Topic and Purpose

My original topic was GeoTweet, which involved a visualizer of world leaders' tweets across a world map. I'll explain below, but I had to pivot my site twice since then due to API obstacles. I have kept a theme of global culture/insight by now instead of having my website centered around a global music playlist that the user can scroll through. The music is called World Music, and it is curated by a random Spotify user, however it is the highest rated world-genre playlist.

Here is a link to it: https://open.spotify.com/playlist/0q6c4fBoe3XFqkPgHo01KT

How It Works, (+) Data

The main page (M2) is the homepage of the website. On that page, basic info about the site can be found. However, if user wants to access the API-driven program, they have to make an account.

Making an account

Making an account is straightforward. Enter the proper information when clicking on the "sign up button". If not entered correctly, several different warning prompts will be displayed. After successfully creating an account, you will automatically be redirected to the program.php file where the music playlist is dynamically displayed.

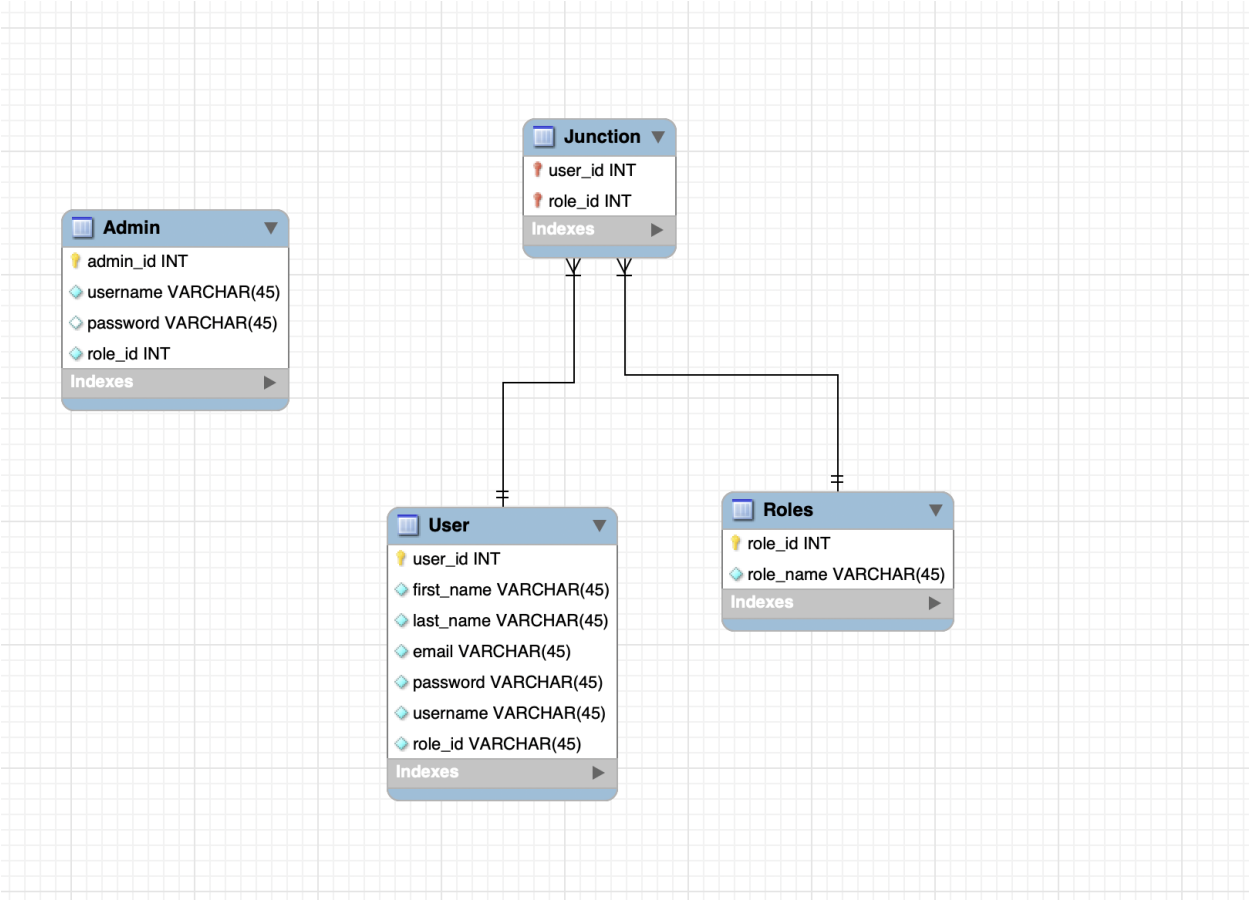
Signing into your account

Same logic as making an account. Will check if your password/username exists in the database. On top of that, if the account you are logging into has a special role assigned to it, AKA 'admin' instead of 'user' (role_id is equal to 2 instead of 1), the username/password will be checked to see if it exists in the Admin database. If it is, you will be redirected to the admin page.

Admin page

The admin page prints all of the information for each row, and then displays a form of each record so that the admin can edit any of the information. **NOTE*** in order to visually see the updates when clicking a submit button for a specific row, you must refresh the page after the page reloads to see the updated table display. The same applies for when you click the delete button to delete a certain row.

Database Design:



User: first name, last name, email, password, username, and role_id. User_id is the primary key and is automatically incremented.

Roles: role_name represents the type of role that the user has. Right now, there are only two roles, 1 being a regular user and 2 being an admin. Role_id is the primary key and is automatically incremented.

Junction: for the future, junction is used to facilitate the relationships between how many roles a user has and how many users categorize under a single role (many-to-many relationship). Right now, it is just configured at a one-to-many relationship for the sake of the assignment, however I designed it like this for a proof of scalability to show that one can add multiple different user roles and users are capable of obtaining more than one role.

Admin: admin_id is an auto incremented primary key. It has its own unique role_id category for its own special labeling system that can be used for future scalability. It possesses a username and password, so that users can be tested if they have admin privileges in the login portal.