CS353 DATABASE SYSTEMS

2017-2018 SPRING SEMESTER

MUSICHOLICS

PROJECT PROPOSAL

GROUP 17

Esra Nur AYAZ

Ezgi ÇAKIR

Metehan KAYA

Miraç Vuslat BAŞARAN

CONTENTS

1. Introduction

Musicholics is a web-based music service allowing users to stream and buy music online anywhere anytime. Musicholics aims to be a breath of fresh air to the field by combining music streaming and social life. Musicholics users can befriend each other, post on others' wall, and gift songs to their loved ones. Unlike currently available music services and social media sites, Musicholics brings them together and provides an innovative way to have fun.

This report aims to introduce reader to details of Musicholics. It consists of requirements, limitations, and Entity/Relationship Model of proposed design in the following parts of the report.

2. Description

To use Musicholics, first, a person has to sign up by entering some typical information such as username, password, email address, etc. After the registration, the user can start listening to music. The users can be friends, and follow each other. Between users, it is also possible that a user can block and unblock another user. Moreover, the user can post on a friend's wall.

A user can have a premium account which enables the user to listen to music even if he is offline. Upgrading the account to premium is optional. Also, the user can buy a track or send it to a friend as a gift.

A user has an opportunity to create a new playlist which is composed of tracks. Moreover, the user can follow playlists of some friends without any effort. This saves the user's time from searching music and adding them to a new playlist. The user can also rate and comment playlists. The user can create playlists with the same name.

Musicholics should provide access to tracks, artists, albums, and publishers. Whenever the user listens to music, it is allowed to go to these pages via links.

3. Why Use a Database System?

Musicholics aims to reach millions of people and provide access to millions of songs. That is a huge amount of data to organize especially when billions of prospective relations between them is considered. Storing data in a not data centered system is always an option. However, size of prospective data indicates that Musicholics should be data centered to be able to function with as less erroneous states as possible. Obviously, we were not the first ones coming up with data management system idea. Commonly used and publicly available database management systems ease managing and maintaining large amounts of data. Such database systems provide strong tools to access, manipulate and create data in a fast and reliable way. In addition to those, establishing relationships between entities is simplified with Entity/Relationship Model databases. Therefore, considering all advantages of database systems, it is sensible to use one.

4. Functional Requirements

4.1. Playlists

Playlists can have multiple followers.

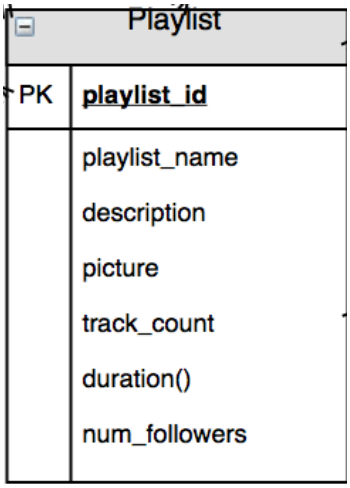
Playlists can have comments.

Playlists can have a picture.

Playlists can have a description.

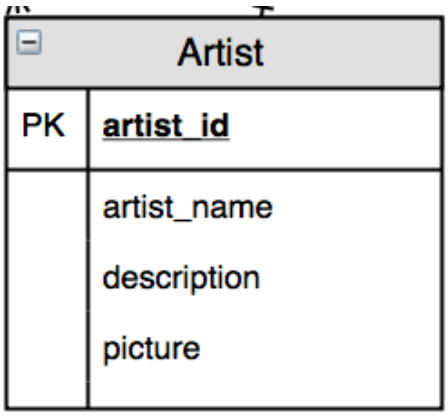
Playlists can have a rating.

Playlists can have multiple tracks from different artists.



Playlists should be able to keep track of when a track was added to them.

4.2. Artists

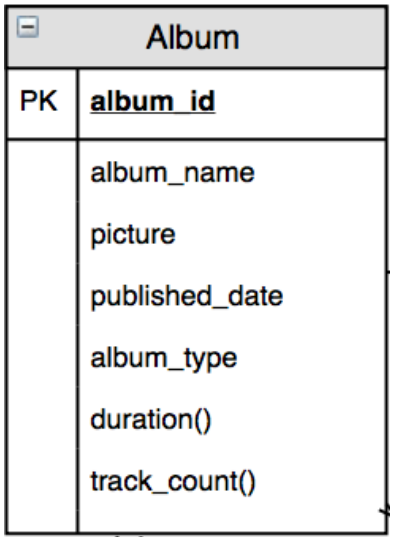


Artists can have multiple tracks.

Artists can have a picture.

Artists can have a description.

4.3. Albums



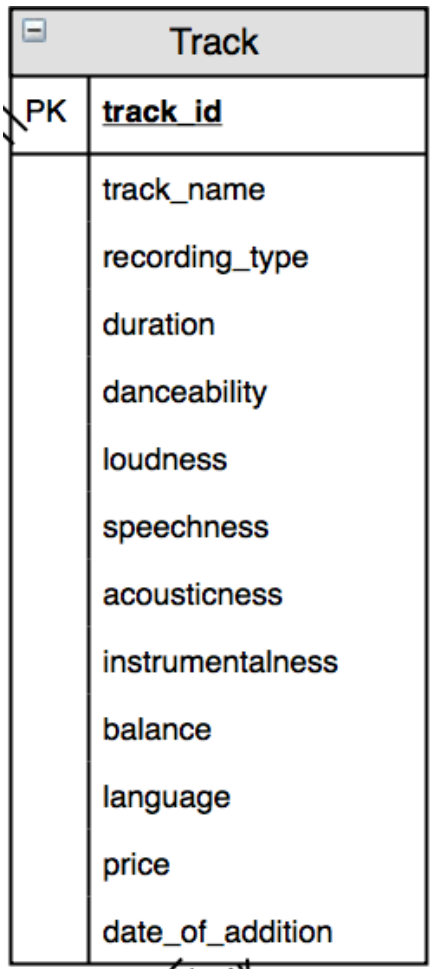
Albums can have multiple artists.

Albums can either be a “Single” or an “Album.”

Albums can have multiple tracks in them.

Albums can have a picture.

4.4. Tracks



Tracks can have varying values for attributes such as danceability, spechness, acousticness, instrumentalness, and balance.

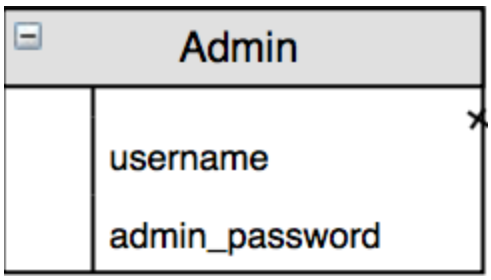
Tracks can have multiple artists.

Tracks can be “live” or “studio recording.”

Tracks can have different prices.

Tracks can have a picture.

4.5. Admins



The admin should be able to add, delete or modify information about tracks.

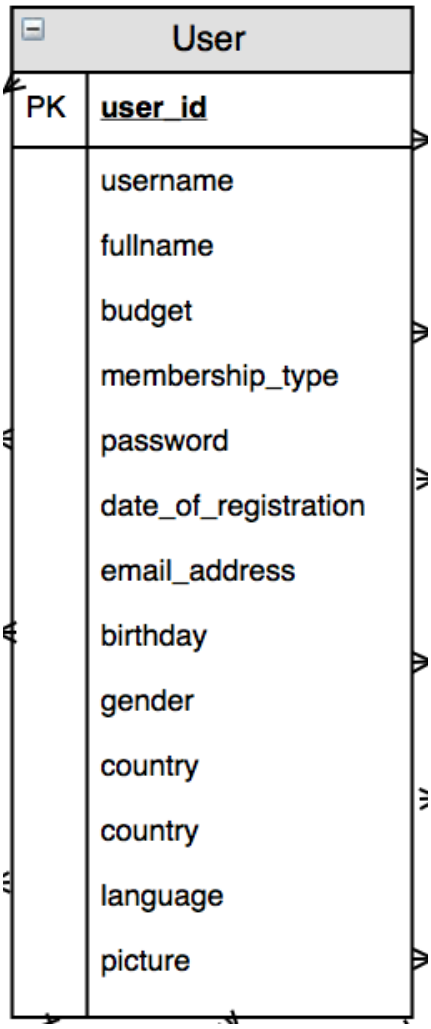
The admin should be able to add, delete or modify information about albums.

The admin should be able to add, delete or modify information about artists.

The admin should be able to add, delete or modify information about publishers.

The admin should be able to ban a user if he posts inappropriate content etc.

4.6. Users



The user should be able to update her profile information.

The user should be able to listen to tracks.

The user should be able to change her membership type

The user should be able to purchase tracks.

The user should be able to create playlists.

The user should be able to to follow playlists.

The user should be able to add tracks to her playlists.

The user should be able to rate other users' playlists.

The user should be able to comment on other users' profile.

The user should be able to comment on other users' playlists.

The user should be able to block other users.

The user should be able to friend other users.

The user should be able to gift a track to her friends.

The user should be able to search for tracks, albums, artists, playlists and other users.

The user should be able to have a picture.

5. Non-Functional Requirements

5.1. Reliability

Musicholics has to be available 24/7 and 365 days except scheduled maintenance which should take minimal time. Also, Musicholics should prevent users from data loss during actions by doing regular backups.

5.2. Security

Musicholics should not allow any security issues. It has to keep users safe, and this can be handled by applying multiple advanced hash functions on some important information like passwords. Payments for tracks must be done in a secure way.

5.3. Usability

Musicholics should have straight-forward and attractive user interface for the sake of easy usage. A user should find whatever he wants with a few clicks. Also, Musicholics should not have any complicated, or unnecessary functionalities.

5.4. Portability

To reach a high number of users, Musicholics needs to work on different operating systems, browsers such as Chrome, Mozilla, etc., and most of the smartphones, tablets, and different operating systems. According to the device, auto-adjust should be handled correctly.

5.5. Performance

For Musicholics, it is important to react fast to inputs from users. Defined operations such as searching, sorting, data modification should run fast. Response time should not be higher than 2 second. Also, big amount of data and high number of active users should not lead to a late response.

5.6. Authentication

Application includes the end-user and administrative interfaces. Given permissions to the users should be specified and prevent them from doing some extra actions that they are not allowed.

5.7. Scalable

Thanks to what Musicholics provides, it will be a popular music service which has to support a high number of users at a time. It should deal with multiple connections.

5.8. Capacity

Musicholics should be able to store big amounts of data including user information, playlists, tracks, albums, artists, publishers, etc. Capacity should be large enough so that registrations, addition of new tracks, etc. can be possible.

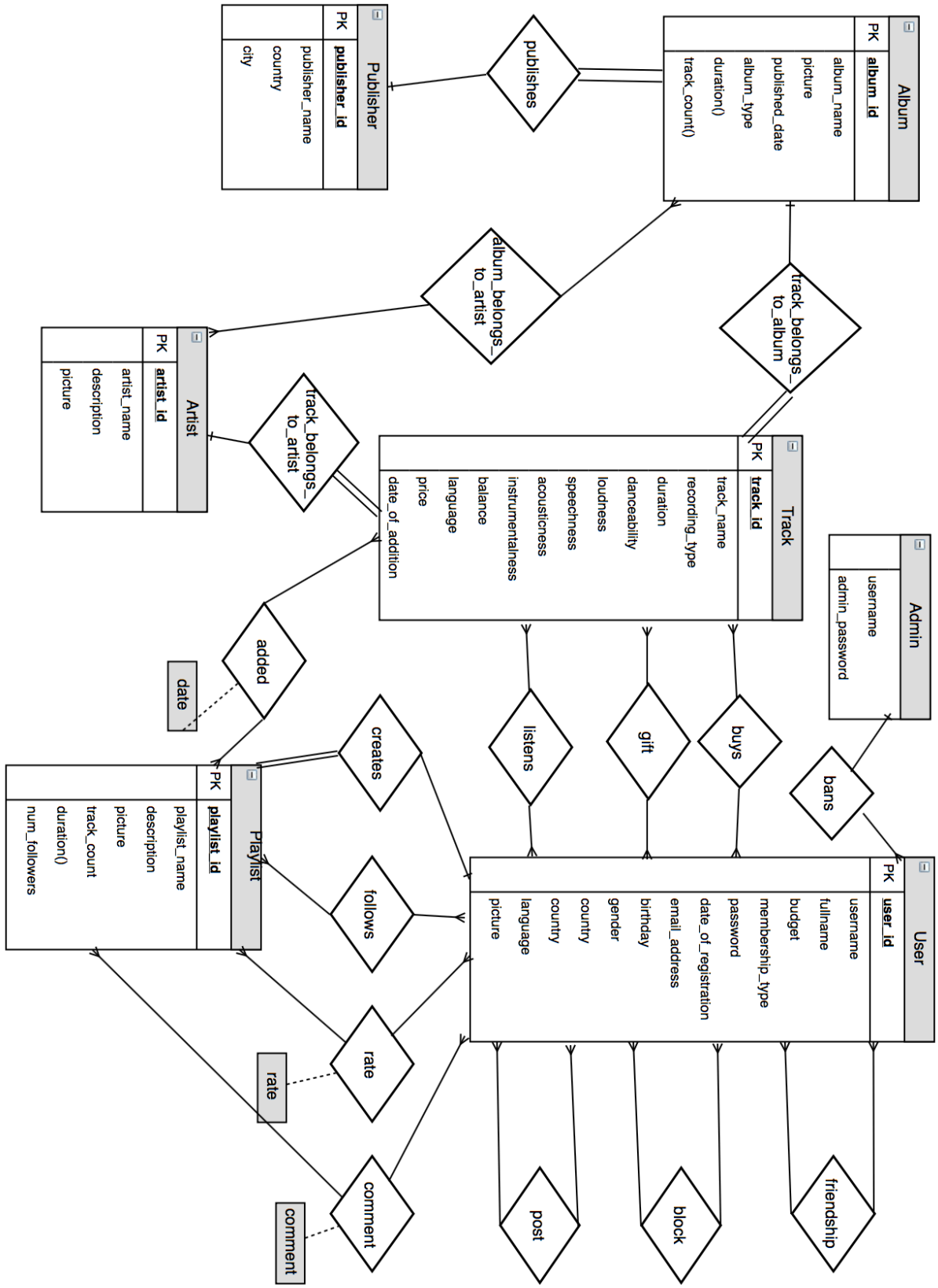
6. Limitations

Users can search for tracks, albums, artists, playlist and other users; friend other users; block other users; listen to tracks; see friend's activities; create playlists and add songs to them; follow playlists; comment on other users and playlists; give ratings to playlists; purchase tracks and update their own profile information. However, they cannot friend themselves, follow their own playlists, add songs to other users' playlists, update other people's profile information or do anything that involves a user that blocked him/her.

Additionally, depending on a user's membership type, i.e. "Premium" or "Free, she might be restricted in which songs she can listen to. This does not include the tracks she has purchased.

Admins are the administrators and moderators of the application. They can add or delete tracks, albums, artists and publishers. They can also modify any information about them. Admins can also ban a user if she posts inappropriate content.

7. Entity-Relationship Diagram



7.1. Track_belongs_to_album

This relation is between album and track entities. There is a 1-to-many relationship and total participation. It represents that a track have to belong to only one album and album may consist of multiple tracks.

7.2. Album_belongs_to_artist

This relation is between album and artist entities. There is many-to-many relationship. It means that an artist can have one or more album. Besides, an album may belong to a single artist or multiple artists.

7.3. Track_belongs_to_artist

This relation is between artist and track entities. There is a 1-to-many relationship and total participation. This means that a track must belong to a artist. Besides, an artist may have multiple tracks.

7.4. Publishes

This relation is between publisher and album entities. . There is a 1-to-many relationship and total participation. An album is published by only single publisher and a publisher may publish multiple albums.

7.5. Added

This relation is between playlist and track entities. There is many-to-many relationship. This relation represents that it is stored when a track is added to a playlist with “date” attribute. Multiple track may be added to multiple playlists.

7.6. Creates

This relation is between user and playlist entities. There is a 1-to-many relationship and total participation. It is interpreted as a playlist must be created by only one user. Additionally, a user may create multiple playlists.

7.7. Follows

This relation is between user and playlist entities. It is interpreted as a playlist can be followed by many users. Additionally, a user may follow multiple playlist.

7.8. Rate

This relation is between user and playlist entities. There is a many-to-many relationship. It is interpreted as following: a playlist can be rated by many users, likewise a user may rate one or more playlists.

7.9. Comment

This relation is between user and playlist entities. There is a many-to-many relationship. It means that a playlist can be commented by many users, likewise a user may make a comment one or more playlists.

7.10. Posts

This relation is between user and user entities. There is a many-to-many relationship. It represents that multiple users are able to post something on multiple users' profiles.

7.11. Blocks

This relation is between user and user entities. There is a many-to-many relationship. It means multiple users are able to block multiple users.

7.12. Friendship

This relation is between user and user entities. There is a many-to-many relationship. It is interpreted that multiple users are able to be friend with multiple users.

7.13. Buys

This relation is between user and track entities. There is a many-to-many relationship. It is interpreted that multiple users are able to buy multiple tracks.

7.14. Gift

This relation is between user and track entities. There is many-to-many relationship. It is interpreted that a user is able to buy a track or multiple tracks in order to give as a gift to another user. Likewise, a track can be bought to be given as gift by many users.

7.15. Listens

This relation is between user and track entities. There is many-to-many relationship. It means that a user is able to listen a track or multiple tracks. Likewise, a track can be listened by one or many users.

7.16. Bans

This relation is between user and admin entities. There is 1-to-many relationship. It means that an admin is able to ban a single user or multiple users. A user can be banned only one admin.

8. Online Access

Musicholics is available online at github.com/miracvbasaran/Musicocholics