




North South University
Department of Electrical and Computer Engineering

CSE327 PROJECT REPORT

Project Name: Koe

Group Members:

- Shadman Tahmid Arib (2312201642)
- Habiba Alam Raisa (2231272642)
- Sakibul Samir Rafi (2232080642)
- Omar Ahmed Sakib (2022006042)

GitHub Repository:  <https://github.com/Calambrito/Koe>

July 20, 2025

1 Project Description

Koe is meant to be an android application built for the purpose of music streaming and management developed using Google's **flutter** framework. Drawing inspiration from Spotify our project aims to provide for its users a feature rich, intuitive, and highly customizable interface that allows for the music streaming and managing experience to seamlessly blend into their daily activities.

The app will now only interact with two different actors, a **Listener** and an **Admin**. A **Listener** is a person who interacts with the app as a customer, listening to music, searching for new music and even making their own playlists. Another entity that will interact with the app is the admin. The admin will be responsible for creating URLs from our cloud storage of choice **AWS S3** and then adding these songs to the database for the **Listeners** to enjoy. In a future update, an abstraction layer can be added such that an **Artist** can directly register and upload their mp3 files that will be automatically be uploaded to a cloud storage solution and the URL automatically stored in our database.

A **Listener** will be initially greeted by a get started prompt that will lead them into creating their first playlist. Then they will be able to navigate to a discovery interface where they can search for songs. They can search for songs by artist or by song name, a further solution may be added to allow for search by genre. The songs can be played from the discovery search results or added to a playlist for easier access in the future.

Each playlist is by default going to loop in order of song priority. The **Listener** at any time can move songs up and down to change their order of priority. They can even set a single song to loop within their playlist. A **Listener** will be able to configure their app such that it can change themes for each playlist to correspond to the set of emotions they may feel while constructing such a playlist.

Once a certain artist is added to a **Listener's** playlist, they are automatically considered to be subscribed to that artist. Whenever our **Admin** uploads a song that is released by that particular artist. Each user currently subscribed to that artist will get a notification about the new song being released.

Songs and relevant data will be stored in either an SQLite database or a config.json file. For large chunks of data like the data required for song and playlist objects, we will use the database and for smaller bits of information like startup info and theme info we will opt for config files to reduce database dependency and complexity.

Koe's development will involve the use of atleast 6 design patterns to make our code as efficient, readable and maintainable as possible. Our goal is to design the application's backend using **C++** to handle the **OOP** (Object Oriented Programming) part that will be responsible for the implementation of most of these design patterns while keeping things as fast and efficient as possible.

Ultimately, our goal is to:

- Develop a full-featured music streaming platform
- Provide a personalized and dynamic UI/UX experience
- Allow audio playback, user-curated playlists, and artist profiles
- Minimize bugs through clean and structured implementation
- Implement robust user authentication and authorization for safety

In the future, if we manage to scale up, we plan to include features such as:

- Allow artists to open their own profiles and upload music directly
- Add song thumbnails
- Mood-based music recommendations
- Displaying lyrics while a song is playing
- Add more themes including animated themes
- Allow heavier customization of UI elements

End Users

Our end users will be anyone who is enthusiastic about listening to music.

Functional Requirements

1. Login & Registration

- Users can create their own account.
- Users can log into their account.
- The system will show an error message if the user tries to log in with the wrong password.

2. User Roles

- Admins can manage content by adding or removing songs.
- Listeners can listen to music or songs.
- Listeners can subscribe.
- Artists can send notifications.

3. Song Management

- The system will allow admins to upload songs with metadata.
- Admins will be able to remove songs.

4. Subscription and Notification System

- Listeners will be able to subscribe to an artist.
- Artists will be able to send notifications (e.g., for a new song) to their subscribers.

5. Search and Discovery

- The system will allow users to search songs based on song name, artist, or genre.

6. Themes

- The system shall allow users to select themes from preset options (saved via config files).

7. Song Playback

- Users will be able to play songs.
- Users can enable or disable the loop feature.

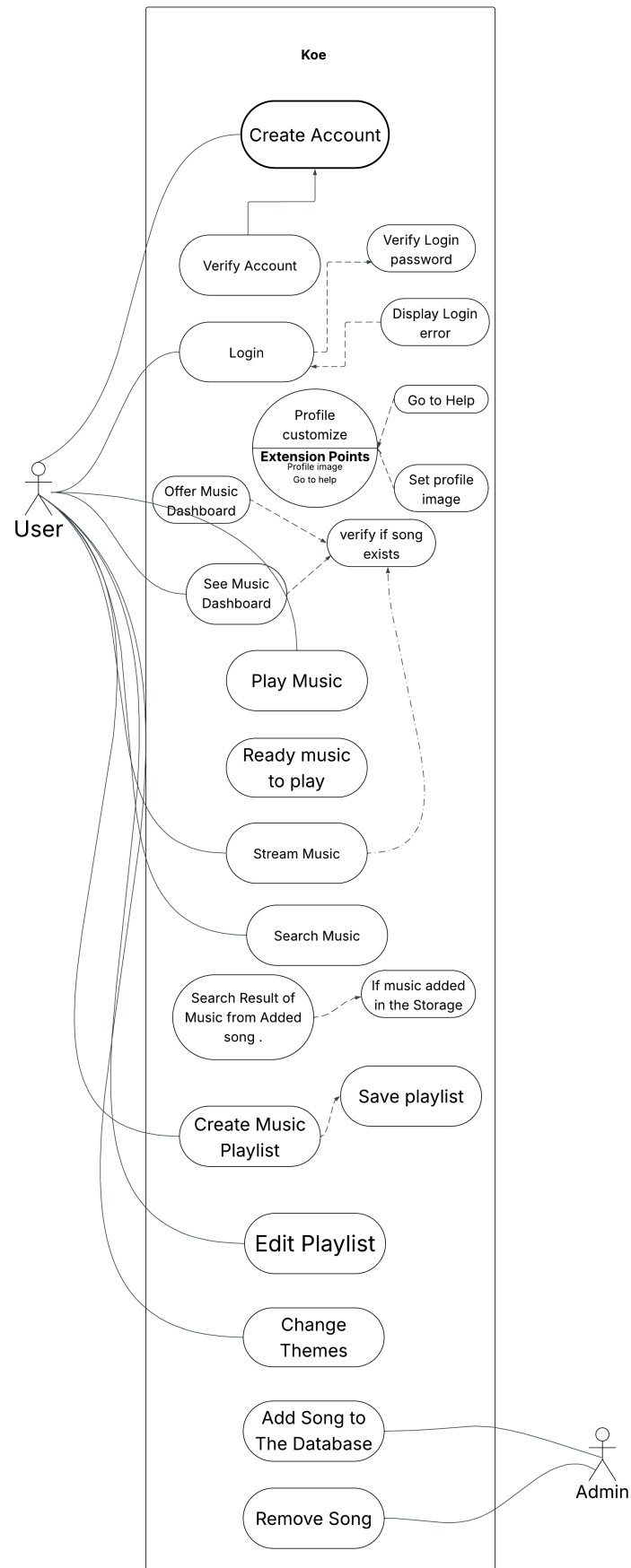
8. Playlist

- Users will be able to create their own playlists.

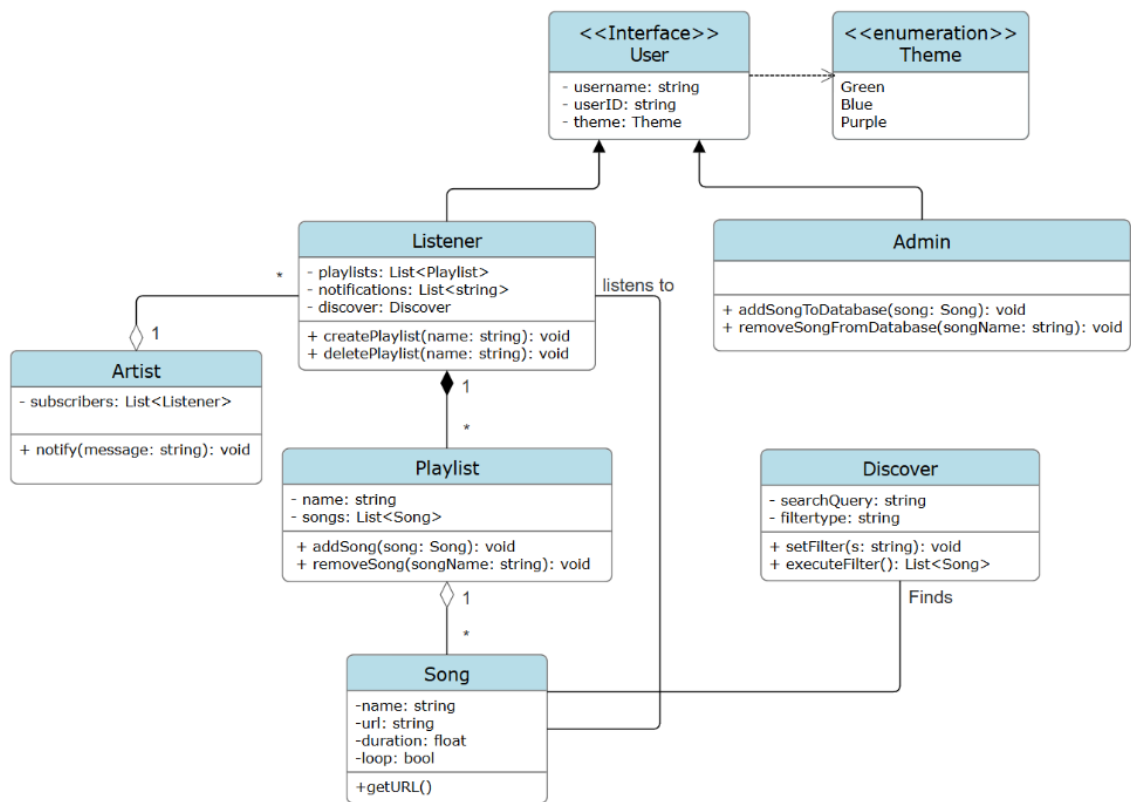
Non-Functional Requirements

- **Performance Requirement:** The system will provide fast response times to users.
- **Usability Requirement:** The user interface will be user-friendly and easy to navigate.
- **Reliability Requirement:** All data will be stored correctly and will be recoverable.
- **Privacy and Safety Requirement:** The system will require authenticated login and passwords for all users. User data will remain safe and secure.
- **Portability Requirement:** The application should run on multiple Android devices with varying specifications.
- **Delivery Requirement:** We are aiming to finish the project within the timeline.

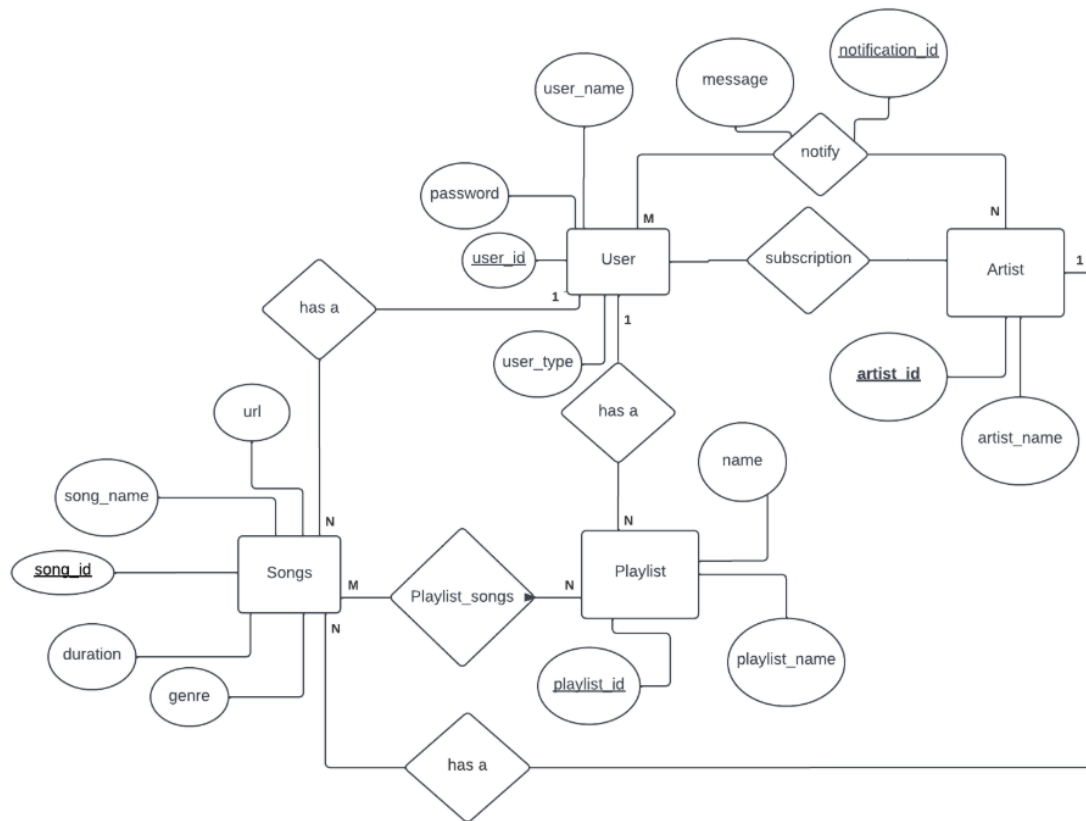
2 USE CASE DIAGRAM:



3 Class Diagram:



4 Entity relationship diagram:



5 Sequence diagram:

