Bilkent University

Department of Computer Engineering

Database Systems Project

Project Name: Musify

Proposal Report

Berfu Deniz Kara	21201662
Burak Alaydın	21603427
Sarp Tekin	21600813
Vural Doğan Akoğlu	21602479

Instructor: Özgür Ulusoy

Teaching Assistant: Arif Usta

Proposal Report, Oct 14, 2020

Table of Contents

1. Introduction

This is the proposal report for the CS 353 Database Systems term project. This project is designed to create an audio streaming platform for the artists and users. This proposal report details the project description, functional and non-functional requirements, limitations and entity-relationship model for the database system. Progress of the project can be followed through this website: https://sarptekin.github.io/Cs-353/

2. Project Description

This project is a web-based music track platform for people who want to have. There are users, artists, songs, playlists, and albums. Artists can release their songs and albums on the app with their price. Users have to buy the songs and albums to play them. Users can search for songs or albums with a specific genre that they like. Users can also comment and like the songs and albums. In addition, users can add friends and share the playlists that they create with their friends. A user can see his/her friends' actions and give them a song or album.

2.1 Why a Database System Is Needed

Songs and user information such as friends, genres, likes and comments contain a vast amount of information that our app should handle. Information about these components are not just static. They are also changed and updated in time (new songs can be bought by the user which means new comments and new likes). In addition, there are actions to be recorded such as which song user listens. It is remarkably difficult to store and update such a huge amount of information without an automated system. Therefore, we use a database to handle this difficulty and regulate certain actions that users perform.

2.2 How a Database System Is Going to Be Used

The database is going to provide structure to manage all data related to all user activities between songs and users. We are going to use the database management system so as to search for the relevant data regarding

users' demands by way of queries. We are going to update the database considering different releases from artists such as songs and albums by creating new entries and we will update the system for friendship changes, likes, and comments. The database is going to provide data about the most liked or the most purchased songs and albums. Thus, users can find the most popular songs, or the ones that are their choice of genre. Additionally, we are going to use the database triggers to relate distinct elements to each other to provide a fluid and functional system.

3. Requirements

3.1 Functional Requirements

3.1.1 User Requirements

- Users will have username, genre preferences and account balance.
- Users will be able to search songs and albums.
- Users will be able to buy songs and albums.
- Users can create multiple playlists and categorize them.
- Users can play songs individually or in a playlist.
- Users will be able to send friend requests to other users.
- Users will be able to see the activities of other users which are on their friend list.
- Users can leave feedback on songs and albums.
- Users will be able to report other users.

3.1.2 Artist Requirements

- Artists will have all the features a regular user has and an artist name.
- Artists will have songs and albums.
- Artists will be able to add songs and albums.
- Artists will determine the price for songs and albums.

- Artists will be able to feature their songs and albums.

3.1.3 Song Requirements

- Songs will have a unique song-id, name, length, price, genre, release date, comments, album and artist name.

3.1.4 Album Requirements

- Albums will have a unique album-id, name, songs, genre, release date, total number of songs and comments.

3.1.5 Playlist Requirements

- Playlists will have a unique playlist-id, name and genre.

3.1.6 Activity Requirements

- Activities will have activity-id to distinguish them.

3.2 Non-Functional Requirements

3.2.1 Performance

- Process of buying songs and adding them to the playlists must be done in under 3 seconds.
- “Play” command for songs, albums or playlists must respond in under 2 second.
- The server should be able to handle multiple online users. Interacting with the app should be fast for a better user experience.

3.2.2 Reliability

- Users must always be able to login to the system with matching id and password.
- Transaction process (buying songs or albums) must always work as expected.
- A user must always be able to play the songs which s/he has.

- Users must be able to change their nicknames but not their usernames.
- Regular users must not be able to add songs.
- Artists must not be able to add the exact same song which already exists in the system.

3.2.3 Maintainability

- Each feature should be divided into smaller parts as much as possible.
- A malfunctioning feature should not block other irrelevant features/processes.
- The project should be well documented and commented on.

3.2.4 User Friendly Interface

- Sign-up and sign-in page should be simple.
- Four main features, owned songs, playlists, friend lists and search button, should be easily located by a user.

3.3 Pseudo Requirements (Constraints)

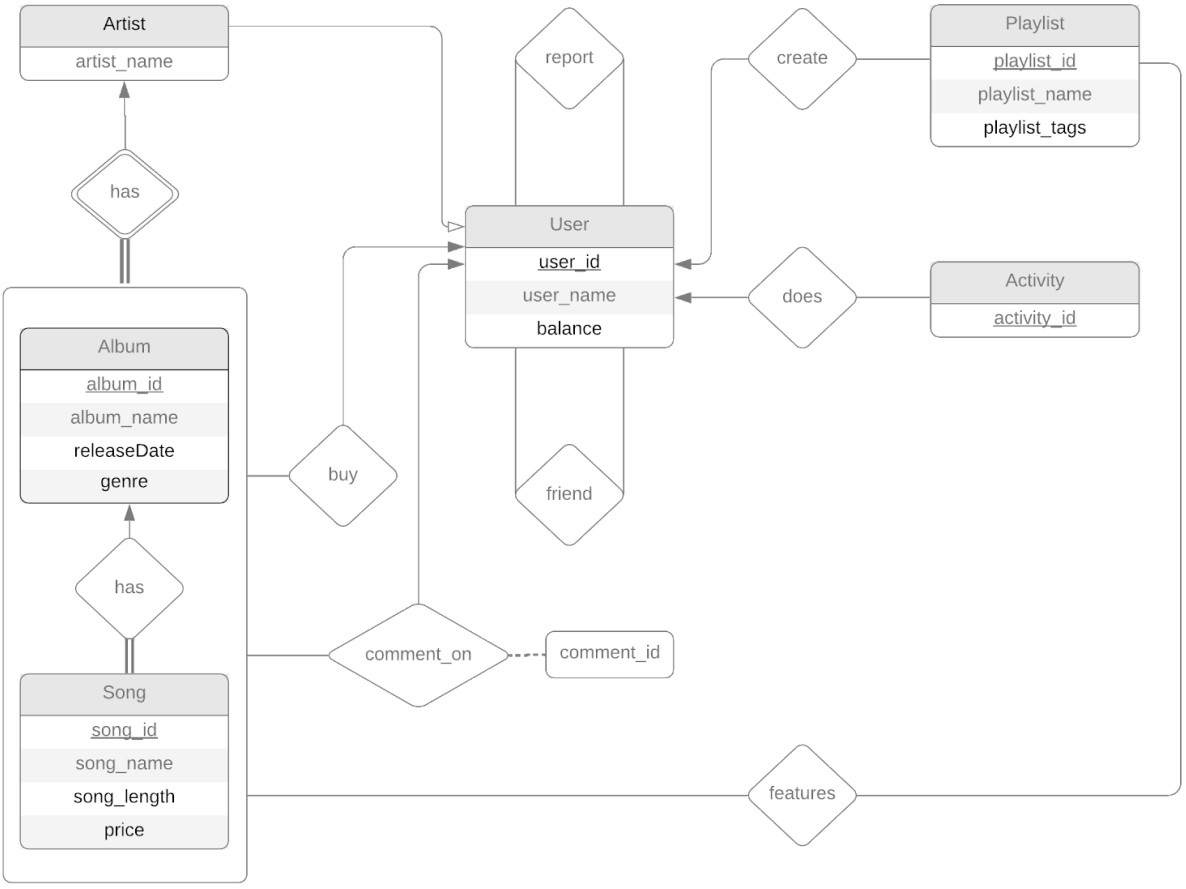
- We will use MySQL as Structured Query Language.
- We will use PHP for back-end development.
- HTML, CSS and Javascript will be used for our website.
- We plan to use AJAX to interactively communicate between webpage and database.

4. Limitations

- Regular users can sign-up with a username and password.
- In addition to that, artists should sign-up with their real names or the names they are known within the industry.
- The password must have at least 5 characters which consist of at least one letter and one number.
- Users can only buy songs if they have enough money on their account.

- A user can create only 20 playlists and have 100 friends at most.
- A user can only see other users' status if they are in his/her friend list.
- Users who have not bought a song or an album cannot comment on that particular song or album.
- Only an artist account can add songs and albums to the system.
- Artists can feature a particular song or an album for the determined price.

5. Entity - Relationship Model



6. Conclusion

This is a music platform like Spotify. Users will be able to play any song they own and they do not have to pay monthly subscriptions. They can buy songs and

entire albums. The system allows artist accounts to add songs and albums. In general, this system consists of the interactions between users, artists, albums and songs.

The report has explained what the project topic is, why and how a database system is involved in it. It has functional, non-functional requirements and limitations. In the E/R diagram, we have shown the relationships between the entity sets which have primary keys and other attributes. Below, you can find the link to the website of the project for additional information.

Website : https://sarptekin.github.io/CS-353_Project/