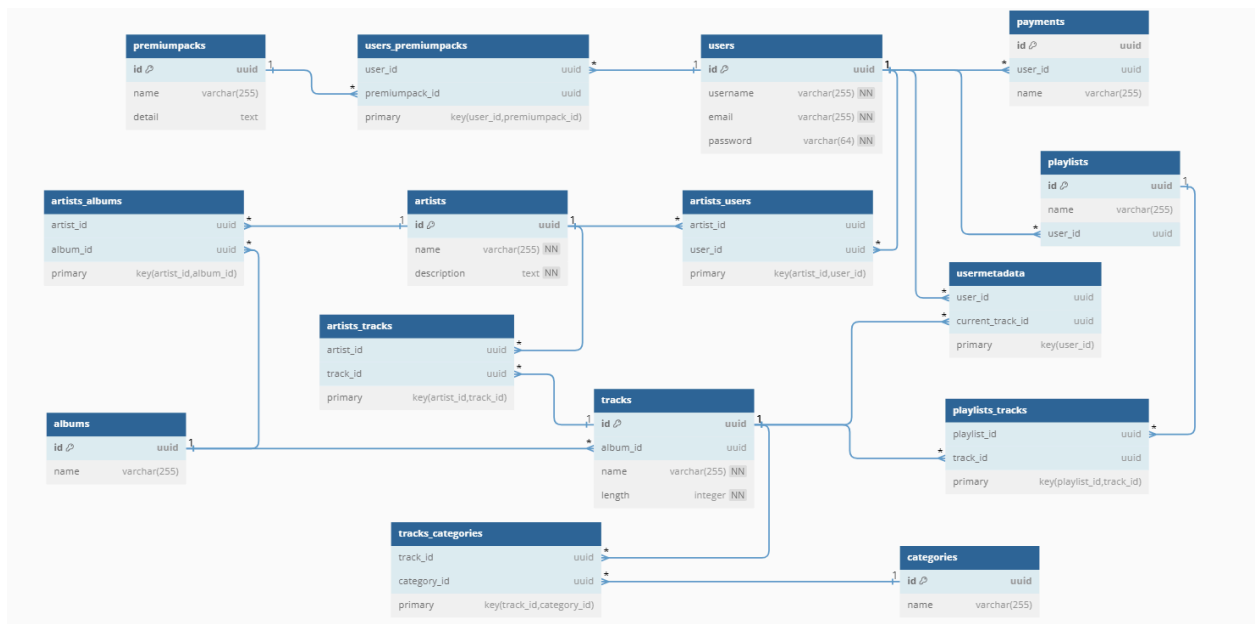


Figure 29: Demo download track

4. Database



5. Others

5.1. Sequence Diagrams

5.1.1. Create an account

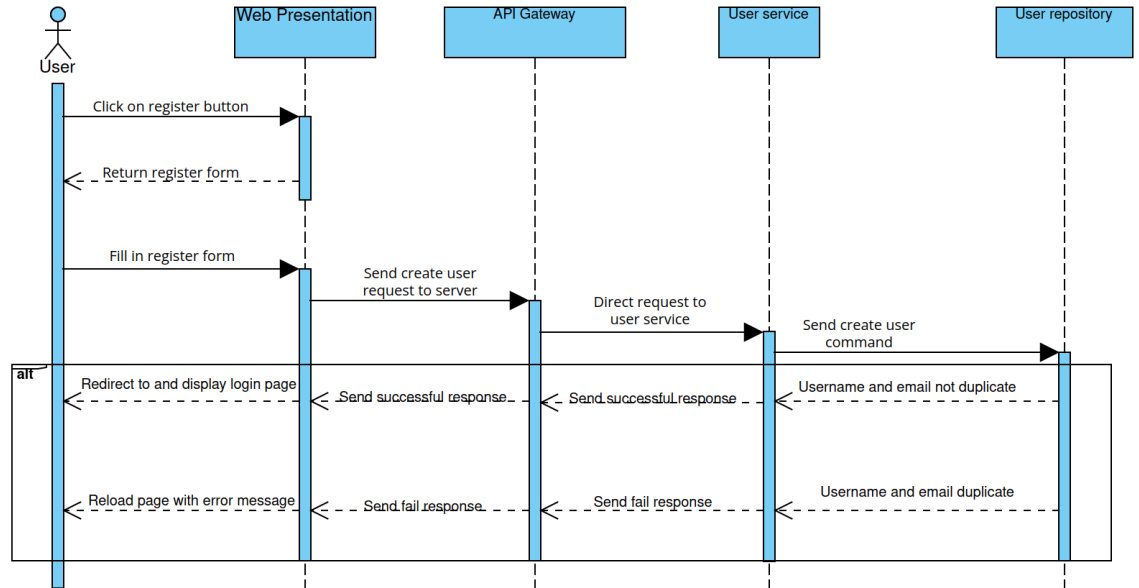


Figure 3: Account registration sequence diagram

5.1.2. Log in

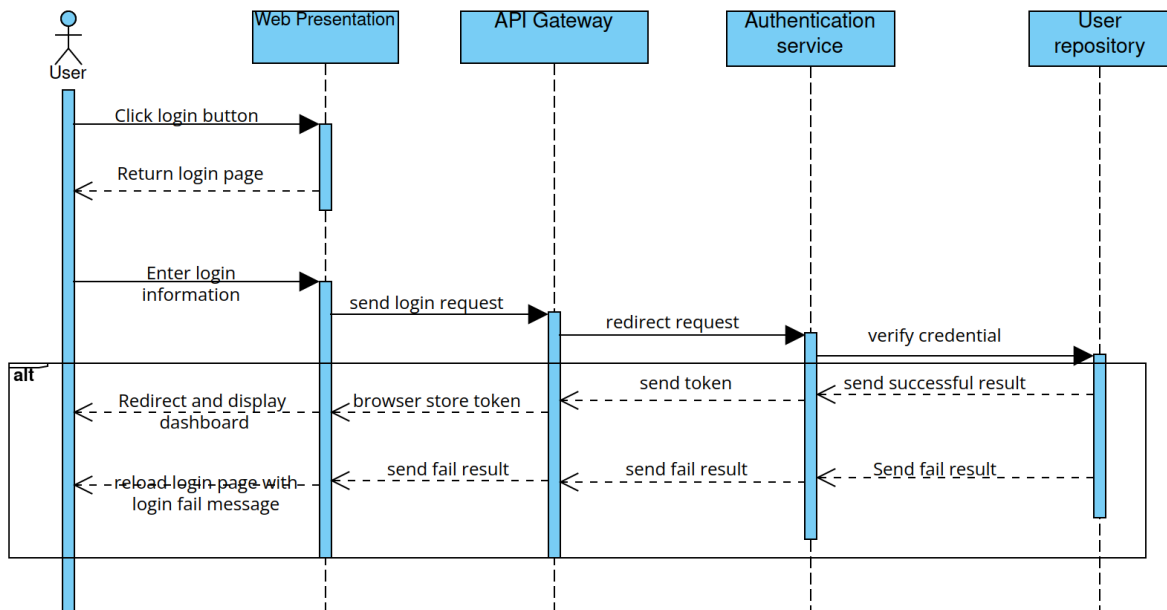


Figure 4: Login sequence diagram

5.1.3. Play the track

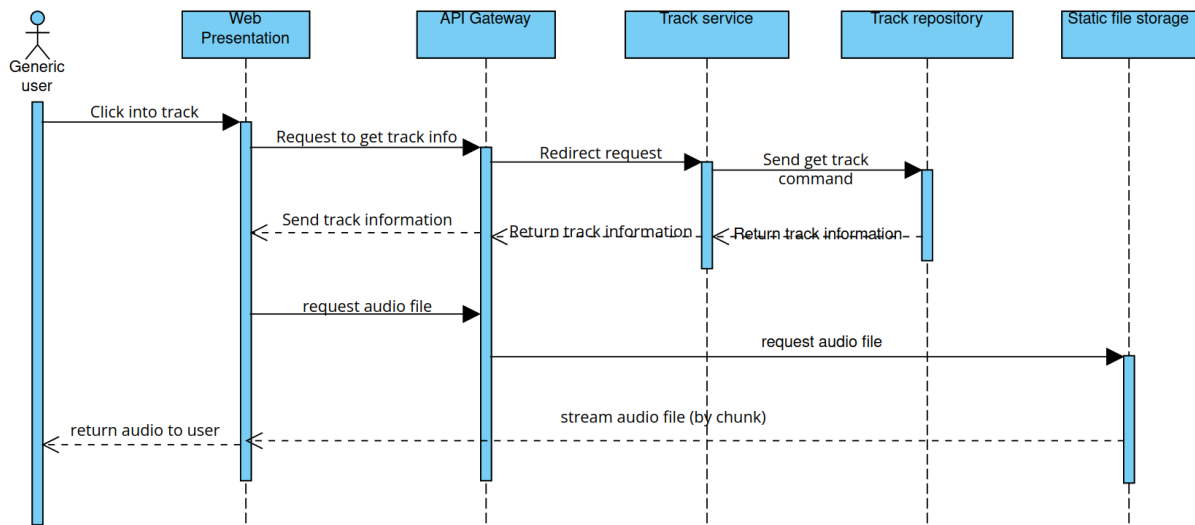


Figure 5: Playing Track sequence diagram

5.1.4. Search the tracks, artists and genres

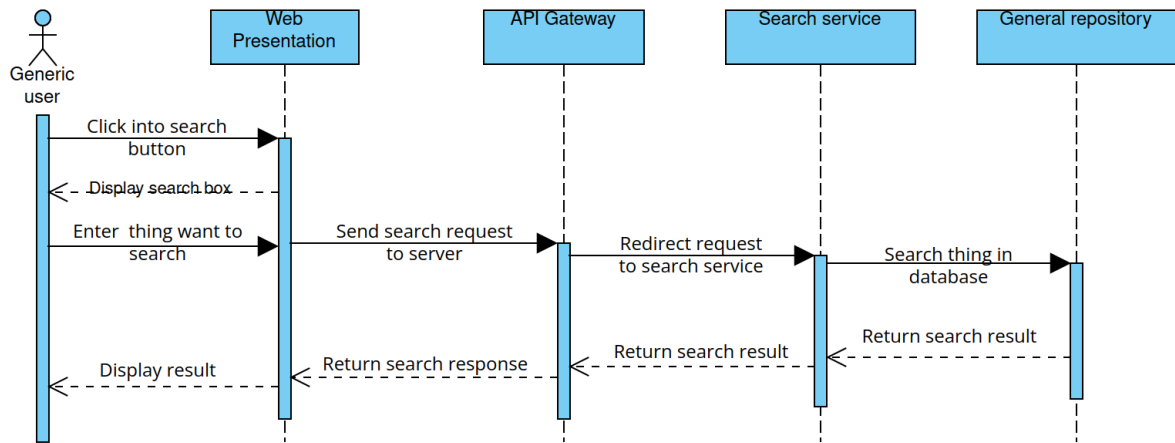


Figure 6: Search Engine sequence diagram

5.1.5. Create playlists

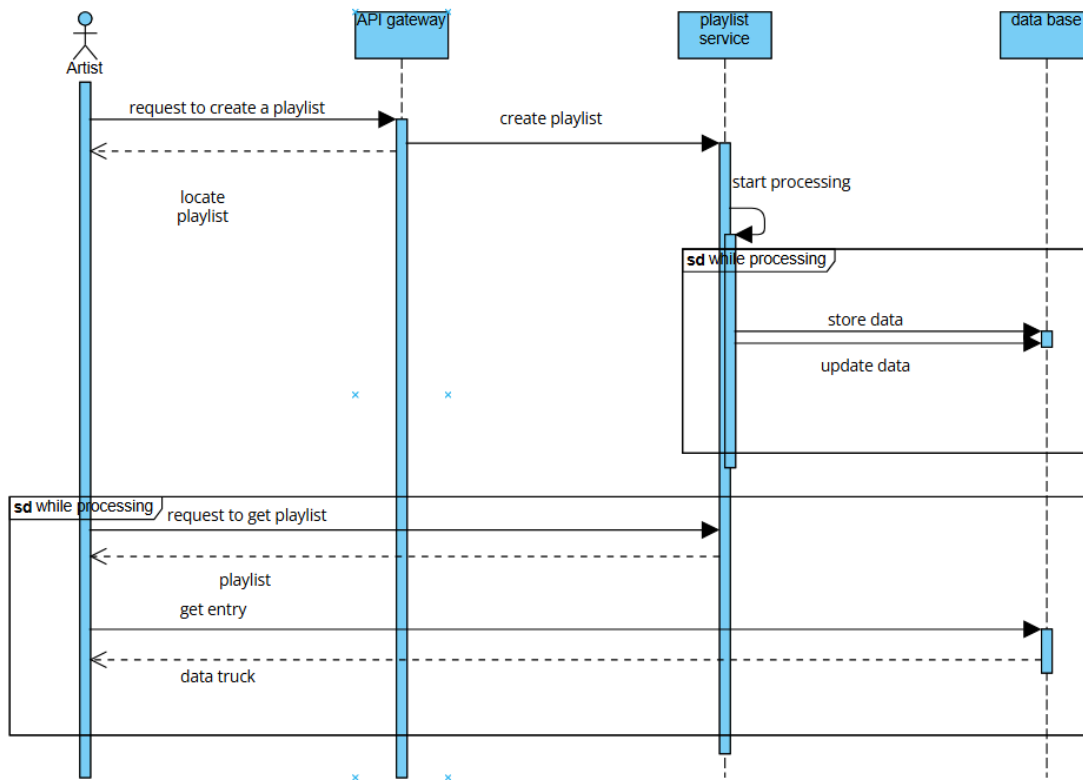


Figure 7: Playlist creation sequence diagram

5.1.6. Manage playlists

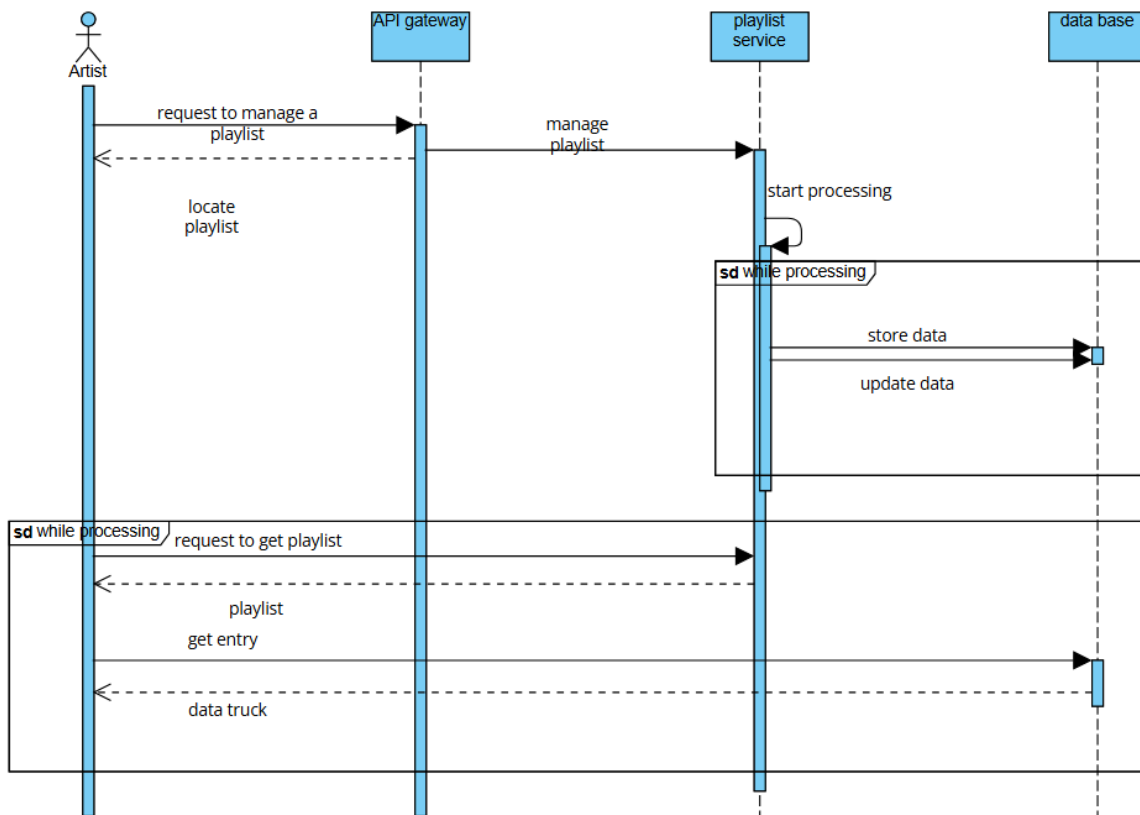


Figure 8: Playlist management sequence diagram

5.1.7. Upload the track

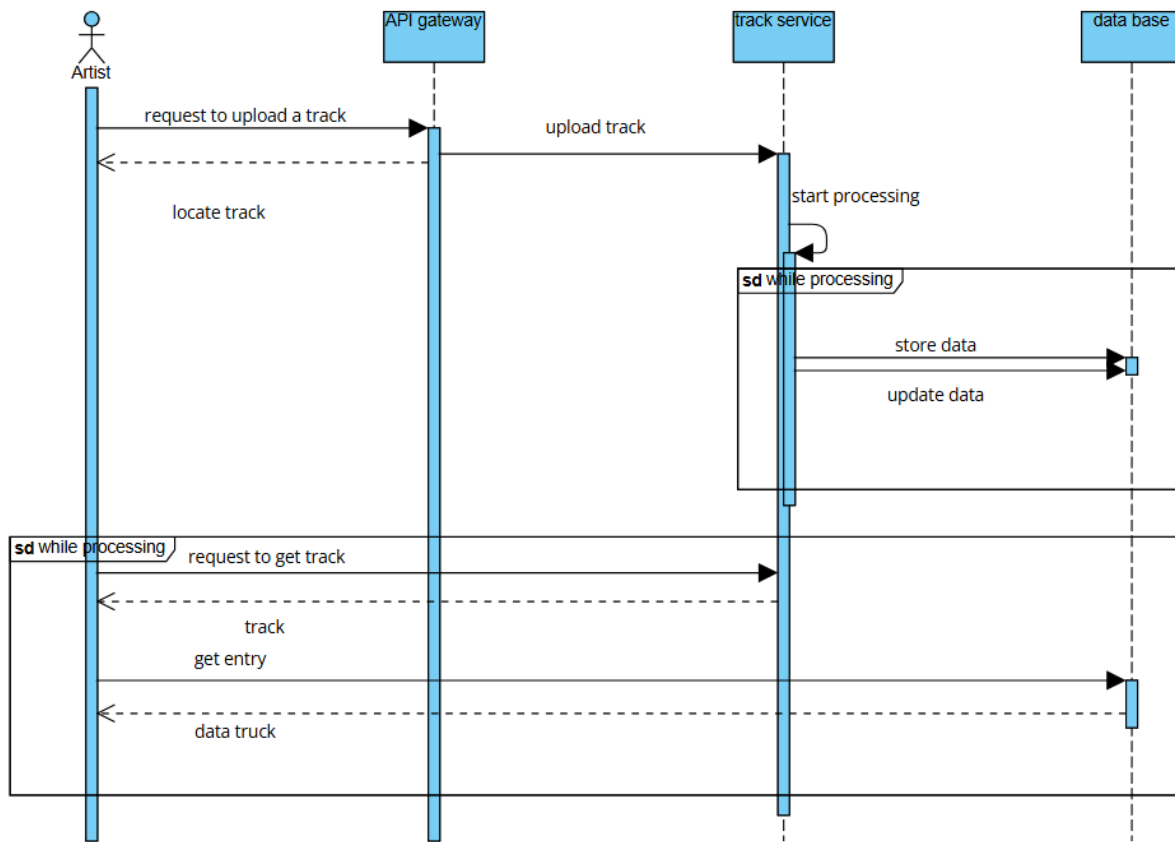


Figure 9: Track Upload sequence diagram

5.1.8. Manage track

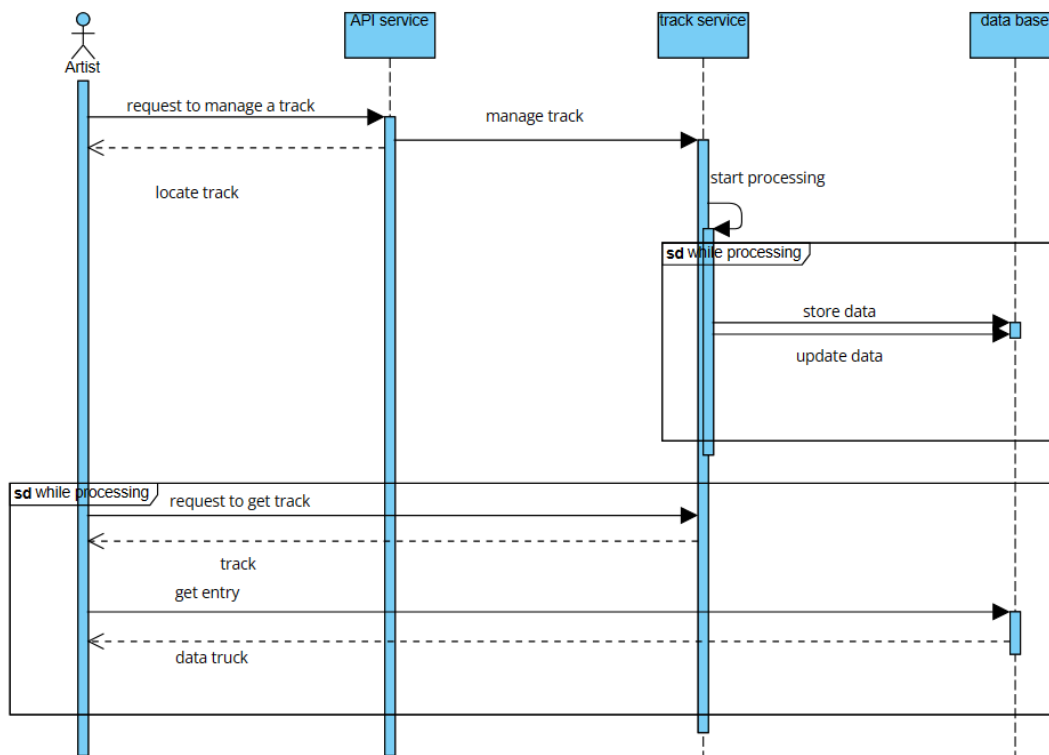


Figure 10: Track management sequence diagram

5.1.9. Create album

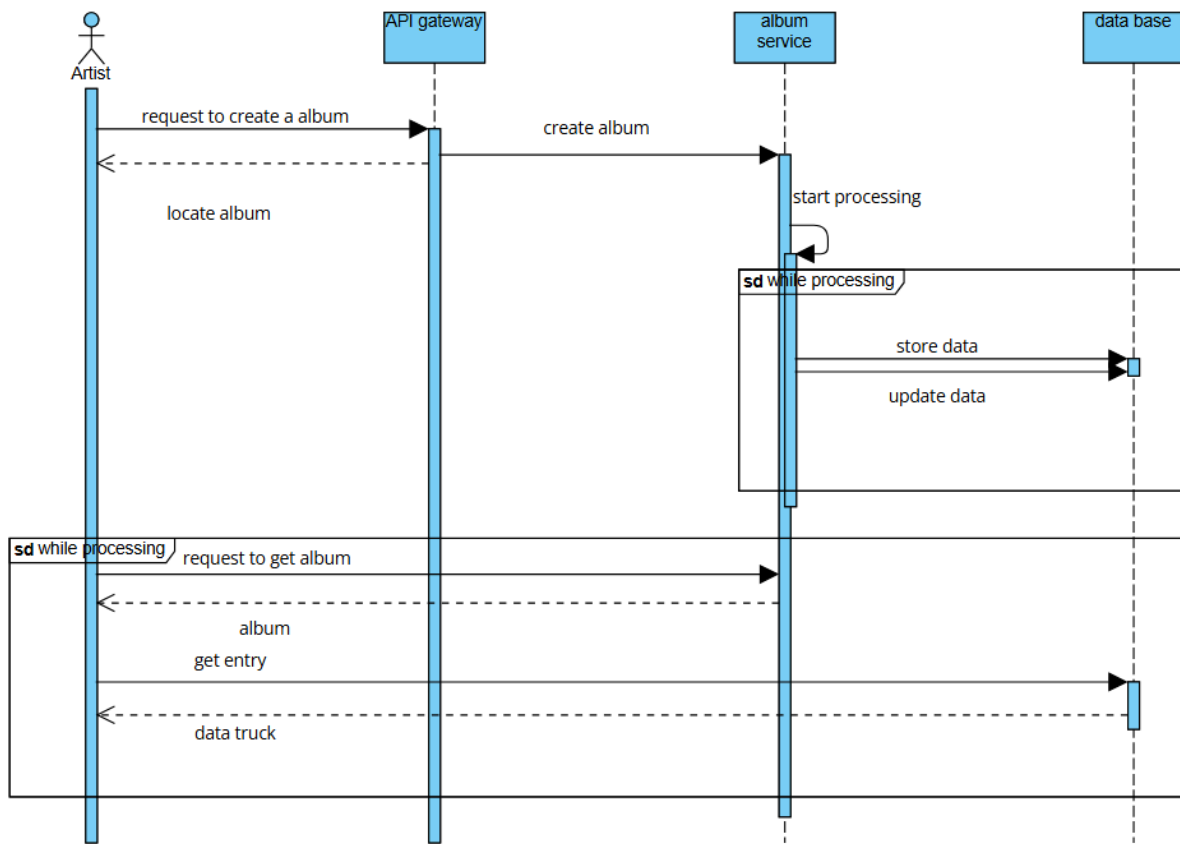


Figure 11: Album creation sequence diagram

5.1.10. Manage album

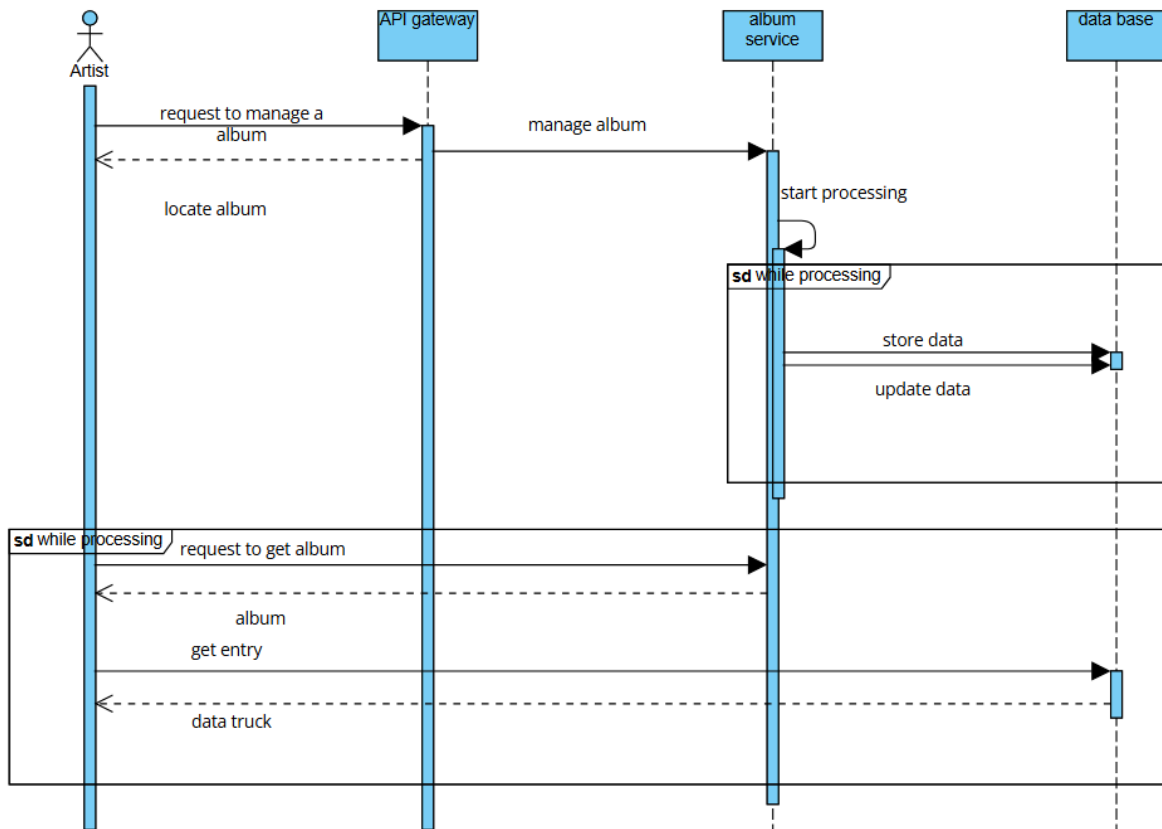


Figure 12: Album management sequence diagram

5.2. Class Diagrams

5.2.1. Sign up

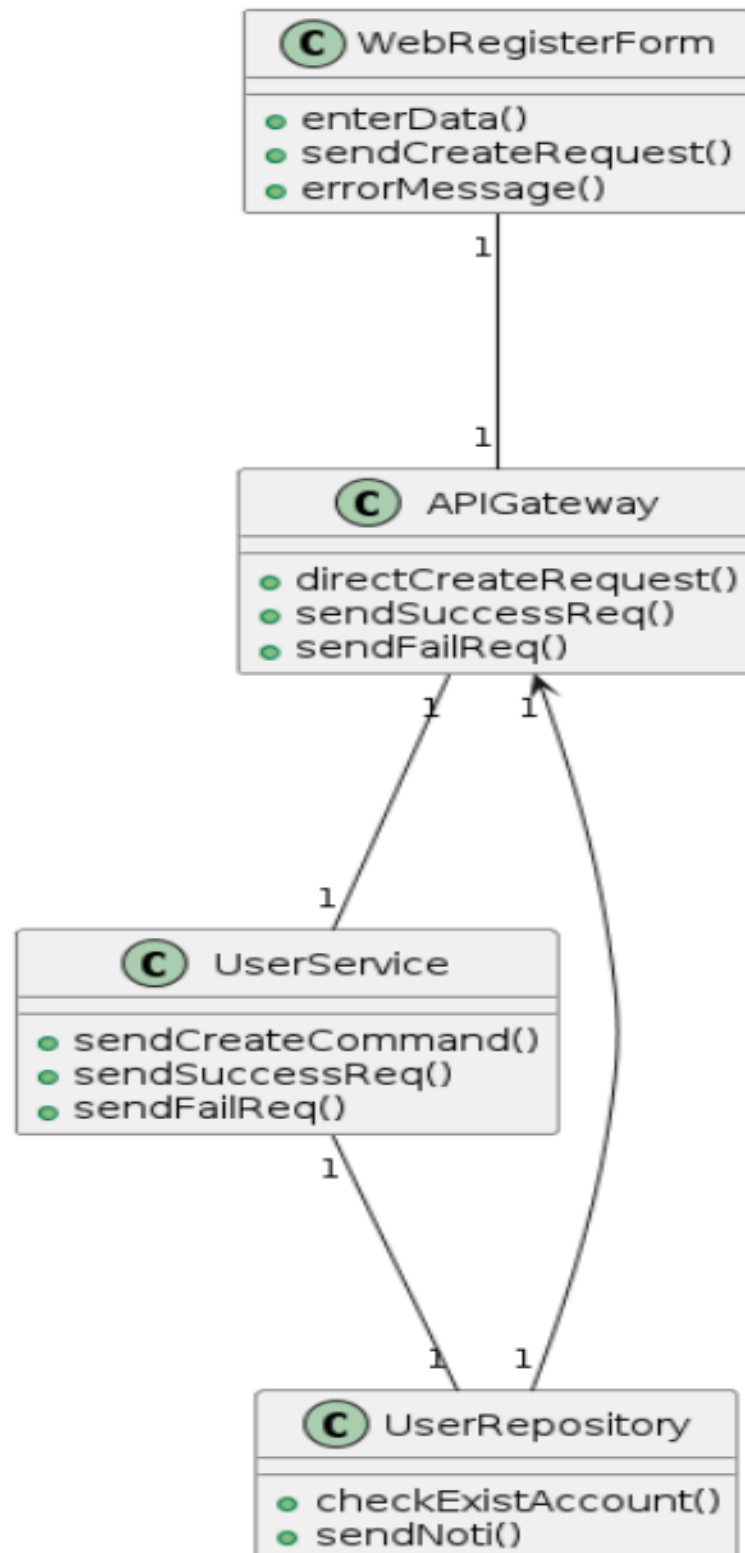


Figure 14: Account registration class diagram

5.2.2. Log in

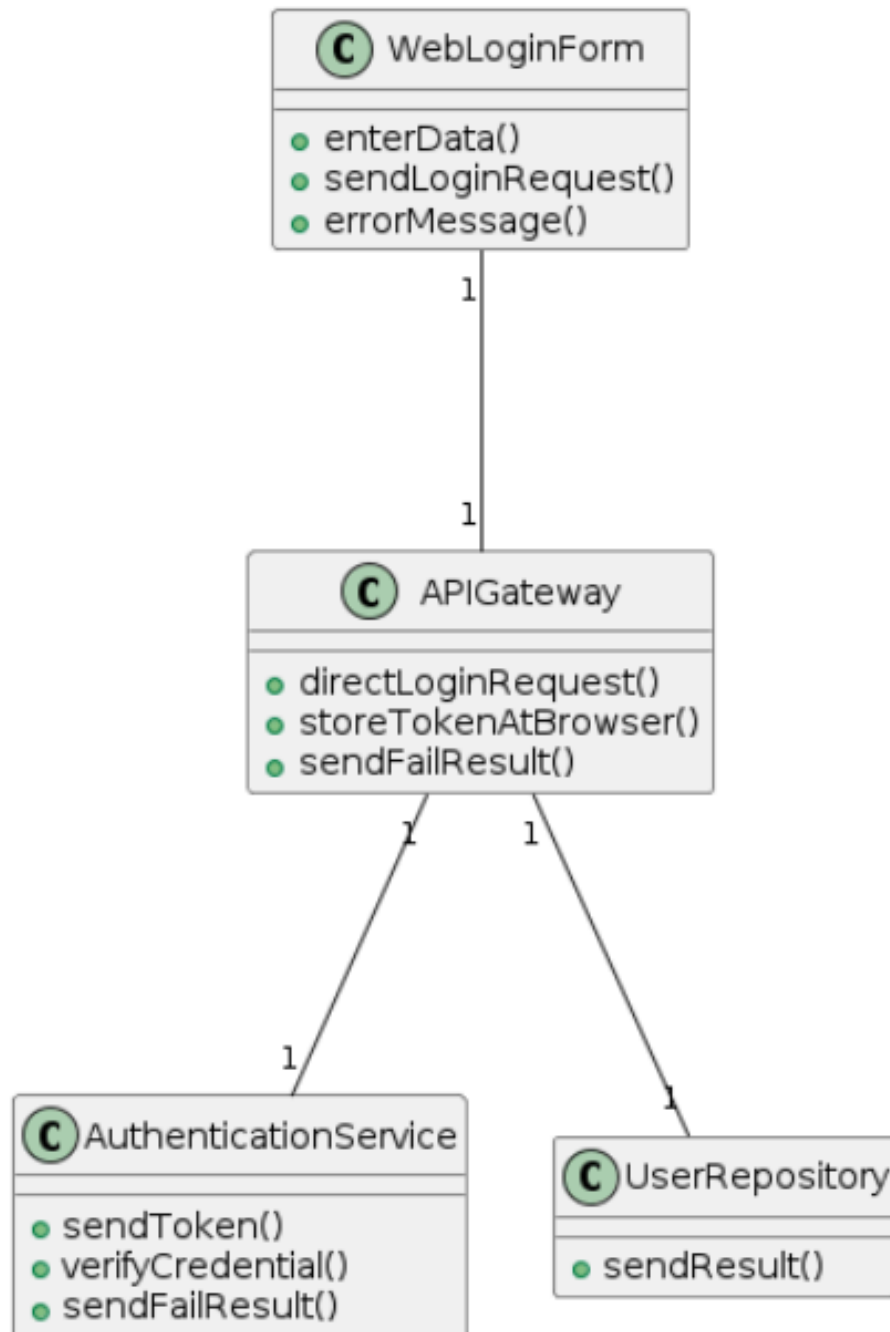


Figure 15: Login class diagram

5.2.3. Play the tracks

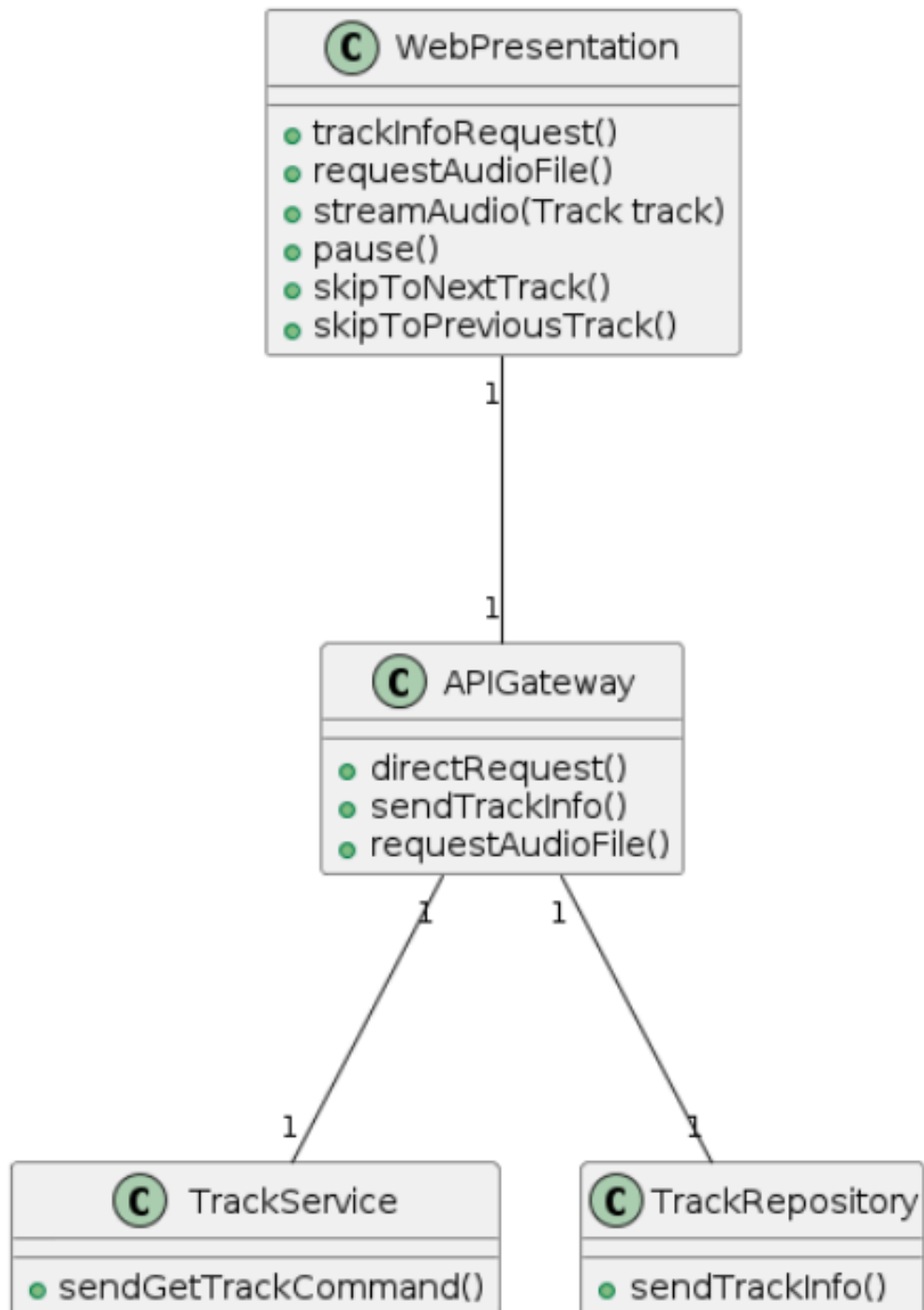


Figure16 : Playing Track class diagram

5.2.4. Search tracks, artists and genres

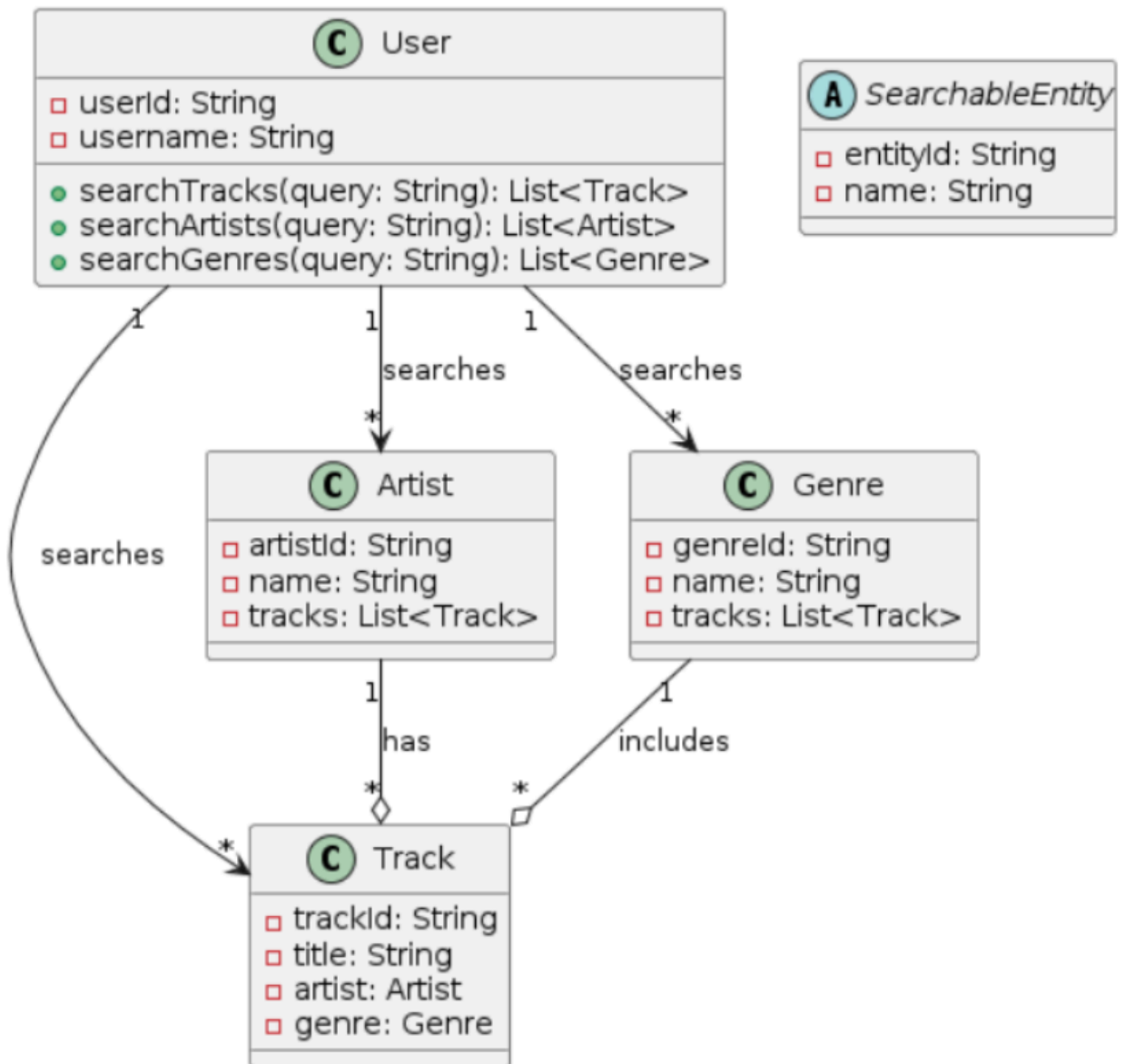


Figure 17: Search Engine class diagram

5.2.5. Download tracks

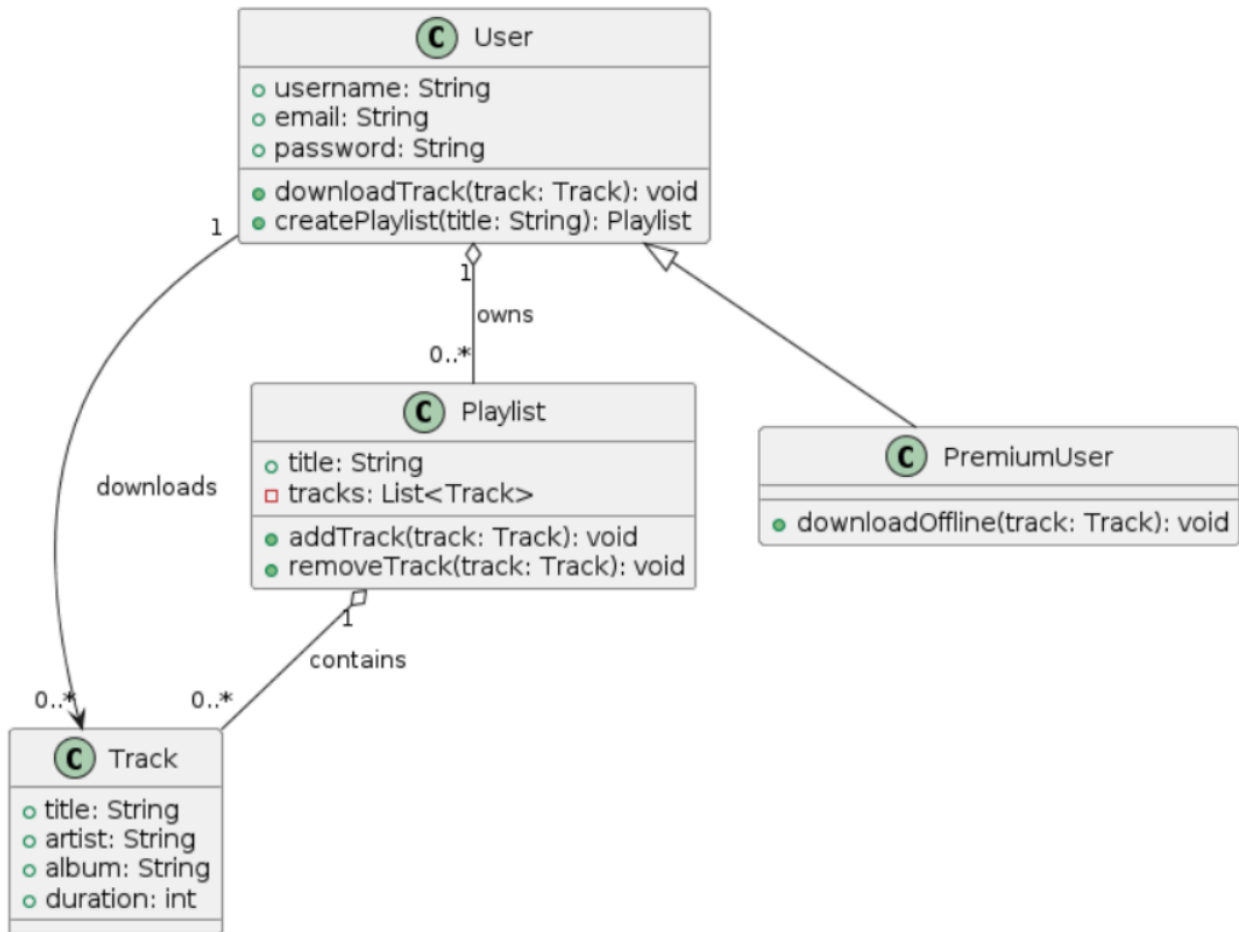


Figure 18: Track download class diagram

5.2.6. Create playlists

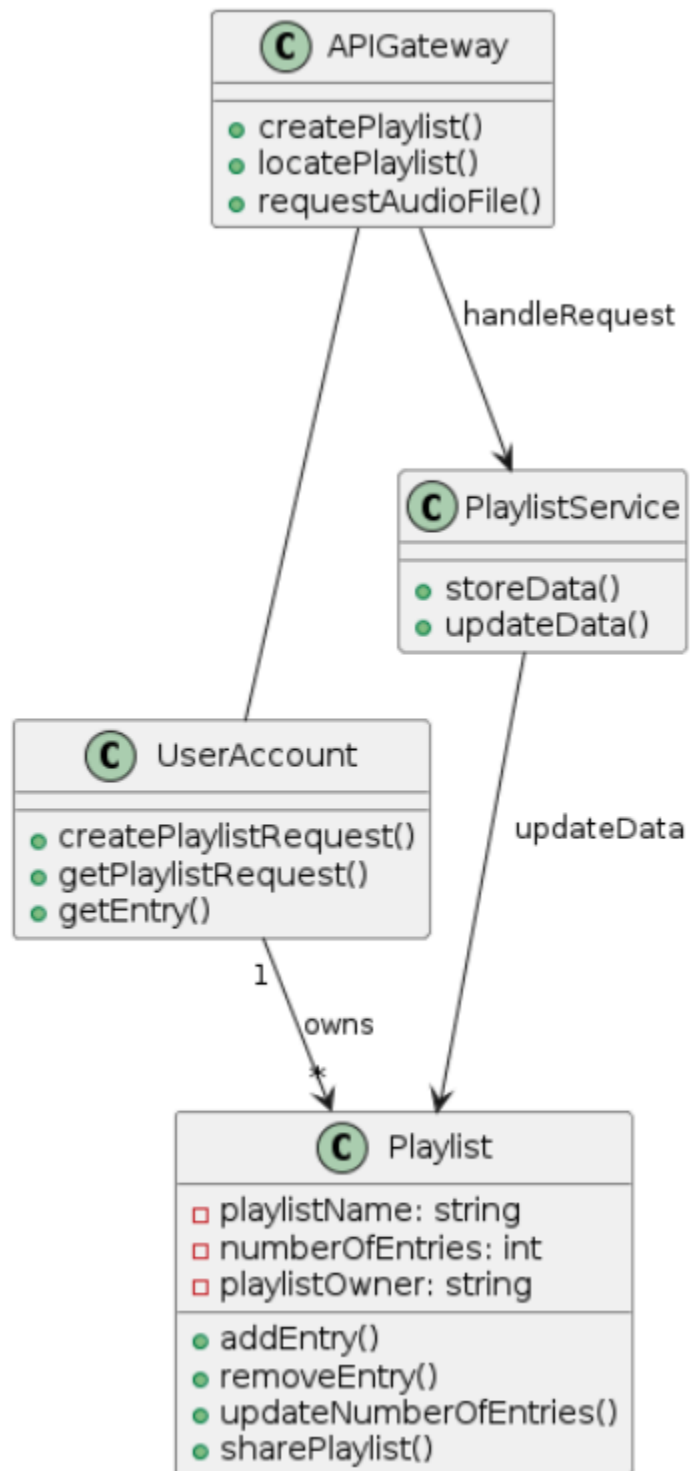


Figure 19: Playlist Creation class diagram

5.2.7. Upload tracks

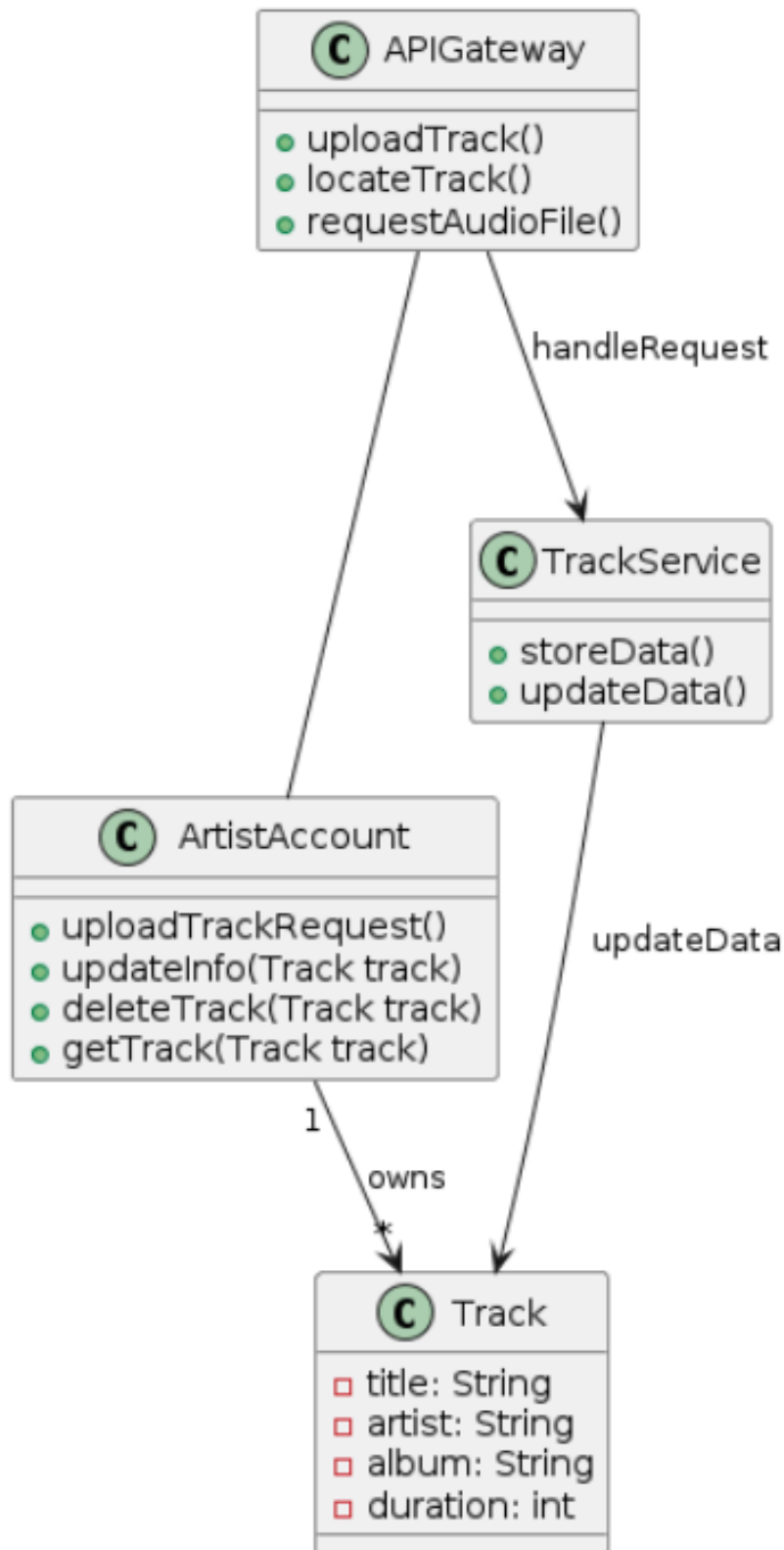


Figure 20: Track Upload class diagram

5.2.8. Create album

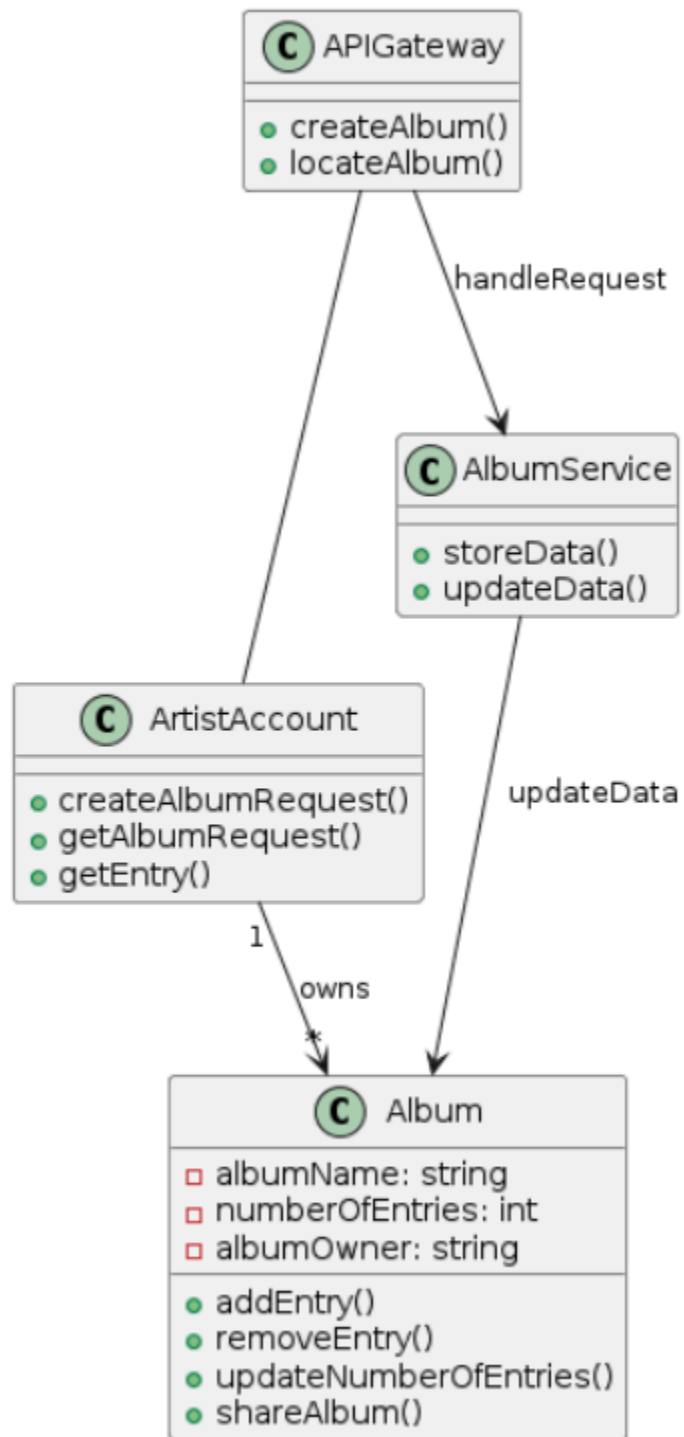


Figure 21: Album creation class diagram

5.2.9. Upgrade account

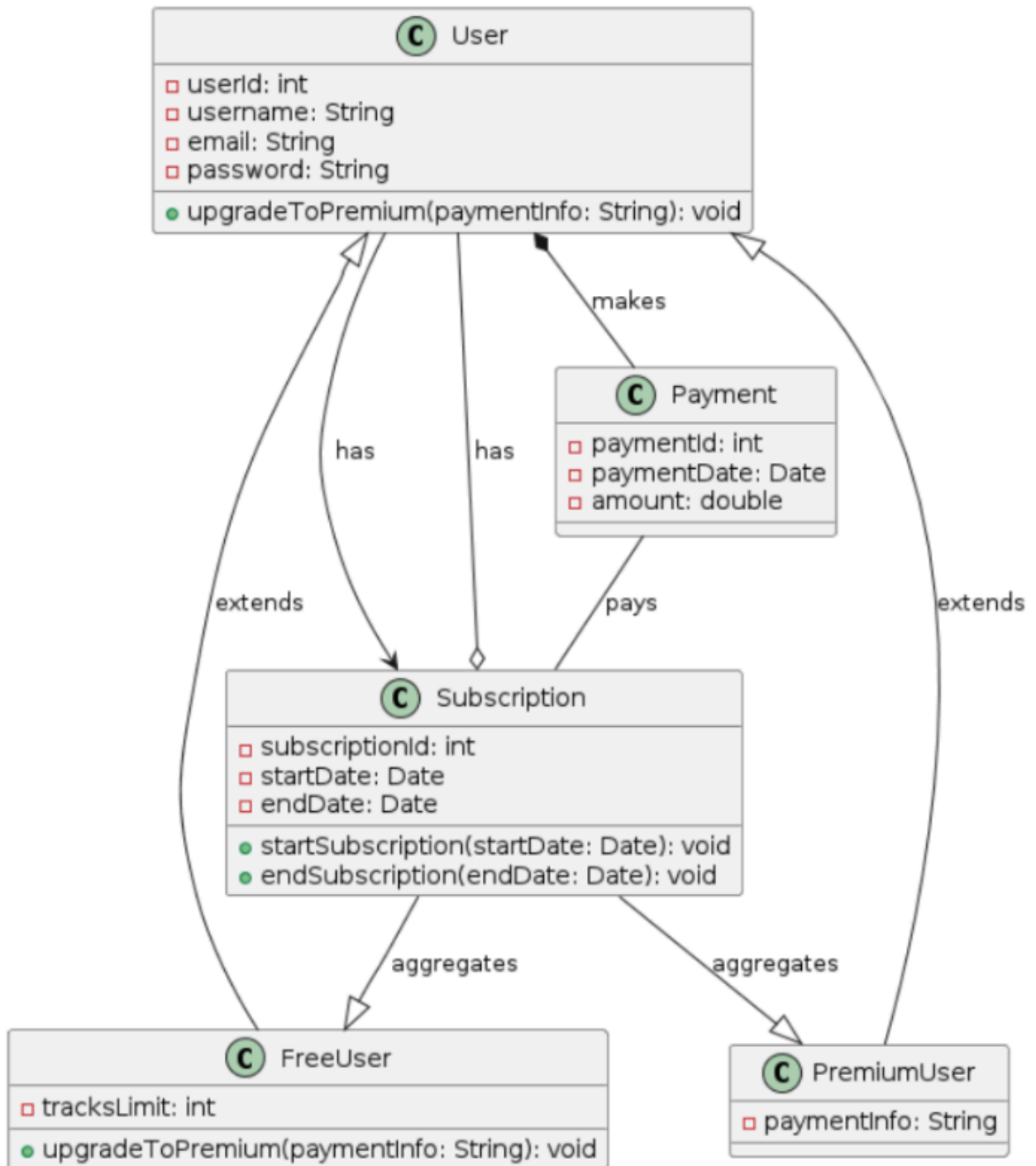


Figure 22: Account upgrade class diagram

