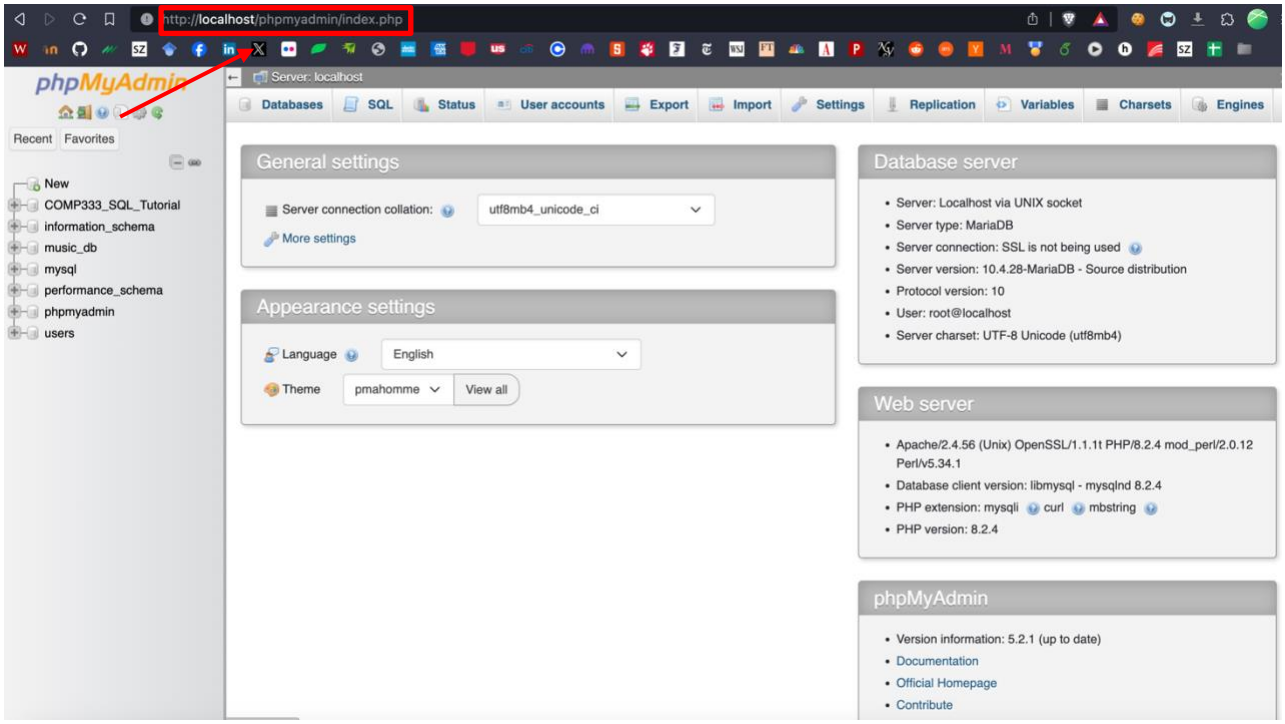
Sebastian Zimmeck

COMP 333: Software Engineering 2025
Homework 2: Backend (LAMP, CRUD, APIs, PHP, MySQL)

Learning goals: In this homework you will learn:
(1) designing and implementing a relational database scheme using the MySQL database language,
(2) based on the LAMP stack, designing and implementing a CRUD web app gluing a MySQL database to an HTML frontend via the scripting language PHP, and
(3) setting up development and production environments as well as deploying your web app.

Problem 1 [5 Points]

Follow the instructions from the tutorial on the class website to set up your local XAMPP development environment. For every team member, please take a screenshot of your local phpMyAdmin interface as shown below and include it in your readme.md on GitHub (described in more details in Problem 2 below).



Problem 2 [90 Points]

Having created your landing page you now decide to develop your full stack CRUD web app with a MySQL and PHP backend. Add this backend to your landing page and evolve it into a CRUD app. In particular, your app is intended to allow a user to do the following (using the example of a music rating app; adapt the CRUD functionality for the domain of your app accordingly):

- **Create** (e.g., a new song in a database with title, artist, and rating)
- **Read** (e.g., a list of songs from the database with title, artist, and rating)
- **Update** (e.g., title, artist, and rating of an existing song in the database)
- **Delete** (e.g., a song (title, artist, and rating) from the database)

In addition, implement user registration, login, and logout functionality. The above CRUD functionality should be in addition to creating and reading existing usernames and passwords in your database. It is not necessary that people can update and delete their usernames and passwords.

We work from first principles. Thus, your app should only use HTML, PHP, and MySQL without relying on any libraries, frameworks, or external scripts. Those can be added later on in the coming assignments. At this point, the focus is on the structure and basic functionality of your app.

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 this homework.

The organization of your code is up to you. For example, you could have one PHP file for each of the CRUD functions plus some additional files (e.g., for logging in and out).

You are required to use GitHub for version control and organization of your team's work using the issue tracker, branches, commits, and pull requests as described in the previous homework. There is no particular number of uses for each of these GitHub features as those are dependent on your individual app and style. However, it is necessary for you to demonstrate proper use of GitHub to not lose points. For example, just two issues or branches is clearly insufficient. Please use the same repo that you created for your landing page.

Your code should be sufficiently commented. **Your repo should contain a readme.md explaining how to install your app**, how to use the different features of your app, and satisfying the requirements that are explicitly mentioned here (e.g., screenshots of your development environment as explained above and **SQL queries to recreate your database and table structures** as explained below).

You should develop your app in the local development environment you created with XAMPP and later deploy it to your production environment on InfinityFree.

Your database, which you should call `app-db`, should contain, at a minimum, two tables, which you should create via SQL queries in phpMyAdmin.

You should have a `users` table, as shown below. Your other tables depend on your app. Below is an example `ratings` table illustrating the idea for a music rating app.

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

username	password
Amelia-Earhart	Youaom139&yu7
Otto	StarWars2*

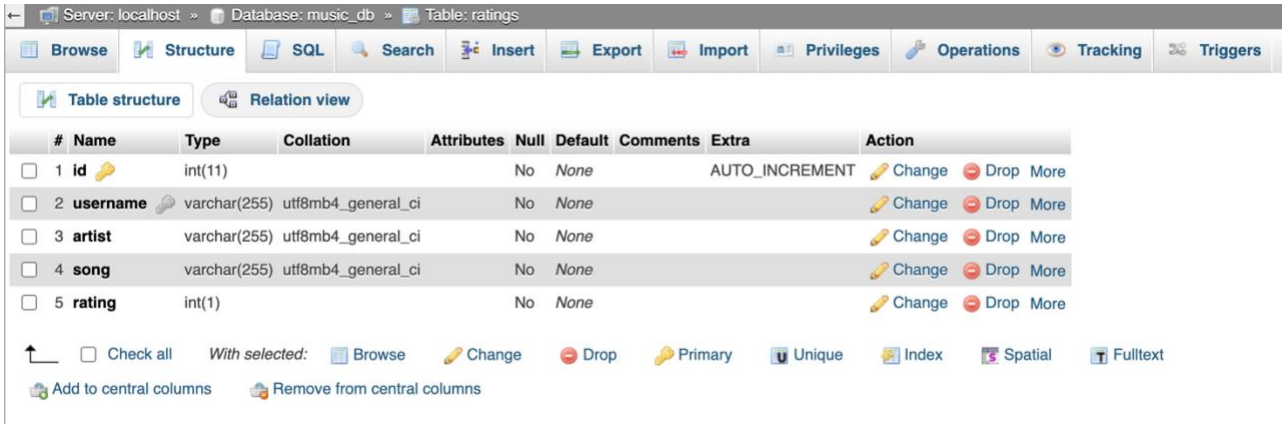
`ratings` table (primary key: `id`, foreign key: `username`)

id	username	artist	song	rating
1	Amelia-Earhart	Aimee Mann	Freeway	3
2	Amelia-Earhart	Bill Evans	Days of Wine and Roses	4
3	Otto	Bill Evans	Days of Wine and Roses	5

Each of your tables should have at least two columns.

The id attribute in the `ratings` table is an integer that is consecutively increased by one as a tuple (row) is being added. Please use such incrementing in your tables as well. Incrementing should be done via MySQLs autoincrement (AI) feature accessible via phpMyAdmin.

In terms of types, as an example, you can use the `varchar(255)` type for `username`, `password`, `song`, and `artist`. You can use `int(11)` and `int(1)` and type for `id` and `rating`.



Your database must remain consistent. In particular, `username` must be a primary key in the users table, and `username` must be a foreign key, as applicable, in your additional tables (e.g., in case of the music rating app, the `ratings` table). For example, 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. Thus, the ratings table has `id` as primary key and `username` as foreign key. All your tables must have a primary key.

There are different ways to add keys (during table creation or via SQL queries). Please add respective code to your SQL queries or explain to us otherwise how we can add the keys to your tables. Please include screenshots of your table structures in your readme.

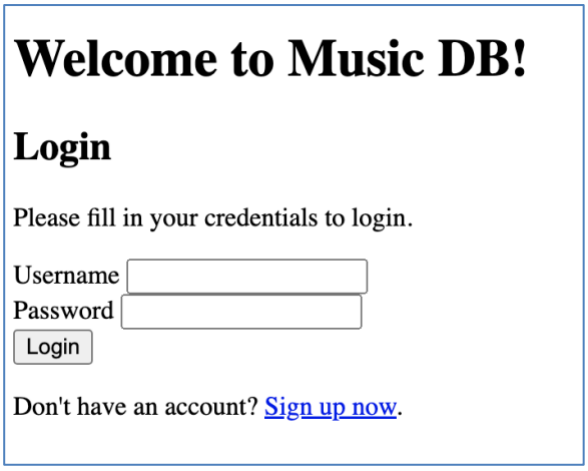
Here are further requirements that your app has to satisfy:

- People should be required to register and login with a username and password. 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. The submitted username and password should be written to the `users` table of your `app-db` database. Once people are logged in they should remain authenticated via a PHPSESSID token. Users should be able to log out as well.
- The registration form should ask people to enter their password twice to ensure that they are not mistaken about what they entered. If the passwords do not match, the user should be prompted again.
- If a user is already logged in and requests the registration page, they should be redirected to the index page, in the music rating example, the page with the overview of all rated songs. Your app would have an overview page with data read from your database as well.
- Show at the top of each page in your app which user is logged in.
- In your app all users should be able to see all data all users entered. However, only the user who entered the data should be able to update and delete it. In the example, of the music rating app, songs (title, artist, and rating) should be displayed for all users in

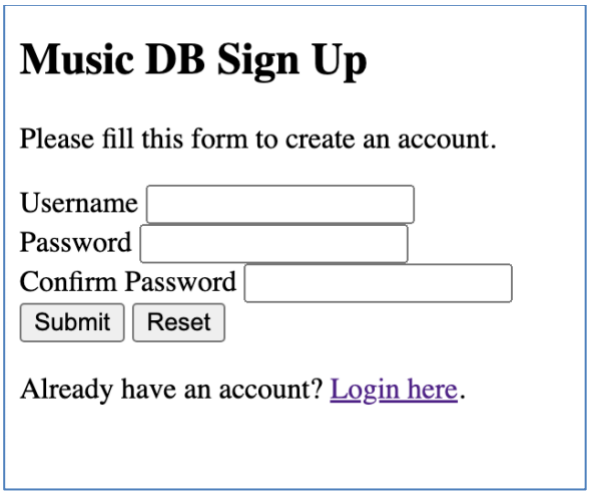
the database. If Sally entered “While we’re young” and Don entered “Sweet Jane,” both Sally and Don will see the same song list containing “While we’re young” and “Sweet Jane.” However, only Sally should be able to update and delete “While we’re young” while only Don should be able to update and delete “Sweet Jane.” You can assume that users will only enter unique entries, i.e., no duplicates.

- Validate via the frontend that users cannot leave any form fields blank. Their password must be at least 10 characters long. Also ensure via the frontend that all other values satisfy your constraints, e.g., for the music rating app a rating must be a one-digit integer. An error message should appear if a user does not conform to any of your requirements. Your requirements must also make sense, e.g., for the music rating app the rating cannot be open-ended but constrained to, say, integer values 0 to 9.
- All SQL queries must be parameterized to avoid SQL injection attacks.
- Passwords must 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/). See the tutorial for how you hash and salt a password in PHP.
- Readme: **Please make it easy for us to run and understand your app.** Include in your readme how we can set up your app locally, especially, SQL queries for generating the database of your app. We will follow your instructions for running your app. If we cannot run your app by simply following your instructions, you will lose points. **Thus, please ensure that your instructions are intelligible, complete, and can be performed by someone not familiar with your app.**

Here is how your app could look like. It does not need to be visually appealing, but all CRUD functionality must be fully implemented.



Login.



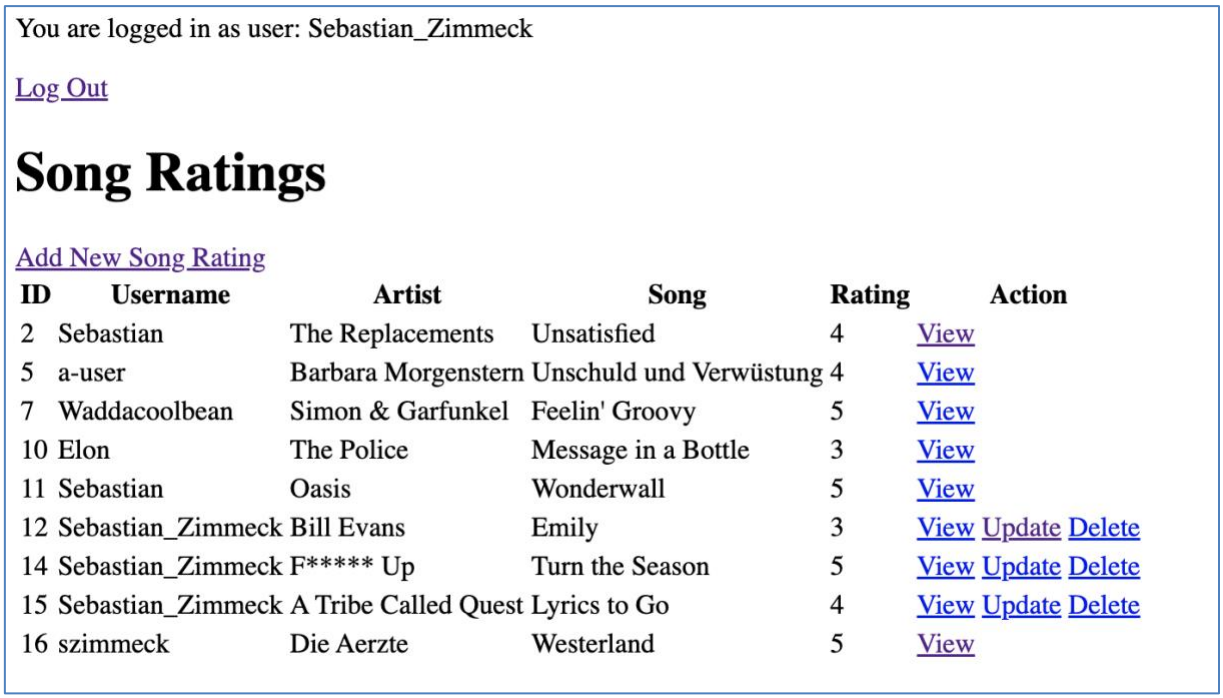
Music DB Sign Up

Please fill this form to create an account.

Username
Password
Confirm Password
Submit Reset

Already have an account? Login here.

Registration.



You are logged in as user: Sebastian_Zimmeck

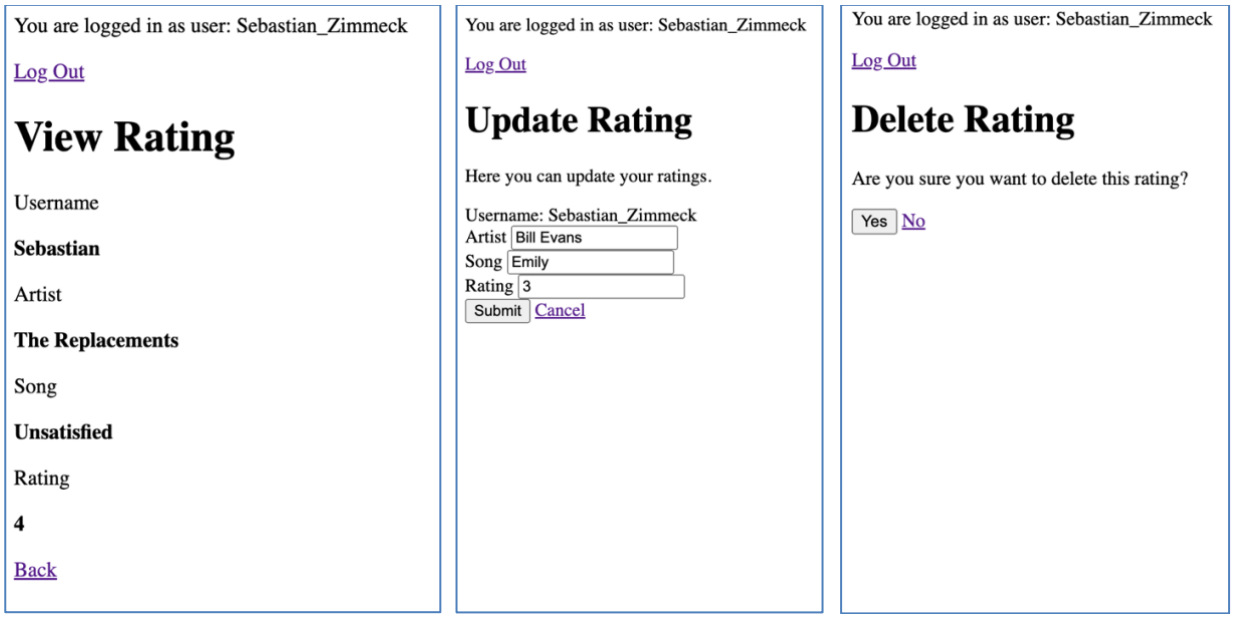
Log Out

Song Ratings

Add New Song Rating

ID	Username	Artist	Song	Rating	Action
2	Sebastian	The Replacements	Unsatisfied	4	View
5	a-user	Barbara Morgenstern	Unschuld und Verwüstung	4	View
7	Waddacoolbean	Simon & Garfunkel	Feelin' Groovy	5	View
10	Elon	The Police	Message in a Bottle	3	View
11	Sebastian	Oasis	Wonderwall	5	View
12	Sebastian_Zimmeck	Bill Evans	Emily	3	View Update Delete
14	Sebastian_Zimmeck	F***** Up	Turn the Season	5	View Update Delete
15	Sebastian_Zimmeck	A Tribe Called Quest	Lyrics to Go	4	View Update Delete
16	szimmeck	Die Aerzte	Westerland	5	View

Overview with all songs all users entered and rated.



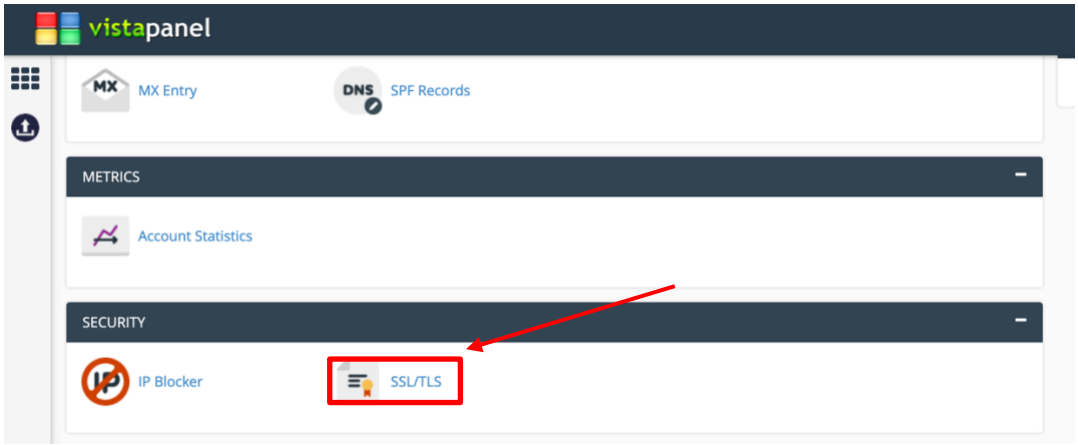
The view, update, and delete rating views.

Problem 3 [5 Points]

Follow the instructions from the tutorial on the class website to sign up for and deploy your web app on https://www.infinityfree.com/. Provide the URL of your deployed site in your readme. Please create a v1.0.0 release on GitHub with the deployed code.

Secure the traffic of your web app by deploying an SSL/TLS certificate. Once your certificate is deployed HTTP requests going to and responses coming from your web app will be encrypted during transmission.

Click on Control Panel on the home screen of your InfinityFree account. You should see the interface below. Click on SSL/TLS and follow the instructions to set up a certificate. The certificate is free. It is not necessary that you spend any money.



After a few minutes, you should now see in your browser the little green padlock next to the URL bar of your site. There is a message on InfinityFree that says it may take 48 hours or so. However, my experience is that it happens within one hour or so. If you are still waiting, there is a good chance that you created your certificate but did not actually deploy it. Creation and deployment are two separate steps, and you may still need to do the latter.

You do not need to deploy any certificate on your local development environment server.

Submission

Please give the following people – course assistants and instructor – access to your GitHub repo (write permission) by **3/7/2025, 10 am** (via Settings -> Collaborators and teams -> Manage access -> Add people). This is all you have to do for submitting your work. No submission on Moodle or via email is necessary. We will have all information we need from the version history.

- lgartrell11
- samir-cerrato
- dadak-dom
- SebastianZimmeck

We will grade your work based on the last commit before the homework due date (or you can let us know the ID of a particular commit before the due date that you would like to use as the basis for the grading).

Important: In your readme, provide a percentage estimate of how much each teammate contributed, e.g., 50/50, 60/40, etc. Grades will be adjusted accordingly. If you do not provide an estimate, we may assume a 50/50 estimate. In any case, **we will look at your version history** and make our determination based on what we see and the estimate that you provide. **It is not possible to challenge a grade with the argument that you contributed more than what you said in your submission.**