Ryerson University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Software Design Architecture
Course Number:	COE692
Semester/Year (e.g.F2016)	W2024

Instructor:	Faezeh Ensan

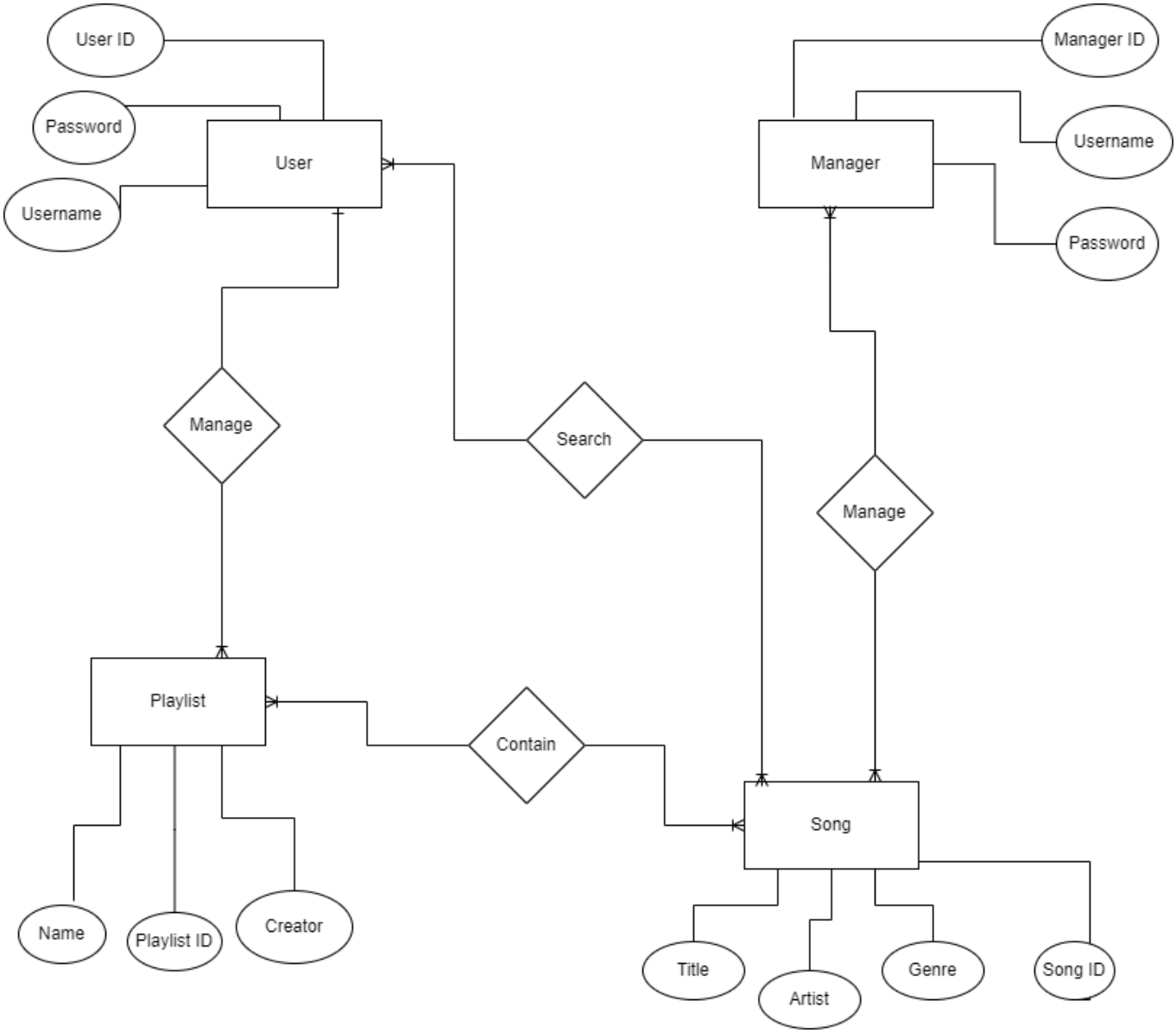
Assignment/Lab Number:	3
Assignment/Lab Title:	

Submission Date:	February 25, 2024
Due Date:	February 25, 2024

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Cao	Andy	501105786	02	A.C
Chiu	Danielle	501116551	02	D.C

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

ER Diagram:



Description

This ER diagram consists of four entities: User, Manager, Playlist, and Song. A user has a username, password, and a unique user ID. Similarly, a manager has a username, password, and a unique manager ID. Playlists have names, creators, and playlist IDs. There is a one-to-many relationship between User and Playlist, meaning that a user can manage many playlists, but each playlist can only be managed by one user. A song has a title, artist, genre, and song ID. There is a many-to-many relationship between playlists and songs; a song can be in multiple playlists, and multiple playlists can contain the same song. There is also a many-to-many Search relationship between User and Song; a song can be searched by multiple users, and a user can search for multiple songs. There is a many-to-many relationship between manager and song, which means that a manager can manage different songs, and a song can be managed by multiple managers.

SQL Table Commands

```sql
CREATE TABLE USER (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    password VARCHAR(250) NOT NULL
);
```

```sql
CREATE TABLE MANAGER (
    manager_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(100) NOT NULL,
    password VARCHAR(250) NOT NULL
);
```

```sql
CREATE TABLE PLAYLIST (
    playlist_id INT AUTO_INCREMENT PRIMARY KEY,
    playlist_name VARCHAR(100) NOT NULL,
    creator_id INT NOT NULL,
    FOREIGN KEY (creator_id) REFERENCES USER(user_id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE SONG (
    song_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    artist VARCHAR(100) NOT NULL,
    genre VARCHAR(50) NOT NULL
);
```

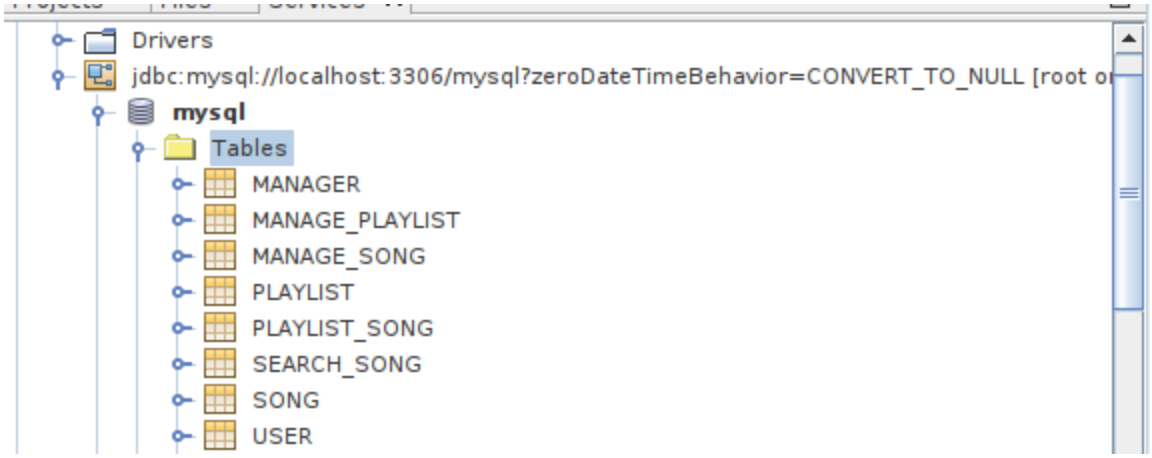
```sql
CREATE TABLE MANAGE_SONG (
    manager_id INT,
    song_id INT,
    PRIMARY KEY (manager_id, song_id),
    FOREIGN KEY (manager_id) REFERENCES MANAGER(manager_id),
    FOREIGN KEY (song_id) REFERENCES SONG(song_id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE MANAGE_PLAYLIST (
    user_id INT,
    playlist_id INT,
    PRIMARY KEY (user_id, playlist_id),
    FOREIGN KEY (user_id) REFERENCES USER(user_id) ON DELETE CASCADE,
    FOREIGN KEY (playlist_id) REFERENCES PLAYLIST(playlist_id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE SEARCH_SONG (
    user_id INT,
    song_id INT,
    searched_datetime DATETIME NOT NULL,
    PRIMARY KEY (user_id, song_id, searched_datetime),
    FOREIGN KEY (user_id) REFERENCES USER(user_id) ON DELETE CASCADE,
    FOREIGN KEY (song_id) REFERENCES SONG(song_id) ON DELETE CASCADE
);
```

```sql
CREATE TABLE PLAYLIST_SONG (
    playlist_id INT,
    song_id INT,
    PRIMARY KEY (playlist_id, song_id),
    FOREIGN KEY (playlist_id) REFERENCES PLAYLIST(playlist_id) ON DELETE CASCADE,
    FOREIGN KEY (song_id) REFERENCES SONG(song_id) ON DELETE CASCADE
);
```

SQL Tables


Drivers
jdbc:mysql://localhost:3306/mysql?zeroDateTimeBehavior=CONVERT_TO_NULL [root o
mysql
Tables
MANAGER
MANAGE_PLAYLIST
MANAGE_SONG
PLAYLIST
PLAYLIST_SONG
SEARCH_SONG
SONG
USER