Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-452-M2024/it202-api-project-milestone-2-2024-m24/grade/ajz27

IT202-452-M2024 - [IT202] API Project Milestone 2 2024 M24

Submissions:

Submission Selection

1 Submission [active] 7/30/2024 10:36:54 PM

Instructions

COLLAPSE

Implement the Milestone 2 features from the project's proposal document: https://docs.google.com/document/d/1XE96a8DQ52Vp49XACBDTNCq0xYDt3kF29cO88EWVwfo/view Make sure you add your ucid/date as code comments where code changes are done All code changes should reach the Milestone2 branch Create a pull request from Milestone2 to dev and keep it open until you get the output PDF from this assignment. Gather the evidence of feature completion based on the below tasks. Once finished, get the output PDF and copy/move it to your repository folder on your local machine. Run the necessary git add, commit, and push steps to move it to GitHub Complete the pull request that was opened earlier Create and merge a pull request from dev to prod Upload the same output PDF to Canvas

Branch name: Milestone2

Tasks: 23 Points: 10.00

Define Appropriate Tables for Data (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

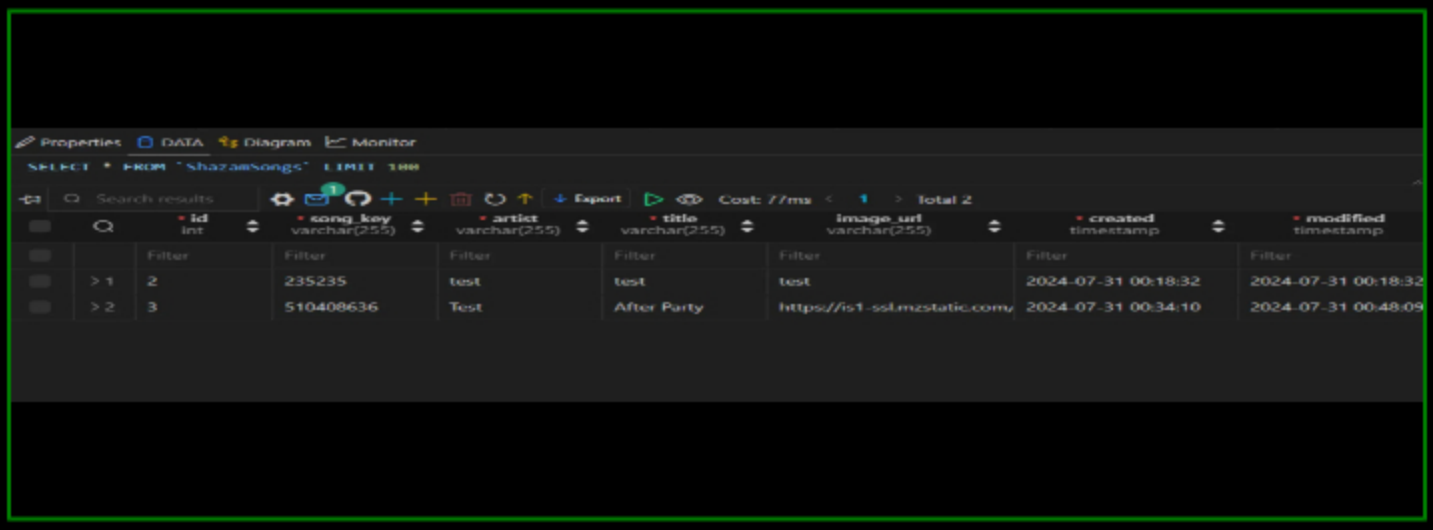
Text: Screenshots of Table SQL

Checklist *The checkboxes are for your own tracking

#	Points	Details

☐ #1	1	Table(s) should have the 3 core columns we'll always be using (id, created, modified) plus additional columns for the incoming API data
☐ #2	1	Columns should be logical and thought out (not valid to have a single field of JSON data or similar

#1) Screenshot of the table (you can duplicate this subtask as needed)



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

Explanation (required) ✓
Explain the columns and what data they represent (briefly), also note any normalization that may have been necessary
PREVIEW RESPONSE
id - entry identification
song_key - the key for a specific song as assigned by shazam
artist - artist of song
title - title of song
image_url - the url as given by shazam for the cover art of the song
created/modified - timestamps

COLLAPSE

Task #2 - Points: 1
Text: Add the pull request link for the branch related to this feature

Details:
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/35

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

+ ADD ANOTHER URL

Data Creation Page (2 pts.)

^COLLAPSE^

^COLLAPSE^

Task #1 - Points: 1

Text: Screenshots of the creation page

Details:
Heroku dev url must be visible in all relevant screenshots

#1) Show potentially valid data filled in for the custom creation page

Caption (required) ✓
Describe/highlight what's being shown
valid creation data

#2) Show how the API data is fetched for API data (must be server-side)

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

```php
$data = [
    "term" => $term,
    "locale" => $locale,
    "offset" => $offset,
    "limit" => $limit
]; <- #14-19 $data =
$endpoint = "https://shazam.p.rapidapi.com/search";
$isRapidAPI = true;
$rapidAPIHost = "shazam.p.rapidapi.com";
// ajz27 7/29
```

Caption (required) ✓

Describe/highlight what's being shown

fetch api code

Explanation (required) ✓

Briefly explain the code

PREVIEW RESPONSE

The paramters $term, $locale, $offset, and $limit are fetched from teh URL parameters with default values assigned if they are not provided. these parameters build the $data array which specifies the search criteria foor the API request.

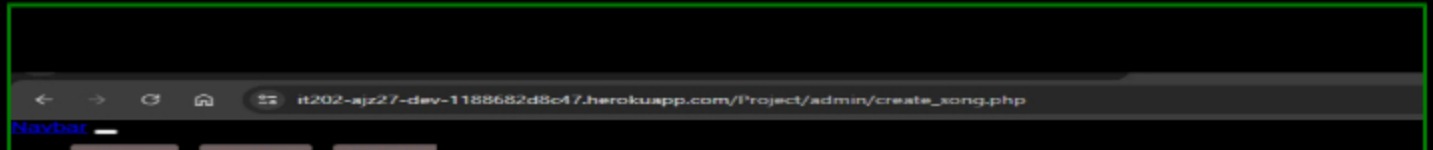
#3) Show examples of validation messages

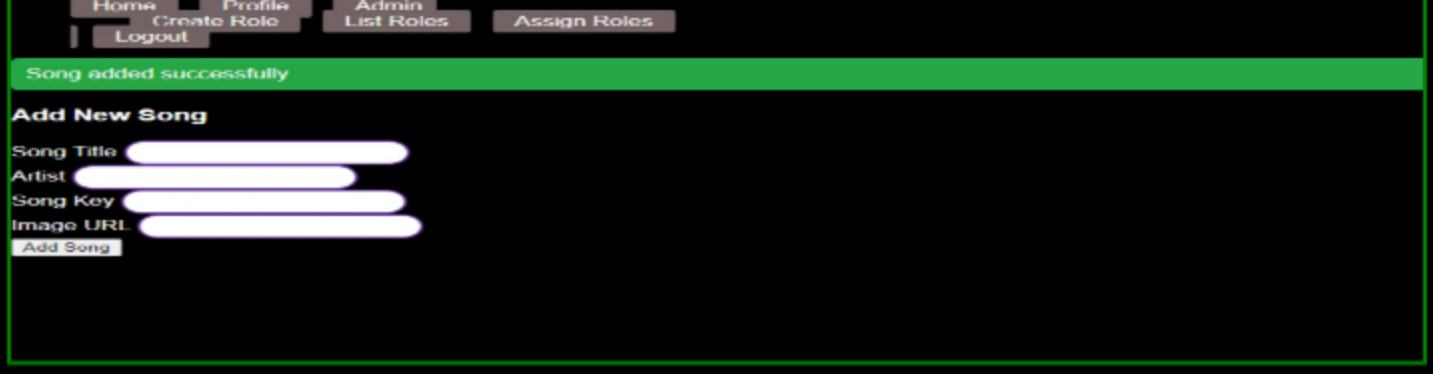
Caption (required) ✓

Describe/highlight what's being shown

track added successfully

#4) Show an example of successful creation message





Caption (required) ✓

Describe/highlight what's being shown

creation successful

#5) Design/Style should be considered (i.e., bootstrap, custom css, etc)

Caption (required) ✓

Describe/highlight what's being shown

styling: title, subtitle, picture embed

Explanation (required) ✓

Briefly explain your design choices

PREVIEW RESPONSE

The design here is organized to show the song name, artist name, key for the song, and the cover art for the song.

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of creation page code

Details:

Include ucid/date comments for each code screenshot

#1) Form should have correct data types for each property being requested

```
<div class="container-fluid">
    <h3>Add New Song</h3>
    <form method="POST">
        <div class="mb-3">
            <label for="title" class="form-label">Song Title</label>
            <input type="text" class="form-control" id="title" name="title" required>
        </div>
        <div class="mb-3">
            <label for="artist" class="form-label">Artist</label>
            <input type="text" class="form-control" id="artist" name="artist" required>
        </div>
        <div class="mb-3">
            <label for="song_key" class="form-label">Song Key</label>
            <input type="text" class="form-control" id="song_key" name="song_key" required>
        </div>
        <div class="mb-3">
            <label for="image_url" class="form-label">Image URL</label>
            <input type="text" class="form-control" id="image_url" name="image_url" required>
        </div>
        <button type="submit" class="btn btn-primary">Add Song</button>
    </form>
</div>
//ajz27 7/29
```

Caption (required) ✓

Describe/highlight what's being shown

form code for data creation

Explanation (required) ✓

Briefly talk about each field type and why it was chosen

PREVIEW RESPONSE

The only fields necessary were text fields as each column only needs text.

#2) Form should have correct validation for each field (HTML, JS, and PHP)

```
    try {
        $stmt = $db->prepare($query);
        $stmt->execute($params);
        flash("Song added successfully", "success");
    } catch (PDOException $e) {
        if ($e->errorInfo[1] == 1062) {
            flash("A song with the same artist and title already exists", "warning");
        } else {
            error_log("Error adding song: " . var_export($e, true));
            flash("An error occurred", "danger");
        }
    } // #28-55 catch (PDOException $e)
} // #9-26 if ($_SERVER["REQUEST_METHOD"] == "POST")
?>

<div class="container-fluid">
    <h3>Add New Song</h3>
    <form method="POST">
        <div class="mb-3">
            <label for="title" class="form-label">Song Title</label>
            <input type="text" class="form-control" id="title" name="title" required>
        </div>
        <div class="mb-3">
            <label for="artist" class="form-label">Artist</label>
            <input type="text" class="form-control" id="artist" name="artist" required>
        </div>
        <div class="mb-3">
            <label for="song_key" class="form-label">Song Key</label>
            <input type="text" class="form-control" id="song_key" name="song_key" required>
        </div>
        <div class="mb-3">
            <label for="image_url" class="form-label">Image URL</label>
            <input type="text" class="form-control" id="image_url" name="image_url" required>
        </div>
        <button type="submit" class="btn btn-primary">Add Song</button>
    </form>
</div>
//ajz27 7/29
```

Caption (required) ✓

Describe/highlight what's being shown
form validation

Explanation (required) ✓
Briefly explain the validations

PREVIEW RESPONSE

The PHP validation is done by checking if a song with the same artist and title already exist. In addition, there is an HTML validation in requiring each field to be filled before it can be submitted successfully.

#3) Successful creation should have a user-friendly message

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $title = $_POST["title"];
    $artist = $_POST["artist"];
    $song_key = $_POST["song_key"];
    $image_url = $_POST["image_url"];

    $db = getDB();
    $query = "INSERT INTO ShazamSongs (title, artist, song_key, image_url) VALUES (:title, :artist, :song_ke
    $params = [
        ":title" => $title,
        ":artist" => $artist,
        ":song_key" => $song_key,
        ":image_url" => $image_url
    ]; <- #17-22 $params =

    try {
        $stmt = $db->prepare($query);
        $stmt->execute($params);
        flash("Song added successfully", "success");
    } catch (PDOException $e) {
        if ($e->errorInfo[1] == 1062) {
            flash("A song with the same artist and title already exists", "warning");
        } else {
            error_log("Error adding song: " . var_export($e, true));
            flash("An error occurred", "danger");
        }
    } <- #28-35 catch (PDOException $e)
} <- #9-36 if ($_SERVER["REQUEST_METHOD"] == "POST")
//ajz27 7/29
```

Caption (required) ✓
Describe/highlight what's being shown
successful creation message

Explanation (required) ✓
Explain how duplicate/existing data is handled

PREVIEW RESPONSE

A check is added in the table that prevents two entries with the same artist and title existing. This is reflected to the user by catching a PDO exception and flashing the message.

#4) Any errors should have user-friendly messages

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $title = $_POST["title"];
    $artist = $_POST["artist"];
    $song_key = $_POST["song_key"];
    $image_url = $_POST["image_url"];

    $db = getDB();
    $query = "INSERT INTO ShazamSongs (title, artist, song_key, image_url) VALUES (:title, :artist, :song_ke
    $params = [
        ":title" => $title,
        ":artist" => $artist,
        ":song_key" => $song_key,
        ":image_url" => $image_url
    ]; <- #17-22 $params =

    try {
        $stmt = $db->prepare($query);
```

```
            $stmt->execute($params);
            flash("Song added successfully", "success");
        } catch (PDOException $e) {
            if ($e->errorInfo[1] == 1062) {
                flash("A song with the same artist and title already exists", "warning");
            } else {
                error_log("Error adding song: " . var_export($e, true));
                flash("An error occurred", "danger");
            }
        } <- #28-35 catch (PDOException $e)
    } <- #9-36 if ($_SERVER["REQUEST_METHOD"] == "POST")
    //ajz27 7/29
```

Caption (required) ✓

Describe/highlight what's being shown

error messages

Explanation (required) ✓

Briefly describe the scenarios

PREVIEW RESPONSE

Error messages are added if an entry cannot be submitted.

#5) Include the form/process for fetching API data

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Briefly explain the steps (include how duplicates are handled)

PREVIEW RESPONSE

Missing text

#6) Include some indicator between custom data and API data

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

Explanation (required)
Briefly mention what the indicator is (i.e., api_id if the API has ids or is_api as a boolean-like column, etc)
PREVIEW RESPONSE
Missing text

#7) Include any other rules like role guards and login checks

```
if (!has_role("Admin")) {
    flash("You don't have permission to view this page", "warning");
    die(header("Location: $BASE_PATH" . "/home.php"));
}
//ajz27 7/29
```

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

Explanation (required) ✓
Briefly explain the logic/reasoning
PREVIEW RESPONSE
The create custom entry is temporarily role guarded by the admin role. This will change in the future.

COLLAPSE

Task #3 - Points: 1

Text: Screenshot of records from DB

#1) Show at least one record fetched from the API

	> 3	4	530552117	Pierce Fulton	It's Only 10 Days...	https://is1-ssl.mzstatic.com/ 2024-07

Caption (required) ✓

Describe/highlight what's being shown (note what differs from a custom record)

record from db, note there is a cover art pulled from the api

#2) Show at least one record created via the creation form

	> 4	5	332589235890235890	test2	test2	testing 2024-07

Caption (required) ✓

Describe/highlight what's being shown (note what differs from the API record)

record from creation form, there is no cover art pulled

^COLLAPSE ^

Task #4 - Points: 1

Text: Add related links

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Include the heroku prod link for this page
☐ #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/35

Url

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

+ ADD ANOTHER URL

^COLLAPSE ^

Data List Page (many entities) (2 pts.)

^COLLAPSE ^

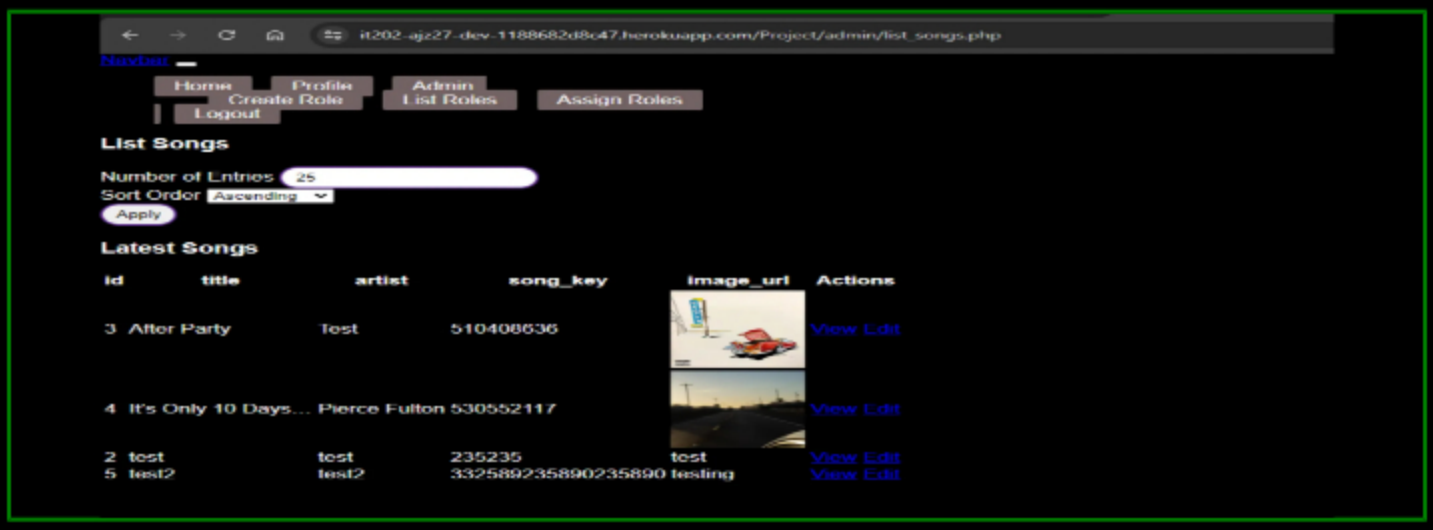
Task #1 - Points: 1

Text: Screenshots of the list page

Details:

Heroku dev url must be visible in all relevant screenshots

#1) Show the page of your entities listed (have a reasonable number shown)



Caption (required) ✓

Describe/highlight what's being shown

list songs

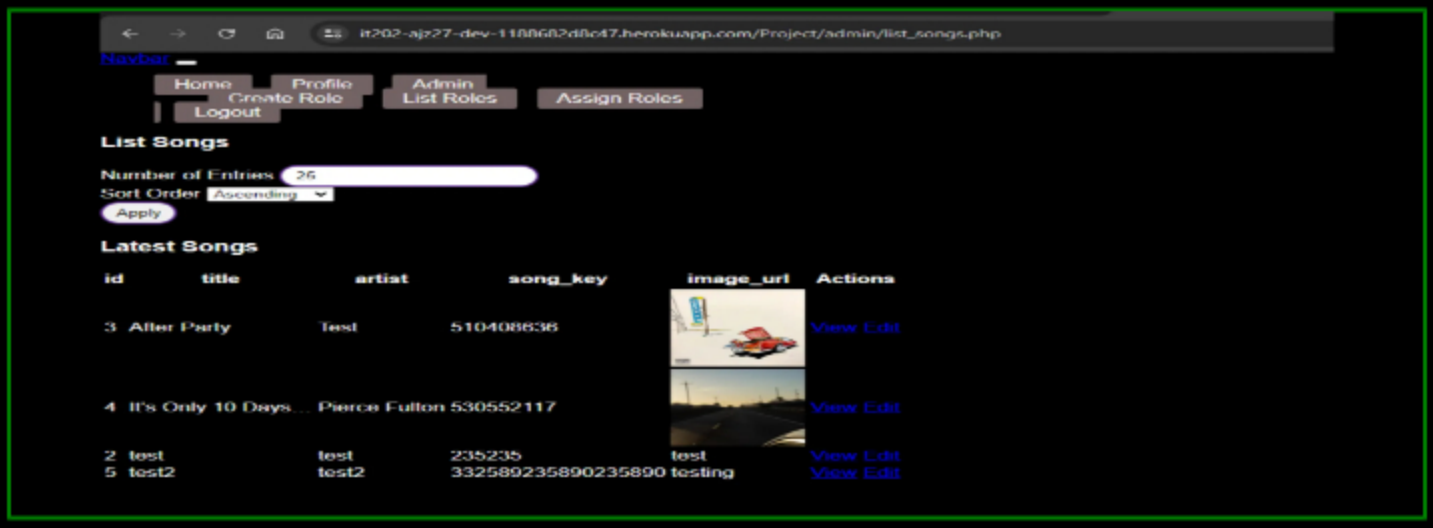
Explanation (required) ✓

Briefly describe the page

PREVIEW RESPONSE

List all songs currently stored in the database.

#2) Show the filter/sort form based on your data and the required limit field



Caption (required) ✓

Describe/highlight what's being shown

sort, limit field

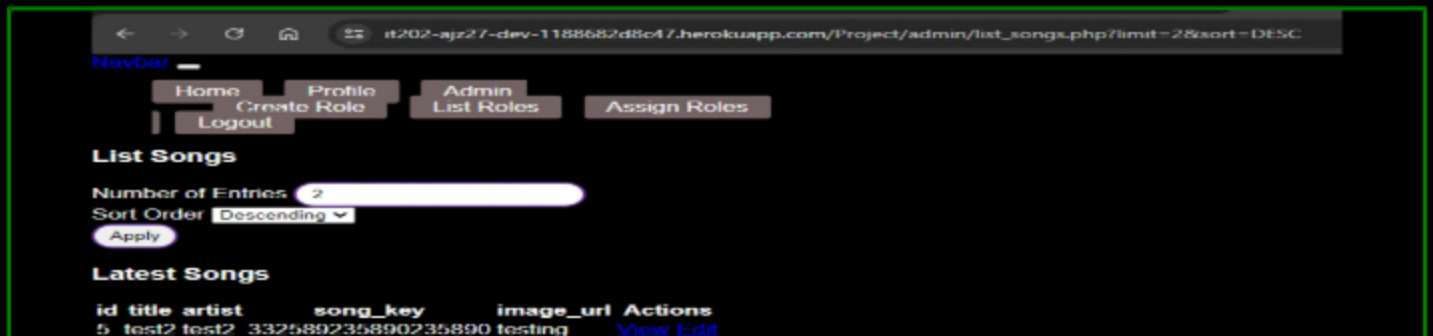
Explanation (required) ✓

Briefly mention what's available to the user

PREVIEW RESPONSE

The two fields limit the number of entries, and the sort order sorts the songs by alphabetic order.

#3) Demonstrate a few varied filters/sorts



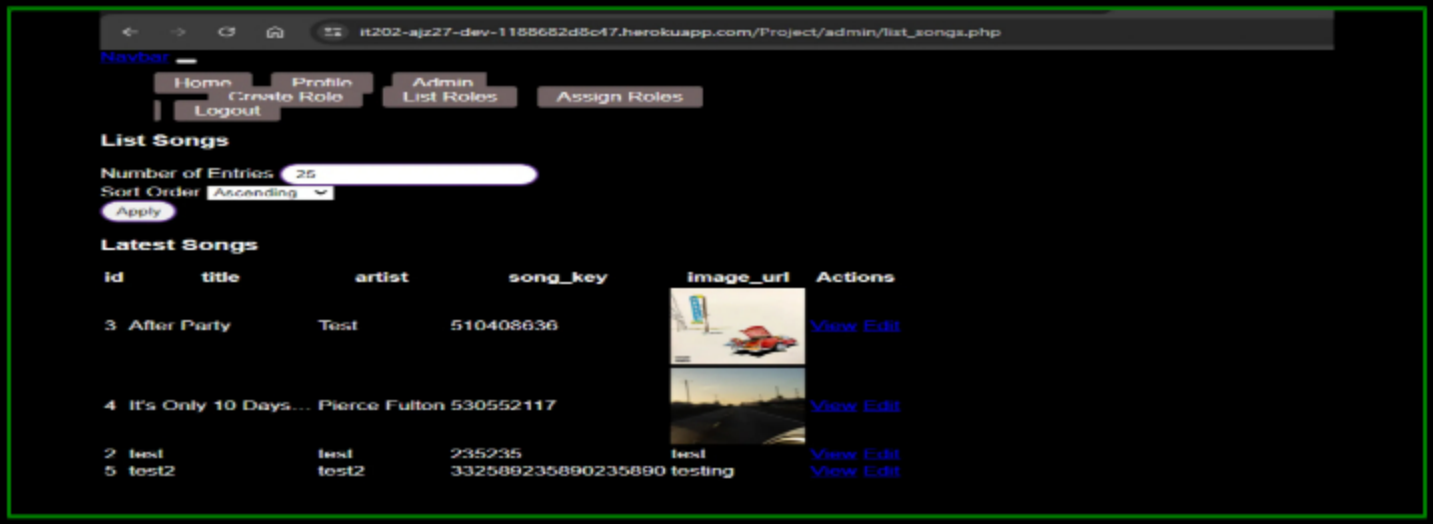
2 test test 235235 test View Edit

Caption (required) ✓
Describe/highlight what's being shown
sort changed to descending alphabetic order, limited to 2 entries

#4) Demonstrate a filter that doesn't have any records (should show an appropriate message)

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

#5) Each list item should have a link of single view (i.e., details), edit, and delete (some of which may only be visible to admin users, include examples from different user roles)

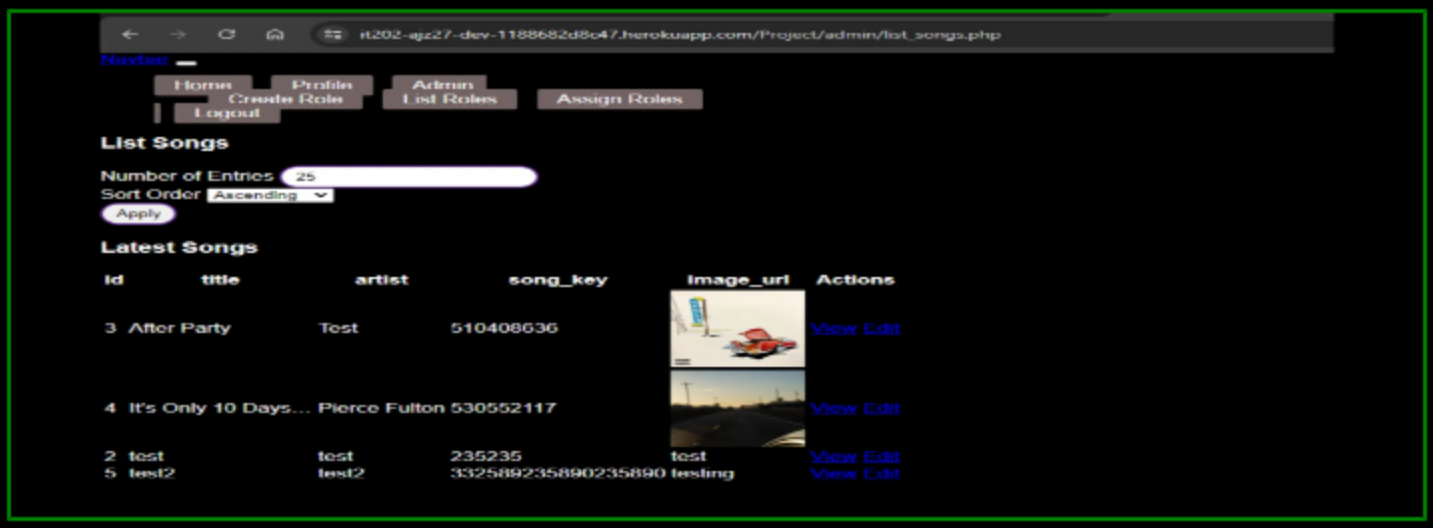


Caption (required) ✓

Describe/highlight what's being shown
view, edit, delete

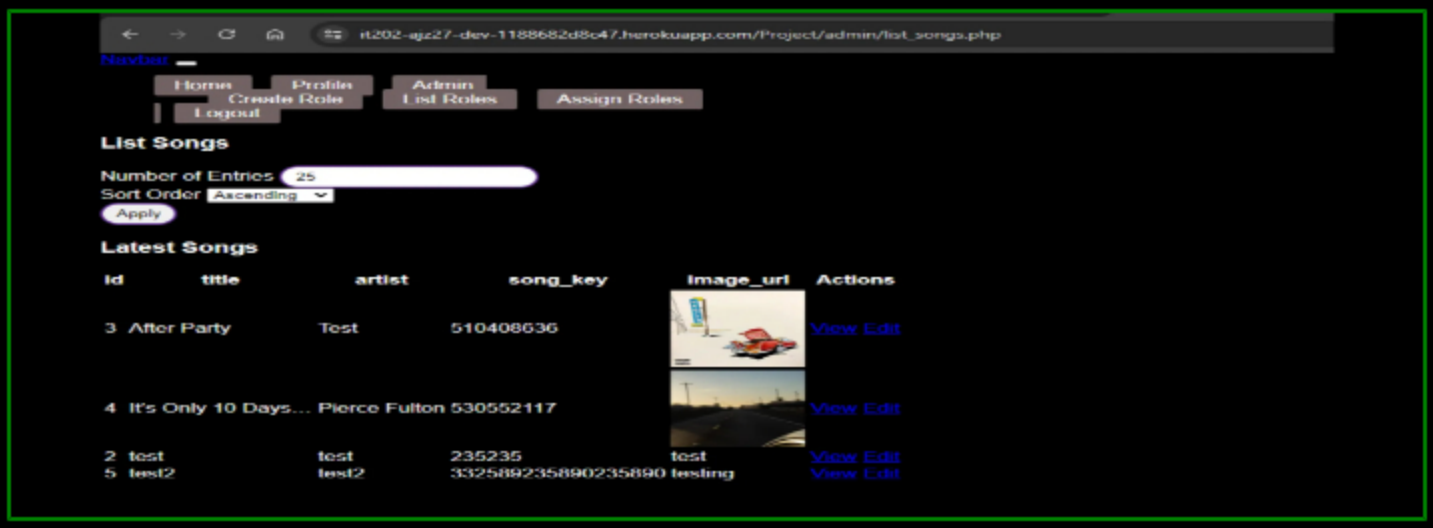
Explanation (required) ✓
Mention which users can interact with the view, edit, and delete links
PREVIEW RESPONSE
Edit/delete are combined into one page. At the moment only admin can view, edit, and delete entries.

#6) Each list item should have a summary of the entity (likely won't be the entire entity data)



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

#7) Design/Style should be considered (i.e., bootstrap, custom css, etc)



Caption (required) ✓

Describe/highlight what's being shown
design

Explanation (required) ✓
Briefly explain your design choices
PREVIEW RESPONSE
filters are kept up top, and table.php was modified to allow for an embed of the cover art.

COLLAPSE

Task #2 - Points: 1
Text: Screenshots of the list page code

Details:
Include ucid/date comments for each code screenshot

#1) Show the filter/sort form generation

```php
$limit = isset($_GET['limit']) ? intval($_GET['limit']) : 25; // default limit to 25
$sort_order = isset($_GET['sort']) ? $_GET['sort'] : 'ASC'; // default sort order to ASC

// validate sort order
$sort_order = strtoupper($sort_order) === 'DESC' ? 'DESC' : 'ASC';
    You, 8 minutes ago • more updates
$query = "SELECT id, title, artist, song_key, image_url FROM ShazamSongs ORDER BY title $
$db = getDB();
$stmt = $db->prepare($query);
$stmt->bindParam(':limit', $limit, PDO::PARAM_INT);
//ajz27 7/29
```

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

Explanation (required) ✓
Briefly explain how the code works (i.e, some stuff may be dynamic)
PREVIEW RESPONSE
The form retreives limit and sort paramters from the query string and then modifies the SQL query to include dynamic limit and order by clauses.

#2) Show the DB query and how the filter/sort is handled (including the restriction on the limit field)

```php
$limit = isset($_GET['limit']) ? intval($_GET['limit']) : 25; // default limit to 25
$sort_order = isset($_GET['sort']) ? $_GET['sort'] : 'ASC'; // default sort order to ASC

// validate sort order
$sort_order = strtoupper($sort_order) === 'DESC' ? 'DESC' : 'ASC';
    You, 8 minutes ago • more updates
$query = "SELECT id, title, artist, song_key, image_url FROM ShazamSongs ORDER BY title $
$db = getDB();
$stmt = $db->prepare($query);
$stmt->bindParam(':limit', $limit, PDO::PARAM_INT);
//ajz27 7/29
```

Caption (required) ✓

Describe/highlight what's being shown

sorting

Explanation (required) ✓

Briefly explain the related logic

PREVIEW RESPONSE

The SQL query has include dynamic limit and order by clauses.

#3) Show how the output is generated and displayed

```php
$table = [
    "data" => $results,
    "title" => "Latest Songs",
    "ignored_columns" => [],
    "edit_url" => get_url("admin/edit_song.php"),
    "view_url" => get_url("admin/view_song.php")
]; <- #34-40 $table =
?>
<div class="container-fluid">
    <h3>List Songs</h3>
    <form method="GET" class="mb-3">
        <div class="form-group">
            <label for="limit">Number of Entries</label>
            <input type="number" name="limit" id="limit" value="<?php echo htmlspecialcha
        </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' : ''; ?
                <option value="DESC" <?php echo $sort_order === 'DESC' ? 'selected' : '';
            </select>
        </div>
        <input type="submit" value="Apply" class="btn btn-primary" />
    </form>
    <?php render_table($table); ?>
</div>
<!-- ajz27 7/29 -->
```

Caption (required) ✓

Describe/highlight what's being shown

output generation

Explanation (required) ✓

Briefly explain the related logic

PREVIEW RESPONSE

The form renders the table and then outputs it in HTML.

#4) Show any restrictions like role guard or login checks

```php
require(__DIR__ . "/../../../partials/nav.php");

if (!has_role("Admin")) {
    flash("You don't have permission to view this page", "warning");
    die(header("Location: $BASE_PATH" . "/home.php"));
}
//ajz27 7/29
```

Caption (required) ✓

Describe/highlight what's being shown

role guard

Explanation (required) ✓

Briefly explain the logic/reasoning

PREVIEW RESPONSE

Only admin is allowed to view the page at the moment.

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 this page
☐ #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/35

URL

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

URL #2

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

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

+ ADD ANOTHER URL

View Details Page (single entity) (1 pt.)

^COLLAPSE^

^COLLAPSE^

Task #1 - Points: 1

Text: Screenshots of the details page

Details:

Heroku dev url must be visible in all relevant screenshots

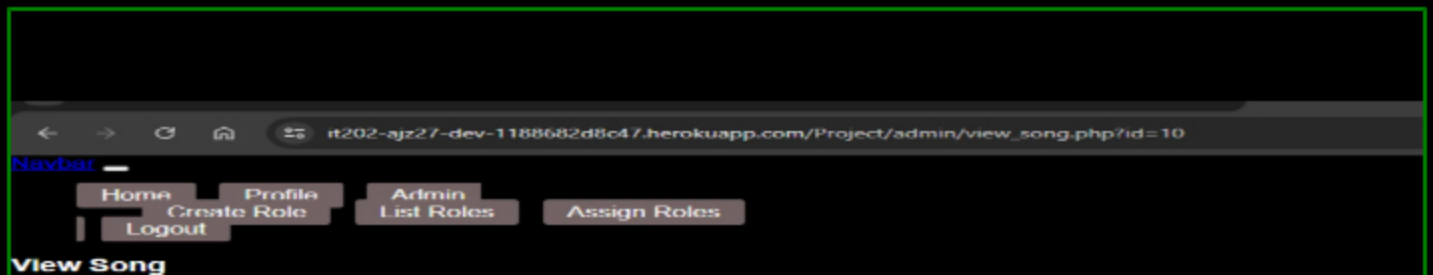
#1) Entity should be fetch by id (via the url)

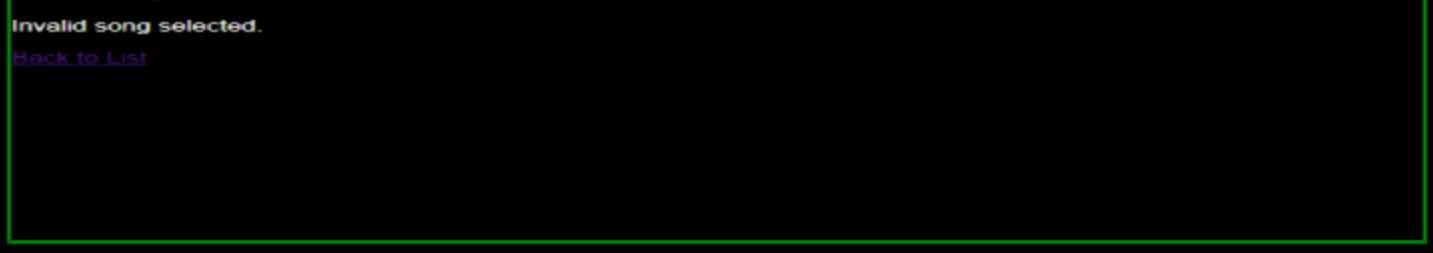
Caption (required) ✓

Describe/highlight what's being shown

single entity page

#2) A missing id should redirect back to the list page with an applicable message





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

#3) Design/Style should be considered (i.e., bootstrap, custom css, etc)

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

Explanation (required) ✓
Briefly explain your design choices
PREVIEW RESPONSE
organized vertically to make it easy to read.

#4) Data shown should be more detailed/inclusive than the summary view

Caption (required) ✓

Describe/highlight what's being shown

single entity view

Explanation (required) ✓

Briefly explain the details shown and how it differs from the list view

PREVIEW RESPONSE

Organized vertically, also shows a larger version of the cover art as well as the full url.

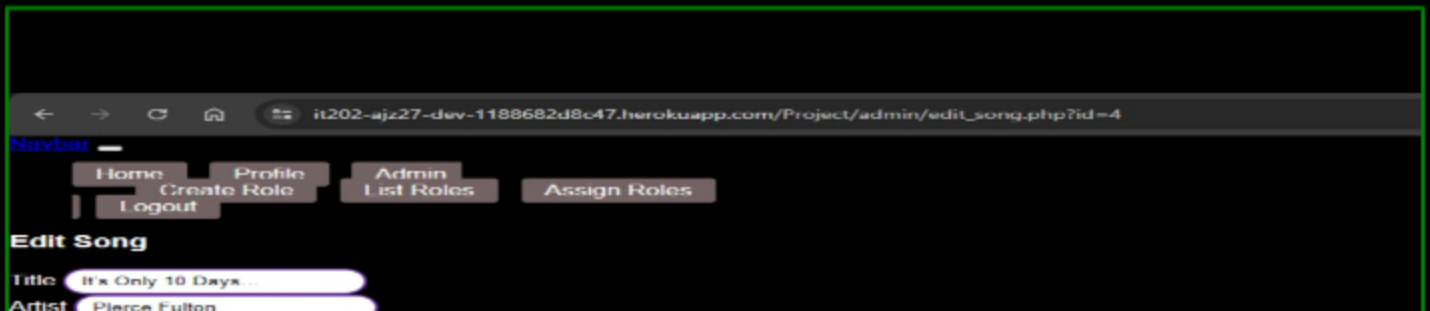
#5) There should be a link to edit the entity (this may be an admin-only thing, but it should be present for the respective role)

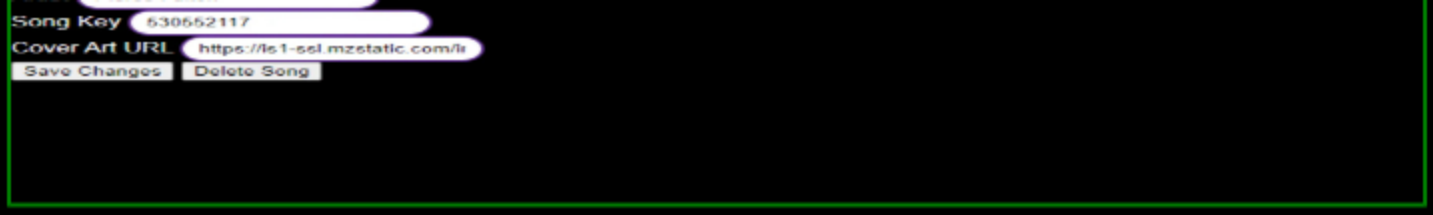
Caption (required) ✓

Describe/highlight what's being shown

edit/delete page

#6) There should be a link to delete the entity (this may be an admin-only thing, but it should be present for the respective role)





Caption (required) ✓
Describe/highlight what's being shown
edit/delete page

^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshots of the details page code

ℹ Details:
Include ucid/date comments for each code screenshot

#1) Show how id is fetched

```php
$isValid = false;

if (isset($_GET["id"])) {
    $id = $_GET["id"];
    $db = getDB();
    $stmt = $db->prepare("SELECT * FROM ShazamSongs WHERE id = :id");
    try {
        $stmt->execute([":id" => $id]);
        $song = $stmt->fetch(PDO::FETCH_ASSOC);
        if ($song) {
            $isValid = true;
        }
    } catch (PDOException $e) {
        error_log("Error fetching song: " . var_export($e, true));
        flash("Error loading song", "danger");
    }
} <- #14-28 if (isset($_GET["id"]))
//ajz27 7/29
```

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

Explanation (required) ✓
Briefly explain the logic and how it's handled when the property is missing or not "valid"

PREVIEW RESPONSE

The script fetches the ID. If an error occurs it will log and flash the error. If an invalid ID is shown it will display "invalid song selected" in HTML.

#2) Show the DB query to get the record

```php
$isValid = false;

if (isset($_GET["id"])) {
    $id = $_GET["id"];
    $db = getDB();
    $stmt = $db->prepare("SELECT * FROM ShazamSongs WHERE id = :id");
    try {
        $stmt->execute([":id" => $id]);
        $song = $stmt->fetch(PDO::FETCH_ASSOC);
        if ($song) {
            $isValid = true;
        }
    } catch (PDOException $e) {
        error_log("Error fetching song: " . var_export($e, true));
        flash("Error loading song", "danger");
    }
} <- #14-28 if (isset($_GET["id"]))
//ajz27 7/29
```

Caption (required) ✓

Describe/highlight what's being shown

query

Explanation (required) ✓

Briefly explain the logic

PREVIEW RESPONSE

The query selects from ShazamSong the requested ID.

#3) Show the code related to presenting the data and showing the links

```php
$table = [
    "data" => $results,
    "title" => "Latest Songs",
    "ignored_columns" => [],
    "edit_url" => get_url("admin/edit_song.php"),
    "view_url" => get_url("admin/view_song.php")
]; <- #34-40 $table =
//ajz27 7/29
```

Caption (required) ✓

Describe/highlight what's being shown

shows the links

Explanation (required) ✓

Briefly explain the logic

PREVIEW RESPONSE

The links are added redirecting to their respective php pages.

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 this page
☐ #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/35

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

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

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

+ ADD ANOTHER URL

Edit Data Page (2 pts.)

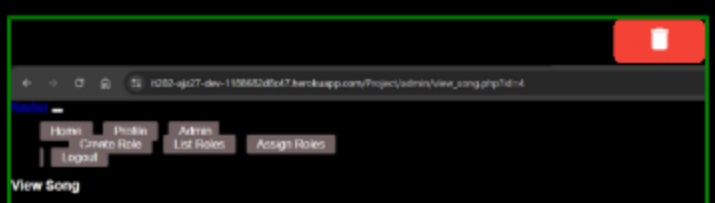
COLLAPSE

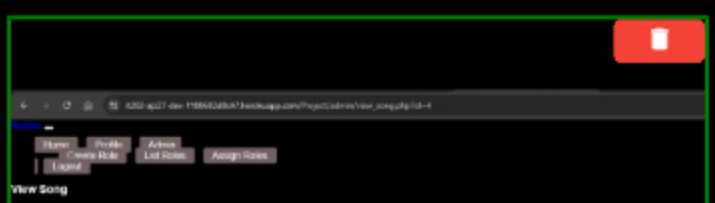
Task #1 - Points: 1

Text: Screenshots of the edit page

Details:
Heroku dev url must be visible in all relevant screenshots

#1) Show before and after screenshots of data you'll edit (two screenshots required)





Caption (required) ✓

Describe/highlight what's being shown

changed title

Explanation (required) ✓

Briefly explain what you changed

PREVIEW RESPONSE

the title was changed

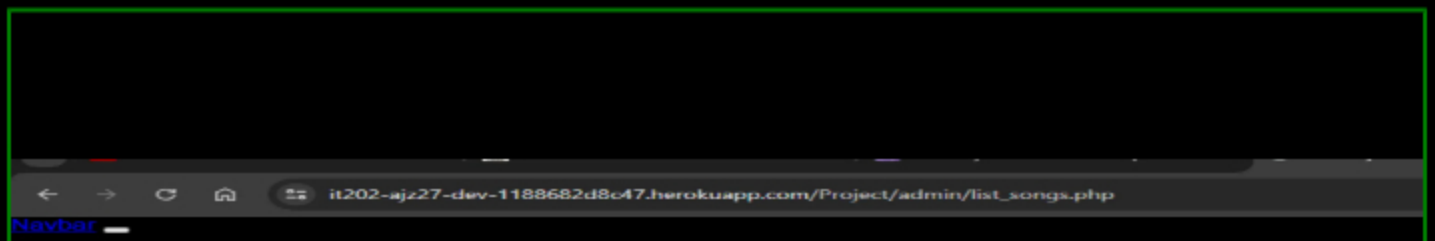
#2) Show examples of validation messages

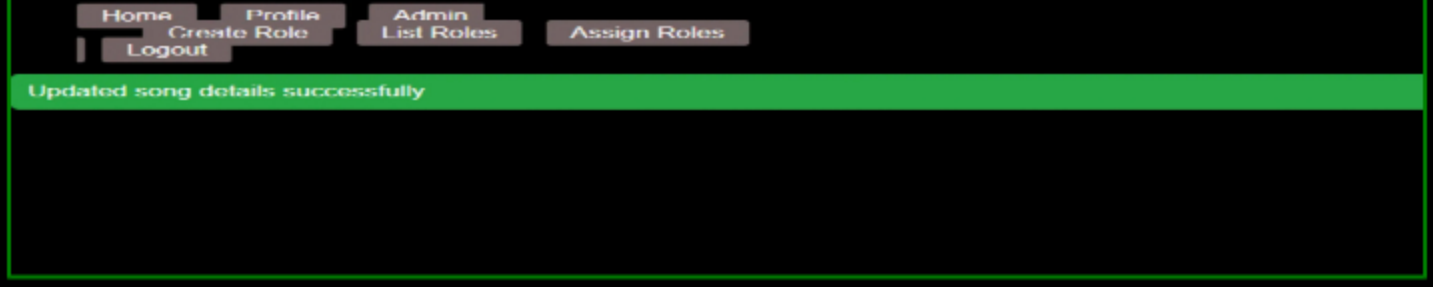
Caption (required) ✓

Describe/highlight what's being shown

validaiton message

#3) Show an example of successful edit messages





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

#4) Design/Style should be considered (i.e., bootstrap, custom css, etc)

Caption (required) ✓
Describe/highlight what's being shown
edit song design

Explanation (required) ✓
Briefly explain your design choices
PREVIEW RESPONSE
Simple fields for changing the respective data entries.

COLLAPSE

Task #2 - Points: 1
Text: Screenshots of edit page code

Details:
Include ucid/date comments for each code screenshot

#1) Form should have correct data types for each property being requested

```
<div class="container-fluid">
    <h3>Edit Song</h3>
    <?php if ($isValid) : ?>
        <form method="POST">
            <div class="form-group">
                <label for="title">Title</label>
                <input type="text" name="title" id="title" value="<?php se($song, 'title'); ?>" class="form-
            </div>
            <div class="form-group">
                <label for="Artist">Artist</label>
                <input type="text" name="Artist" id="Artist" value="<?php se($song, 'Artist'); ?>" class="fo
            </div>
            <div class="form-group">
                <label for="song_key">Song Key</label>
                <input type="text" name="song_key" id="song_key" value="<?php se($song, 'song_key'); ?>" cla
            </div>
            <div class="form-group">
                <label for="image_url">Cover Art URL</label>
                <input type="text" name="image_url" id="image_url" value="<?php se($song, 'image_url'); ?>"
            </div>
            <div class="form-group">
                <button type="submit" name="save" class="btn btn-primary">Save Changes</button>
                <button type="submit" name="delete" class="btn btn-danger" onclick="return confirm('Are you
            </div>
        </form>
    <?php else : ?>
        <p>Invalid song selected.</p>
    <?php endif; ?>
</div>
<!-- ajz27 7/29 -->
```

Caption (required) ✓

Describe/highlight what's being shown

data types

Explanation (required) ✓

Briefly explain the data type choices

PREVIEW RESPONSE

The only data types needed are text.

#2) Form should have correct validation for each field (HTML, JS, and PHP)

```
<div class="container-fluid">
    <h3>Edit Song</h3>
    <?php if ($isValid) : ?>
        <form method="POST">
            <div class="form-group">
                <label for="title">Title</label>
                <input type="text" name="title" id="title" value="<?php se($song, 'title'); ?>" class="form-
            </div>
            <div class="form-group">
                <label for="Artist">Artist</label>
                <input type="text" name="Artist" id="Artist" value="<?php se($song, 'Artist'); ?>" class="fo
            </div>
            <div class="form-group">
                <label for="song_key">Song Key</label>
                <input type="text" name="song_key" id="song_key" value="<?php se($song, 'song_key'); ?>" cla
            </div>
            <div class="form-group">
                <label for="image_url">Cover Art URL</label>
                <input type="text" name="image_url" id="image_url" value="<?php se($song, 'image_url'); ?>"
            </div>
            <div class="form-group">
                <button type="submit" name="save" class="btn btn-primary">Save Changes</button>
                <button type="submit" name="delete" class="btn btn-danger" onclick="return confirm('Are you
            </div>
        </form>
    <?php else : ?>
        <p>Invalid song selected.</p>
    <?php endif; ?>
</div>
<!-- ajz27 7/29 -->
```

Caption (required) ✓

Describe/highlight what's being shown

validation

Explanation (required) ✓

Briefly explain the validations

PREVIEW RESPONSE

HTML validation requires the fields to be populated.

#3) Successful edit should have a user-friendly message

```php
if (isset($_POST["save"]) && $isValid) {
    $title = se($_POST, "title", "", false);
    $Artist = se($_POST, "Artist", "", false);
    $song_key = se($_POST, "song_key", "", false);
    $image_url = se($_POST, "image_url", "", false);

    $db = getDB();
    $stmt = $db->prepare("UPDATE ShazamSongs SET title = :title, Artist = :Artist, song_key = :song_key, ima
    try {
        $stmt->execute([
            ":title" => $title,
            ":Artist" => $Artist,
            ":song_key" => $song_key,
            ":image_url" => $image_url,
            ":id" => $id
        ]); <- #39-45 $stmt->execute
        flash("Updated song details successfully", "success");
        header("Location: list_songs.php");
        exit;
    } catch (PDOException $e) {
        error_log("Error updating song: " . var_export($e, true));
        flash("Error updating song", "danger");
    }
}
//ajr27 7/29 <- #38-53 if (isset($_POST["save"]) && $isValid)
```

Caption (required) ✓

Describe/highlight what's being shown

messages

Explanation (required) ✓

Provide a high-level step-by-step of how the fetching of the record, populating the form, and the update works and gets changed in your DB

PREVIEW RESPONSE

Fetching is done checking if the id parameter is set in the URL. Then a database connection is made using getDB(). An SQL statement is prepared to sleect the song with the given ID and executed. The result is then fetched if found. The page then updates the DB with an UPDATE query.

#4) Any errors should have user-friendly messages

```php
if (isset($_POST["save"]) && $isValid) {
    $title = se($_POST, "title", "", false);
    $Artist = se($_POST, "Artist", "", false);
    $song_key = se($_POST, "song_key", "", false);
    $image_url = se($_POST, "image_url", "", false);

    $db = getDB();
    $stmt = $db->prepare("UPDATE ShazamSongs SET title = :title, Artist = :Artist, song_key = :song_key, ima
    try {
        $stmt->execute([
            ":title" => $title,
            ":Artist" => $Artist,
            ":song_key" => $song_key,
            ":image_url" => $image_url,
            ":id" => $id
        ]); <- #39-45 $stmt->execute
        flash("Updated song details successfully", "success");
```

```
            exit;
        } catch (PDOException $e) {
            error_log("Error updating song: " . var_export($e, true));
            flash("Error updating song", "danger");
        }
    }
    //ajz27 7/29 <- #30-53 if (isset($_POST["save"]) && $isValid)
```

Caption (required) ✓

Describe/highlight what's being shown

errors

Explanation (required) ✓

Briefly explain the possible errors

PREVIEW RESPONSE

Possible errors are if the song cannot be fetched or is invalid, or if the song cannot be updated.

#5) Include any other rules like role guards and login checks

```php
if (!has_role("Admin")) {
    flash("You don't have permission to view this page", "warning");
    die(header("Location: $BASE_PATH" . "/home.php"));
}
//ajz27 7/29
```

Caption (required) ✓

Describe/highlight what's being shown

role guard

Explanation (required) ✓

Briefly explain the logic/reasoning

PREVIEW RESPONSE

The edit page is temporarily role guarded for now.

COLLAPSE

Task #3 - Points: 1

Text: Screenshot of records from DB

#1) Show a before and after screenshot of the record (two screenshots)

Caption (required) ✓

Describe/highlight what's being shown

changes

Explanation (required) ✓

Explain what differs

PREVIEW RESPONSE

Title changed.

COLLAPSE

Task #4 - Points: 1

Text: Add related links

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Include the heroku prod link for this page
☐ #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/edit_song.php

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

URL #2

https://github.com/ajz27/ajz27-IT202-MC/pull/35

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

+ ADD ANOTHER URL

Delete Handling (1 pt.)

^COLLAPSE ^

^COLLAPSE ^

Task #1 - Points: 1

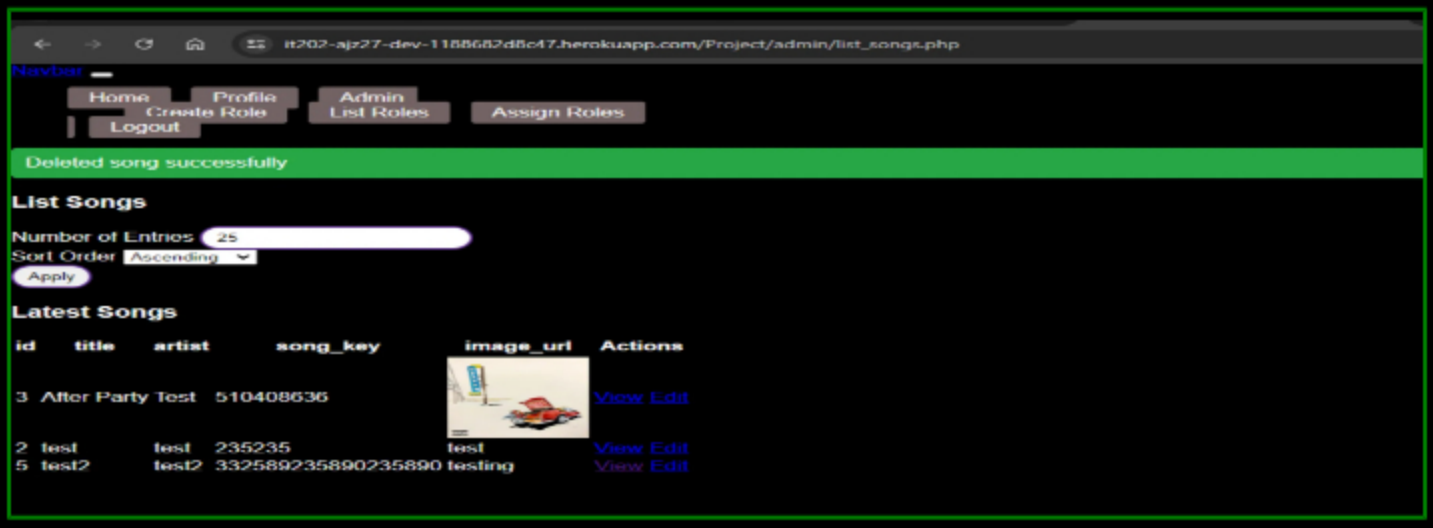
Text: Screenshots related to delete

Details:

Heroku dev url must be visible in all relevant screenshots

Include ucid/date comments for each code screenshot

#1) Show the success message of a delete



Caption (required) ✓

Describe/highlight what's being shown

successful delete

#2) Show any error messages of a failed delete (like id not being passed)

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

Explanation (required)
Explain in concise steps how this logically works
PREVIEW RESPONSE
Missing text

#3) Show the code related to the delete processing

```php
56
57  if (isset($_POST["delete"]) && $isValid) {
58      $db = getDB();
59      $stmt = $db->prepare("DELETE FROM ShazamSongs WHERE id = :id");
60      try {
61          $stmt->execute([":id" => $id]);
62          flash("Deleted song successfully", "success");
63          header("Location: list_songs.php");
64          exit;
65      } catch (PDOException $e) {
66          error_log("Error deleting song: " . var_export($e, true));
67          flash("Error deleting song", "danger");
68      }
69  }
70  //ajz27 7/29 <- #57-69 if (isset($_POST["delete"]) && $isValid)
```

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

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The page takes the request from the button to delete the song, and then the query is prepared to delete the song from the table.

#4) Explain the delete logic

Explanation (required) ✓
Is it a soft or hard delete? Are there any necessary roles or restrictions? (can only delete their data, can only be done by admin, etc)?
PREVIEW RESPONSE

PREVIEW RESPONSE

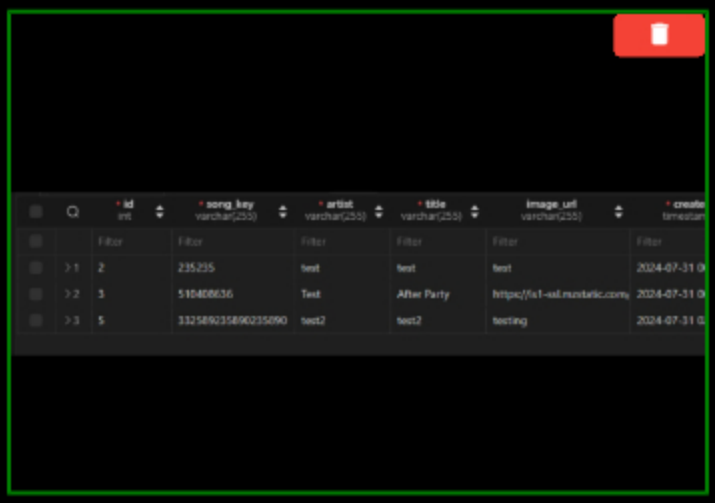
This is a hard delete.. It is currently restricted to admin usage only.

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the data

#1) Show a before and after screenshot of the DB data (two screenshots)



Caption (required) ✓

Describe/highlight what's being shown (note precisely what changed)

before/after

COLLAPSE

Task #3 - Points: 1

Text: Add the pull request link for the branch related to this feature

Details:

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/35

URL

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

URL #2

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

URL

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

ADD ANOTHER URL

Misc (1 pt.)

COLLAPSE

COLLAPSE

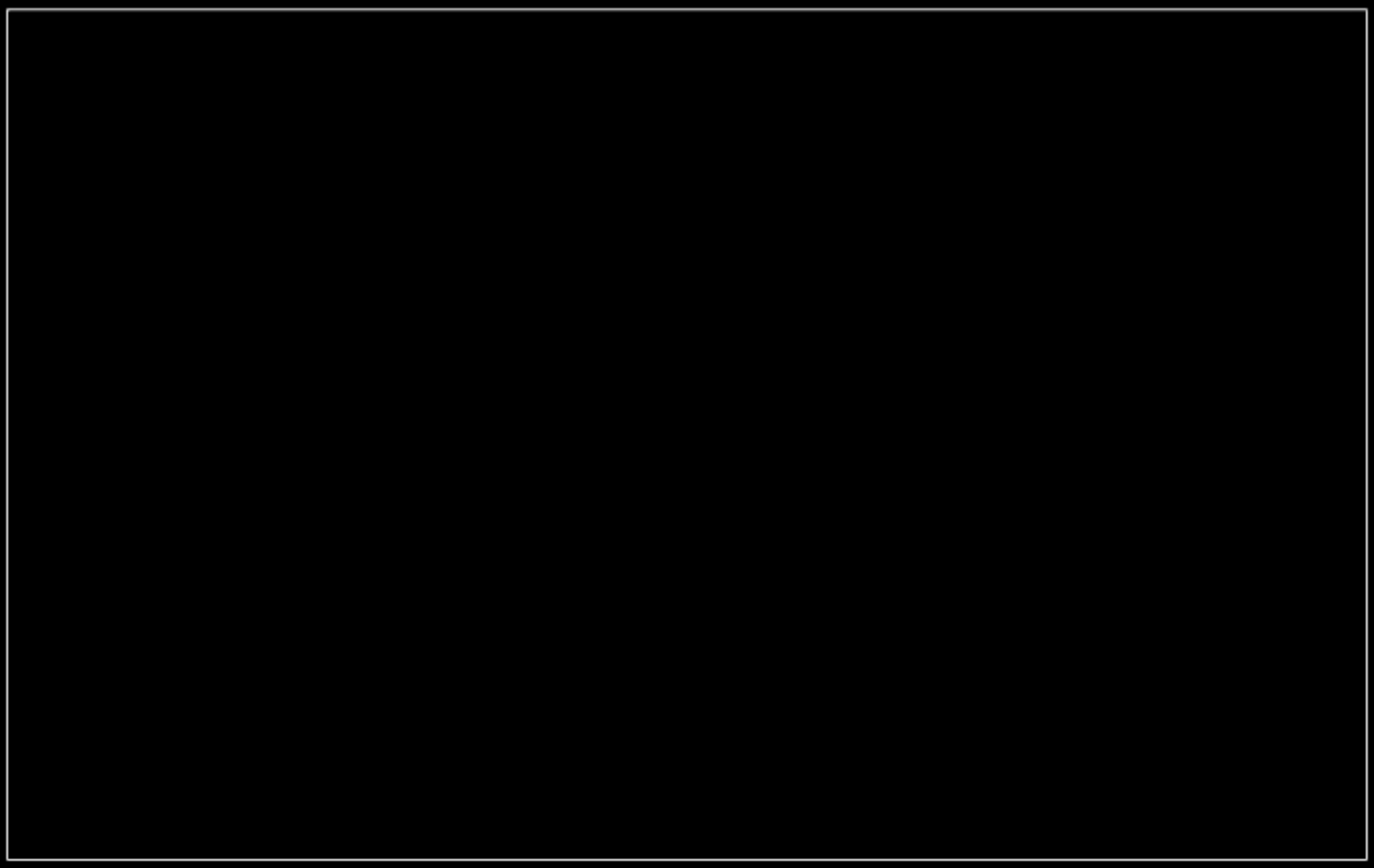
Task #1 - Points: 1

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

Missing URL

URL

+ ADD ANOTHER URL

COLLAPSE

Task #3 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

Had issues getting user specific tables to work, consolidated all under admin role for now.

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

End of Assignment

End of Assignment