Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-452-M2024/it202-api-project-milestone-3-2024-m24/grade/ajz27

IT202-452-M2024 - [IT202] API Project Milestone 3 2024 m24

Submissions:

Submission Selection

1 Submission [active] 8/4/2024 11:01:20 PM

Instructions

COLLAPSE

Overview Video: https://youtu.be/-4hlb9MXrQE

1. Implement the Milestone 3 features from the project's proposal document: https://docs.google.com/document/d/1XE96a8DQ52Vp49XACBDTNCq0xYDt3kF29cO88EWVwfo/viev
2. Make sure you add your ucid/date as code comments where code changes are done
3. All code changes should reach the Milestone3 branch
4. Create a pull request from Milestone3 to dev and keep it open until you get the output PDF from this assignment.
5. Gather the evidence of feature completion based on the below tasks.
6. Once finished, get the output PDF and copy/move it to your repository folder on your local machine.
7. Run the necessary git add, commit, and push steps to move it to GitHub
8. Complete the pull request that was opened earlier
9. Create and merge a pull request from dev to prod
10. Upload the same output PDF to Canvas

Branch name: Milestone3

Tasks: 21 Points: 10.00

API (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Data Related to Users

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	What's the concept/association?
#2	1	What sort of relationship is it (one to many, many to one, many to many, etc)
#3	1	Note any other considerations

Response:

1. The concept is a simple music database where a user can look up songs through Shazam's API and review them.
2. It is a one to many relationship. One user is associating with multiple entities (songs).

COLLAPSE

Task #2 - Points: 1

Text: Updating Entities

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	When an update occurs either manually or from the API how does it affect associated data?
#2	1	Do users see the old data, new data, does data need to be reassociated, etc?

Response:

1. Associated data is either overwritten or added depending on action.
2. Users will only see new data.

Handle Data Association (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the code

Details:

Option 1: Related pages will have a button to do association (like favorites or similar),
Option 2: a separate page will be used to associate entities to a user by some other user (like assignment of entities)
Include ucid/date comments for each code screenshot

#1) Show the related code

Caption (required) ✓

Describe/highlight what's being shown

association code

Explanation (required) ✓

Explain in concise steps how this logically works and mention which option your application handles regarding association

PREVIEW RESPONSE

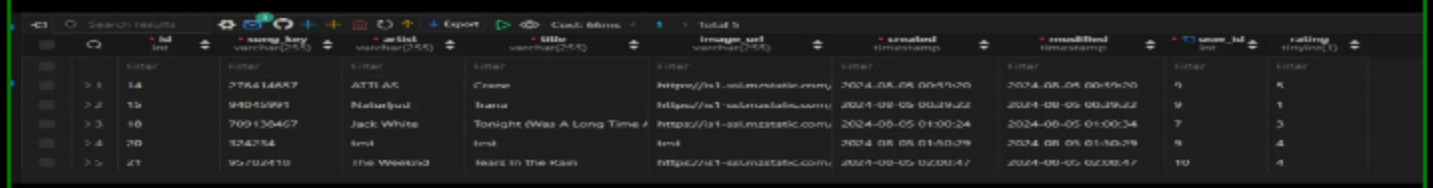
Association is done through a user_id column in the table for saving songs. The user_id corresponds with the id from the users table, and when a song is added it also adds the ID of the user into the user_id column. This is the first option and the user associates data themselves.

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the association table(s)

#1) Show the table(s) you made to handle the associations (Should have some example data)



Caption (required) ✓
Describe/highlight what's being shown
Association table

Explanation (required) ✓
Describe each column/association table
PREVIEW RESPONSE
The user_id column is tied to the id column from the users table. All other columns contain respective song and rating data.

COLLAPSE

Task #3 - Points: 1
Text: Add related links

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Include the heroku prod link for the page that creates the association
#2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1
https://it202-ajz27-prod-be806f4746c9.herokuapp.com/Project/add_song_to_database.php
URL: https://it202-ajz27-prod-be806f4746c9.herokuap

URL #2
https://github.com/ajz27/ajz27-IT202-MC/pull/38
URL: https://github.com/ajz27/ajz27-IT202-MC/pull/3

+ ADD ANOTHER URL

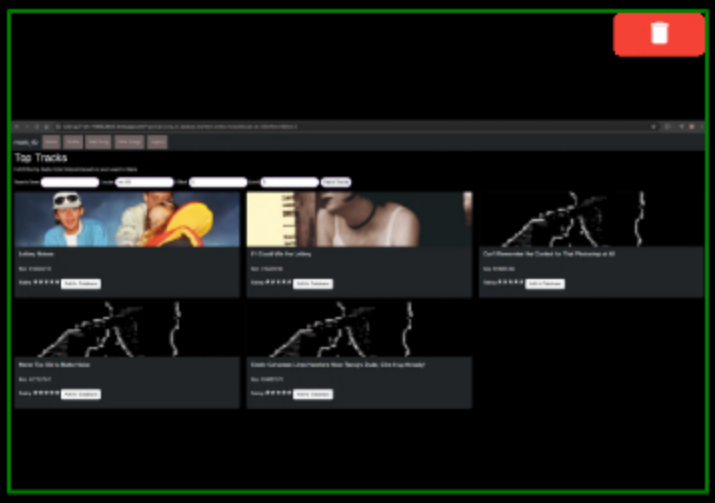
Current User's Association Page (2 pts.)
COLLAPSE

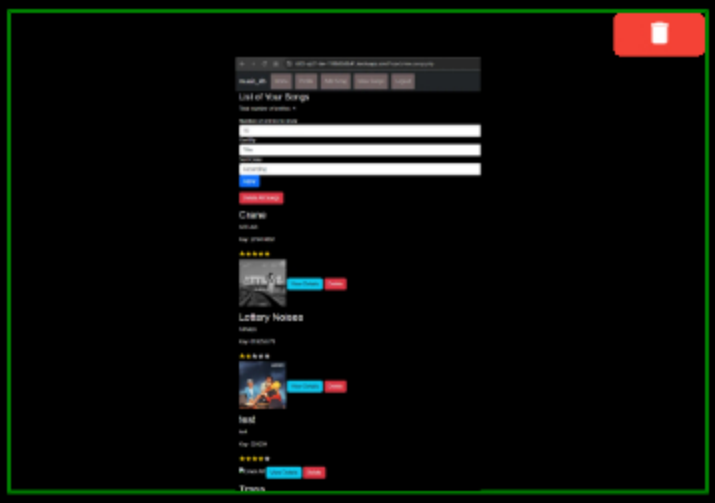
Task #1 - Points: 1
Text: Screenshots of this page

Details:
Make sure the heroku dev url is visible in the address bar

Some screenshots may be repeated in subtasks, but ensure they highlight the specific subtask requirement.

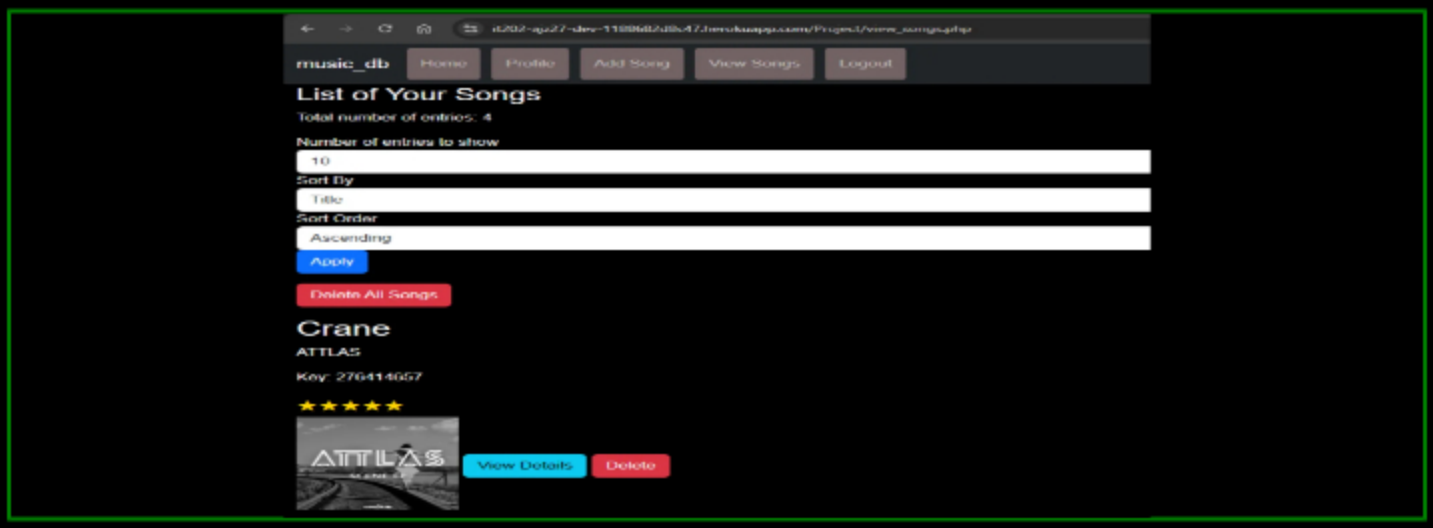
#1) Show the summary of the results with relevant information per entity





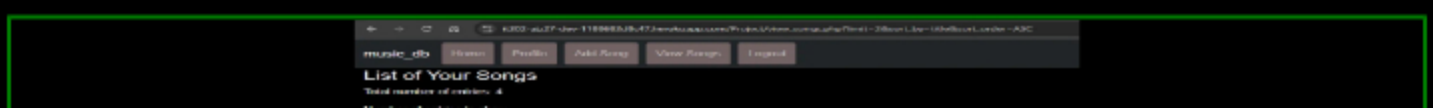
Caption (required) ✓
Describe/highlight what's being shown
summary of results

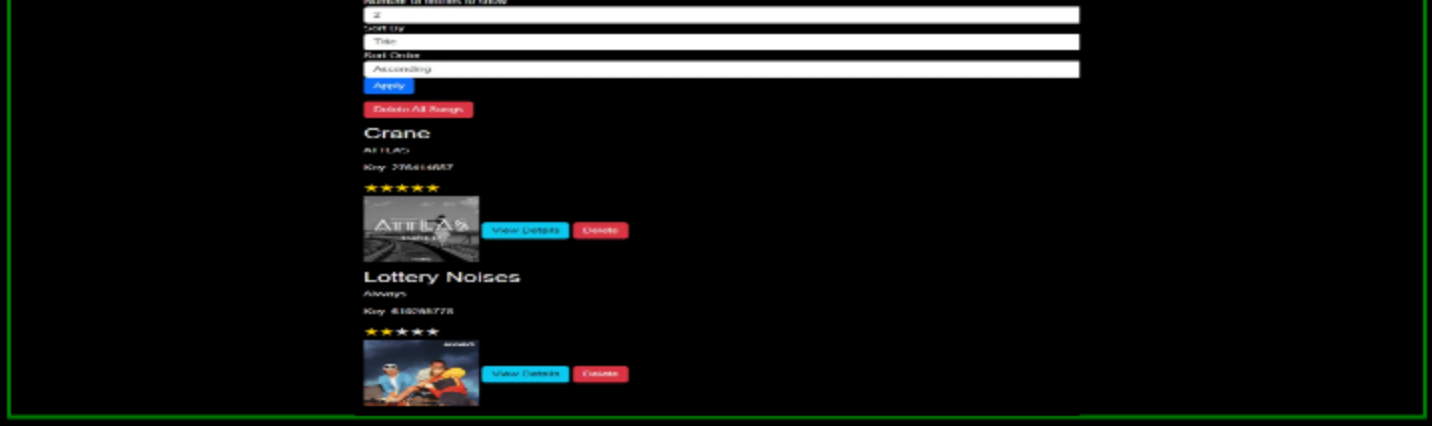
#2) Show the single view buttons/links, delete button/links, and delete all button/link



Caption (required) ✓
Describe/highlight what's being shown
view, delete, delete all

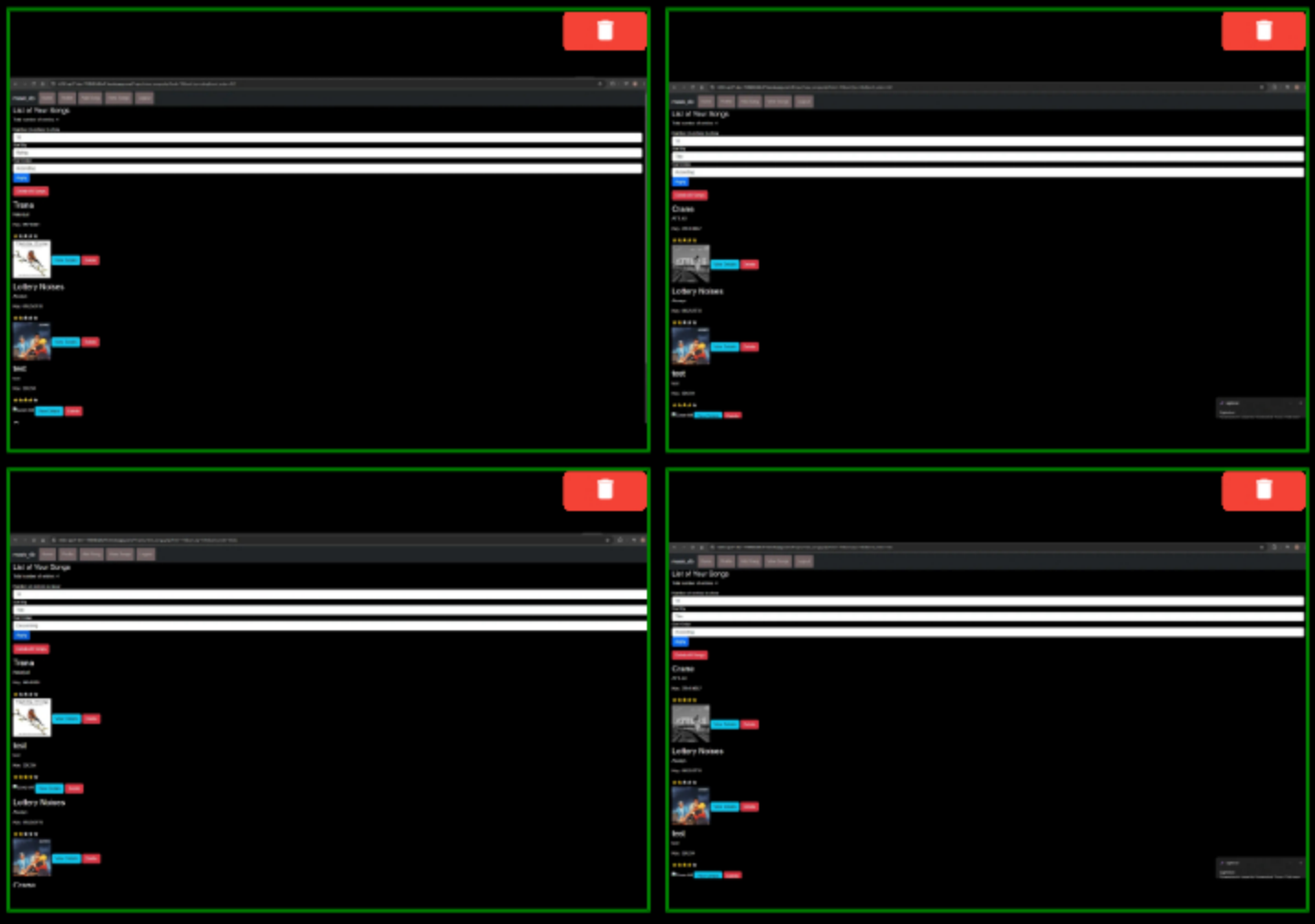
#3) Show variations of the number of shown items count and show the count of total number of associated items to the user





Caption (required) ✓
Describe/highlight what's being shown
filter variation

#4) Show variations of the filter/sort including no results (proper message should be visible)



Caption (required) ✓
Describe/highlight what's being shown
filter/sort

^COLLAPSE^

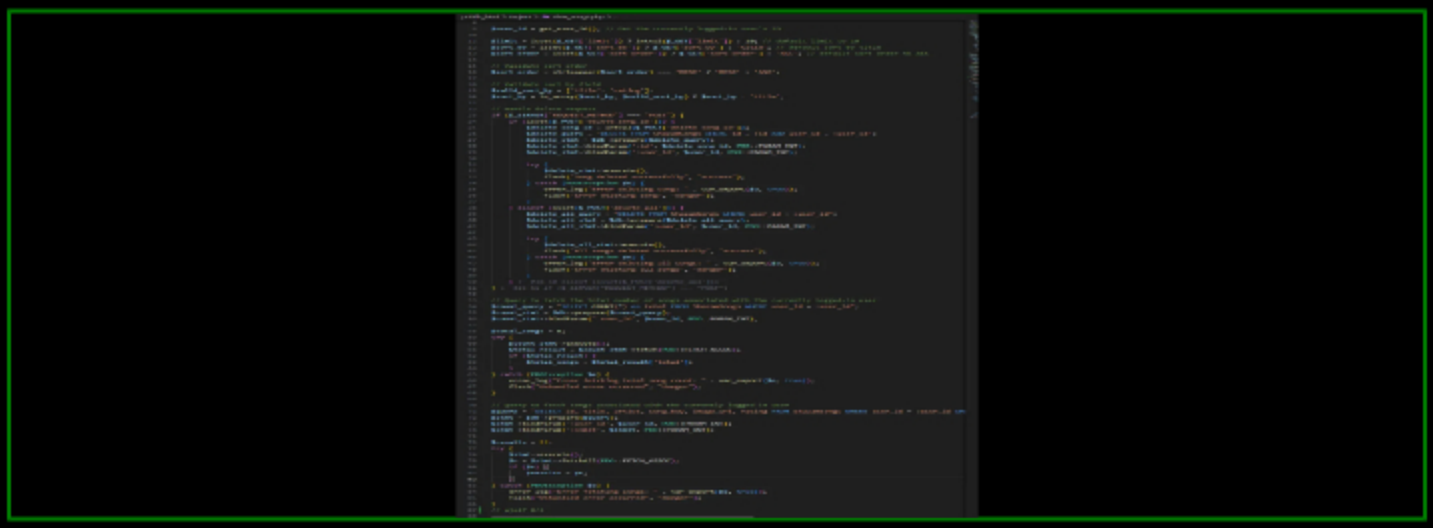
Task #2 - Points: 1

Text: Screenshot the code

Details:

Include ucid/date comments for each code screenshot

#1) Show the code related to fetching the user's associations (including the query)



Caption (required) ✓

Describe/highlight what's being shown

user associations

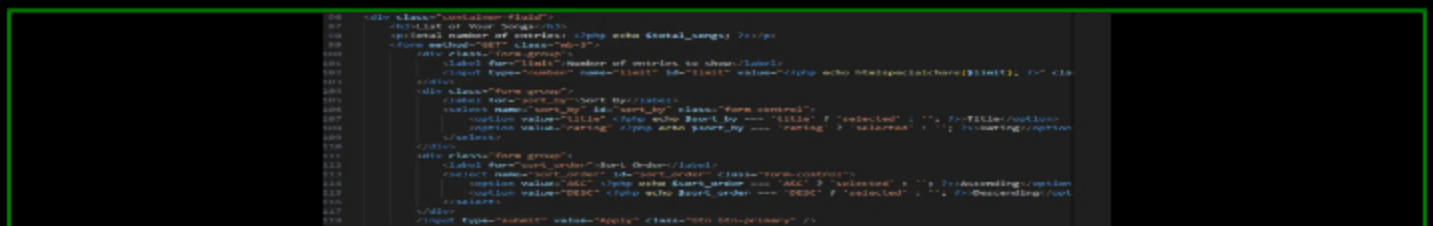
Explanation (required) ✓

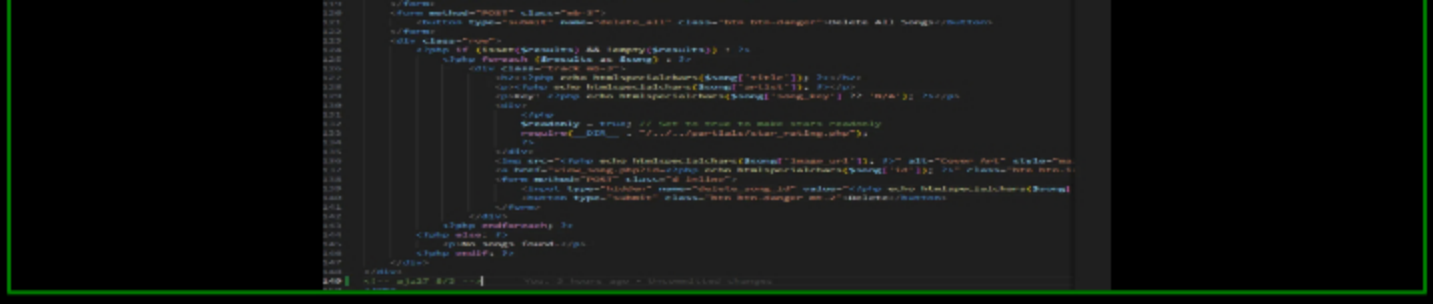
Explain in concise steps how this logically works and mention how you determine the result list (include the association logic and filters)

PREVIEW RESPONSE

The user is first authenticated, making sure they are logged in. Then the ID of the currently logged in user is fetched through get_user_id. Songs are associated with the user through the user_id column in the table. Each song in the result set has a user_id that maches the logged in user. The songs are fetched using bindParam as a security measure to insert user_id and limit into the query. The results are the nrendered.

#2) Show the code related to the display of the results





Caption (required) ✓

Describe/highlight what's being shown

display code

Explanation (required) ✓

Explain in concise steps how this logically works

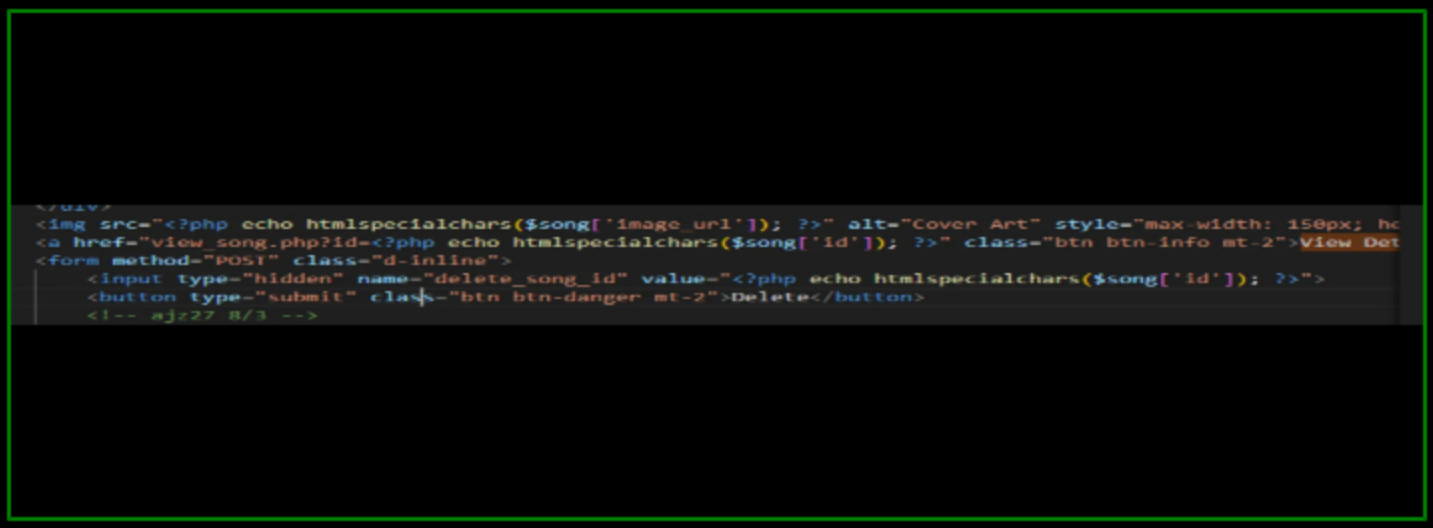
PREVIEW RESPONSE

Div class with container-fluid makes it a responsive container, and

displays the title of the page. A paragraph displays the total number of songs retrieved from the database. Filter and sort forms are offered, and then the delete all songs function uses a POST form with a button to delete all songs associated with the current user. The results display is done by checking if $results is set tand not empty and then iterating through each song. Various song details are

displayed, including a star rating system. If $results is empty, then a message is displayed saying "No songs found."

#3) Each record should have a button/link for single view



Caption (required) ✓

Describe/highlight what's being shown

single view

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

An anchor tag pointing to view_song,php and a query parameter id set to the song's ID is used. This link directs the user to the detailed view of the song.

#4) Each record should have a button/link for delete (this may be an admin-only thing but should be present for the specific role) Note: this is to delete the relationship and not the specific entity

```php
<img src="<?php echo htmlspecialchars($song['image_url']); ?>" alt="Cover Art" style="max-width: 150px; hei
<a href="view_song.php?id=<?php echo htmlspecialchars($song['id']); ?>" class="btn btn-info mt-2">View Deta
<form method="POST" class="d-inline">
    <input type="hidden" name="delete_song_id" value="<?php echo htmlspecialchars($song['id']); ?>">
    <button type="submit" class="btn btn-danger mt-2">Delete</button>
```

Caption (required) ✓

Describe/highlight what's being shown

delete button

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

This is doen through a hidden input field with name=delete_song_id and value set to the song's ID.

#5) Show the logic for deleting all associations for the user (this may be admin-only but should be present for the specific role)

```php
// Handle delete request
if ($_SERVER["REQUEST_METHOD"] === "POST") {
    if (isset($_POST['delete_song_id'])) {
        $delete_song_id = intval($_POST['delete_song_id']);
        $delete_query = "DELETE FROM ShazamSongs WHERE id = :id AND user_id = :user_id";
        $delete_stmt = $db->prepare($delete_query);
        $delete_stmt->bindParam(':id', $delete_song_id, PDO::PARAM_INT);
        $delete_stmt->bindParam(':user_id', $user_id, PDO::PARAM_INT);

        try {
            $delete_stmt->execute();
            flash("Song deleted successfully", "success");
        } catch (PDOException $e) {
            error_log("Error deleting song: " . var_export($e, true));
            flash("Error deleting song", "danger");
        }
    } elseif (isset($_POST['delete_all'])) {
        $delete_all_query = "DELETE FROM ShazamSongs WHERE user_id = :user_id";
        $delete_all_stmt = $db->prepare($delete_all_query);
        $delete_all_stmt->bindParam(':user_id', $user_id, PDO::PARAM_INT);

        try {
            $delete_all_stmt->execute();
            flash("All songs deleted successfully", "success");
        } catch (PDOException $e) {
            error_log("Error deleting all songs: " . var_export($e, true));
            flash("Error deleting all songs", "danger");
        }
    } <- #38-50 elseif (isset($_POST['delete_all']))
} <- #23-51 if ($_SERVER["REQUEST_METHOD"] === "POST")
// ajax 8/3
```

Caption (required) ✓

Describe/highlight what's being shown

delete all

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

This checks for if delet_all is set in the POST request. The SQL query is then prepared and executed.. Either a success message is flashed or a fail message is flashed and the error is logged.

#6) Show the logic related to the count of all associated items to the user (even the ones not shown in

#6) Show the logic related to the count of all associated items to the user (even the ones not shown in the filtered results)

```php
// ajz27 8/3
// Query to fetch the total number of songs associated with the currently logged-in user
$count_query = "SELECT COUNT(*) as total FROM ShazamSongs WHERE user_id = :user_id";
$count_stmt = $db->prepare($count_query);
$count_stmt->bindParam(':user_id', $user_id, PDO::PARAM_INT);

$total_songs = 0;
try {
    $count_stmt->execute();
    $total_result = $count_stmt->fetch(PDO::FETCH_ASSOC);
    if ($total_result) {
        $total_songs = $total_result['total'];
    }
} catch (PDOException $e) {
    error_log("Error fetching total song count: " . var_export($e, true));
    flash("Unhandled error occurred", "danger");
}
```

Caption (required) ✓

Describe/highlight what's being shown

total count

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

An SQL query is prepared to count al lsong associations for the current user based on their user ID. A variable $total_songs is initalized with a default of 0 to count songs associated with the user. The query is then executed and displayed.

#7) Show the logic related to the count of the items on the page (this value should change based on the filter applied)

```php
//ajz27 8/3
$displayed_songs_count = count($results);

$table = [
    "data" => $results,
    "title" => "Your Songs",
    "ignored_columns" => [],
    "view_url" => get_url("view_song.php") // URL to the song detail page
]; <- #90-95 $table =
?>
<div class="container-fluid">
    <h3>List of Your Songs</h3>
    <p>Total number of entries: <?php echo $total_songs; ?></p>
    <p>Number of entries displayed: <?php echo $displayed_songs_count; ?></p>
```

Caption (required) ✓

Describe/highlight what's being shown

result display

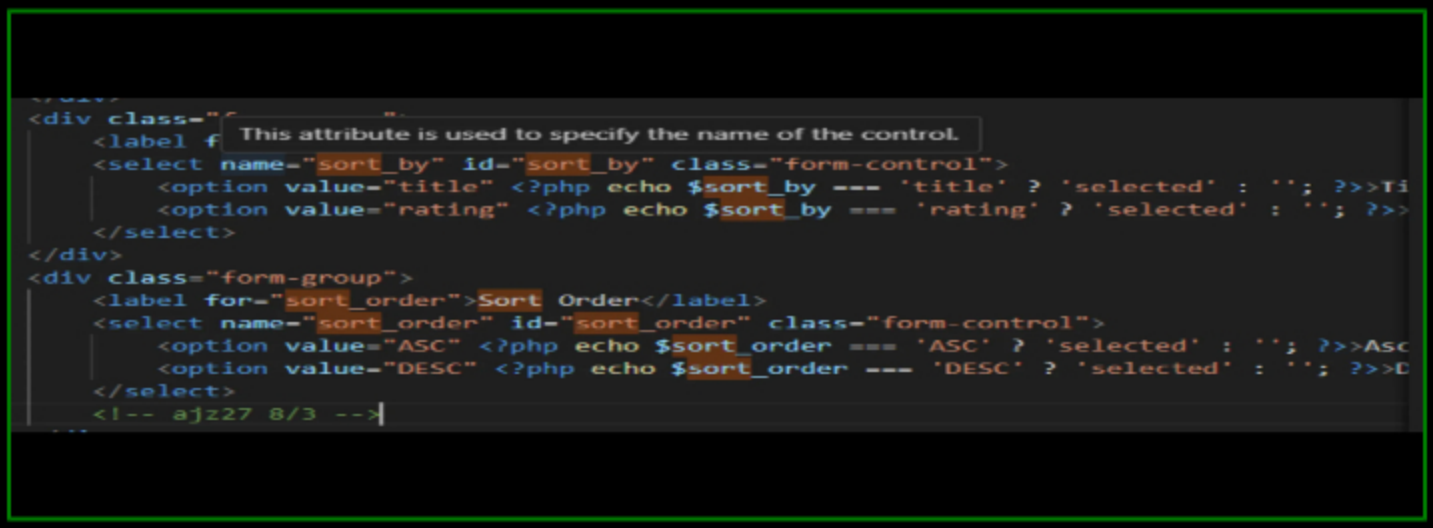
Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The $displayed_songs_count variable counts $results and then displays it.

#8) Show the logic related to filter/sort (limit should be constrained to 1-100 otherwise default to 10)



Caption (required) ✓

Describe/highlight what's being shown

sort/filter

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

Two sorting options are provided. The user can sort by rating and alphabetic order of the title. A filter for limiting the amount of entries is also present.

COLLAPSE

Task #3 - Points: 1

Text: Add related links

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Include the heroku prod link for the page that creates the association
#2	1	Add the pull request link for the branch related to this feature Note: the link should end with /#. Same pull request shouldn't be used for each feature

URL #1

https://github.com/ajz27/ajz27-IT202-MC/pull/3

https://github.com/ajz27/ajz27-IT202-MC/pull/39

URL #2
https://it202-ajz27-prod-be806f4746c9.herokuapp.com/Project/view_songs.php

https://it202-ajz27-prod-be806f4746c9.herokuap

+ ADD ANOTHER URL

All Users Association Page (likely an admin page) (2 pts.)

^COLLAPSE ^

^COLLAPSE ^

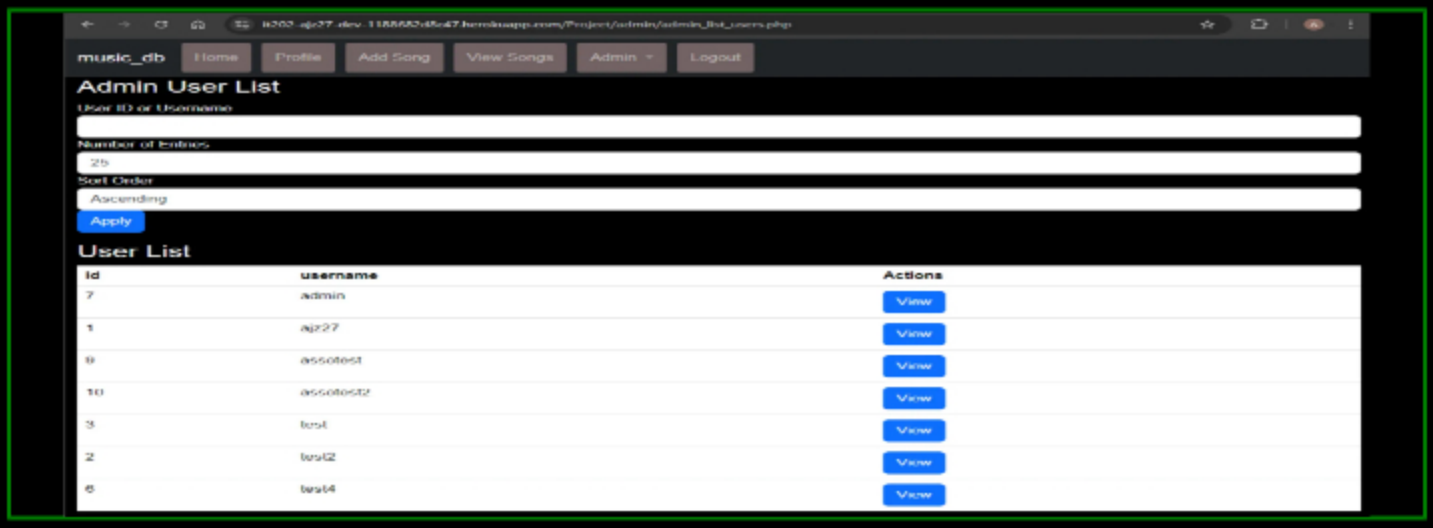
Task #1 - Points: 1

Text: Screenshots of this page

Details:
Make sure the heroku dev url is visible in the address bar

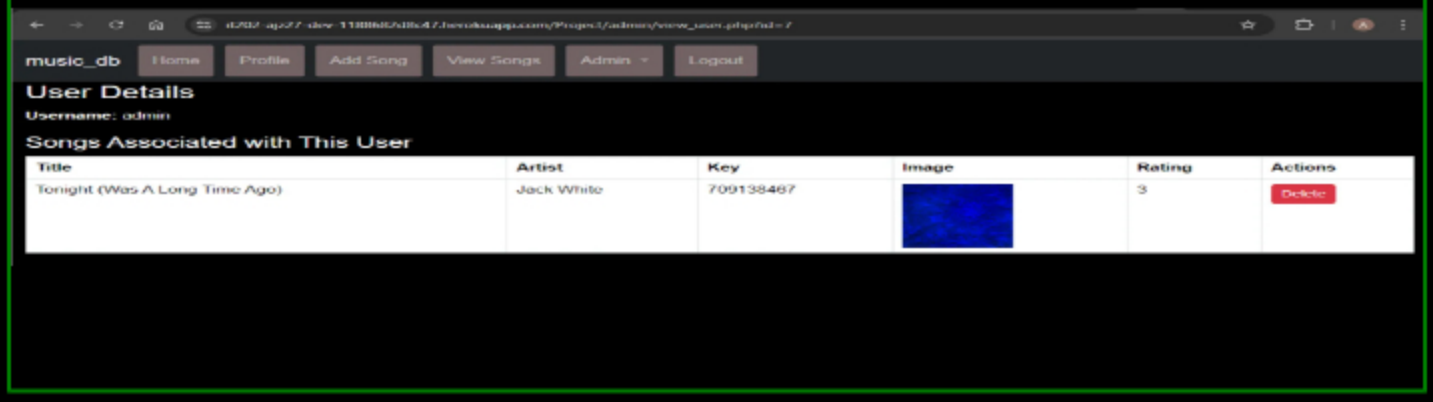
Some screenshots may be repeated in subtasks, but ensure they highlight the specific subtask requirement.

#1) Show the summary of the results with relevant information per entity



Caption (required) ✓
Describe/highlight what's being shown
admin user list

#2) Show the single view buttons/links, delete button/links



Caption (required) ✓

Describe/highlight what's being shown

delete link, view link in above screenshot

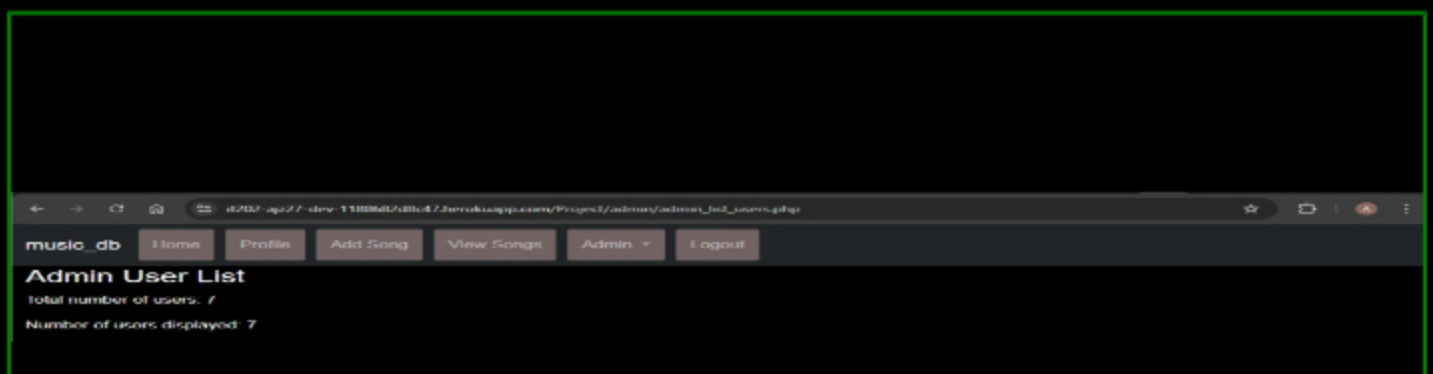
#3) Show the username related to the specific entity and that it's clickable

Caption (required)

Describe/highlight what's being shown

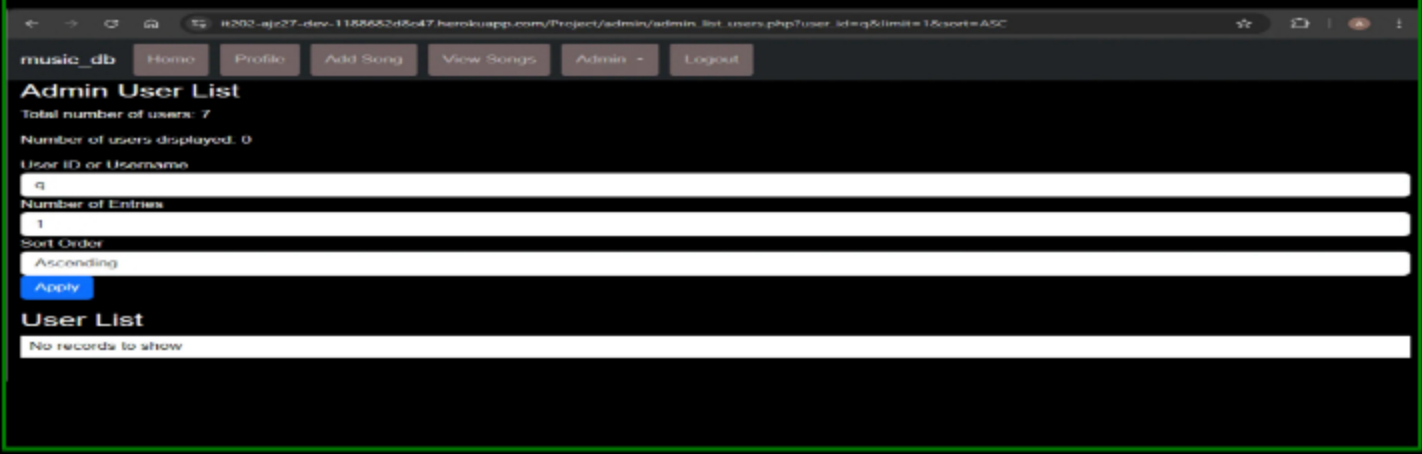
Missing caption

#4) Show variations of the number of shown items count and show the count of total number of associated items



Caption (required) ✓
Describe/highlight what's being shown
variation

#5) Show variations of the filter/sort including no results (proper message should be visible)



Caption (required) ✓
Describe/highlight what's being shown
filter/sort

^COLLAPSE ^

Task #2 - Points: 1
Text: Screenshot the code

Details:
Include ucid/date comments for each code screenshot

#1) Show the code related to fetching all associations (including the query)

```php
// Build query to fetch users
$query = "SELECT id, username FROM Users";
$params = [];

// Add search condition if search_term is provided
if (!empty($search_term)) {
    $query .= " WHERE id LIKE :search_term OR username LIKE :search_term";
    $params[':search_term'] = "%$search_term%";
}

// Add sorting and limit
$query .= " ORDER BY username $sort_order LIMIT :limit";
$stmt = $db->prepare($query);
```

```php
// Bind search_term if it is used
if (isset($params[':search_term'])) {
    $stmt->bindParam(':search_term', $params[':search_term'], PDO::PARAM_STR);
}

$results = [];
$displayed_users_count = 0;
try {
    $stmt->execute();
    $r = $stmt->fetchAll(PDO::FETCH_ASSOC);
    if ($r) {
        $results = $r;
        $displayed_users_count = count($results);
    }
} catch (PDOException $e) {
    error_log("Error fetching users: " . var_export($e, true));
    flash("Unhandled error occurred", "danger");
}
// ajz27 8/3
```

Caption (required) ✓

Describe/highlight what's being shown

fetch all associations

Explanation (required) ✓

Explain in concise steps how this logically works and mention how you determine the result list (include the association logic and filters)

PREVIEW RESPONSE

The base query is set to select id and username from the USers table. The search condition is then added to append the query to filter results based on the user ID or username.. Then the query is appended with an ORDER BY clause to sort by username in the specified order.. FInally, the query is prepared with PDO and bindParam and executed.

#2) Show the code related to the display of the results

```php
<div class="container-fluid">
    <h3>Admin User List</h3>
    <p>Total number of users: <?php echo $total_users; ?></p>
    <p>Number of users displayed: <?php echo $displayed_users_count; ?></p>
    <form method="GET" class="mb-3">
        <div class="form-group" id="search_value">
            <label for="user_id">User ID or Username</label>
            <input type="text" name="user_id" id="user_id" value="<?php echo htmlspecialchars($search_term ?
        </div>
        <div class="form-group">
            <label for="limit">Number of Entries</label>
            <input type="number" name="limit" id="limit" value="<?php echo htmlspecialchars($limit); ?>" cla
        </div>
        <div class="form-group">
            <label for="sort">Sort Order</label>
            <select name="sort" id="sort" class="form-control">
                <option value="ASC" <?php echo $sort_order === 'ASC' ? 'selected' : ''; ?>>Ascending</option
                <option value="DESC" <?php echo $sort_order === 'DESC' ? 'selected' : ''; ?>>Descending</opt
            </select>
        </div>
        <input type="submit" value="Apply" class="btn btn-primary" />
    </form>
    <?php render_table($table); ?>
    <!-- ajz27 8/3 -->
```

Caption (required) ✓

Describe/highlight what's being shown

code for display of results

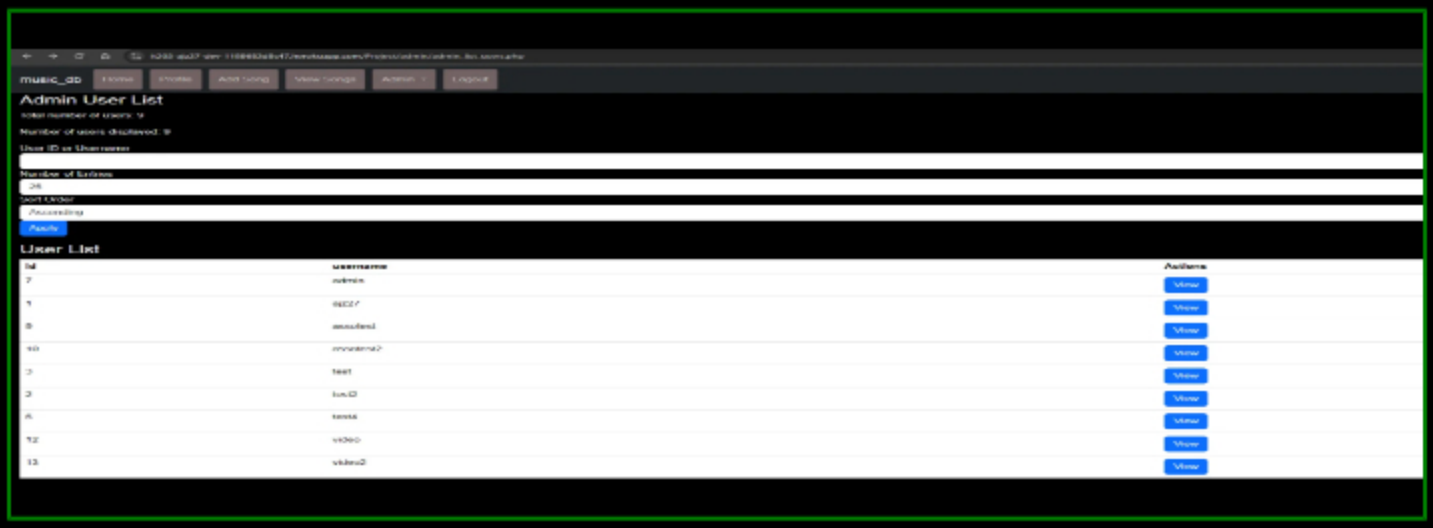
Explanation (required) ✓

Explain in concise steps how this logically works and mention the logic for handling the username requirements

PREVIEW RESPONSE

The results are in a fluid container which contain the input fields for the user id, limit for results, and the sort function. The results are rendered in a table using the professor's table script.

#3) Each record should have a button/link for single view



Caption (required) ✓

Describe/highlight what's being shown

record single view

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

Clicking the button pulls the information from the database about the user's songs.

#4) Each record should have a username field that is clickable to go to the user's profile page

```
<d>
    <?php if ($col === 'username' && $_view_url) : ?>
        <a href="<?php echo get_url('profile.php'); ?>?id=<?php se($row['id']); ?>"><?php
    <?php elseif (filter_var($row[$col], FILTER_VALIDATE_URL) && in_array($col, ['coverar
        <img src="<?php se($row[$col]); ?>" alt="Cover Art" style="max-width: 100px; max-
    <?php else : ?>
        <?php se($row[$col]); ?>
    <?php endif; ?>
    <!-- ajz27 8/3 -->
```

Caption (required) ✓

Describe/highlight what's being shown

clickable usernames

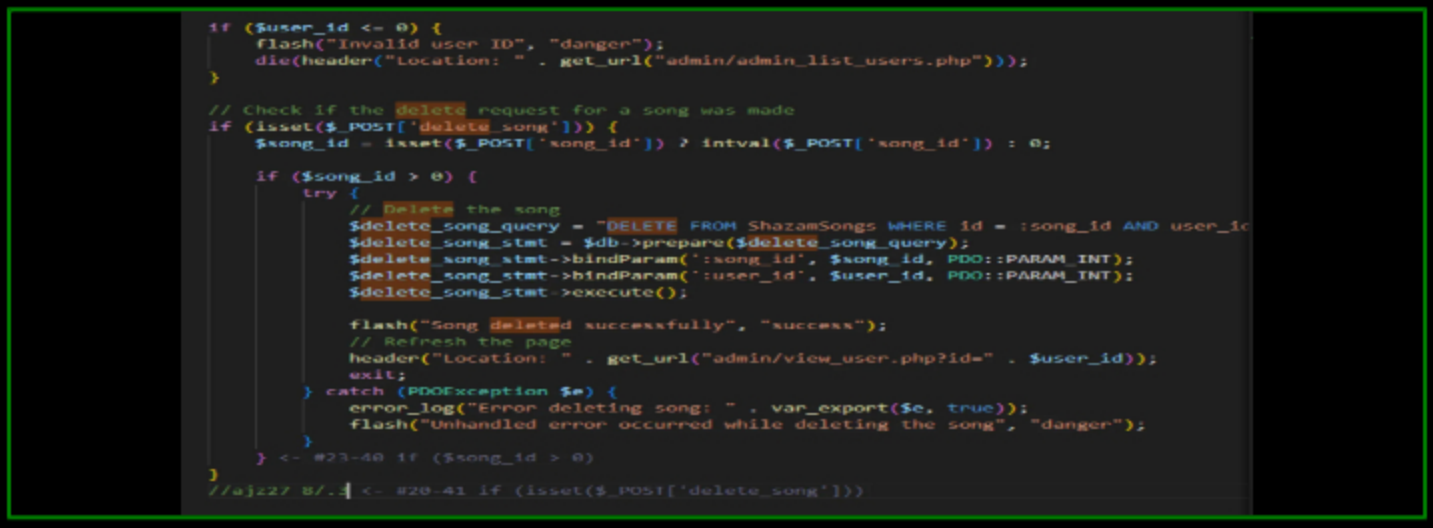
Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

When the username link is clicked, it redirects to the appropriate URL for the profile.

#5) Each record should have a button/link for delete (this may be an admin-only thing but should be present for the specific role) Note: this is to delete the relationship and not the specific entity



```php
if ($user_id <= 0) {
    flash("Invalid user ID", "danger");
    die(header("Location: " . get_url("admin/admin_list_users.php")));
}

// Check if the delete request for a song was made
if (isset($_POST['delete_song'])) {
    $song_id = isset($_POST['song_id']) ? intval($_POST['song_id']) : 0;

    if ($song_id > 0) {
        try {
            // Delete the song
            $delete_song_query = "DELETE FROM ShazamSongs WHERE id = :song_id AND user_id
            $delete_song_stmt = $db->prepare($delete_song_query);
            $delete_song_stmt->bindParam(':song_id', $song_id, PDO::PARAM_INT);
            $delete_song_stmt->bindParam(':user_id', $user_id, PDO::PARAM_INT);
            $delete_song_stmt->execute();

            flash("Song deleted successfully", "success");
            // Refresh the page
            header("Location: " . get_url("admin/view_user.php?id=" . $user_id));
            exit;
        } catch (PDOException $e) {
            error_log("Error deleting song: " . var_export($e, true));
            flash("Unhandled error occurred while deleting the song", "danger");
        }
    } <- #23-40 if ($song_id > 0)
}
//ajz27 8/3 <- #20-41 if (isset($_POST['delete_song']))
```

Caption (required) ✓

Describe/highlight what's being shown

delete record

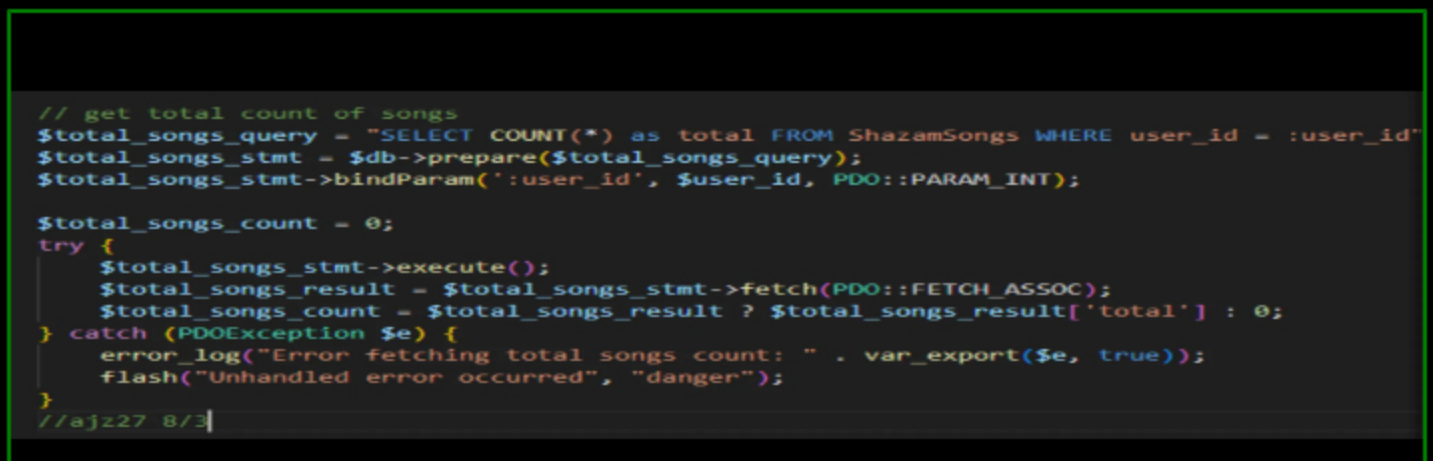
Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

When the delete button is clicked on the record, it uses a query to delete the related song from ShazamSongs.

#6) Show the logic related to the count of all associated items (even the ones not shown in the filtered results)



```php
// get total count of songs
$total_songs_query = "SELECT COUNT(*) as total FROM ShazamSongs WHERE user_id = :user_id"
$total_songs_stmt = $db->prepare($total_songs_query);
$total_songs_stmt->bindParam(':user_id', $user_id, PDO::PARAM_INT);

$total_songs_count = 0;
try {
    $total_songs_stmt->execute();
    $total_songs_result = $total_songs_stmt->fetch(PDO::FETCH_ASSOC);
    $total_songs_count = $total_songs_result ? $total_songs_result['total'] : 0;
} catch (PDOException $e) {
    error_log("Error fetching total songs count: " . var_export($e, true));
    flash("Unhandled error occurred", "danger");
}
//ajz27 8/3
```

Caption (required) ✓

Describe/highlight what's being shown

count

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The count works by using a query to select all the songs from ShazamSongs with the appropriate user ID.

#7) Show the logic related to the count of the items on the page (this value should change based on the filter applied)

```
<p><strong>Songs Displayed:</strong> <?php echo count($songs); ?></p>
<table class="table table-bordered">
    <thead>
        <tr>
            <th>Title</th>
            <th>Artist</th>
            <th>Key</th>
            <th>Image</th>
            <th>Rating</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
    <!-- ajz27 8/3 -->
```

Caption (required) ✓

Describe/highlight what's being shown

count of items on the page

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The count here just echoes the count of the displayed songs.

#8) Show the logic related to filter/sort (should include a partial match for username) (limit should be constrained to 1-100 otherwise default to 10)

```
// Add search condition if search_term is provided
```

```php
if (!empty($search_term)) {
    $query .= " WHERE id LIKE :search_term OR username LIKE :search_term";
    $params[':search_term'] = "%$search_term%";
}
//ajz27 8/5
```

Caption (required) ✓

Describe/highlight what's being shown

filter/sort

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The code uses a query to find partial matches from the database using LIKE.

COLLAPSE

Task #3 - Points: 1

Text: Add related links

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Include the heroku prod link for the page that creates the association
☐ #2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1

https://github.com/ajz27/ajz27-IT202-MC/pull/40

URL

https://github.com/ajz27/ajz27-IT202-MC/pull/4

URL #2

https://it202-ajz27-prod-be806f4746c9.herokuapp.com/Project/admin/admin_list_users.php

URL

https://it202-ajz27-prod-be806f4746c9.herokuap

+ ADD ANOTHER URL

Unassociated Page (2 pts.)

COLLAPSE

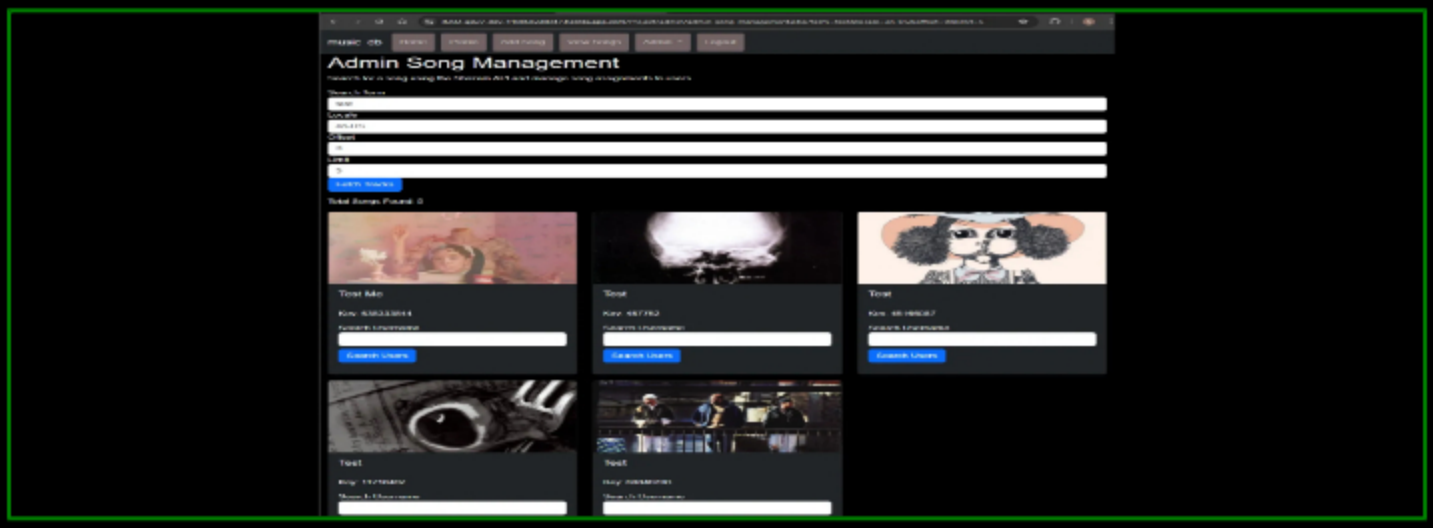
Task #1 - Points: 1

Text: Screenshots of this page

Details:
Make sure the heroku dev url is visible in the address bar

Some screenshots may be repeated in subtasks, but ensure they highlight the specific subtask requirement.

#1) Show the summary of the results with relevant information per entity

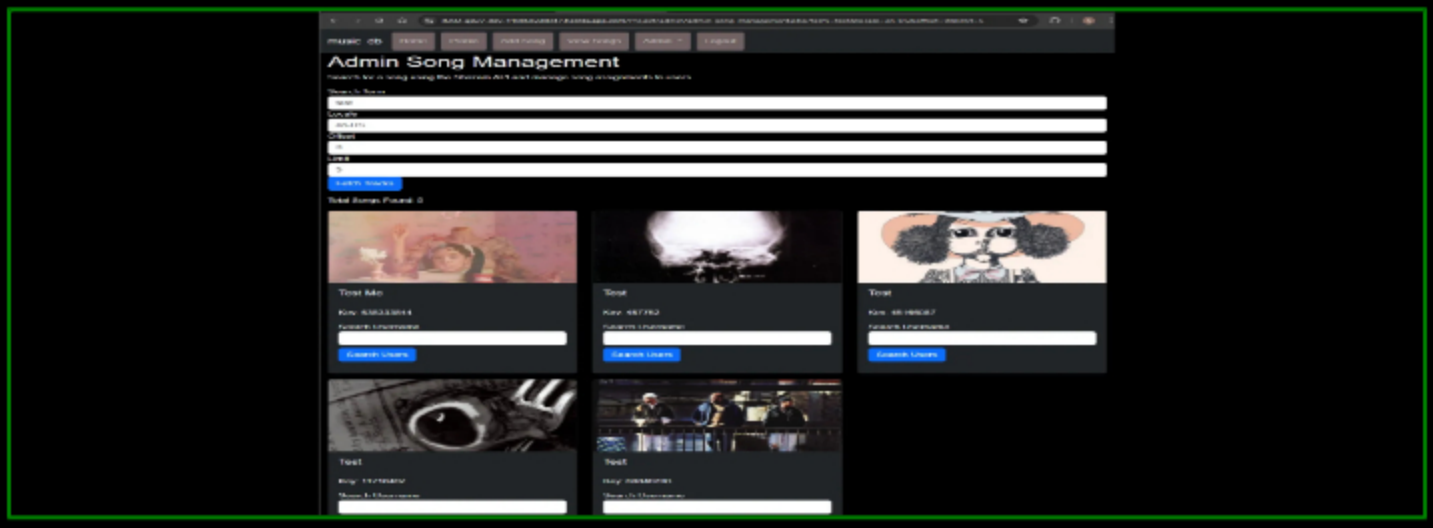


Caption (required) ✓
Describe/highlight what's being shown
Admin song management page

#2) Show the single view buttons/links

Caption (required)
Describe/highlight what's being shown
Missing caption

#3) Show variations of the number of shown items count and show the count of total number of associated items



Caption (required) ✓

Describe/highlight what's being shown

Admin song management page

#4) Show variations of the filter/sort including no results (proper message should be visible)

Caption (required)

Describe/highlight what's being shown

Missing caption

COLLAPSE

Task #2 - Points: 1

Text: Screenshot the code

Details:

Include ucid/date comments for each code screenshot

#1) Show the code related to fetching all unassociated entities (including the query)

```php
if (isset($_GET["term"])) {
    $term = $_GET["term"];
    $locale = isset($_GET["locale"]) ? $_GET["locale"] : "en-US";
    $offset = isset($_GET["offset"]) ? intval($_GET["offset"]) : 0;
    $limit = isset($_GET["limit"]) ? intval($_GET["limit"]) : 5;

    $data = [
        "term" => $term,
        "locale" => $locale,
        "offset" => $offset,
        "limit" => $limit
    ]; <- #19-24 $data =
    $endpoint = "https://shazam.p.rapidapi.com/search";
    $isRapidAPI = true;
    $rapidAPIHost = "shazam.p.rapidapi.com";

    $result = get($endpoint, "STOCK_API_KEY", $data, $isRapidAPI, $rapidAPIHost);

    error_log("Response: " . var_export($result, true));
    if (se($result, "status", 400, false) == 200 && isset($result["response"])) {
        $result = json_decode($result["response"], true);
        $songCount = count($result['tracks']['hits']);
    } else {
        $result = [];
    }
} <- #13-38 if (isset($_GET["term"]))
```

Caption (required) ✓

Describe/highlight what's being shown

code

Explanation (required) ✓

Explain in concise steps how this logically works and mention how you determine the result list (include the unassociated logic and filters)

PREVIEW RESPONSE

Since my project requires to pull data from the API, the unassociated entities are simply results that have pulled from the API that are not associated to a user already.

#2) Show the code related to the display of the results

Caption (required) ✓
Describe/highlight what's being shown
display code

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The code displays the search results, followed by the users who have saved the song as well as the username search and assignment form.

#3) Each record should have a button/link for single view

Caption (required) ✓
Describe/highlight what's being shown
not complete

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
not complete

#4) Show the logic related to the count of all unassociated items (even the ones not shown in the filtered results)

Caption (required)
Describe/highlight what's being shown
Missing caption

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
not complete

#5) Show the logic related to the count of the items on the page (this value should change based on the filter applied)

```
if (se($result, "status", 400, false) == 200 && isset($result["response"])) {
    $result = json_decode($result["response"], true);
    $songCount = count($result['tracks']['hits']);
} else {
    $result = [];
}
//ajz27 8/5
```

Caption (required) ✓
Describe/highlight what's being shown
result count

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This code counts the results returned from the API and displays it on the page.

#6) Show the logic related to filter/sort (should include a partial match for username) (limit should be constrained to 1-100 otherwise default to 10)

```php
function get_users_by_username($db, $username) {
    $query = "SELECT id, username FROM Users WHERE username LIKE :username";
    $stmt = $db->prepare($query);
    $stmt->execute([":username" => "%$username%"]);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
//ajz27 8/5 <- #71-76 function get_users_by_username($db, $username)
```

Caption (required) ✓

Describe/highlight what's being shown

partial match code

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The code uses a query to select a username that is similar to one in teh database.

COLLAPSE

Task #3 - Points: 1

Text: Add related links

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Include the heroku prod link for the page that creates the association
☐ #2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1

https://it202-ajz27-prod-be806f4746c9.herokuapp.com/Project/admin/admin_song_management.php

Url https://it202-ajz27-prod-be806f4746c9.herokuap

URL #2

https://github.com/ajz27/ajz27-IT202-MC/pull/39

Url https://github.com/ajz27/ajz27-IT202-MC/pull/3

+ ADD ANOTHER URL

COLLAPSE

Admin Association Management (like UserRoles) (1 pt.)

COLLAPSE

Task #1 - Points: 1

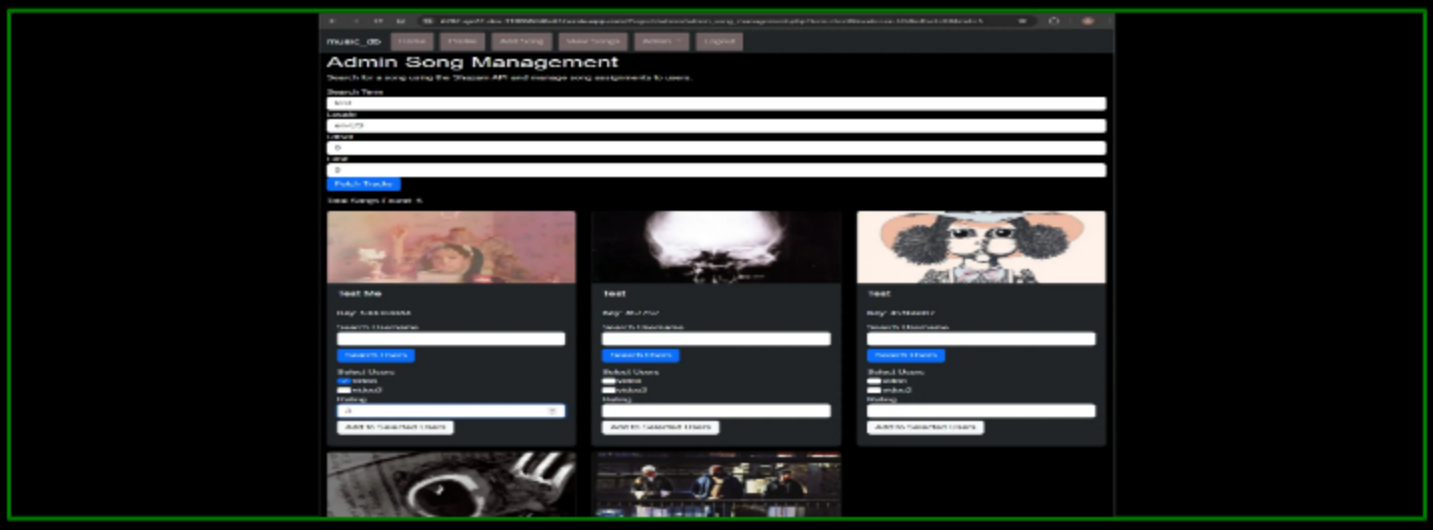
Text: Screenshots of the page

Details:

Make sure the heroku dev url is visible in the address bar

Some screenshots may be repeated in subtasks, but ensure they highlight the specific subtask requirement.

#1) Show the search form with valid data

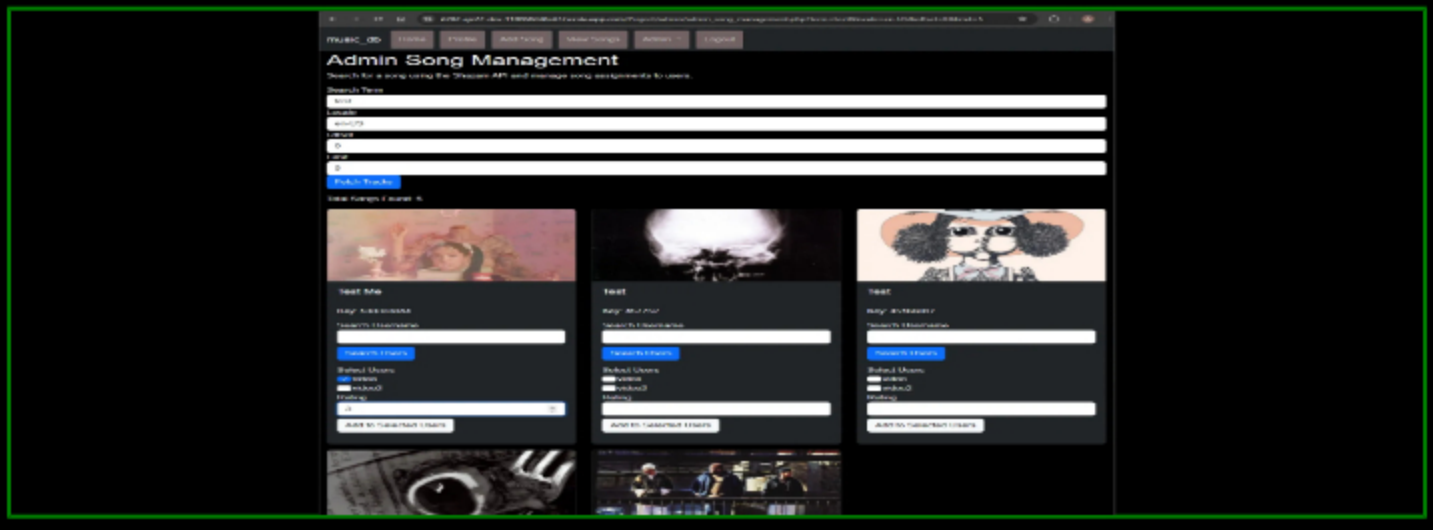


Caption (required) ✓

Describe/highlight what's being shown

Valid search data

#2) Show the results of the search

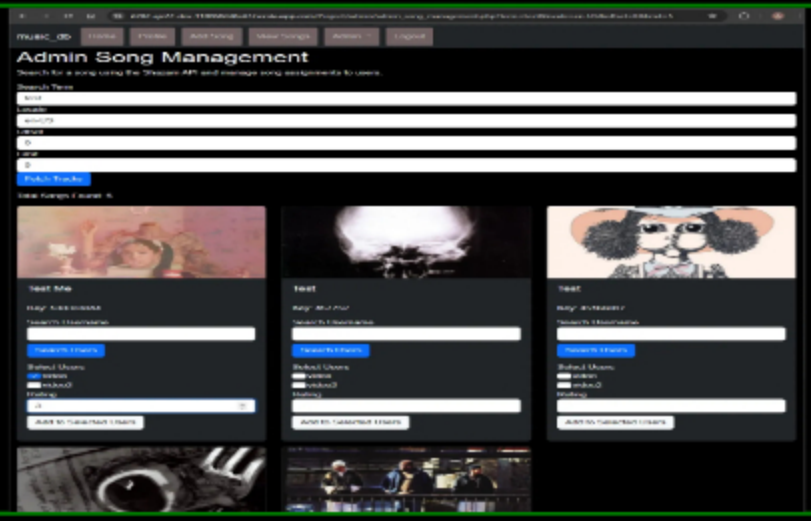


Caption (required) ✓

Describe/highlight what's being shown

search results

#3) Show the result of entities and users being associated and unassociated



Caption (required) ✓

Describe/highlight what's being shown

searhc results

^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshots of the code

Details:

Include ucid/date comments for each code screenshot

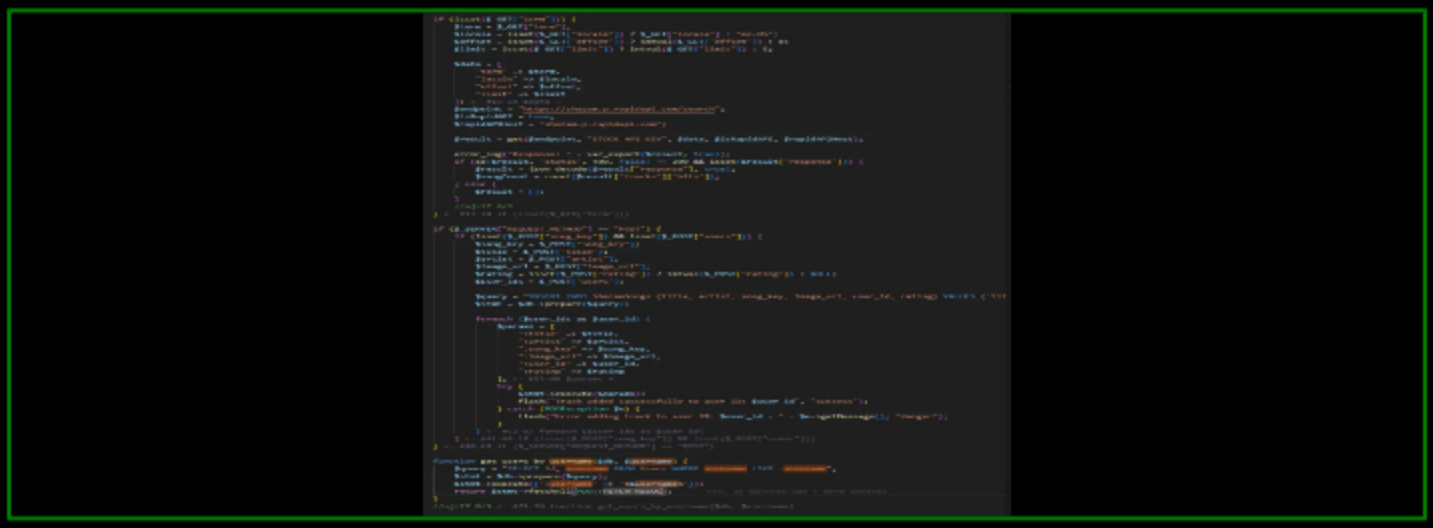
#1) Search form field for finding a partial match of usernames

```
        <div class="form-group">
            <label for="username_search_<?php echo htmlspecialchars($track['track']['key'] ?? ''); ?>">Searcl
            <input type="text" name="username_search" id="username_search_<?php echo htmlspecialchars($track
            <button type="submit" class="btn btn-primary mt-2">Search Users</button>
        </div>
    </form>
    <!-- ajz27 8/5 -->
```

Caption (required) ✓
Describe/highlight what's being shown
partial match of usernames

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This works similarly to the other admin page, where a form is used to find a partial match.

#2) Search form field for finding a partial match of entities



Caption (required) ✓
Describe/highlight what's being shown
partial match of entities

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This is done through the API, which already has built in partial matching.

#3) Code related to getting a max of 25 results for each field (i.e., 25 users and 25 entities limit)

Caption (required) ✓

Describe/highlight what's being shown

incomplete

Explanation (required) ✓

Explain in concise steps how this logically works and describe the steps for the search and how it works for users and entities

PREVIEW RESPONSE

incomplete

#4) Code that generates the checkboxes next to each list (users and entities)

```
<div class="form-group">
    <label>Select Users</label>
    <?php foreach ($users as $user) : ?>
        <div class="form-check">
            <input type="checkbox" class="form-check-input" name="users[]" value="<?php echo $user['id']; ?>"
            <label class="form-check-label"><?php echo htmlspecialchars($user['username']); ?></label>
        </div>
    <?php endforeach; ?>
</div>
<!-- ajz27 8/5 -->
```

Caption (required) ✓

Describe/highlight what's being shown

checkboxes

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

This uses a checkbox input in the card to select users to add a song to.

#5) Code related to submitting the checkbox lists

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST["song_key"]) && isset($_POST["users"])) {
        $song_key = $_POST["song_key"];
        $title = $_POST["title"];
        $artist = $_POST["artist"];
        $image_url = $_POST["image_url"];
        $rating = isset($_POST["rating"]) ? intval($_POST["rating"]) : NULL;
        $user_ids = $_POST["users"];

        $query = "INSERT INTO ShazamSongs (title, artist, song_key, image_url, user_id, rating) VALUES (:tit
        $stmt = $db->prepare($query);

        foreach ($user_ids as $user_id) {
            $params = [
                ":title" => $title,
                ":artist" => $artist,
                ":song_key" => $song_key,
                ":image_url" => $image_url,
                ":user_id" => $user_id,
                ":rating" => $rating
            ]; <- #53-60 $params =
            try {
                $stmt->execute($params);
                flash("Track added successfully to user ID: $user_id", "success");
            } catch (PDOException $e) {
                flash("Error adding track to user ID: $user_id - " . $e->getMessage(), "danger");
            }
        } <- #52-67 foreach ($user_ids as $user_id)
    } <- #41-68 if (isset($_POST["song_key"]) && isset($_POST["users"]))
}
//ajz27 8/5 <- #40-69 if ($_SERVER["REQUEST_METHOD"] == "POST")
```

Caption (required) ✓

Describe/highlight what's being shown

checkbox lists

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

When teh form is submitted, it checks if song_key and users are set in the $POST array. Then the insert query is prepared with placeholders for song details and user IDs. For each selected user ID, the query is executed to insert the song details into ShazamSongs with the user ID and rating.

#6) Code related to applying the associations upon submission (i.e., add the relationship if it doesn't exist and remove the relationship if it does exist)

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    if (isset($_POST["song_key"]) && isset($_POST["users"])) {
        $song_key = $_POST["song_key"];
        $title = $_POST["title"];
        $artist = $_POST["artist"];
        $image_url = $_POST["image_url"];
        $rating = isset($_POST["rating"]) ? intval($_POST["rating"]) : NULL;
        $user_ids = $_POST["users"];

        $query = "INSERT INTO ShazamSongs (title, artist, song_key, image_url, user_id, rating) VALUES (:tit
        $stmt = $db->prepare($query);

        foreach ($user_ids as $user_id) {
            $params = [
                ":title" => $title,
                ":artist" => $artist,
                ":song_key" => $song_key,
                ":image_url" => $image_url,
                ":user_id" => $user_id,
                ":rating" => $rating
            ]; <- #53-60 $params =
            try {
                $stmt->execute($params);
                flash("Track added successfully to user ID: $user_id", "success");
            } catch (PDOException $e) {
                flash("Error adding track to user ID: $user_id - " . $e->getMessage(), "danger");
            }
        } <- #52-67 foreach ($user_ids as $user_id)
    } <- #41-68 if (isset($_POST["song_key"]) && isset($_POST["users"]))
}
//ajz27 8/5 <- #40-69 if ($_SERVER["REQUEST_METHOD"] == "POST")
```

Caption (required) ✓

Describe/highlight what's being shown

submission code

Explanation (required) ✓

Explain in concise steps how this logically works and describe the steps for the associate/unassociate logic for the combination of users and entities

PREVIEW RESPONSE

When teh form is submitted, it checks if song_key and users are set in the $POST array. Then the insert query is prepared with placeholders for song details and user IDs. For each selected user ID, the query is executed to insert the song details into ShazamSongs with the user ID and rating. This is the same code as above because the unassociation is in the admin page.

COLLAPSE

Task #3 - Points: 1

Text: Add related links

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Include the heroku prod link for the page that creates the association
☐ #2	1	Add the pull request link for the branch related to this feature Note: the link should end vpitll/#. Same pull request shouldn't be used for each feature

URL #1
https://github.com/ajz27/ajz27-IT202-MC/pull/40

URL
https://github.com/ajz27/ajz27-IT202-MC/pull/4

URL #2
https://it202-ajz27-prod-be806f4746c9.herokuapp.com/Project/admin/admin_song_management.php

URL
https://it202-ajz27-prod-be806f4746c9.herokuap

ADD ANOTHER URL

COLLAPSE

Misc (1 pt.)

COLLAPSE

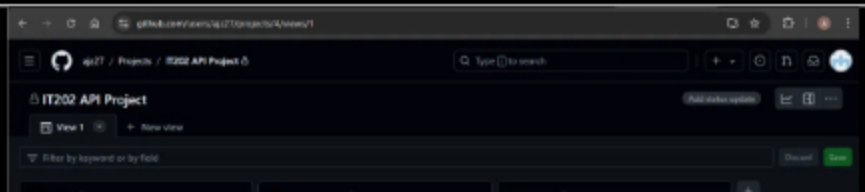
Task #1 - Points: 1

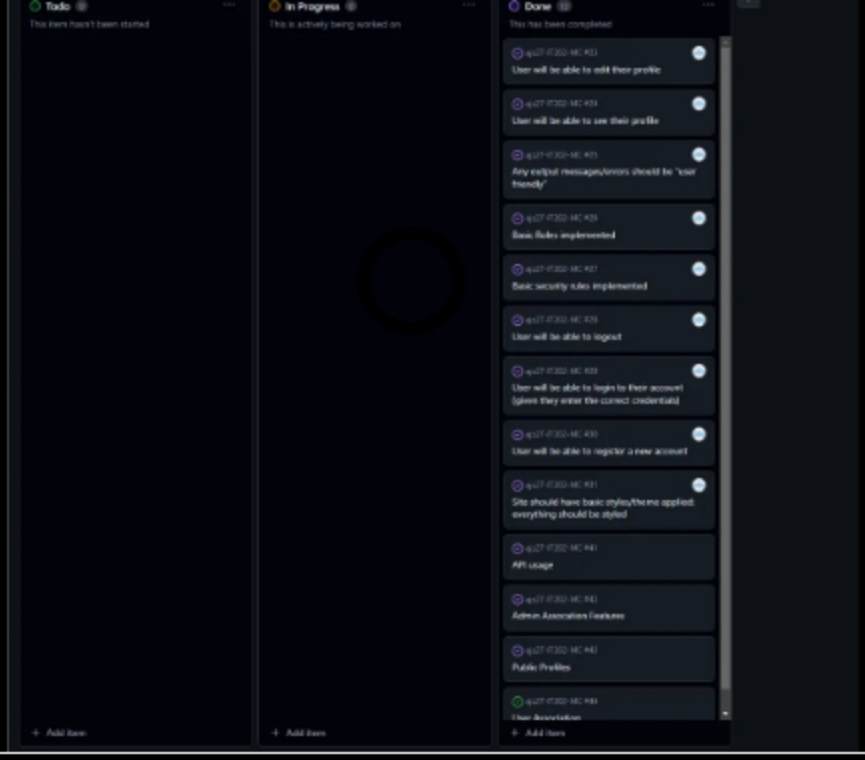
Text: Screenshot of your project board from GitHub (tasks should be in the proper column)

Task Screenshots:

Gallery Style: Large View

Small Medium Large





Missing Caption

COLLAPSE

Task #2 - Points: 1

Text: Provide a direct link to the project board on GitHub

URL #1

https://github.com/users/ajz27/projects/4/views/1

url

https://github.com/users/ajz27/projects/4/view

+ ADD ANOTHER URL

COLLAPSE

Task #3 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

This milestone and project as a whole were very challenging. However, I did learn quite a bit about web development. While its not something I would do as a career, it is good to have the knowledge.

COLLAPSE

Task #4 - Points: 1

Text: WakaTime Screenshot

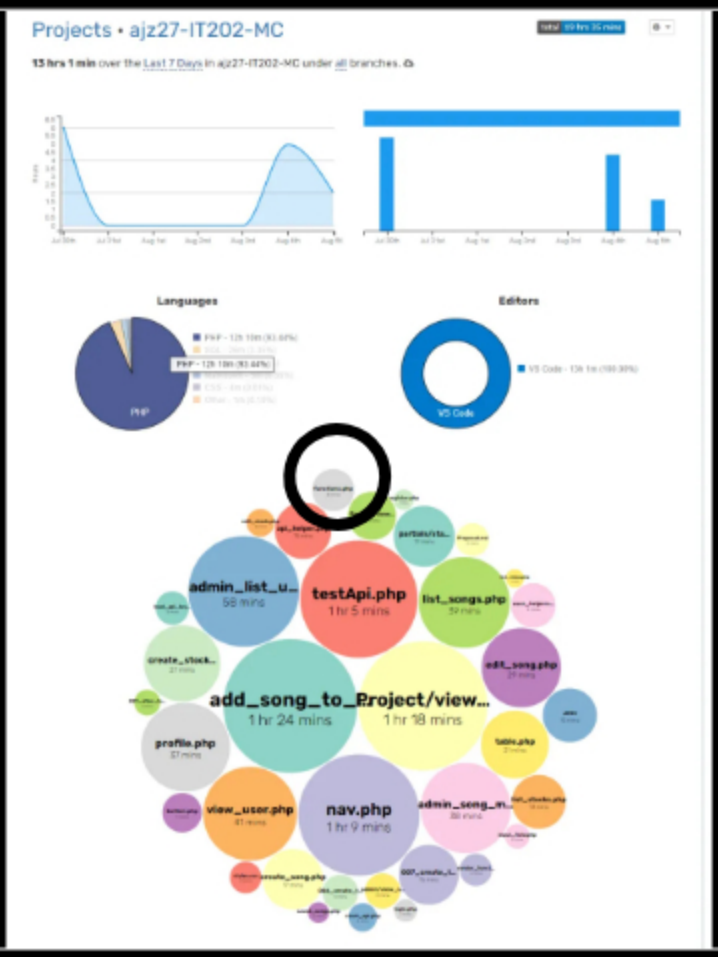
Details:

Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

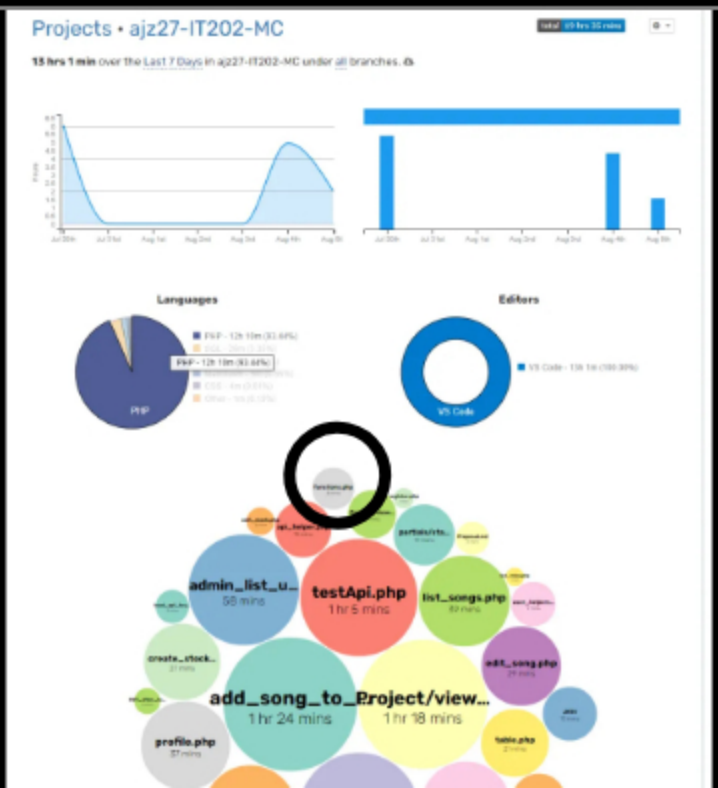
Task Screenshots:

Gallery Style: Large View

Small Medium Large



Missing Caption



Missing Caption

End of Assignment