Sebastian Zimmeck

COMP 333: Software Engineering, Spring 2022
Homework 2: Backend (PHP, SQL, Django)

Homeworks 2 – 4 will cover an ongoing app development project. A crucial part is to work together as group. It is recommended that you use GitHub to organize your work and keep track of the evolution of your codebase.

Check out the code examples, tutorials, and slides on the class website. They are intended to get you started and illustrate core ideas covered in the homeworks.

Problem 1 [35 Points]

If you have not yet done so, install a local web server on your computer with XAMPP. For install instructions see the relational-databases-tutorial on the class website.

(a) After finishing your landing page, you get to work on creating your music rating web app. Using phpMyAdmin create a new database, `music-db`, for your music platform with the three tables shown below: `users`, `ratings`, and `artists`. Include the sample data given below.

`users` table (primary key: `username`)

username	password
Amelia-Earhart	Youaom139&yu7
Otto	StarWars2*

`ratings` table (primary key: `id`)

id	username	song	rating
1	Amelia-Earhart	Freeway	3
2	Amelia-Earhart	Days of Wine and Roses	4
3	Otto	Days of Wine and Roses	5
4	Amelia-Earhart	These Walls	4

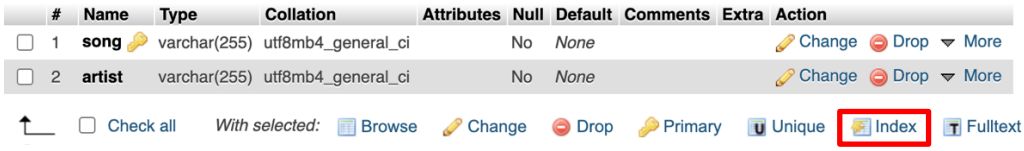
`artists` table (primary key: song)

song	artist
Freeway	Aimee Mann
Days of Wine and Roses	Bill Evans
These Walls	Kendrick Lamar

You can assume that `usernames` and `songs` are unique. In other words, no two `songs` or `usernames` are the same.

The id attribute in the `ratings` table is just an integer that is consecutively increased by one as a tuple (row) is being added.

You can use the `varchar(255)` type for `username`, `password`, `song`, and `artist`. You can use the `int(1)` type for `id` and `rating`.



There are different ways to add keys. Especially, you can use `Index` to add a foreign key. If you create keys in a way that is not based on SQL queries but, for example, via phpMyAdmin settings, please tell us exactly what you did so that we can recreate your database on our end.

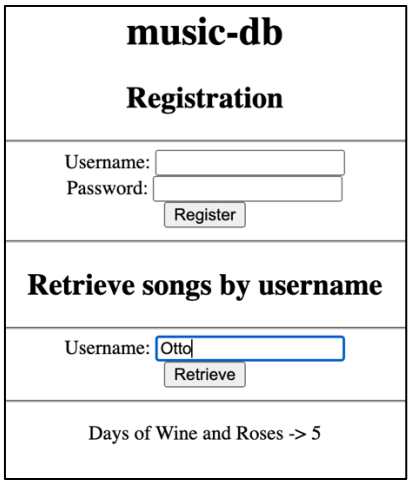
Your database must remain consistent. For example, note that `username` is a foreign key in the `ratings` table. This means that the `username` attribute cannot be deleted in the `ratings` table unless it would be deleted first in the `users` table, where `username` is a primary key. Here is one way of dealing with this situation. If you delete an attribute in one table that is a key in another, i.e., a foreign key, you can set the foreign key attribute so that `on_delete=models.CASCADE`. If you opt for a different solution, please explain.

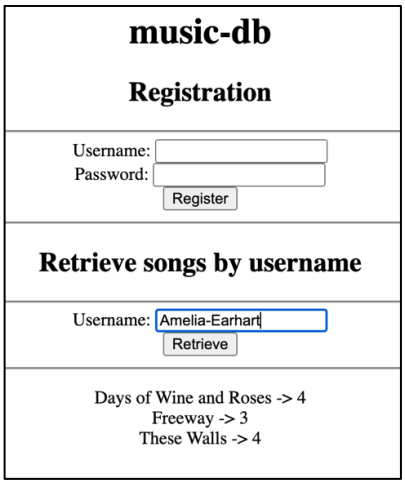
We are recreating your database on our end. The easier you can make it for us the better. If we have difficulty recreating your database, there will be a delay in grading and possibly a point deduction, e.g., for incomplete documentation. Please make sure to test your queries and documentation before you submit your work.

What to submit: Please submit all the SQL queries to create the music-db database with the tables and data shown in the tables in a readme-sql.md file with instructions how to create your database. You do not need to provide any data beyond the data shown in the tables here. We should be able to reproduce your database on our end locally by just running the queries and any instructions you provide.

Sidenote: In a real database passwords should not be stored in cleartext but rather hashed and salted for security reasons (https://www.okta.com/blog/2019/03/what-are-salted-passwords-and-password-hashing/). But for our purposes it is OK to store passwords in cleartext.

(b) Write an index.php script (and possibly other scripts, if any) using HTML, CSS, SQL, and PHP that retrieves data from the `music-db` database that you created per 1(a) via a graphical user interface using HTML forms. Here are some screenshots of results retrieved for the users Otto and Amelia-Earhart.





In particular, your index.php should enable the following functionality:

- **Registration**: It should allow a new user to register with a username and password. The submitted username and password should be written to the users table of your `music-db` database. If a user enters a username that is already taken, the user should be alerted accordingly and asked to pick a different username. Passwords do not need to be unique.
- **Song retrieval**: Upon entering a username, all songs that the user has rated, including the user's rating should be returned. Songs that the user did not rate should not be returned.

  Note, you do not need to populate the database with songs through an HTML form; you can assume it is already populated (i.e., you have put in the data via phpMyAdmin per 1(a)).
- **CSS styling**: Style your site with CSS. You can do it in any way you like. The screenshots are just examples and you do not need to use the same style.

What to submit: Please submit your index.php file (and any other files that you created for this part of the homework).

Problem 2 [65 Points]

After you tried out the classic LAMP stack, you are not satisfied with the results and decide to switch to the modern Django web framework.

If you have not done so yet, set up Django on your computer as discussed in class. It is recommended to use a Python virtual environment. Start a Django project and create a Django app. For instructions see the django-tutorial and Django slides on the class website.

(a) Implement the database schema described per 1(a) and the form under 1(b) with Django in Python as a web app. For this task you need to use models, URLs, views, and templates to create a user form.

(b) Extend the web app that you created per 2(a) with at least one additional table and at least a total of three additional attributes that you use in one or more of your tables. For example, you could add attributes for albums, music genre, year of recording, lyrics, artist biography … . Think about the design of your database and discuss the options in your group. Based on the new table and attributes you added, create at least one additional piece of information a user could retrieve via the form, e.g., upon entering the name of an artist all songs from that artist are retrieved across all users.

What to submit: all files for your Django web app and a readme-django.md with instructions how to run it.

Submission

Please upload all your files in one zip folder to Moodle. You only need to submit one solution per group. It does not matter which student's Moodle account your group is using. The homework is due on **3/4/2021, 10 am**.

Grading

Please try to distribute the work evenly and indicate who did what in the readme(s) you provide. Unless you tell us otherwise, we assume that all students in a group contributed equally and, thus, the grade of each student will be equal as well. It is not possible to challenge a grade after grading based on one student having contributed more than another. If a student contributes less than their equal share, the student's grade will be reduced accordingly. Similarly, if a student contributes more than their equal share, the student's grade will be increased accordingly up to the maximum available grade.