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. Additionally, you have the option to engage in discussions with other individuals who also appreciate the song.

Functional Requirements

The system should enable the following functionalities:

1. Users are able to retrieve random songs and add their own songs to the main music database. These songs can also be linked to various streaming platforms and filtered by genres. Additional song information includes the suggester's identity, location, and their recent suggestions and retrievals. Visiting the suggester's profile after having a song suggested is also possible. Explicit song results can be filtered with a toggle.

2. Access to the system is restricted to registered users only. This functionality is crucial for furthermore music exploration and keeping track of a user's liked songs.

3. Users can view their own profiles, including song retrieval, submission history, and likes. Users can add a bio and link their listening platform profiles to their profiles. Users can also view other users' profiles for furthermore music discovery.

4. Searching for other users and viewing their profiles for recommendations is also possible. On the other hand, viewing and searching the main music database can be toggled on in the user settings, with the default setting off to maintain the element of surprise.

5. Each retrieved song can be liked, disliked and commented on. This can only be done from the main page, after receiving a song retrieval. Comments reported as inappropriate can be removed by admins. The same holds for entire accounts as well.

Non-functional Requirements

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

2. The system must be implemented as a web application using Node.js/Express and MySQL. The installation and configuration should be enabled by running various install scripts.

3. The system must ensure 24/7 operation with an uptime of at least 95%, with Node.js and MySQL components working reliably.

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 songs into the main 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 Node.js 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
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
Searching Users	Yes	Yes	No
Viewing Music Database	Yes	Yes	No
Viewing User Database	Yes	No	No
Viewing Comments Database	Yes	No	No
Viewing Retrieved Songs Database	Yes	No	No
Viewing Reports Database	Yes	No	No

Granting Administrator Privileges	Yes	No	No
Reporting Comments	Yes	Yes	No
Reporting Profiles	Yes	Yes	No
Deleting Own Comment	Yes	Yes	No
Deleting User Comment	Yes	No	No
Deleting Own Account	Yes	Yes	No
Deleting User Account	Yes	No	No
Creating New Account	No	No	Yes
Special Badge	Yes	No	No
Deleting Specific Songs	Yes	No	No

Table 2: Data dictionary

Entity	Description	Name	Type	Description of attribute
Users	System user database	**user-id**	int	Identification number of the user (primary key)
		username	varchar(255)	Username of the user
		password	varchar(255)	Hashed password
		email	varchar(255)	Email of the user
		bio	text/NULL	User bio which is visible on their profile
		location	varchar(255)/ NULL	User location
		platforms	text	User listening platforms
		register-date	date	Registration date of the user
		role	varchar(255)	User role (administrator or not)
Songs	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)	Song genre
		yt-link	varchar(255)	Song YouTube link
		spotify-link	varchar(255)	Song Spotify link
		soundcloud-link	varchar(255)	Song Soundcloud link
		user-id **(Foreign Key)**	int	ID of the user that suggested the song

		upload-date	date	When was the song suggested
		like-count	int/NULL	Number of likes
		comment-count	int/NULL	Number of comments
Comments	Comment on a song	**comment-id**	int	Comment identifier (Primary Key)
		song-id **(Foreign Key)**	int	Comment on which song
		user-id **(Foreign Key)**	int	Comment by who
		content	text	Comment content
		comment-date	date	Comment date
Retrieved Songs	Songs retrieved by users	user-id	int	Song retrieved by
		song-id	int	User retrieved which song
		retrieval-date	date	When was it retrieved
		retrieval-id	int	Identification of retrieval
Reports	Report issued by some user	report-id	int	Report identified
		reported-by **(Foreign Key)**	int	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)

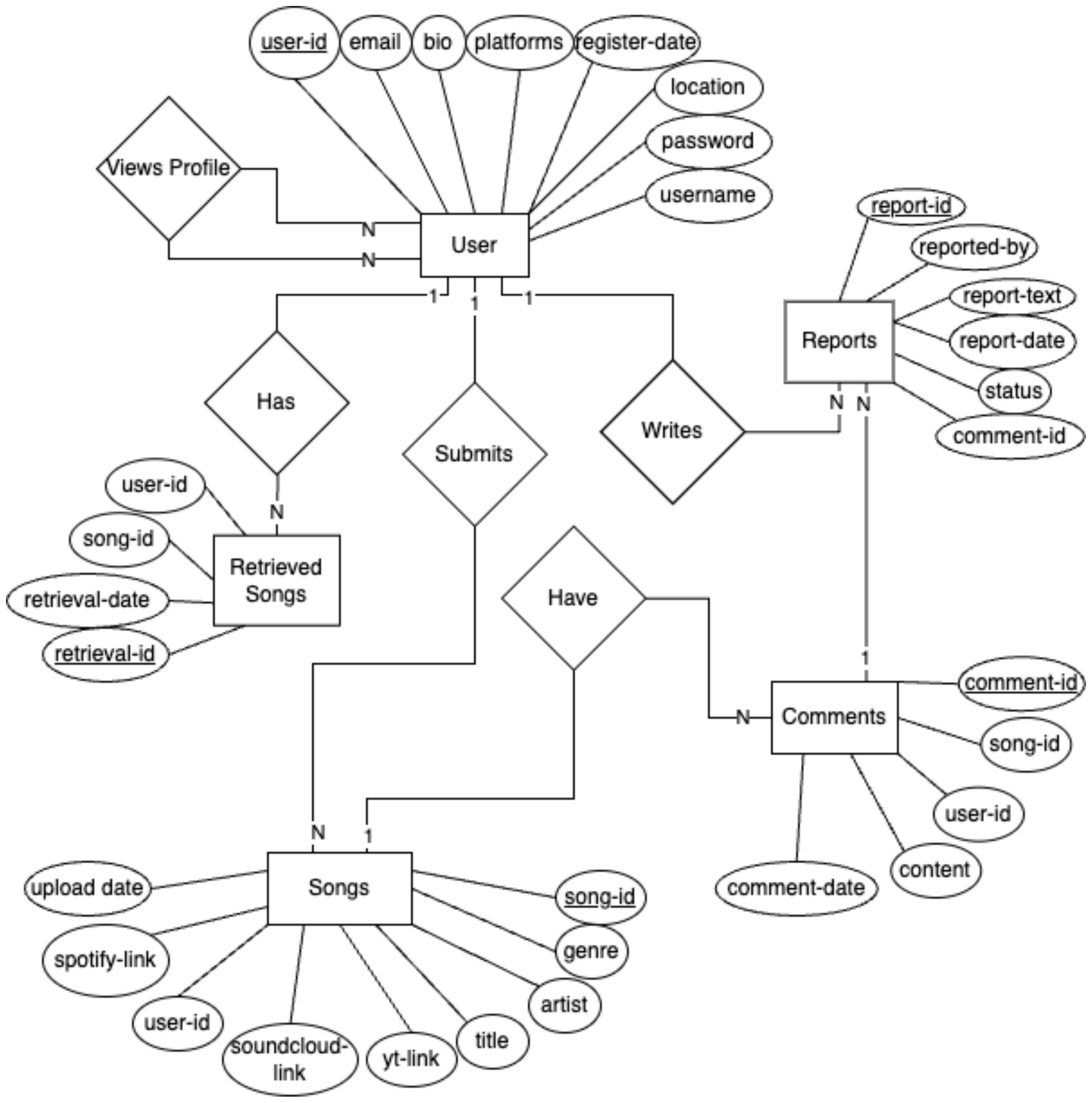

Figure 1: Entity relation diagram

Relational model

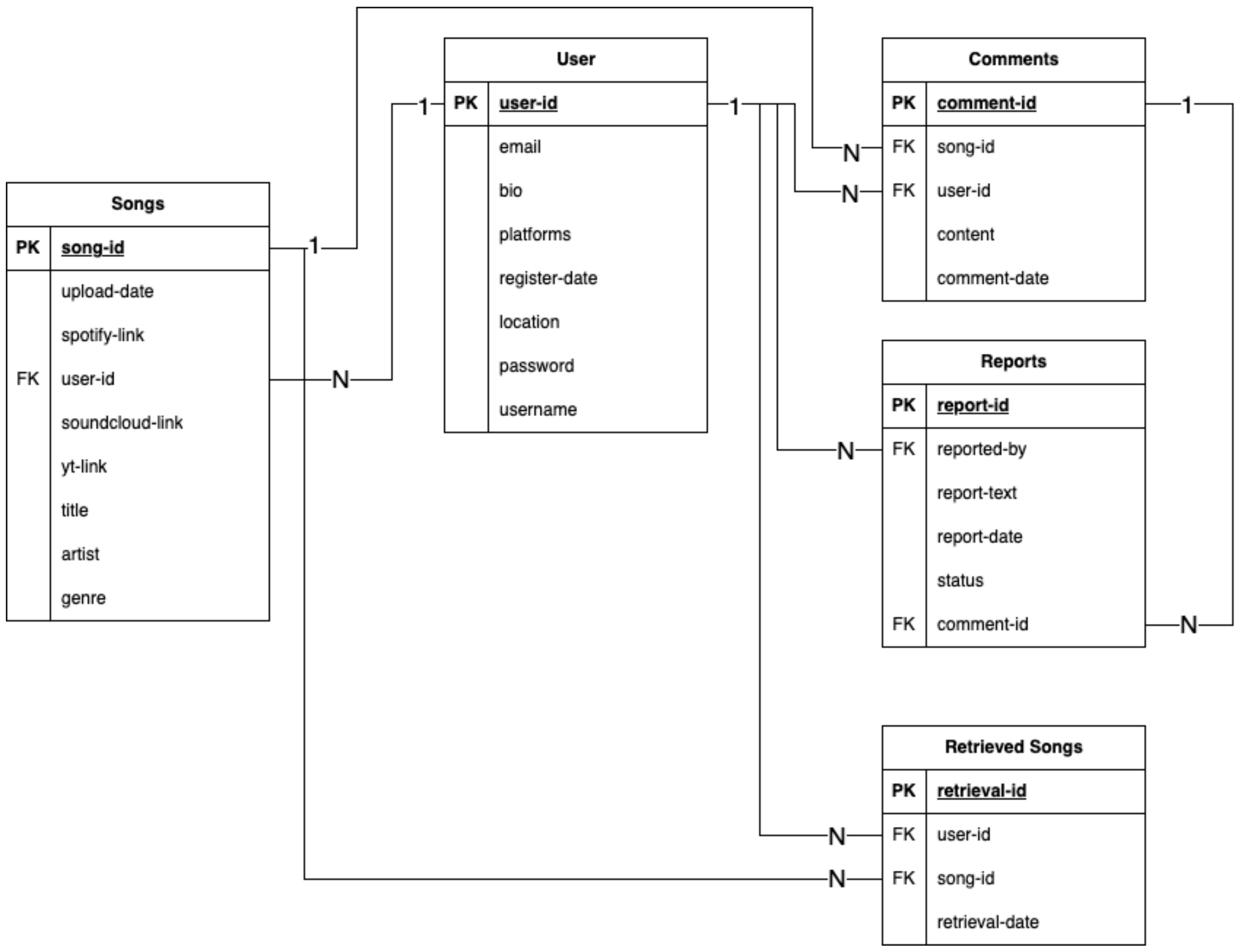

Figure 2: Relational model

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

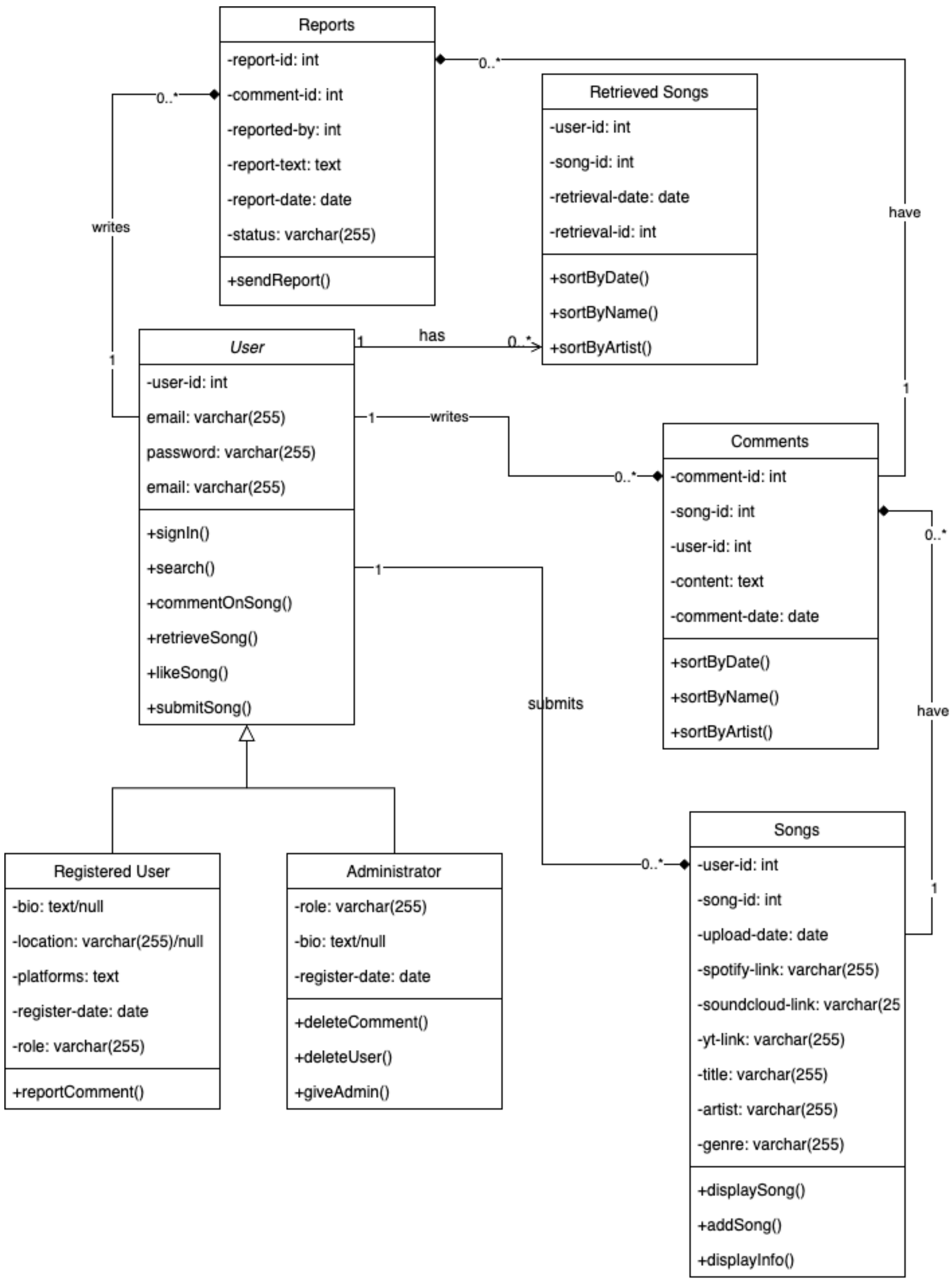

Figure 3: Class Diagram

UML Sequence Diagrams

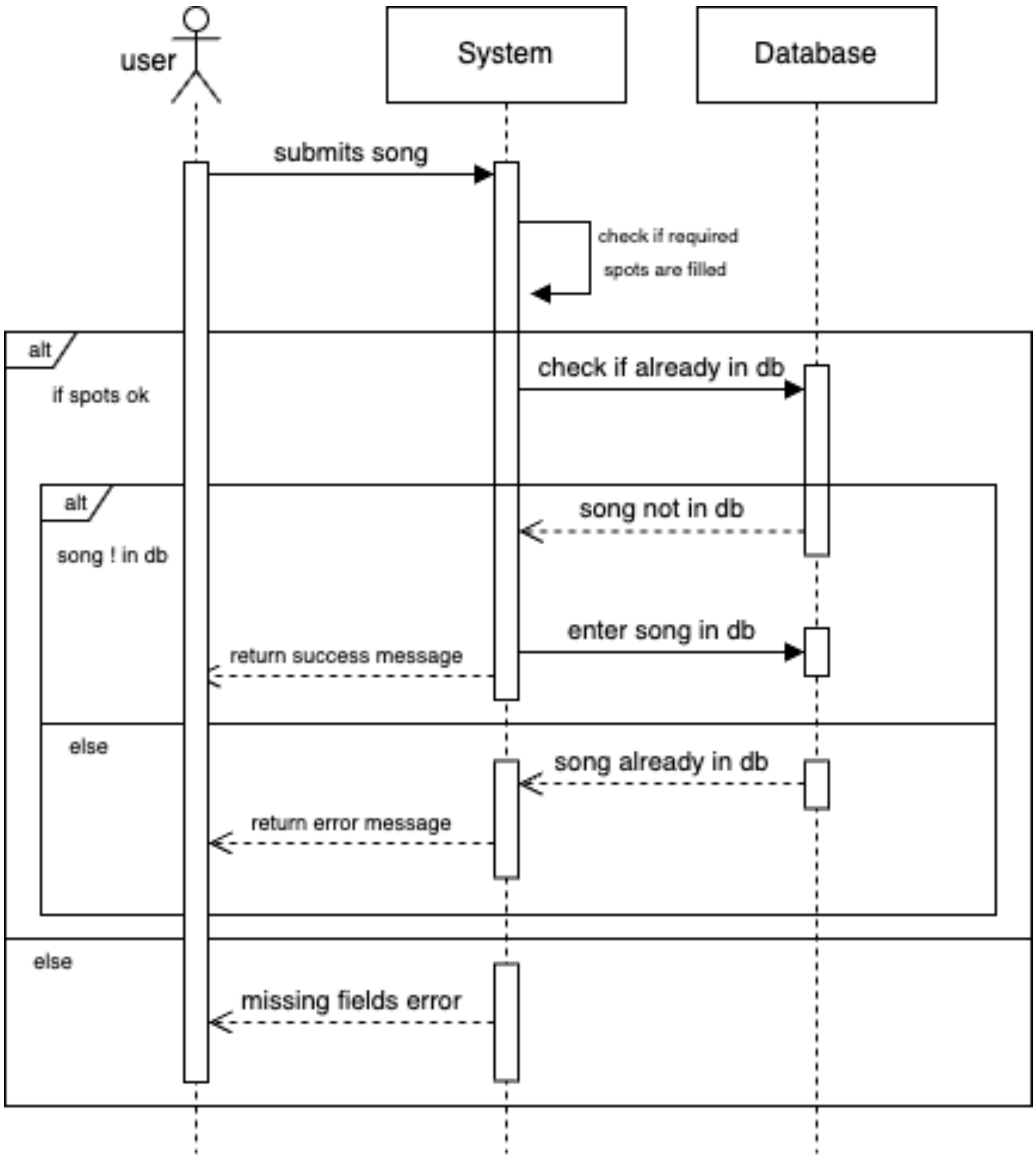

Figure 4: Sequence diagram of submitting a song

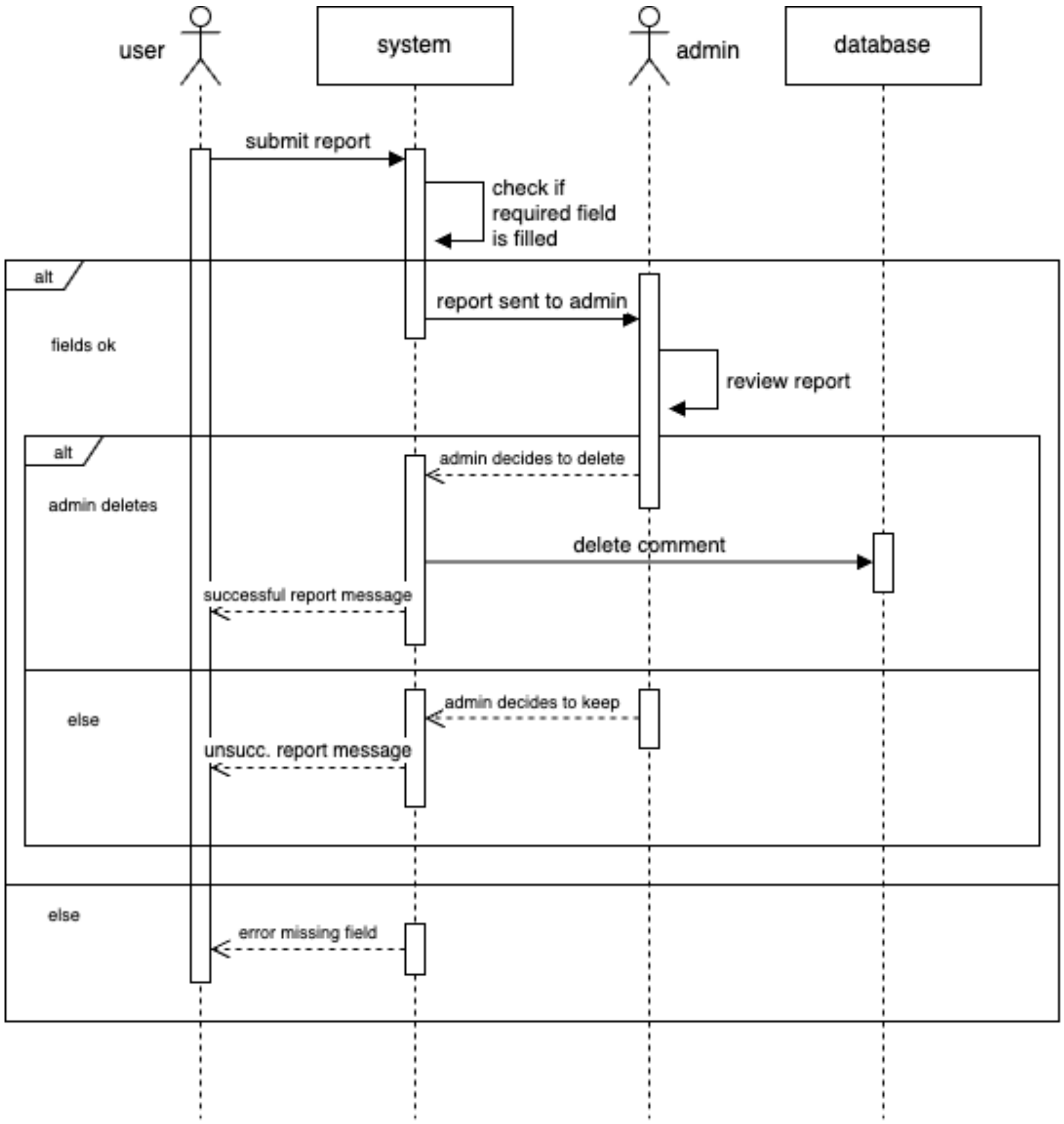

Figure 5: Sequence diagram of submitting a report

Process modelling

Functional decomposition diagram

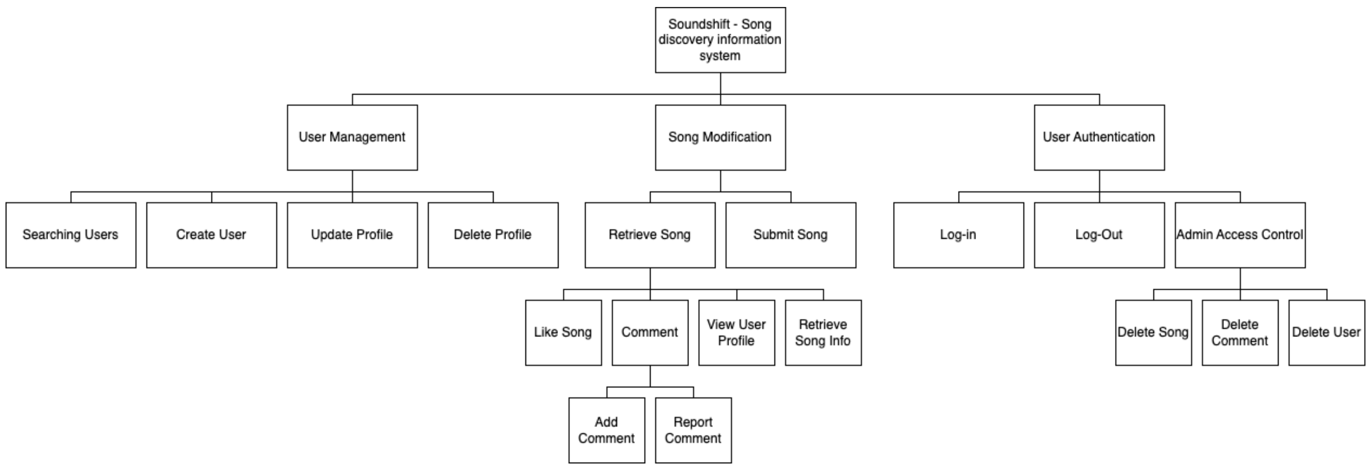

Figure 6: Functional decomposition diagram

Data flow diagram

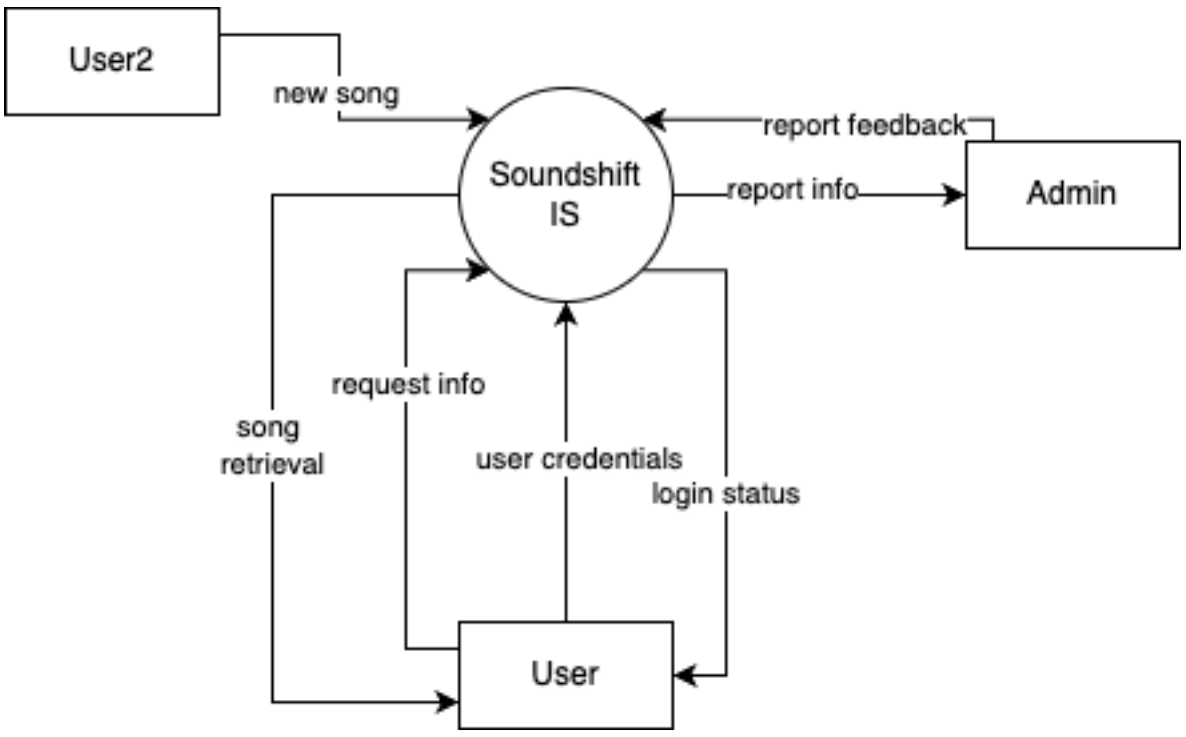

Figure 7: Contextual Data Flow Diagram

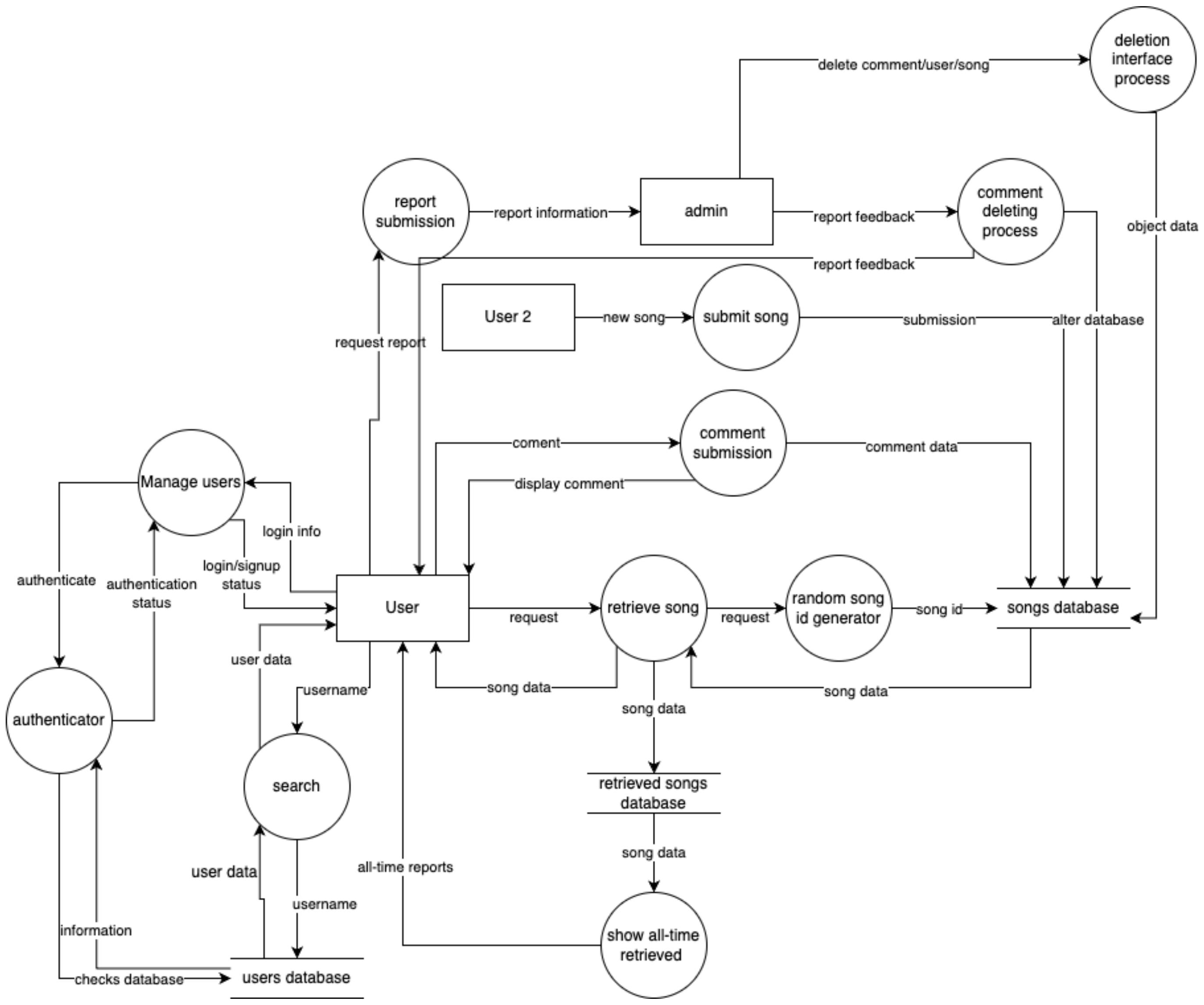

Figure 8: System level data flow diagram

Physical design phase

Physical data model

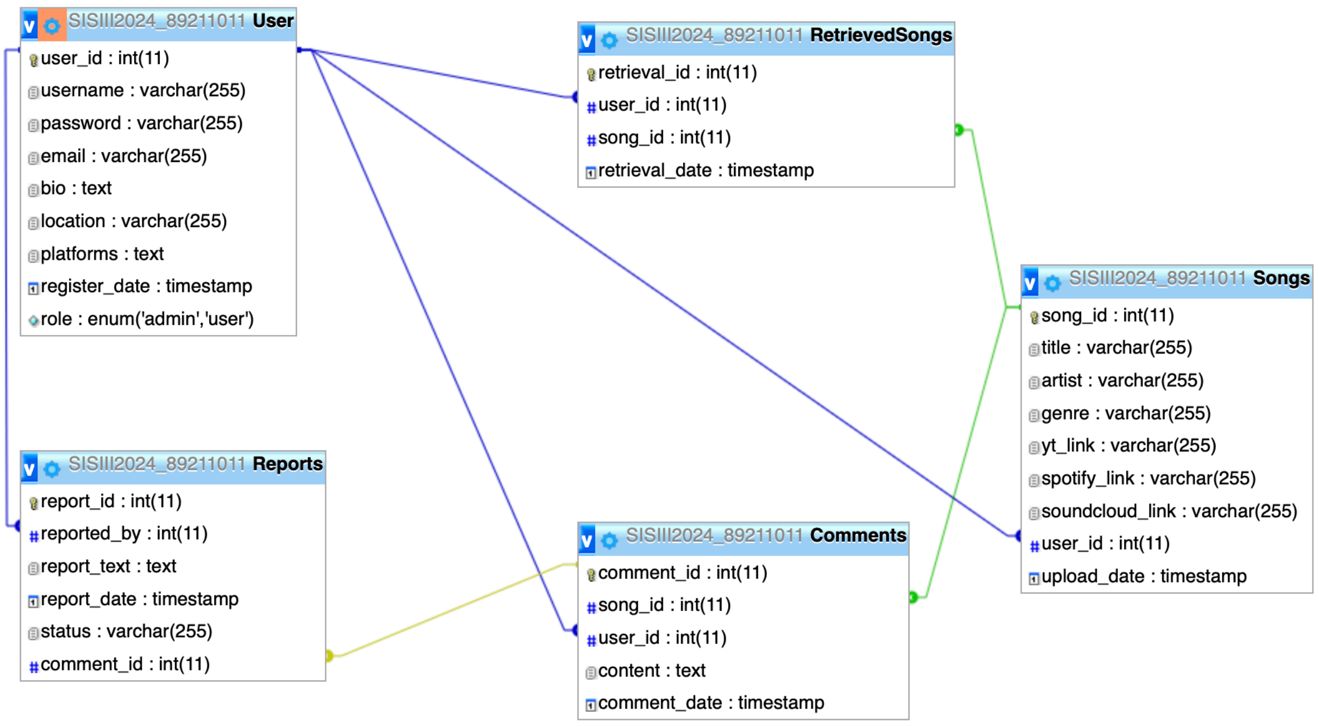

Figure 9: Physical database model