CmpE321 - Project 3

Selen Parlar (selen.parlar@boun.edu.tr)

Deadline: 4 June, 2020, Thursday, 23:55

1 Project Description

In this project, you are supposed to implement a simple song management platform, called *DBtify*, with a web-based user interface. There will be listeners, songs, albums, and artists in the platform. These entities will have the following properties:

- **Listener:** Username and e-mail. They are both unique which means there exists only one listener with a specific username and e-mail address.
- **Artist:** Name and surname. They are not necessarily unique independently but you can assume there exists only one artist with a name and surname couple.
- **Album:** ID, genre, and title. By definition, each ID is unique. Each album must contain at least one song.
- **Song:** ID and title. By definition, each ID is unique. Each song must reside in only one album. Each song may be produced by one or multiple artists.

Two types of people will be using *DBtify*: Listeners and artists. You do not have to implement an authentication mechanism. You are allowed to provide two options for being a listener or an artist. The person can choose what he/she is and cannot perform other type of person's operations.

2 Requirements

Your UI must support the following operations:

- Artists shall be able to add/update/delete albums.
- Artists shall be able to add/update/delete songs in the albums.
- Listeners shall be able to separately view all songs, albums, and artists in *DBtify*.
- Listeners shall be able to view all songs and albums of an artist.
- Listeners shall be able to view all songs of an album.
- Listeners shall be able to like songs and albums.

- Listeners shall be able to view other listeners' liked songs as well as his/her liked songs.
- Listeners shall be able to view popular songs (according to number of likes) of an artist.
- Listeners shall be able to rank all artists by the total number of likes of their songs.
- Listeners shall be able to view songs of a specific genre.
- Listeners shall be able to search a keyword and view the songs that contain this keyword in their titles.
- Listeners shall be able to view the artists who produced a song together. This must be implemented as a **stored procedure**. Parameters of this procedure are the artist's name and surname.
- The system shall have three **triggers**:
  1. When an album is deleted, all the songs in this album must also be deleted.
  2. When a song is deleted, it must be removed from listeners' likes.
  3. When a listener likes an album, all the songs of this album must also be liked by that listener.

3 Notes

- The quality of the web interface does not matter. So, you don't need to style it. The functionality of the system will be evaluated.
- The allowed languages are PHP, Java, JavaScript, and Python. You can use a framework, however, you must write the SQL queries and boot the database server yourself. Note that you should set up the database and create the tables on your own. You are not allowed to use any tool that helps with these parts.
- You are not expected to deploy your system. So, it is fine for it to work on your local.
- There is no restriction for database choice, you can use any **relational** databases. Non-relational databases will not be evaluated at all.

4 Report & Grading

Submissions will be through Moodle. The submission must include your **code** (∼%80) and **ER diagram(s)** (∼%20) describing your system. Your diagram(s) should follow the conventions described in the course material. Otherwise, you should explain clearly what shape corresponds to which concept. The system will be evaluated during a demo session that will be arranged. Demo day(s) will be announced.