Bilkent University

Department of Computer Engineering

CS353 Course Project

TunedIn

Project Proposal

Furkan Kazım Akkurt, Cemal Arda Kızılkaya, Khasmamad Shabanovi, Mehmet Bora Kurucu

Instructor: Özgür Ulusoy
Teaching Assistant(s): Arif Usta

October 14, 2020

1. Introduction

This document is the project proposal for our Music Track Data Management System project, TunedIn. The document starts with the overall description of the project, which contains general information about the project. This section also explains how and why a database system will be used as a part of the project. The document continues with the requirements of the project, where the functional requirements, which define the scope and functionalities of the system are explained, and non-functional requirements, where the criterion that the system should satisfy, are explained. Then the limitations of the system are discussed in terms of constraints and boundaries. Finally, the E/R diagram of the overall system is presented.

2. Project Description

TunedIn is a web application which is designed to be used by two different types of users: artists and general users. Each user will be able to follow other users and see what kind of music they are listening to. The users will be able to purchase songs in an album individually or an entire album and build playlists with the songs they have purchased. The playlists can be organized by genres of songs in them. Artists will be able to create albums and songs, and determine the prices for them. The users will be able to provide feedback on the songs and albums they have purchased, which consists of a score,a title and a description. Users can view and comment on the blog posts posted by the users they follow, which increases the interaction between the users.

2.1. Why Is a Database System Needed?

A music collection management application contains a wide variety of information related to albums, songs, artists, etc. First of all, a single song has a handful of attributes associated with it, such as name, artists, duration, genre, price, album that

it belongs to, and so on. Also, just like a song, an album has many attributes associated with it such as release date, price, feedback, etc. In addition to that, user and playlist information must be stored. In order to meet the data demands of this application, we need an easy and fast way of storing and accessing data. This makes using a database a much more compelling choice over other alternatives, such as a file management system which would cause data inconsistency and redundancy given the scale of our application.

2.2. How a Database System Will Be Used?

The database that is going to be used in this project will benefit keeping track of all the data (such as songs, albums, playlist, and etc.) needed to provide users a more desirable and efficient application. It will be used with a variety of queries depending on what information is needed. For instance, when an artist adds their songs to the application, the database will be updated accordingly to add a new entry to the table which holds records of the songs. Similarly, when a user buys a song or an album, the relationship table with the user table and song or album tables will be updated accordingly. Database triggers can also be used if needed in order to provide a more useful database system.

3. Requirements

3.1. Functional Requirements

The database system will consist of different main entities and the relationship entities between those main ones. Those will be explained in detail in the following part of the report.

3.1.1. User

Users are the artists or the normal users who do not use the application to upload songs.

- Users can sign up with a unique username, password, and email.
- Both types of users can purchase songs and albums.
- Both types of users will be able to create playlists with the songs they have purchased.
- Users can follow other users, including both types of users.
- Both types of users can buy songs in order to listen to them.
- Artists can upload their songs or albums to the system and determine their prices.

3.1.2. Music Object

Music objects are songs and albums together. Music and Album entities will inherit from Music Object, which will make the system more efficient. They will have attributes such as duration, name, and price. Each music object will have at least one artist, but can also have as many artists as required.

- A music object can be played by a user only if it is bought by the user.
- A music object can be in different playlists in the form of songs.
- A music object can be evaluated by the users, thus they will have evaluation scores.

3.1.3. Playlist

Playlists consist of a subset of the songs a user has purchased.

- Both types of users can create playlists from the songs they have purchased.
- Playlists can be categorized by genres. The genres of a playlist is determined by the user who has created it.

3.1.4. Blog Post

The application will be more interactive with blog posts where users can interact with each other easily. Blog posts are created by the users.

- Each blog post will have a unique id.
- Each blog post will have a title that gives an insight about the post.
- Then there is a description of the post, in which users can see the whole post.
- Each blog post is authored by a single user.
- Users will be able comment on the posts of the users they follow.

3.1.5. Feedback

Users will be able to give feedback to either songs or albums. The feedback will have a title, a description, and a score. All given scores by the users will affect the overall score of a music object.

3.1.6. Genre

All music objects, artists, and playlists can have multiple genres which can be used to categorize them across the application.

3.2. Non-functional Requirements

3.2.1. Resilience

For most of the time, the system should not fail and work as intended. If failures occur, they should be handled as quickly as possible without affecting the user experience too much.

3.2.2. Performance

The system must be as quick as possible at all times. Since we have to manage a huge amount of data and most of the operations of our application has to do with the data in some way, operations involving data must be optimized.

3.2.3. Extensibility

The system must be designed in a way that adding new features to the existing system must be easy.

3.2.4. Usability

The system must be easy to use and the users must have a good experience while using TunedIn.

3.2.5. Data Integrity

The data related to the application must be consistent across the database and must represent the actions performed by the users.

3.2.6. Security

The system must be protected against theft of data by third parties and disruption of any of the system services.

3.2.7. Scalability

The system must be capable of serving the growing amount of users as the application gains momentum in the community.

3.3. Pseudo Requirements

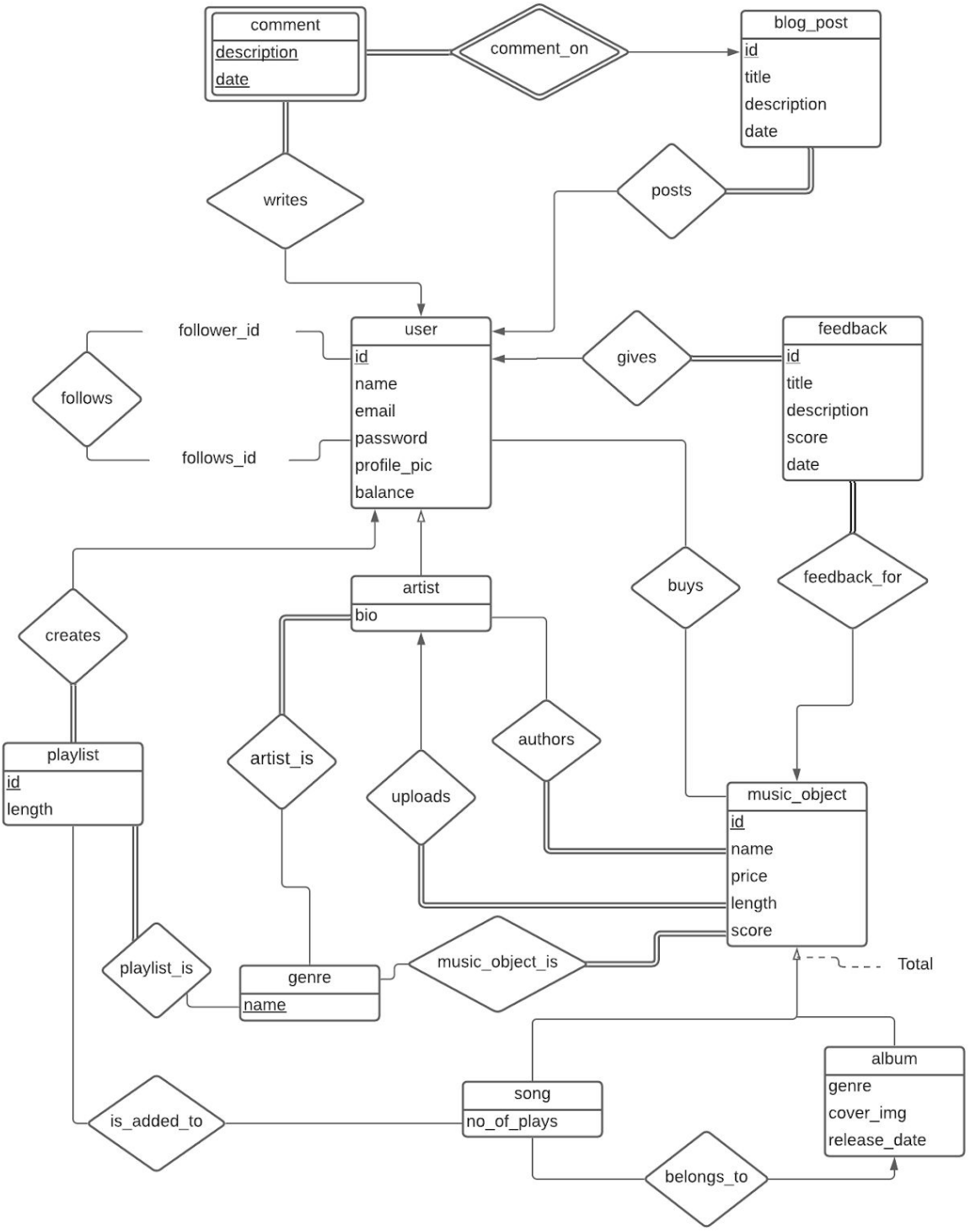
- MySQL will be used as the database management system.
- Backend development will be handled with PHP.
- Frontend development will be handled with HTML, CSS, JavaScript and their useful frameworks such as Bootstrap.

4. Limitations

- Two types of users will be allowed to register, which are default users, who use the application to listen to music, and artists, who will also have the ability to add their albums or single songs to the system.
- Each music object entity should be either a song or an album.

- Users can only play the songs or playlists which they have already bought.
- Users will be able interact with other users after following them.
- Each music object, album or song, should belong to at least one artist.
- Each playlist will belong to at most one user, however, other users can listen to them.
- Users will be able to buy music objects only if they can afford to buy them.
- Users' balance will determine whether they can buy a music object or not.
- Users will be able to see and comment on the blog posts from other users that they follow.

5. E/R Diagram[1]



[1] Note that description and date are partial keys for the comment entity. We opted for a solid underline, since the software we used did not support addition of a dashed underline.

6. Conclusion

TunedIn is a web application which provides users an environment to listen to music, follow their favorite artists and interact with their friends. In this report the aim, significance, and the importance of the application are explained. In addition to that, the significance of having a database management system as a part of the project is mentioned. The requirements are classified as Functional, Nonfunctional, and Pseudo Requirements. In each section detailed information is given. Limitations part is the part where the boundaries of the application are mentioned. Lastly, the E/R diagram is shown to provide a brief but informative look of the application.

7. Website

The report is available at https://kizilkayaarda.github.io/CS353/.