Lizbeth Perez, Jasmine Lau

CSE 111 Checkpoint 1

https://github.com/jasmine-g-lau/discoboxd

DiscoBoxd

Music Rating & Collection Platform

Project Overview

DESCRIPTION:

SOCIAL PLATFORM FOR USERS TO MANAGE & INTERACT WITH A DIGITAL MUSIC LIBRARY

FEATURES:

- LOG LISTENING HISTORY
- WRITE REVIEWS
- CREATE LISTS
- FOLLOW OTHER USERS

Use Cases

FOR GENERAL USERS

1. **BROWSE ALBUMS**

As a user I can browse the album catalog and view album details, including songs, ratings/reviews, artist, and year.

2. **LOG LISTENING HISTORY**

As a user I can log albums I listen to.

3. **RATE ALBUMS**

As a user I can rate an album between 1-5, contributing to the album’s overall rating

4. **REVIEW ALBUMS**

As a user I can write & share detailed review for albums

Use Cases

5. SEARCH MUSIC AND USERS

As a user, I can search for albums, artists, or users using keywords

6. **FOLLOW USERS**

As a user, I can follow other users to see their activity, reviews, and lists

7. **CREATE LISTS**

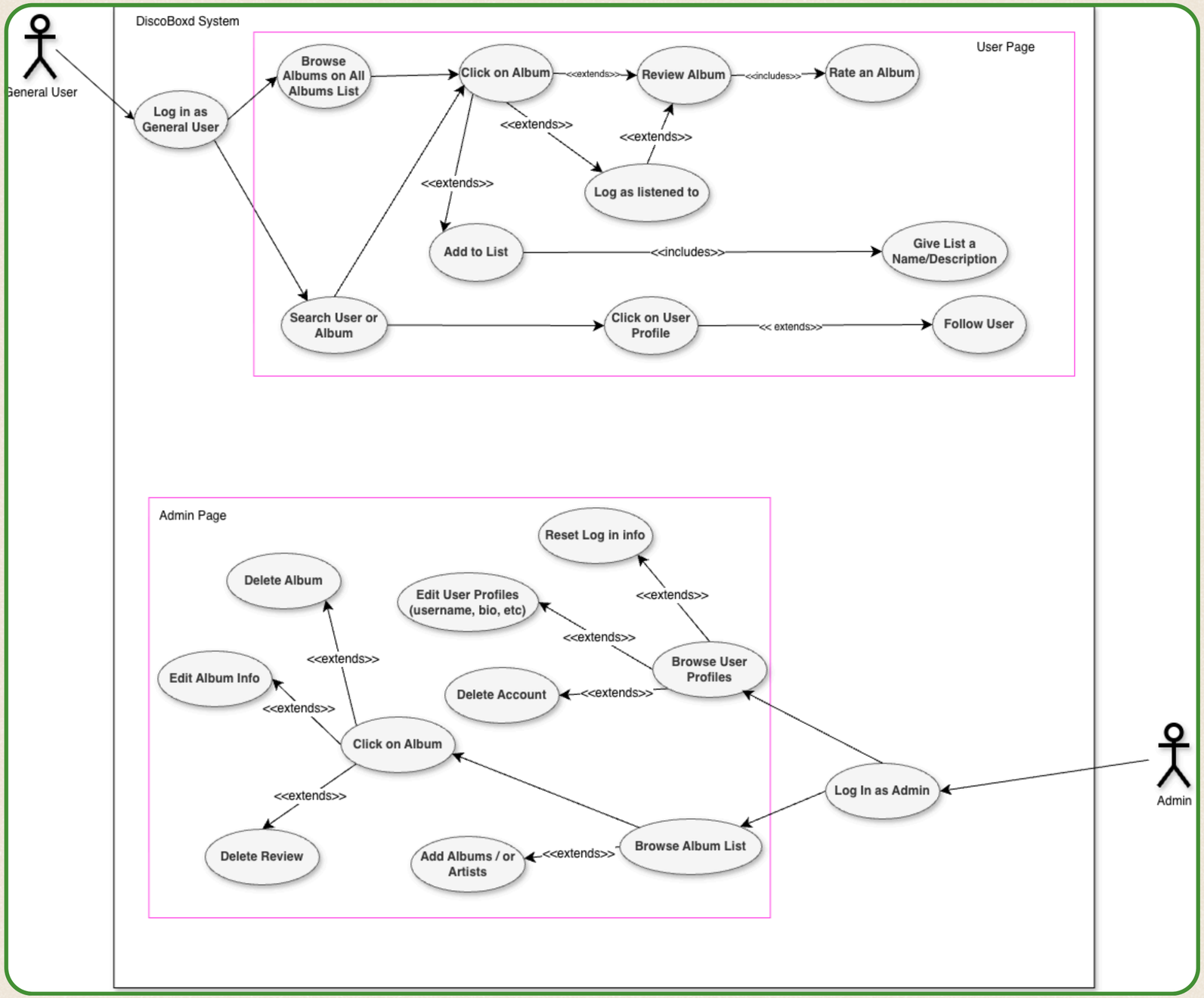
As a user, I can create custom lists, adding albums to lists for personal organization or sharing

FOR ADMIN

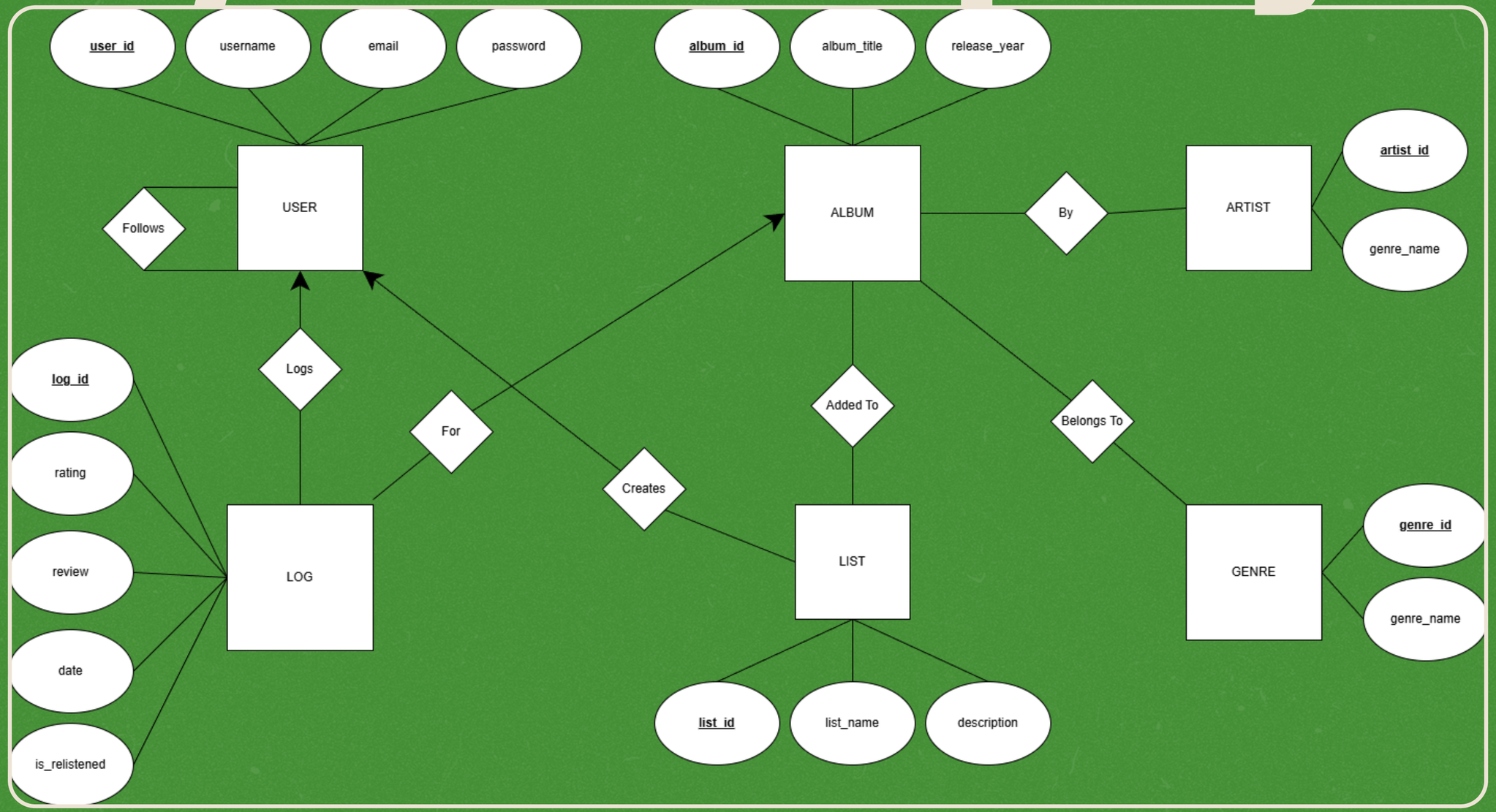
8. MANAGE MUSIC AND USER DATA

As an admin, I can manage albums, user data, ratings, and reviews

UML CASE DIAGRAM



Entity-Relationship Diagram



Entities (6 Entities)

USER
Represents registered users of the platform. Stores user log in information, profile information, and preferences.

ALBUM
Represents music albums with title, release year, and cover art.

ARTIST
Musical artists or bands who create albums.

LIST
User-created collections of albums organized by personal preference.

GENRE
Music genres and subgenres used to categorize albums.

LOG
Users can records when users listen to albums, write reviews, and rate albums they have listened to.

Relationships

(7 relationships, 4 many-many)

FOLLOWS (M:M)
USER-USER
Users can follows many other users

LOGS (1:M)
USER<-LOG
A user logs their watch history (& ratings & reviews), but each list belongs to one user

FOR (1:M)
LOG<-ALBUM
Users can log their listening history to an album. Each log will only be to 1 album

CREATES (1:M)
USER<-LIST
A user creates many lists, but each list belongs to one user

ADDED TO (M:M)
ALBUM-LIST
Lists can contain many albums, and albums can appear in many lists

BELONGS TO (M:M)
ALBUM-GENRE
Albums belong to many genres, and genres contain many albums

By (M:M)
ALBUM-ARTIST
Artists can make many albums, and albums can be made by many artists

ER Diagram to Relations

Strong Entities:
USER(user_id, username, email, password_hash, user_bio, join_date)
ALBUM(album_id, album_title, release_year, cover_image, artist_id)
ARTIST(artist_id, name, artist_bio)
LIST(list_id, user_id, name, description, created_date, is_public)
GENRE(genre_id, genre_name)
LOG(log_id, user_id, album_id, rating, review, date, is_relistened)

Relationships (Junction Tables for M:N):
ADDED_TO(list_id, album_id, position)
BELONGS_TO(album_id, genre_id)
BY(album_id, artist_id)
FOLLOWS(follower_id, following_id, follow_date)

Database SQL Schema (Entities)

```
CREATE TABLE users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL);

CREATE TABLE artists (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL);

CREATE TABLE albums (
    id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(200) NOT NULL,
    release_year INT,
    cover_image VARCHAR(500));

CREATE TABLE genres (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) UNIQUE NOT NULL);

CREATE TABLE lists (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    name VARCHAR(100) NOT NULL,
    is_public BOOLEAN DEFAULT TRUE,FOREIGN KEY (user_id) REFERENCES users(id));

CREATE TABLE logs (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    album_id INT NOT NULL,
    rating INT,
    review TEXT,
    listened_date DATETIME DEFAULT CURRENT_TIMESTAMP,FOREIGN KEY (user_id) REFERENCES
users(id),FOREIGN KEY (album_id) REFERENCES albums(id));
```

Database SQL Schema (Junction Tables)

```
CREATE TABLE user_follows (
    follower_id INT,
    following_id INT,PRIMARY KEY (follower_id, following_id),FOREIGN KEY
(follower_id) REFERENCES users(id),FOREIGN KEY (following_id) REFERENCES
users(id));

CREATE TABLE album_artists (
    album_id INT,
    artist_id INT,PRIMARY KEY (album_id, artist_id),FOREIGN KEY (album_id)
REFERENCES albums(id),FOREIGN KEY (artist_id) REFERENCES artists(id));

CREATE TABLE album_genres (
    album_id INT,
    genre_id INT,PRIMARY KEY (album_id, genre_id),FOREIGN KEY (album_id)
REFERENCES albums(id),FOREIGN KEY (genre_id) REFERENCES genres(id));

CREATE TABLE list_albums (
    list_id INT,
    album_id INT,
    position INT,PRIMARY KEY (list_id, album_id),FOREIGN KEY (list_id)
REFERENCES lists(id),FOREIGN KEY (album_id) REFERENCES albums(id));
```

Let's Work Together and Create Something Great !

CSE111

LIZBETH PEREZ & JASMINE LAU

DISCOBOXD