Information system designed for music discovery - Soundshift

(Project for Systems III)

Valentino Ivanovski

Definition of the problem

The problem Soundshift aims to address is the repetitive and monotonous nature of people's music listening. Many music enthusiasts and everyday listeners often find themselves stuck in a musical rut, looping through the same few songs and struggling to discover new and exciting music.

This issue affects a broad spectrum of listeners, including music enthusiasts seeking fresh and unique musical experiences, musicians looking for inspiration, or casual listeners who use popular streaming platforms during various activities like commuting, working out, studying, or relaxing.

Soundshift provides a user-friendly way for people to make their music experience more diverse. By simply pressing the "Shift" button, users can instantly access a randomly selected song, breaking the cycle of repetitive music listening.

The difference between Soundshift and the music recommendation algorithms of well-known streaming platforms lies in the fact that each Soundshift suggestion is a song cherished by someone, somewhere in the world. It might be their personal favourite or a tune holding a special meaning. Moreover, if you happen to enjoy a suggestion, you can explore the entire catalog of songs recommended by the user who shared that particular track, and even the songs they have liked or retrieved.

Functional Requirements

The system should enable the following functionalities:

1. Prompted to the main page of the information system, access of the users location is requested using the GeoLocation API, which will be stored in the users database. This location is displayed on each of the retrieved songs. The user can decline the location request. Next, the user can submit songs into the main songs database, by entering a song title, artist name, selecting a genre and marking the song as explicit or not. All of this information, including the user location will later on be displayed if someone retrieves the submitted song.

2. Users are able to retrieve random songs from the main music database. The retrieval can be filtered by genre and the song's explicit status. Once a song is retrieved, the user can see a screen of the song with the following details: song title, artist name, genre, explicit status, uploader name (linked profile), date of upload, user location. Under the song information, there will be a YouTube embedding of the song, followed by a like button, different links of the song that take the user to the specific platform they want to listen the song to (options are Spotify, SoundCloud and YouTube), and a comment section in which the user can comment, delete their own comments and report other user's comments. The retrieved song can also be reported.

3. Access to the system is restricted to registered users only. This functionality is crucial in order for users to have insights of the songs that they retrieve, submit or like. Furthermore, each registered account has it's password hashed, for added security. There are two types of users: normal users and administrators. Administrators have a wider range of functionalities compared to normal users,

as mentioned in the next functionalities. When registering, writing credentials that are already in the database will prompt an error. Trying to log in with incorrect credentials will also cause an error.

4. Users can view their own profiles, including song retrieval, submission history, and likes. Their profiles can be edited as such: adding a bio, linking their listening platform profiles and setting profile picture. Other user's profiles can also be viewed for furthermore music discovery. These profiles can be opened by clicking the "Submitted by" link after each submission or by searching a user in the search bar.

5. Searching for users and submitted songs is possible in the system by opening the search route. Here, the user can toggle between song search and user search, and results of either submitted songs or users will show up after entering a query. Clicking on the songs will take the user to the song's Spotify page, while clicking on the users will take you to the user's profile. Here, the user can check whether a certain song is in the database or not.

6. Administrators have extended functionalities alongside the ones from the basic user. They have four new routes, dedicated to filtering, analysing and deleting entries. These routes are the following: Users List, Songs List, Comments List and Reports List. Administrators can grant admin access to normal users from the Users List route, as well as delete users. In the Songs List route, songs can be analysed and deleted, similarly with the Comments List route and the Reports List route. Admins have a special symbol next to their name, making it easy for users to identify them. Also, a small touch to the user interface: administrator profiles have a unique website background compared to the normal user's.

Non-functional Requirements

1. The system can be used by anyone, from any arbitrary location using a personal computer connected to the internet.

2. The system must be implemented as a web application using Node.js, Express.js, Embedded JavaScript and MySQL. The installation and configuration should be enabled by running various install scripts in the terminal.

3. The system must ensure 24/7 operation with an uptime of at least 95%.

4. The system should be compatible with a wide range of web browsers and devices, ensuring a consistent user experience with server-generated content.

5. The system should efficiently load and submit information into the database.

6. The system should respect user privacy and comply with data protection regulations.

7. User data must be securely stored and transmitted, with appropriate encryption and security measures in place to protect sensitive information in both the server backend and the MySQL database.

8. The user interface should be intuitive and responsive, providing a smooth and enjoyable music discovery experience.

9. The system should include documentation for installation, maintenance and integration of the system.

10. The system should be developed and deployed within 3 months of system designs being finalised.

Logical Design

Matrix User role / functions

Table 1: Matrix user role/functions

Functions	Administrator	Registered User	Unregistered User
Creating New Account	No	No	Yes
Retrieving Music	Yes	Yes	No
Submitting Music	Yes	Yes	No
Viewing User Profiles	Yes	Yes	No
Liking Songs	Yes	Yes	No
Commenting on Songs	Yes	Yes	No
Deleting Own Comment	Yes	Yes	No
Searching Users	Yes	Yes	No
Searching Songs	Yes	Yes	No
Reporting Comments	Yes	Yes	No
Reporting Songs	Yes	Yes	No
Granting Administrator Privileges	Yes	No	No
Special Badge	Yes	No	No
Deleting Songs	Yes	No	No
Deleting Specific Users	Yes	No	No
Granting Administrator Privileges	Yes	No	No
Deleting Specific User Comments	Yes	No	No
Deleting Reports	Yes	No	No
Solving/Unsolving Reports	Yes	No	No
View Extra Song/User Data	Yes	No	No
Unique Administrator UI	Yes	No	No

Table 2: Data dictionary

Entity	Description	Name	Type	Description of attribute
		id	int(11)	Identification number of the user (Primary Key)
		username	varchar(255)	Username of the user
		password	varchar(255)	Hashed password

usersNew	System user database	email	varchar(255)	Email of the user
		bio	text/NULL	User bio which is visible on their profile
		location	varchar(255)/ NULL	User location
		spotify_link	varchar(255)/ NULL	Link to user's Spotify account
		applemusic_link	varchar(255)/ NULL	Link to user's Apple Music account
		soundcloud_link	varchar(255)/ NULL	Link to user's SoundCloud account
		admin	tinyint(1)	Administrator or not
		imageUrl	varchar(255)	Image link to the profile picture
songsNew	Main songs database	**song-id**	int	Identification number of the song (Primary Key)
		title	varchar(255)	Song title
		artist	varchar(255)	Song artist
		genre	varchar(255)/ "Other"	Song genre
		yt_link	varchar(255)/ NULL	Song YouTube link
		spotify_link	varchar(255)/ NULL	Song Spotify link
		soundcloud_link	varchar(255)/ NULL	Song Soundcloud link
		user_id **(Foreign Key)**	int	ID of the user that suggested the song
		upload_date	date/NULL	When was the song suggested
		like_count	int/NULL	Number of likes
		comment_count	int/NULL	Number of comments
		explicit	tinyint(1)	Does the song contain explicit content?
		retrLocation	varchar(255)/ "Earth"	Current location of the user when the song was submitted
comments	Comment on a song	**comment_id**	int(11)	Comment identifier (Primary Key)
		song_id **(Foreign Key)**	int(11)	Comment on which song
		user_id **(Foreign Key)**	int(11)	Comment by who
		content	text	Comment content
retrievedSongs	Songs retrieved by	user_id	int(11)	Song retrieved by
		song_id	int(11)	User retrieved which song

retrievedSongs	retrieved by users	retrieval_date	date	When was it retrieved
		retrieval-id	int(11)	Identification of retrieval
reports	Report issued by some user	report-id	int(11)	Report identified
		reported-by **(Foreign Key)**	int(11)	Who was the report submitted by
		report-text	text	Description of report
		report-date	date	When was the report submitted
		status	varchar(255)	Report status
		comment-id **(Foreign Key)**	int/NULL	The comment in question

Entity relational diagram (ERD)

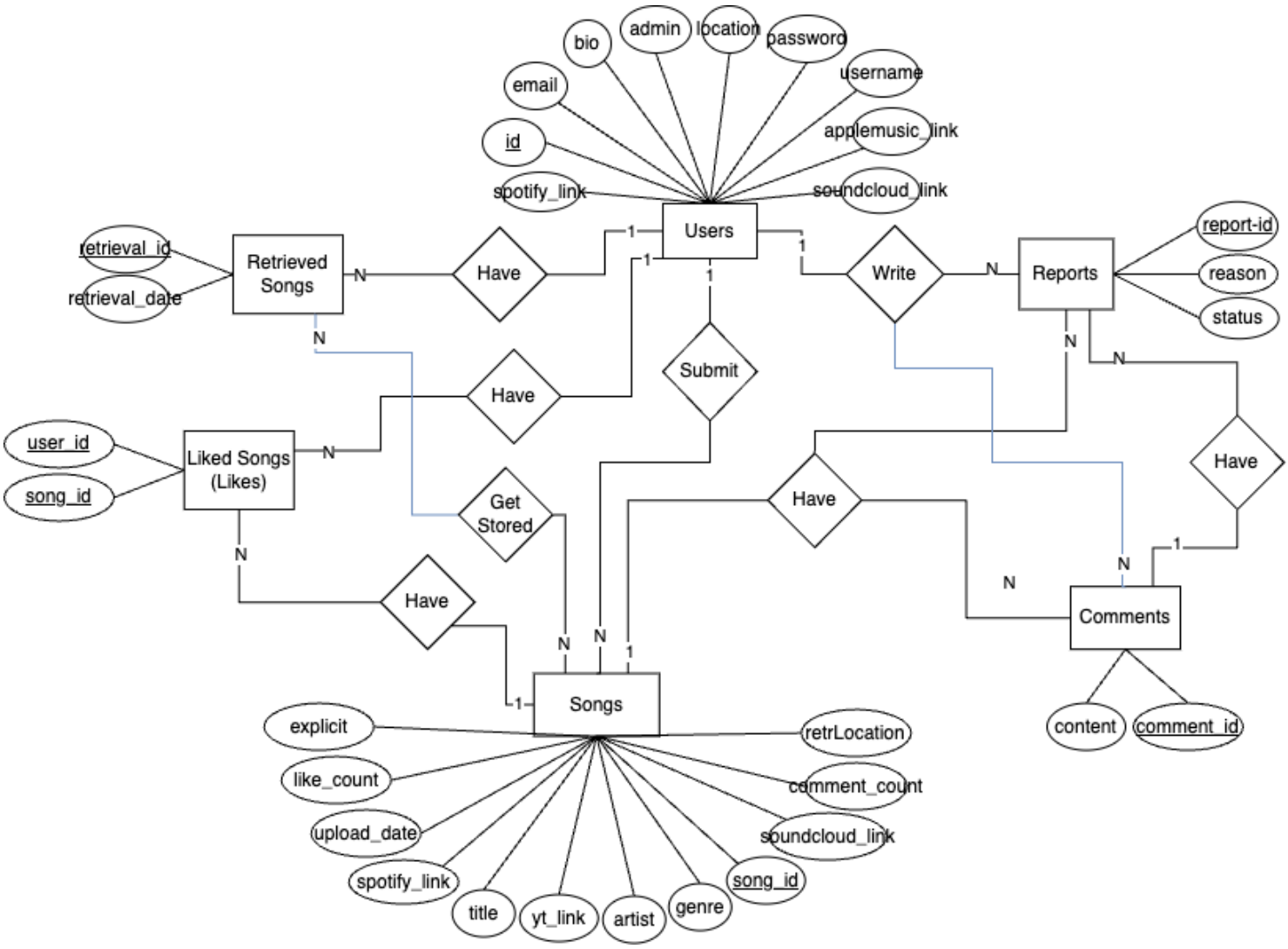

Figure 1: Entity relation diagram

Relational model

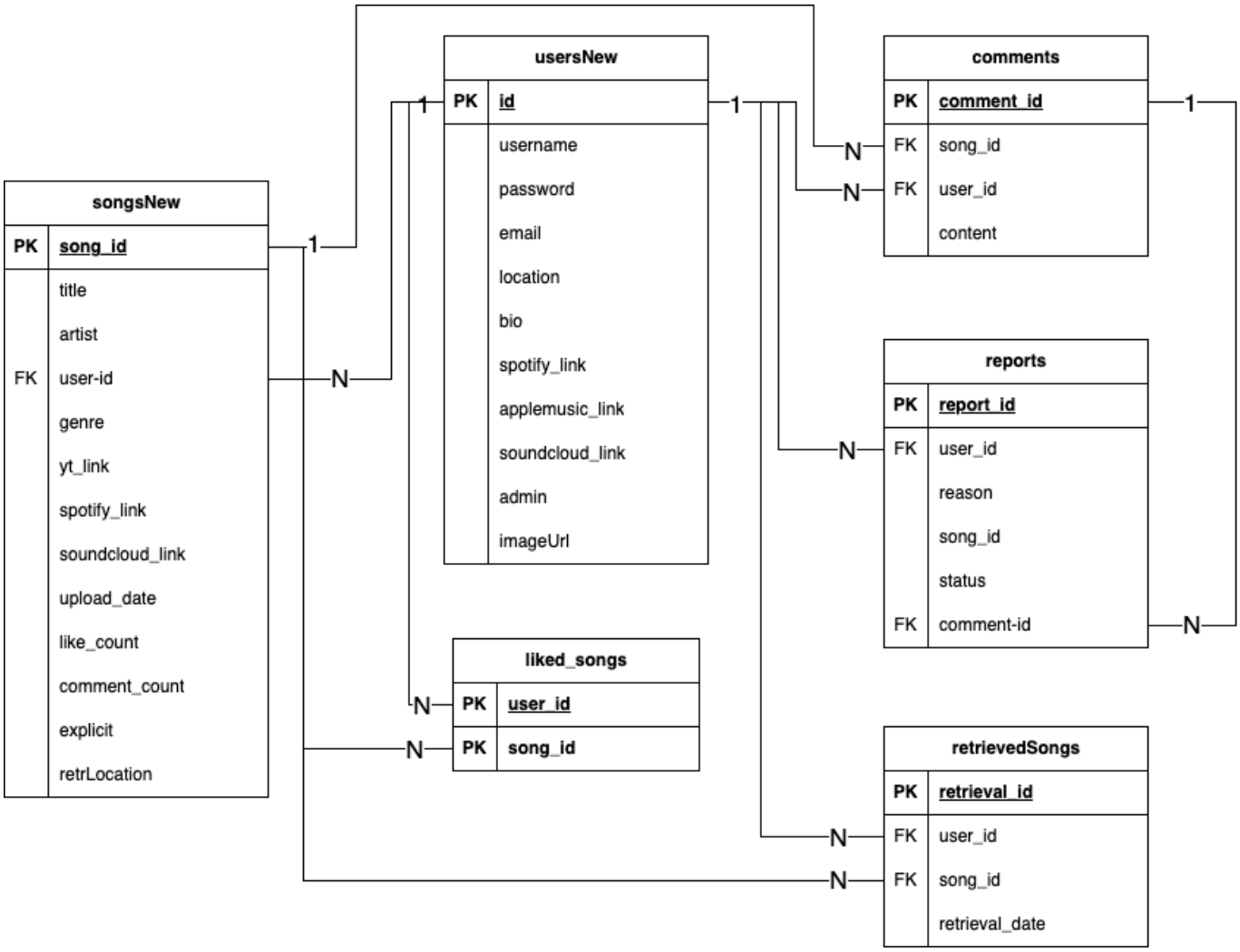

Figure 2: Relational model diagram

The entire database model is in the third normal form (3NF). Each entity has attributes that are fully functionally dependent on the primary key, and there are no transitive dependencies. This means that it is also part of the first and second normal form, which means that all non-prime attributes are fully dependent on their respective primary keys.

Object-oriented Analysis

UML Class diagram

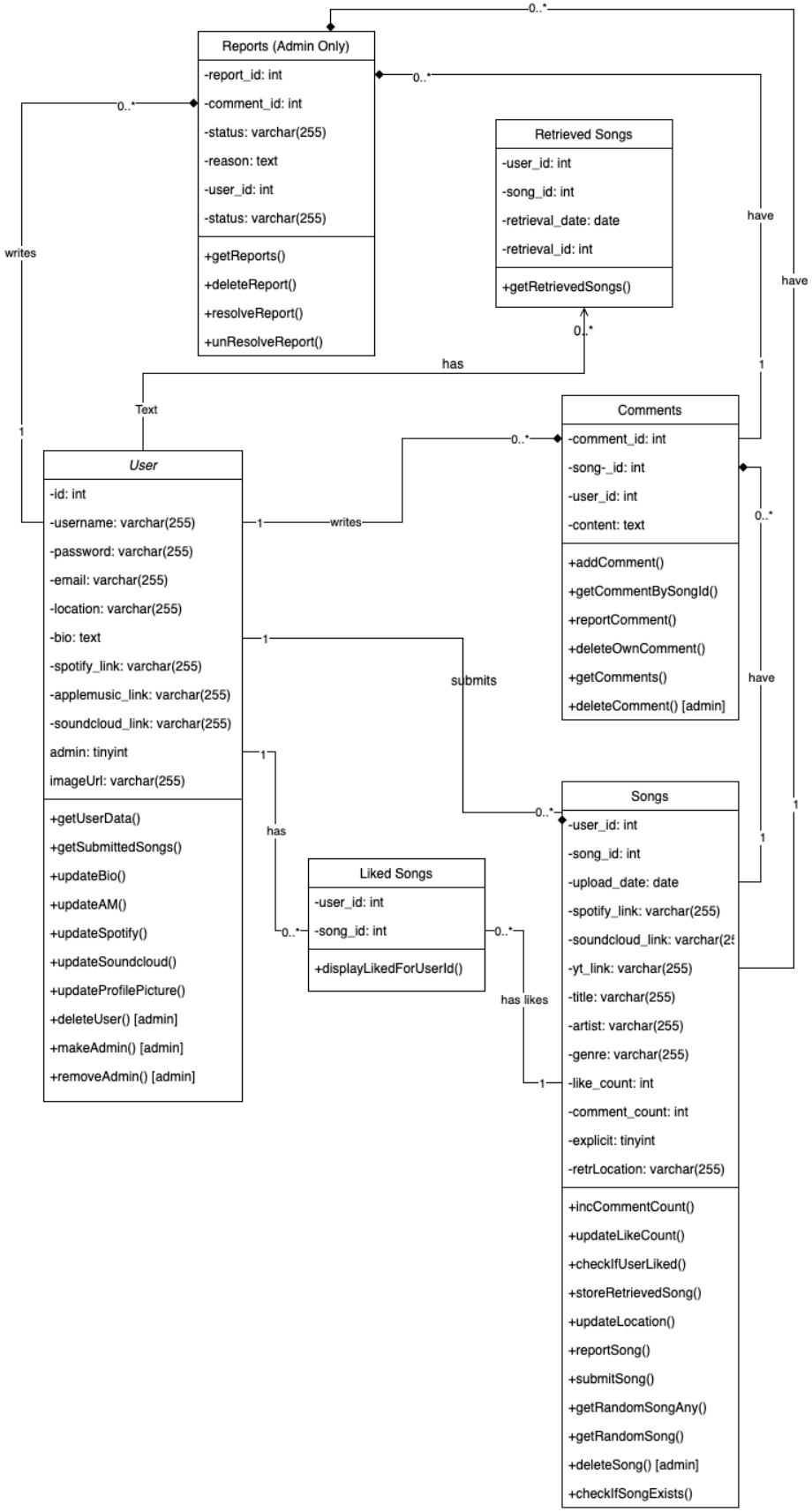

Figure 3: UML Class diagram

UML Sequence Diagrams

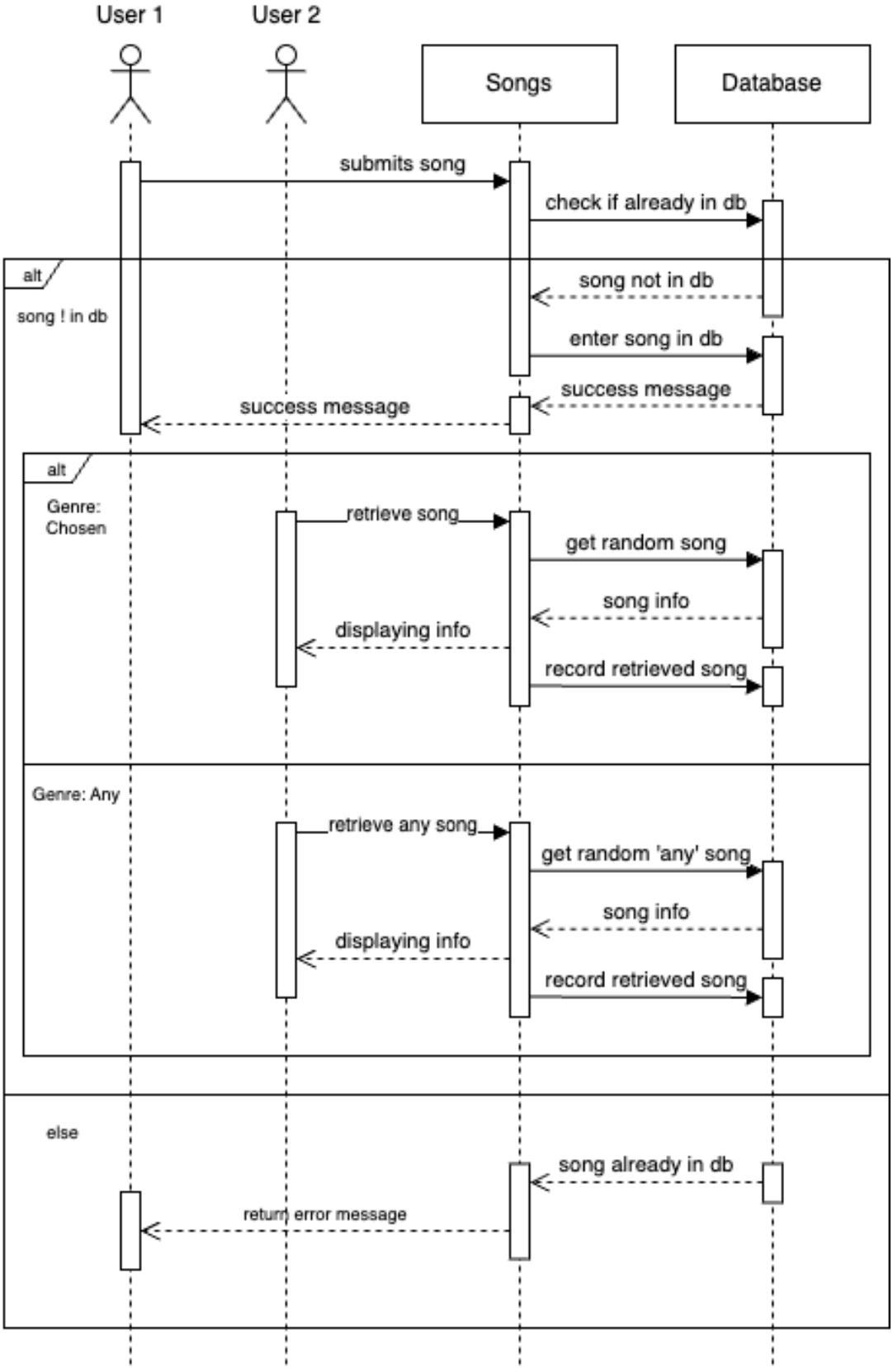

Figure 4: Sequence diagram of submitting and retrieving a song

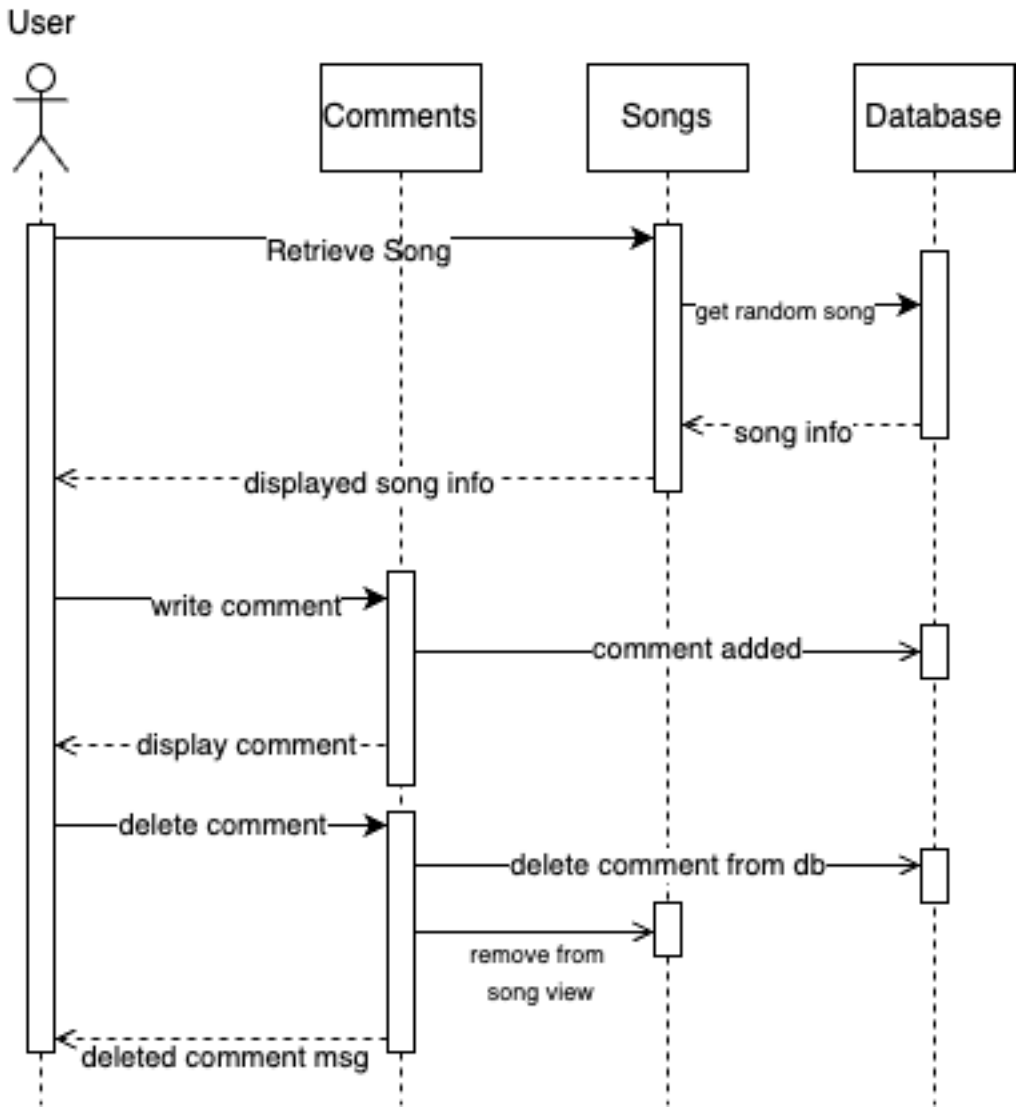

Figure 5: Sequence diagram of adding then deleting a comment

Process modelling

Functional decomposition diagram

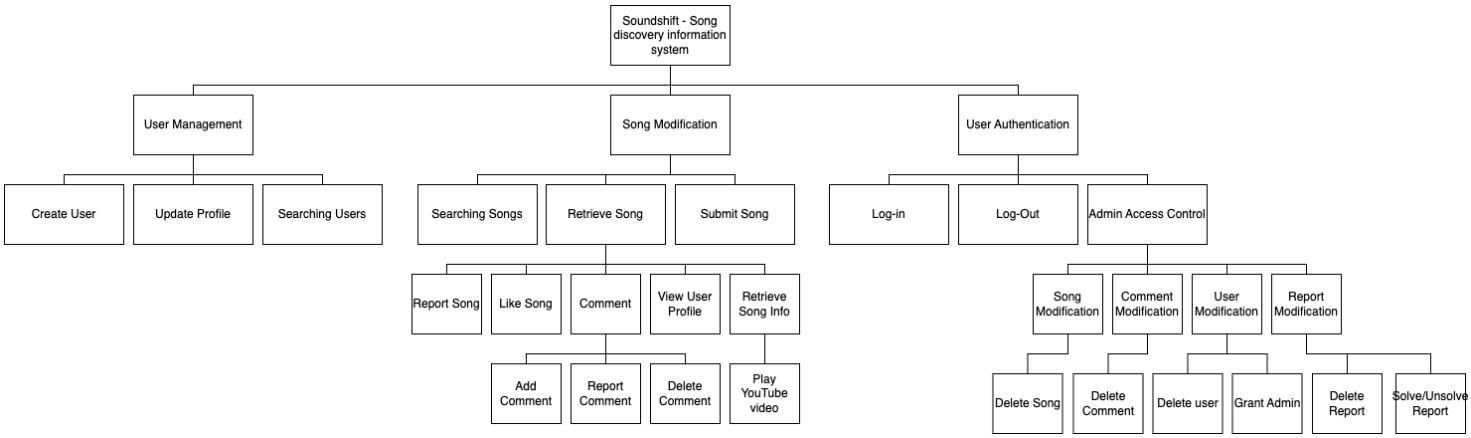

Figure 6: Functional decomposition diagram

Data flow diagram

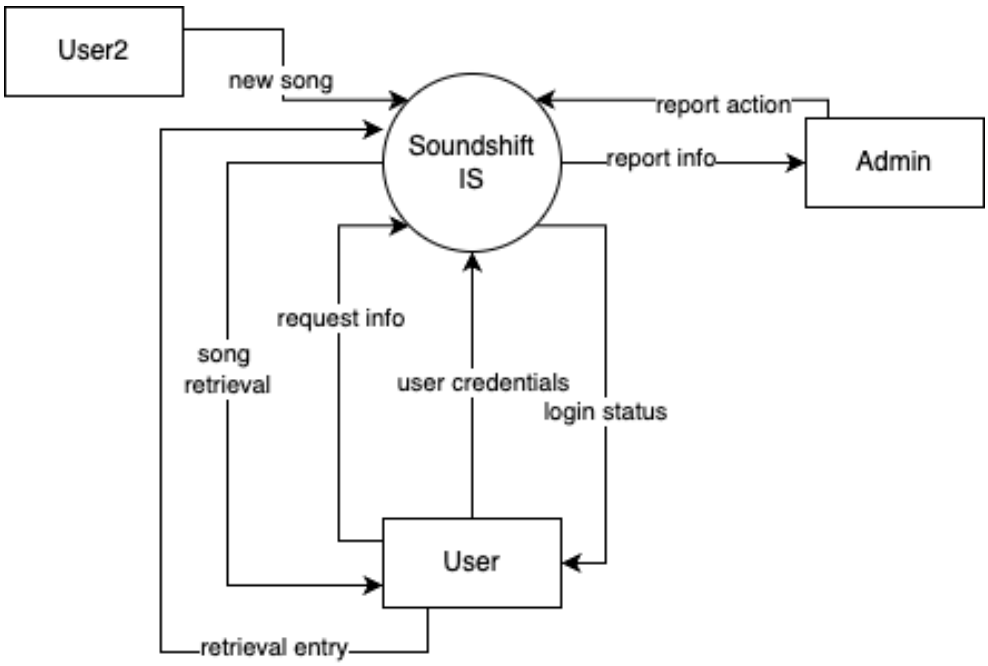

Figure 7: Contextual Data Flow Diagram

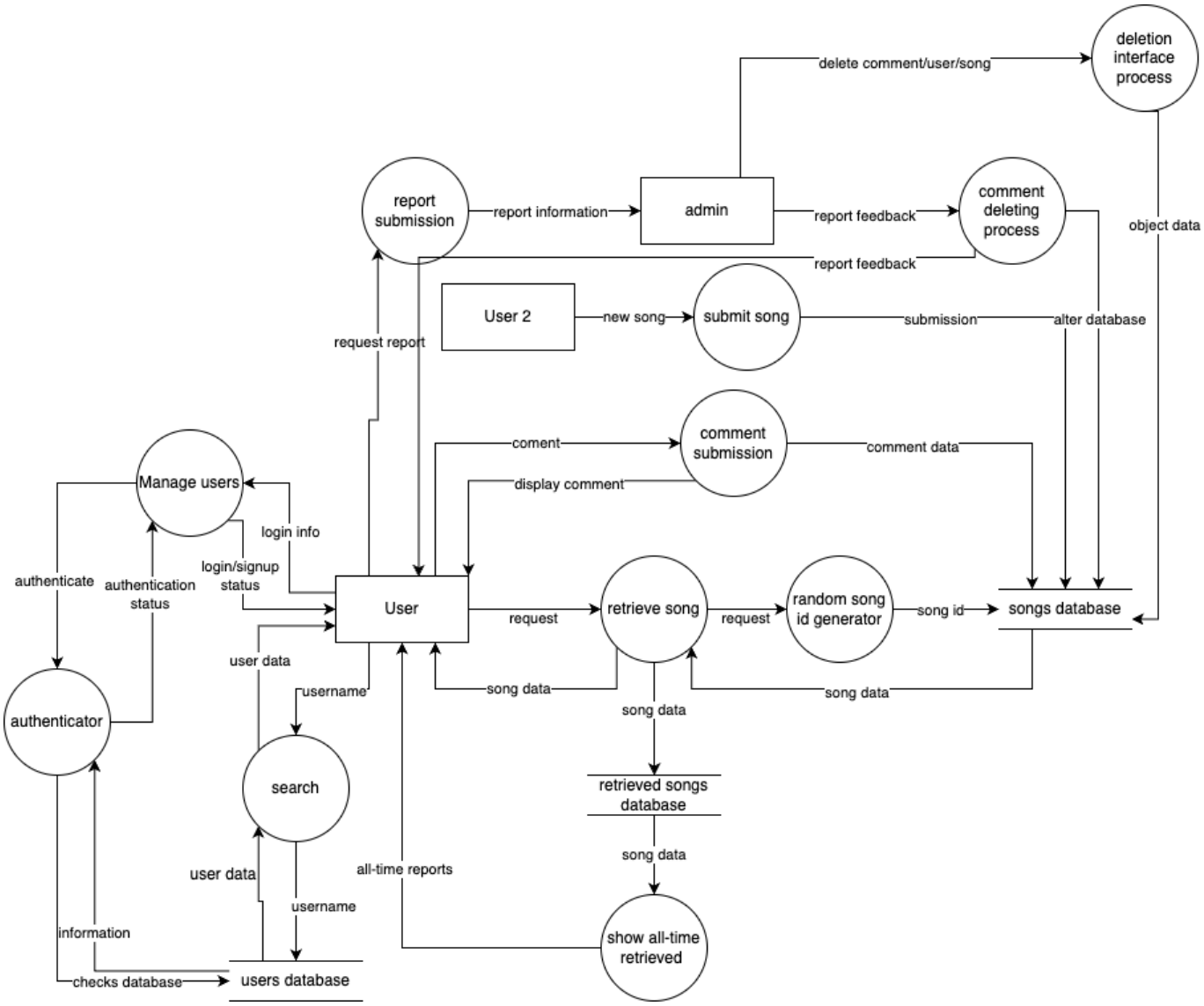

Figure 8: System level data flow diagram

Physical design phase

Physical data model

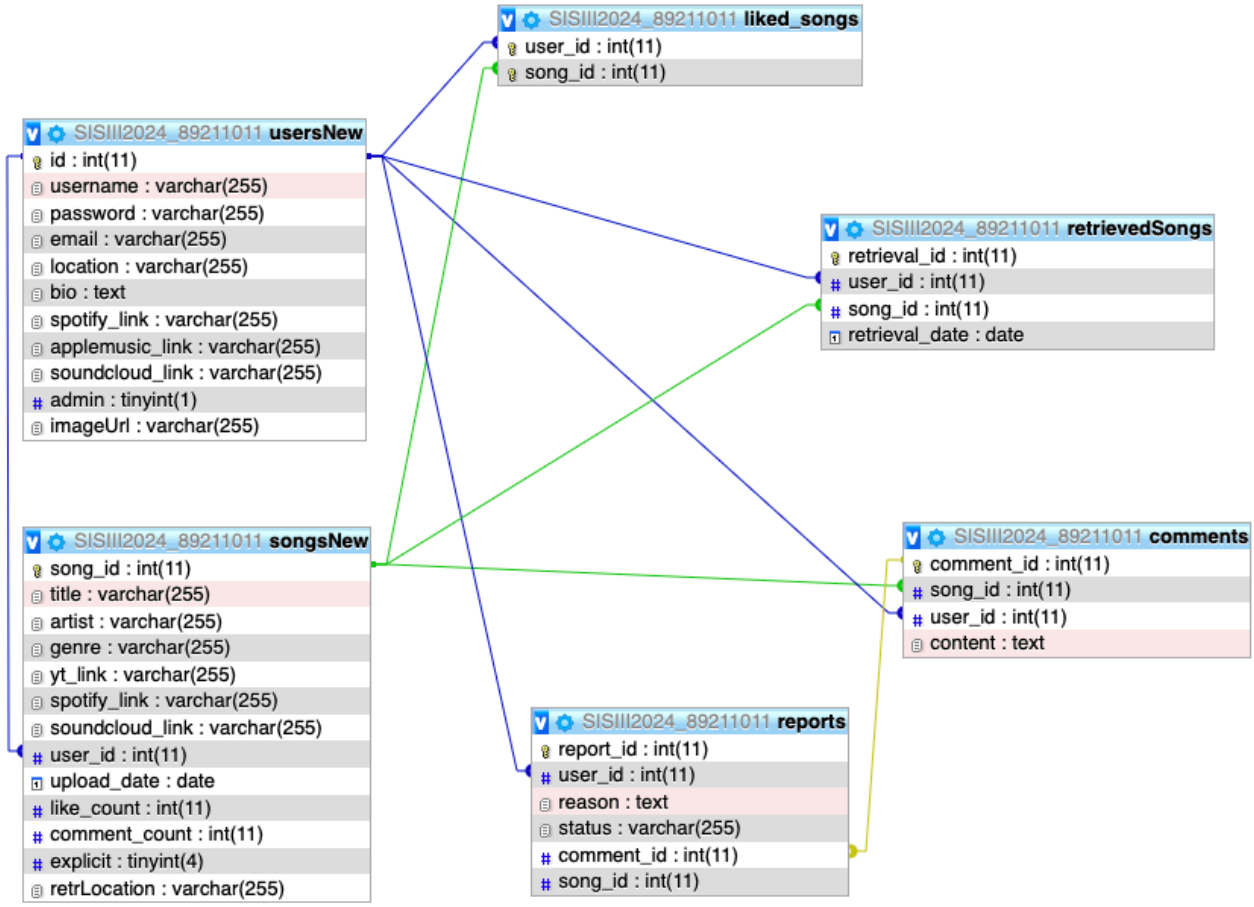

Figure 9: Physical data model