Ihsan Dogramaci Bilkent University

Spring 2018

CS353 - Database Systems

# Term Project Proposal Report

Group 15

Hakan Sarp Aydemir - 21501331

Umut Berk Bilgiç - 21502757

Doruk Çakmakçı - 21502293

Oğuz Göçmen - 21503128

1. 26.02.2018

## Table of Contents

## 1. Introduction

This report is the project proposal of a "Music Playing Database System" which explains primitive functionalities of the given project. The report contains project description in brief, requirements and limitations of the project, as well as the E/R Diagram of the database.

The project description states the aim of the project and clarifies some points about some crucial points and answers to questions such as "Why do we need a Database for a Music Playing Database System?" and "How do we use a Database as a Part of the Project?".

Report continues with "requirements" section which includes functional, non-functional and pseudo-functional requirements. Functional requirements are established by thinking the needs for the end-user. Non-functional requirements include security, performance, reliability and usability goals for our database system. Pseudo-functional requirements include technologies that will be used in the project. After these, "limitations" part comes up that clarifies some constraints and boundaries for the system.

After the "requirements" and "limitations", we provide the E/R Diagram which is the basis of our database. This diagram includes entities, their attributes and relationships with each other. E/R Diagram is generated by considering the requirements for the project.

## 2. Project Description

Sudo is a web-based application for publishing and reaching to people with music as well as listening to music and commenting about it, similar to "Spotify".

The system will contain several entities such as artists, tracks, albums, playlists and users who will be able to create their own playlists, buy songs, comment and share the tracks they listen to. Users will be able to search tracks, albums and artists and find the ones that are best suitable to their music taste.

Artists will be able to publish their brand-new tracks and share information about these tracks as well as their own biography. This published product can be either a single or an album.

### Why do we need a Database for a Music Playing Database System?

This application is in a sense a social platform which is built with the purpose of sharing music with people all around the world. To achieve this purpose, one needs a place that is enough to store all the information of existing accounts, keep track of accounts that is registered to system
and all their public and private information such as their biographies or passwords etc. Moreover, since users will be able to comment on many tracks and share them with their friends, existing data must be clear enough to be manipulated. To control all these vast information, the need for a database becomes more necessary. Apart from storing all the

accounts in the same place, published tracks must also be in this database to reach it whenever wanted; users should be able to listen to their favourite music pieces and share them with friends. To achieve all the desired needs, the database should be connected enough to provide these services to the users of the system.

### How do we use Database as a part of the project?

As mentioned, using a database for such an application is crucial since it will contain a lot of information about every user's account as well as information about published tracks, albums and artists. The database will be constantly updated since there will be many artists in system; hence, there will be many tracks that will be published even for one artist. Users will be able to search artists and tracks; thus, these tracks will be stored in a portion of database and will be accessible all the time when a user searches for it. Users will be able to create their own playlists; in order for the playlists to access tracks, users' accounts must be in touch with tracks, i.e. there will be relationships between tracks and users. The way to do it is that each user will have a library which contains every playlist they create that includes tracks.

For user accounts, utilizing a database is also important. Many people will want to use this application; then a database must be used to enable many concurrent account registrations. With these registrations, we need to be careful about activities that each account is involved in. Each account will be able to "befriend" another account so there must exist a relationship between accounts. In order to keep track of the activities of the user such as determining which album is bought or which comments are made to which playlists should be inspected closely; to achieve this, accounts must also be in a relationship with playlists. Moreover, in order to accommodate a change in users' balance or update/change in account details; another portion of database must include account information to provide the needs of the user.

Holding all these data in a database will enable better user experience; making the application faster and more reliable. Relating accounts with tracks is going to be closely inspected during design and implementation process. Additionally, without any solid database system, the application's search functionalities (or queries) would be inefficient and the scalability of the application would be hindered, i.e. the application would crumble under heavy load.

# 3. Requirements

## 3.1 Functional Requirements

Our database system will include two main end-user types which are User and Artist. Artist entity will inherit all of the attributes of User entity however, it will also have listener_count, name and bio. Related functional requirements can be found below:

- **User**
    - ➢ All users have to authenticate their accounts when logging into the application.
    - ➢ Users will be able to search for tracks, albums, artists and users.
    - ➢ Users will be able to add each other as friends.
    - ➢ Users will be able to create their own playlists.
    - ➢ Users will be able to comment and share about playlists of other users or artists.
    - ➢ Users will be able to update their profiles.
    - ➢ Users will be able to remove their account from system.
    - ➢ Users will have individual libraries to store their playlists and tracks.
    - ➢ Users will have wallets; they can load money and use it for buying tracks or albums later. Also, transaction transcripts will be generated for each cash flow. Each transaction will have a unique identifier.

- **Artist**
    - ➢ Artists will be able to publish albums.
    - ➢ Artist will have total listener counts for each of their songs.
    - ➢ Artists will have bios in their profile.

## 3.2 Non-Functional Requirements

- **User-Friendliness:** System should provide a user-friendly user interface in order to achieve more desirable application.

- **Reliability:** To provide a better service, no data loss must happen. Users must be able to find what they seek in every case; if data is lost for an account, since everything will be dependent on it, most information will be lost for a user. System must prevent this type of situations and must be reliable.

- **Capacity:** Since there will be a lot of data to maintain within system, storage capacity of database must provide enough pace for application.

- **Scalability:** The application must feel responsive regardless the amount of load the database is under at any given moment. This is achieved by making sure that the query systems are efficient and to-the-point.

## 3.3 Pseudo-Functional Requirements

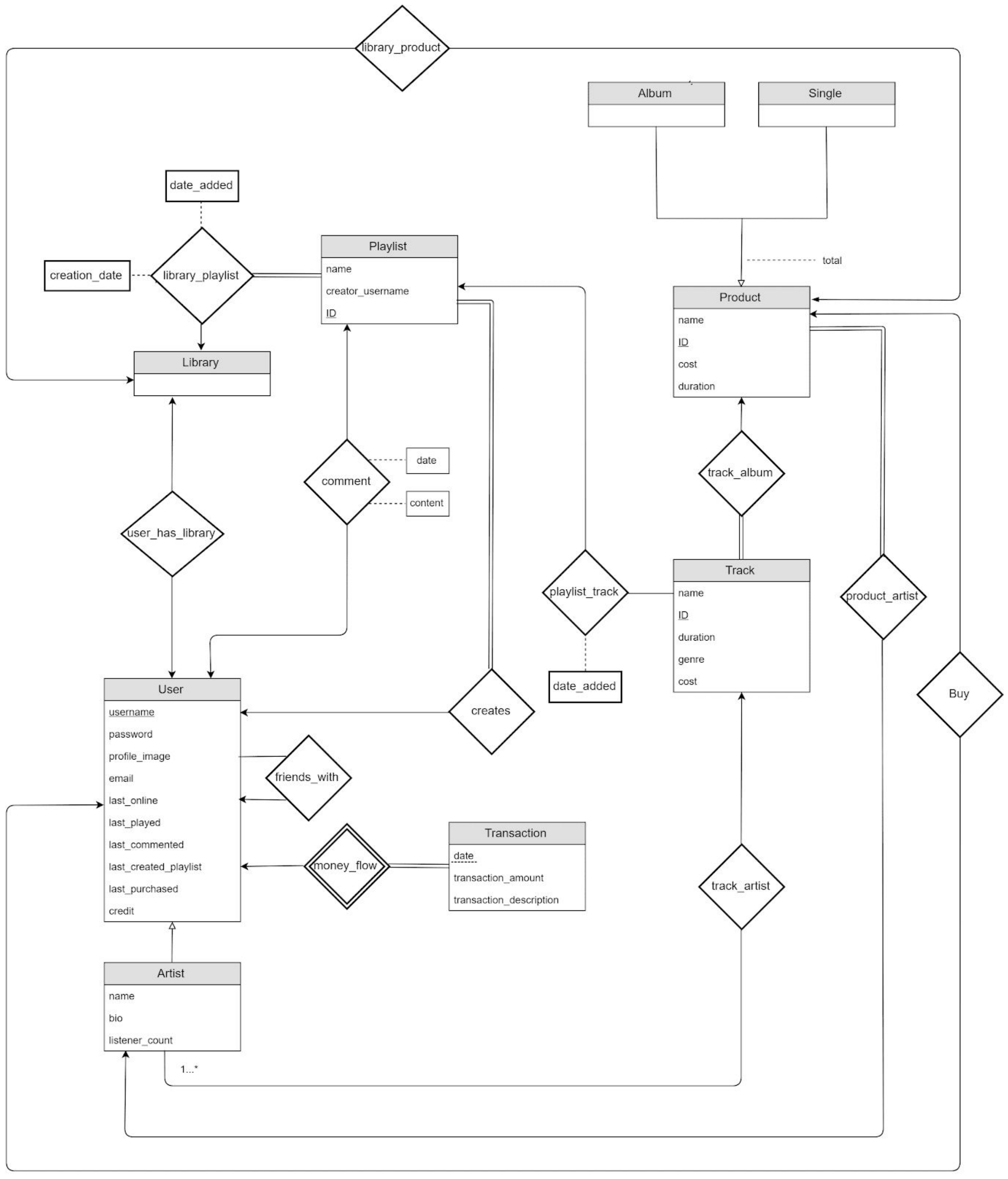
- MySQL will be used for database.
- HTML, CSS, JavaScript, JQuery, Bootstrap will be used for front-end development of website.
- Java and some Python for custom scripting purposes will be used for back-end development of website.

## 4. Limitations

- Customers will not be able to listen to certain tracks without buying them. A short preview could be included for a selection of tracks.
- If a customer does not have enough money in his/her wallet, he/she will not be able to buy a track or album.
- To be in an artist status, one must contact with system administrator to achieve this and validate his/her account. They must provide a product to be displayed on their profile.
- The comments will have a set amount of characters as an upper limit.
- Username and password will have set rules in order to prevent abusive usernames and very unsafe passwords.
- Users will have to validate their accounts via their e-mail if they want to make a purchase.

## 5. E/R Diagram

The E/R Diagram for the database of this project is below:



## 6. Project Webpage

Our project’s source code and anything else regarding to the project will be hosted on a public GitHub repository. The webpage (at umutberkbilgic.github.io/Sudo-Music/) is hosted by GitHub using the GitHub Pages service.