Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-api-project-milestone-2-2024/grade/rr42

IT202-008-S2024 - [IT202] API Project Milestone 2 2024

Submissions:

Submission Selection

1 Submission [active] 4/18/2024 9:40:30 PM

Instructions

COLLAPSE

Implement the Milestone 2 features from the project's proposal document: https://docs.google.com/document/d/1XE96a8DQ52Vp49XACBDTNCq0xYDt3kF29cO88E
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: 29 Points: 10.00

Define Appropriate Tables for Data (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of Table SQL

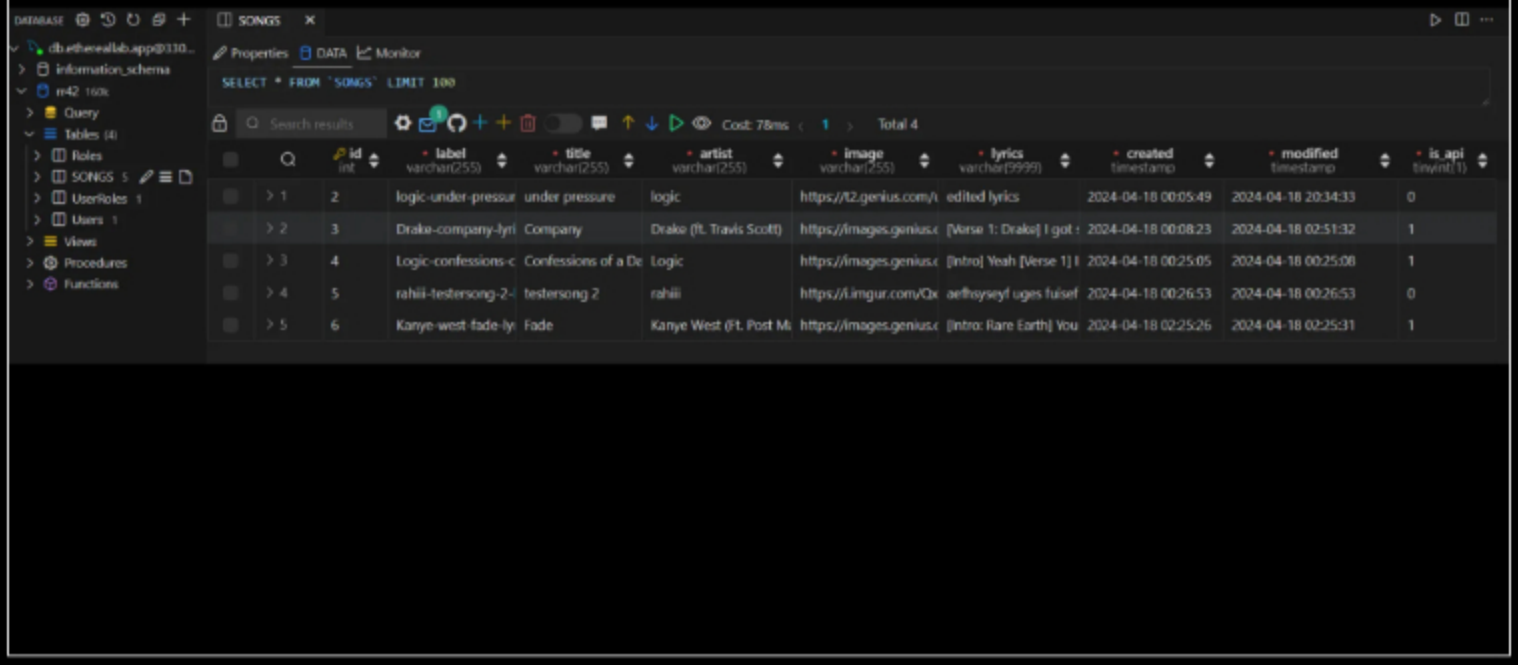
Checklist *The checkboxes are for your own tracking

#	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
#3	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



SONGS table

Checklist Items (3)

#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 Columns should be logical and thought out (not valid to have a single field of JSON data or similar

#3 Clearly caption screenshots

COLLAPSE

Task #2 - Points: 1

Text: Explain the design

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Note the different fields and their purpose
#2	1	Note if you needed to normalize the data into separate tables or if one table fit your needs

Response:

Only one table was necessary for my API data. My custom columns include label, title, artist, image, and lyrics. Label is the unique identifier for a specific song lyric page, and is always in the format " artist-title-'lyrics' ". Title and artist are self-explanatory. Image holds a URL to the image associated with the song, usually an album cover. And lyrics only holds the songs lyrics.

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/rRahmanM6/rr42-it202-008/pull/63

Data Creation Page (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the creation page

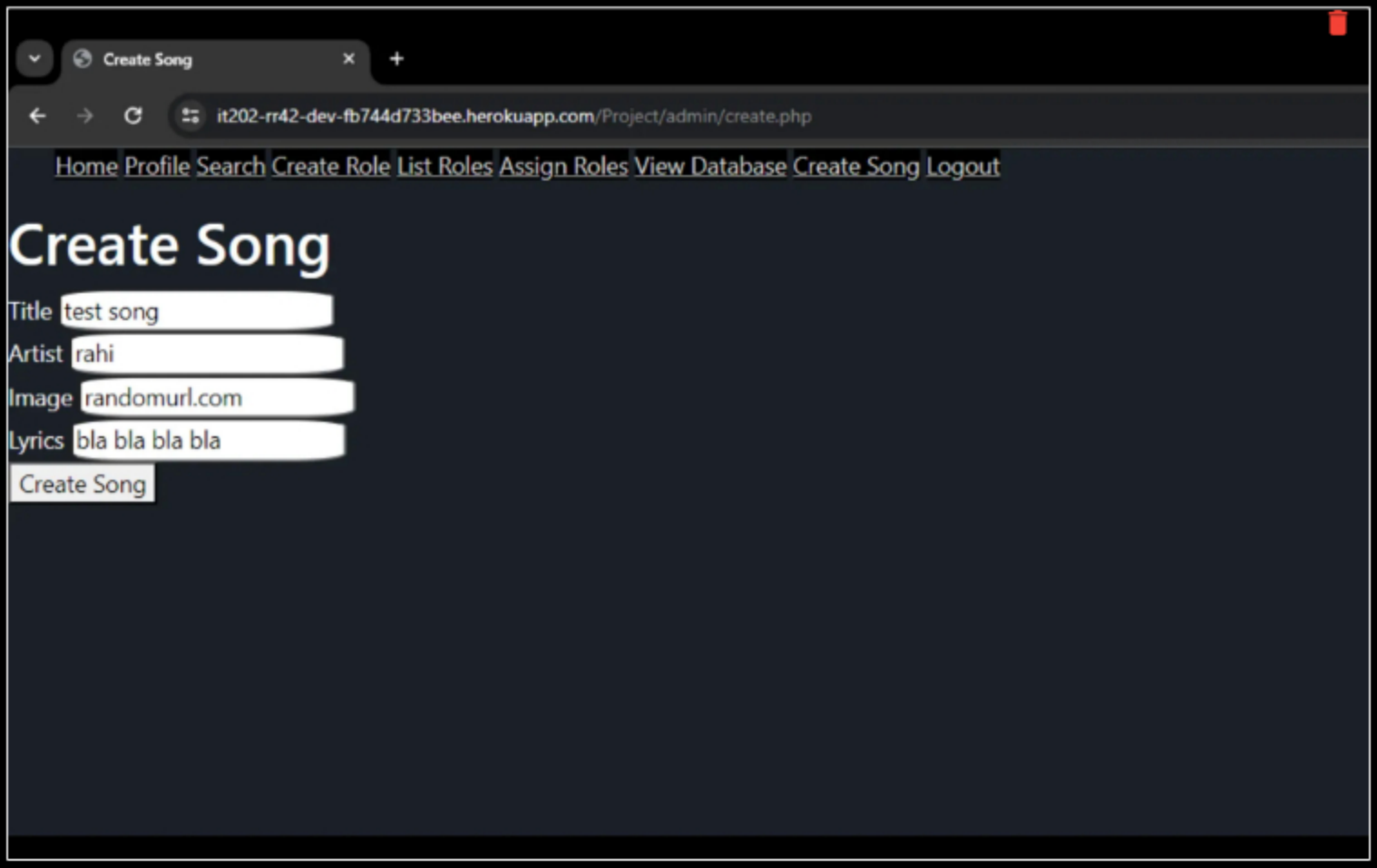
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Show potentially valid data filled in for the custom creation page
#2	1	Show how the API data is fetched for API data (must be server-side)
#3	1	Show examples of validation messages
#4	1	Show an example of successful creation message
#5	1	Design/Style should be considered (i.e., bootstrap, custom css, etc)
#6	1	Make sure the heroku dev url is visible in the address bar
#7	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

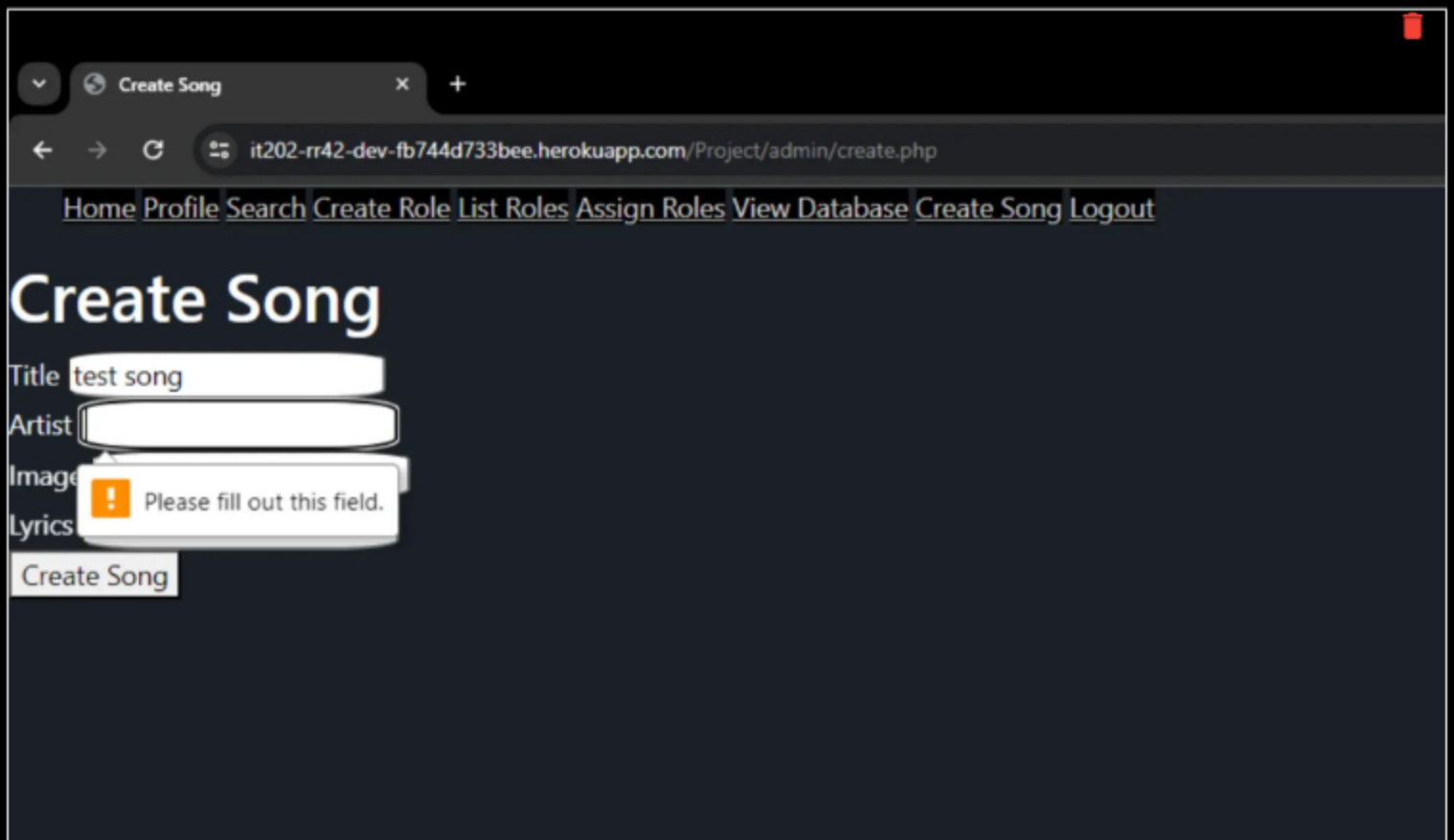
Small Medium Large



Song creation page form filled out

Checklist Items (1)

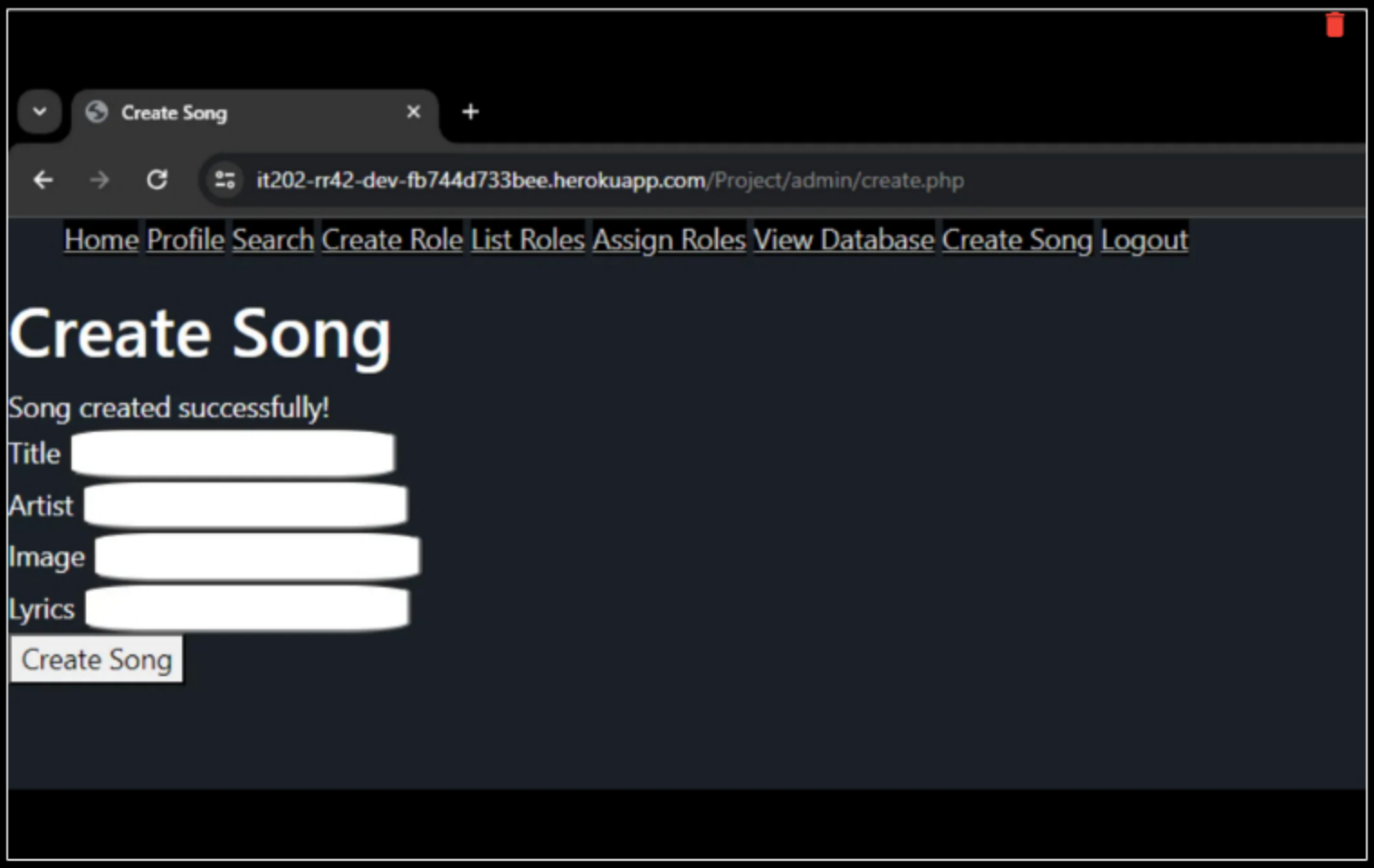
#1 Show potentially valid data filled in for the custom creation page



empty artist field on creation page validation message

Checklist Items (1)

#3 Show examples of validation messages



successful creation message

Checklist Items (1)

#4 Show an example of successful creation message

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of creation page code

Checklist

*The checkboxes are for your own tracking

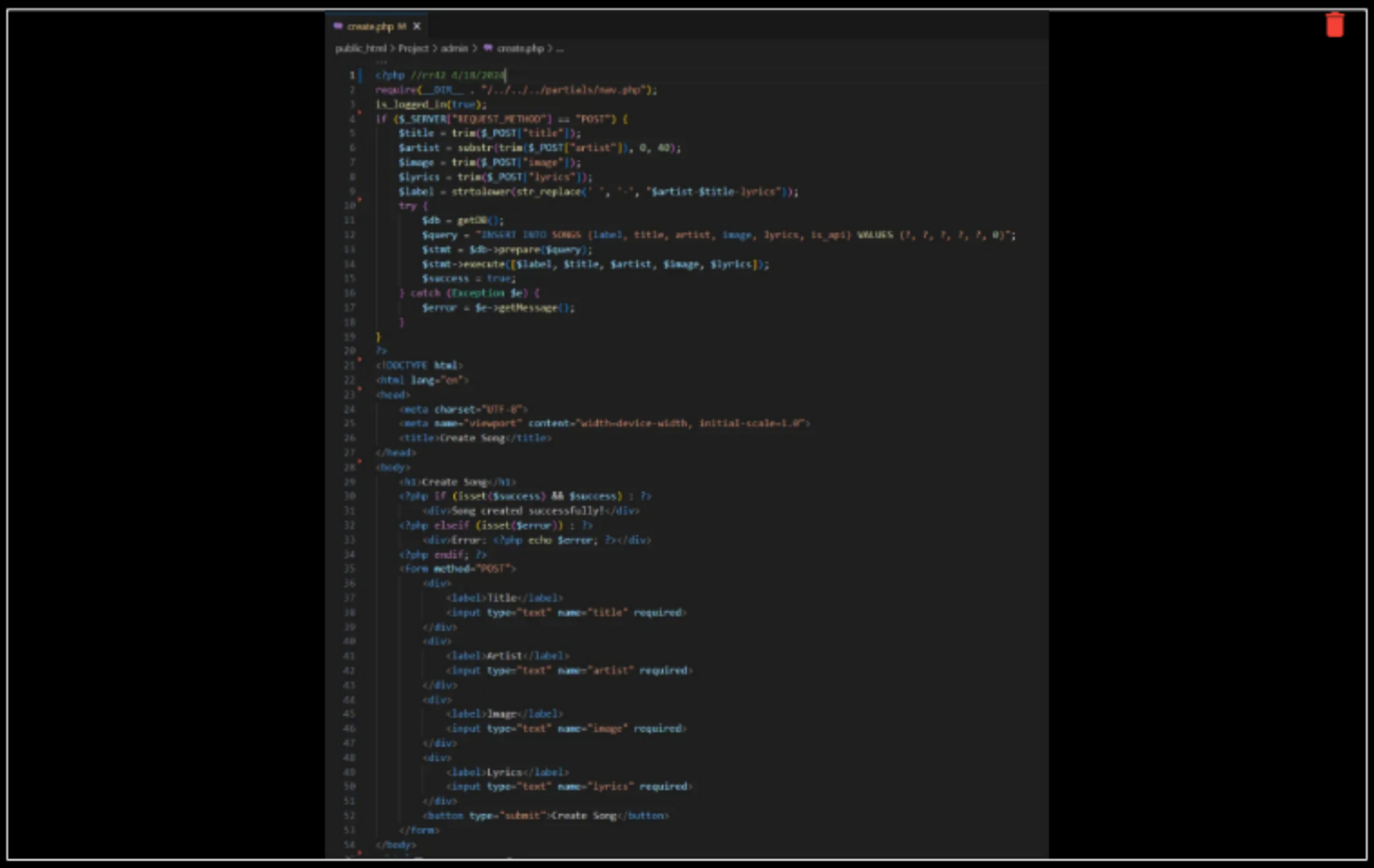
#	Points	Details
☐ #1	1	Form should have correct data types for each property being requested
☐ #2	1	Form should have correct validation for each field (HTML, JS, and PHP)
☐ #3	1	Successful creation should have a user-friendly message
☐ #4	1	Any errors should have user-friendly messages
☐ #5	1	Include the form/process for fetching API data

☐ #6	1	Include some indicator between custom data and API data
☐ #7	1	Include any other rules like role guards and login checks
■ #8	1	Include ucid/date comments for each code screenshot
☐ #9	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```php
<?php //rr42 4/18/2024
require(__DIR__ . "/../../../partials/nav.php");
is_logged_in(true);
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $title = trim($_POST["title"]);
    $artist = substr(trim($_POST["artist"]), 0, 40);
    $image = trim($_POST["image"]);
    $lyrics = trim($_POST["lyrics"]);
    $label = strtolower(str_replace(' ', '-', "$artist-$title-lyrics"));
    try {
        $db = getDB();
        $query = "INSERT INTO SONGS (label, title, artist, image, lyrics, is_api) VALUES (?, ?, ?, ?, ?, 0)";
        $stmt = $db->prepare($query);
        $stmt->execute([$label, $title, $artist, $image, $lyrics]);
        $success = true;
    } catch (Exception $e) {
        $error = $e->getMessage();
    }
}
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Create Song</title>
</head>
<body>
    <h1>Create Song</h1>
    <?php if (isset($success) && $success) : ?>
        <div>Song created successfully!</div>
    <?php elseif (isset($error)) : ?>
        <div>Error: <?php echo $error; ?></div>
    <?php endif; ?>
    <form method="POST">
        <div>
            <label>Title</label>
            <input type="text" name="title" required>
        </div>
        <div>
            <label>Artist</label>
            <input type="text" name="artist" required>
        </div>
        <div>
            <label>Image</label>
            <input type="text" name="image" required>
        </div>
        <div>
            <label>Lyrics</label>
            <input type="text" name="lyrics" required>
        </div>
        <button type="submit">Create Song</button>
    </form>
</body>
```

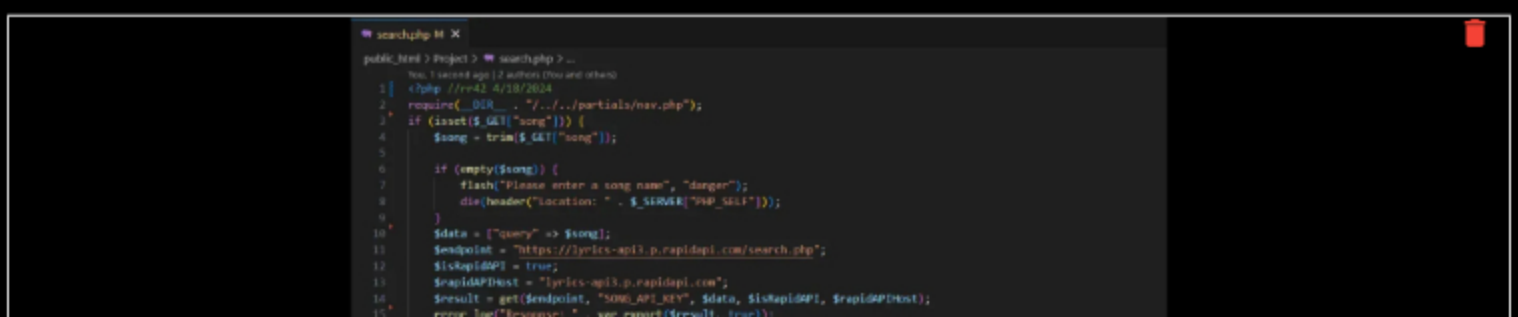
create.php code

Checklist Items (3)

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

#3 Successful creation should have a user-friendly message

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



```php
<?php //rr42 4/18/2024
require(__DIR__ . "/../../partials/nav.php");
if (isset($_GET["song"])) {
    $song = trim($_GET["song"]);

    if (empty($song)) {
        flash("Please enter a song name", "danger");
        die(header("Location: " . $_SERVER["PHP_SELF"]));
    }
    $data = ["query" => $song];
    $endpoint = "https://lyrics-api3.p.rapidapi.com/search.php";
    $isRapidAPI = true;
    $rapidAPIHost = "lyrics-api3.p.rapidapi.com";
    $result = get($endpoint, "SONG_API_KEY", $data, $isRapidAPI, $rapidAPIHost);
```

```php
    if (se($result, "status", 400, false) == 200 && isset($result["response"])) {
        $result = json_decode($result["response"], true);
    } else {
        $result = [];
    }
    if (!empty($result)) {
        $db = getDB();
        $firstSong = $result[0];
        $songName = $firstSong['name'];
        $artist = $firstSong['artist'];
        $image = $firstSong['image'];
        $songId = $firstSong['id'];

        $stmt = $db->prepare("SELECT * FROM SONGS WHERE label = ?");
        $stmt->execute([$songId]);
        $song = $stmt->fetch();

        if (!$song) {
            $query = "INSERT INTO SONGS (label, title, artist, image, lyrics, is_api) VALUES (?, ?, ?, ?, '', 1)";
            $stmt = $db->prepare($query);
            $stmt->execute([$songId, $songName, $artist, $image]);

            $lyricsEndpoint = "https://lyrics-api3.p.rapidapi.com/lyrics.php?id=$songId";
            $lyricsResult = get($lyricsEndpoint, "SONG_API_KEY", [], true, "lyrics-api3.p.rapidapi.com");

            if (se($lyricsResult, "status", 400, false) == 200 && isset($lyricsResult["response"])) {
                $lyricsData = json_decode($lyricsResult["response"], true);
                if (isset($lyricsData['lyrics'])) {
                    $lyrics = $lyricsData['lyrics'];
                    $query = "UPDATE SONGS SET lyrics = ? WHERE label = ?";
                    $stmt = $db->prepare($query);
                    $stmt->execute([$lyrics, $songId]);
                }
            }
        }
    }
}

    You, yesterday • added create page
```

search.php code

Checklist Items (4)

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

#4 Any errors should have user-friendly messages

#5 Include the form/process for fetching API data

#6 Include some indicator between custom data and API data

search.php M

public_html > Project > search.php > div.container-fluid > div.row

```php
<div class="container-fluid">  <!--rd2 4/18/2024-->
    <h1>Song Lyrics</h1>
    <form>
        <div>
            <label>Song</label>
            <input name="song" />
            <input type="submit" value="Fetch Song" />
        </div>
    </form>
    <div class="row">
        <?php if (isset($result) && !empty($result)) : ?>
            <?php
            $firstSong = $result[0];
            $songName = $firstSong['name'];
            $artist = $firstSong['artist'];
            $image = $firstSong['image'];
            $firstSongId = $firstSong['id'];
            $db = getDB();
            $stmt = $db->prepare("SELECT * FROM SONGS WHERE label = ?");
            $stmt->execute([$firstSongId]);
            $song = $stmt->fetch();

            if ($song) {
                $lyrics = $song['lyrics'];
            } else {
                $lyricsEndpoint = "https://lyrics-api3.p.rapidapi.com/lyrics.php?id=$firstSongId";
                $lyricsResult = get($lyricsEndpoint, "SONG_API_KEY", [], true, "lyrics-api3.p.rapidapi.com");    You, yesterday • added create page

                if (se($lyricsResult, "status", 400, false) == 200 && isset($lyricsResult["response"])) {
                    $lyricsData = json_decode($lyricsResult["response"], true);
                    if (isset($lyricsData['lyrics'])) {
                        $lyrics = $lyricsData['lyrics'];
                        $query = "INSERT INTO SONGS (label, title, artist, image, lyrics, is_api) VALUES (?, ?, ?, ?, ?, 1)";
                        $stmt = $db->prepare($query);
                        $stmt->execute([$firstSongId, $songName, $artist, $image, $lyrics]);
                    } else {
                        $lyrics = "Lyrics not available.";
                    }
                } else {
                    $lyrics = "Failed to fetch lyrics.";
                }
            }
            echo "<h2>$songName</h2>";
            echo "<p>Artist: $artist</p>";
            echo "<img src='$image' alt='Song Image' style='max-width: 400px; max-height: 400px;'>";
            echo "<pre>$lyrics</pre>";
            if (has_role("Admin")) {
                $editSongId = $song['id'];
                echo "<div class='admin-actions'>";
                echo "<a href='edit.php?id=$editSongId' class='btn btn-primary'>Edit</a>";
                echo "<form method='POST' class='delete-form'>";
                echo "<input type='hidden' name='delete_id' value='$firstSongId'>";
                echo "<button type='submit' class='btn btn-danger' onclick='return confirm(\"Are you sure you want to delete this song?\")'>Delete</button>";
                echo "</form>";
                echo "</div>";
```

search.php code (continued)

Checklist Items (0)

COLLAPSE

Task #3 - Points: 1

Text: Screenshot of records from DB

Checklist

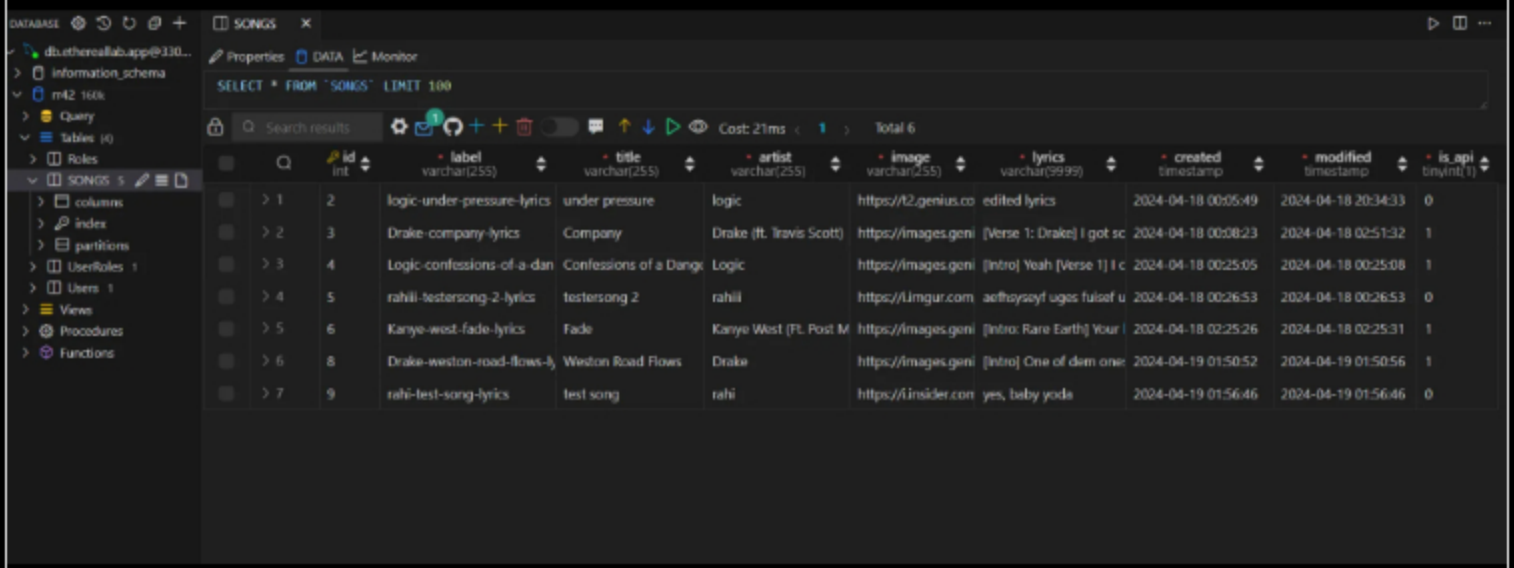
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show at least one record fetched from the API
#2	1	Show at least one record created via the creation form
#3	1	Note what differs
#4	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Records 2, 5, and 9 are both custom created using the create form. The is_api field is set to 0 accordingly.

Checklist Items (3)

#1 Show at least one record fetched from the API

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

#3 Note what differs

^COLLAPSE^

Task #4 - Points: 1

Text: Explain the process

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Provide a high-level step-by-step of how fetching API data works and gets added to your DB (include how duplicates are handled)
☐ #2	1	Provide a high-level step-by-step of how creating custom data works and gets added to your DB (include how duplicates are handled)
☐ #3	1	Briefly describe the validations for the applicable fields
☐ #4	1	Describe how duplicate data is handled

Response:

My application uses two endpoints, a Search Song endpoint that returns song info (id, title, artist, image), and a Get Song Lyrics endpoint that returns the lyrics to a specified song.

The user first submits a search query with the name of a song and its artist. Then, the song endpoint is invoked with the search query as a parameter. The API returns a number of songs in an array with the first item in the array being the most relevant song. The 'id' key of the first array object is then used to invoke the Get Song Lyrics endpoint, which returns the title and the lyrics in an array.
After invoking the first endpoint, my code checks if the 'id' exists in my DB, if so, the lyrics are pulled from the DB, otherwise, the second endpoint is called, and the new song is stored in the DB. This prevents duplicates from being added into the database.

Creation: The user fills out the form and the data is sent to the server-side with a POST request. Validation is in place to ensure no fields are left blank. The data is processed and a unique label is generated. The song is inserted into the database with an INSERT query.

^COLLAPSE^

Task #5 - 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-rr42-prod-dc703459f313.herokuapp.com/Project/admin/create.php

URL #2
https://github.com/rRahmanM6/rr42-it202-008/pull/64

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

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the list page

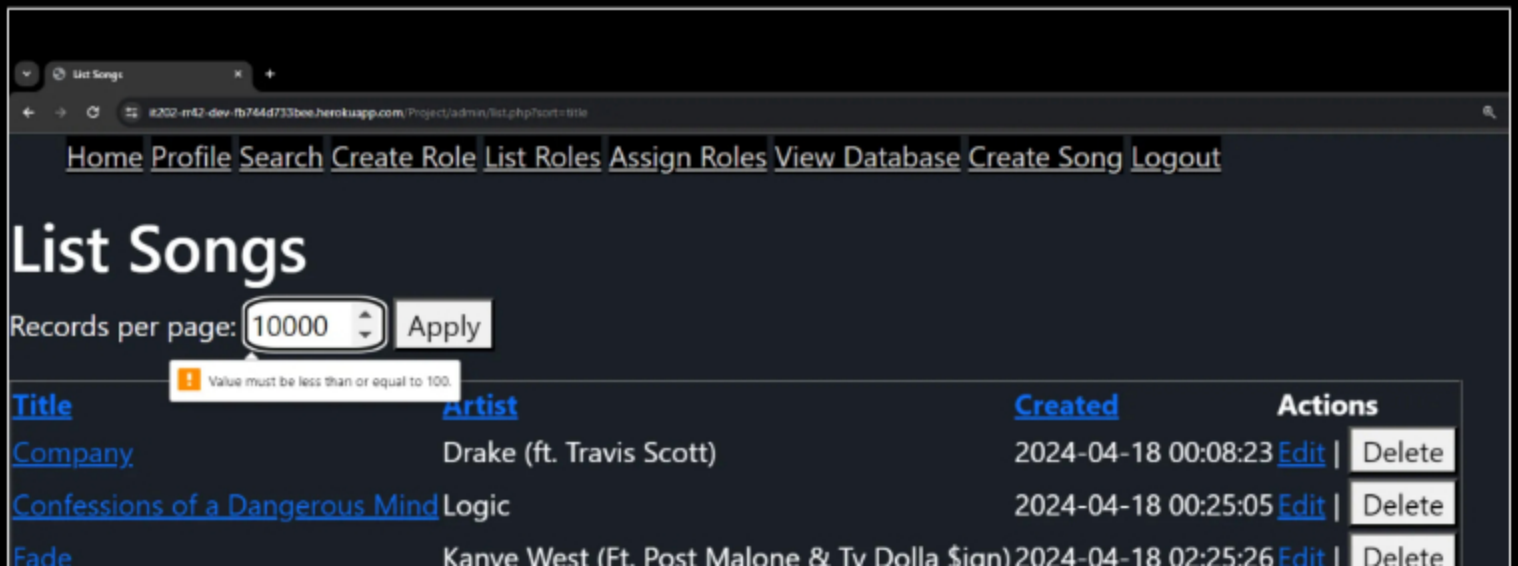
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the page of your entities listed (have a reasonable number shown)
#2	1	Show the filter/sort form based on your data and the required limit field
#3	1	Demonstrate a few varied filters/sorts
#4	1	Demonstrate a filter that doesn't have any records (should show an appropriate message)
#5	1	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)
#6	1	Each list item should have a summary of the entity (likely won't be the entire entity data)
#7	1	Design/Style should be considered (i.e., bootstrap, custom css, etc)
#8	1	Make sure the heroku dev url is visible in the address bar
#9	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



test song	rahi	2024-04-19 01:56:46	Edit | Delete
testersong 2	rahiii	2024-04-18 00:26:53	Edit | Delete
under pressure	logic	2024-04-18 00:05:49	Edit | Delete
Weston Road Flows	Drake	2024-04-19 01:50:52	Edit | Delete

List page sorted alphabetically by title. Attempted to show 10000 record per page, but warning shown to user to ensure value must be between 0-100. Clicking artist or created will sort the list appropriately.

Checklist Items (4)

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

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

#3 Demonstrate a few varied filters/sorts

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

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the list page code

Checklist

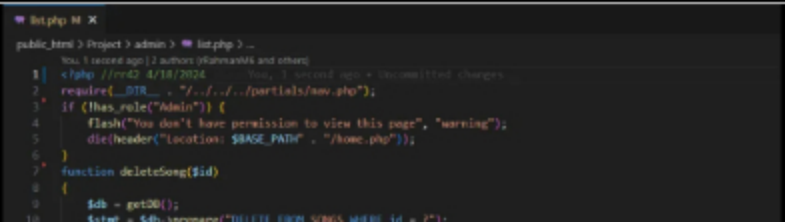
*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Show the filter/sort form generation
☐ #2	1	Show the DB query and how the filter/sort is handled (including the restriction on the limit field)
☐ #3	1	Show how the output is generated and displayed
☐ #4	1	Show any restrictions like role guard or login checks
☐ #5	1	Include ucid/date comments for each code screenshot
☐ #6	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```php
    $stmt->execute([$id]);
}
if (isset($_POST['delete_id'])) {
    $id = $_POST['delete_id'];
    deleteSong($id);
    header("Location: $BASE_PATH" . "/list.php");
    exit;
}
$perPage = 10;
if (isset($_GET['perPage']) && is_numeric($_GET['perPage'])) {
    $perPage = max(0, min(100, $_GET['perPage']));
}
$orderBy = "title";
if (isset($_GET['sort'])) {
    $sort = strtolower($_GET['sort']);
    if (in_array($sort, ['title', 'artist', 'created'])) {
        $orderBy = $sort;
    }
}
$db = getDB();
$stmt = $db->prepare("SELECT id, title, artist, created FROM SONGS ORDER BY $orderBy LIMIT $perPage");
$stmt->execute();
$songs = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>List Songs</title>
</head>

<body>
    <h1>List Songs</h1>
    <form method="GET">
        <label for="perPage">Records per page:</label>
        <input type="number" id="perPage" name="perPage" value="<?php echo $perPage; ?>" min="0" max="100">
        <button type="submit">Apply</button>
    </form>
    <table border="1">
        <tr>
            <th><a href="?sort=title">Title</a></th>
            <th><a href="?sort=artist">Artist</a></th>
```

list.php

Checklist Items (0)

```php
            <th><a href="?sort=artist">Artist</a></th> <!--rr42 4/18/2024!-->
            <th><a href="?sort=created">Created</a></th>
            <th>Actions</th>
        </tr>
        <?php foreach ($songs as $song) : ?>
            <tr>
                <td><a href="../search.php?song=<?php echo urlencode($song['title'] . ' ' . $song['artist']); ?>"><?php echo $song['title']; ?></a></td>
                <td><?php echo $song['artist']; ?></td>
                <td><?php echo $song['created']; ?></td>
                <td>
                    <a href="edit.php?id=<?php echo $song['id']; ?>">Edit</a> |
                    <form method="POST" style="display: inline;">
                        <input type="hidden" name="delete_id" value="<?php echo $song['id']; ?>">
                        <button type="submit" onclick="return confirm('Are you sure you want to delete this song?')">Delete</button>
                    </form>
                </td>
            </tr>
        <?php endforeach; ?>

    </table>
</body>

</html>
```

list.php (continued)

Checklist Items (0)

COLLAPSE

Task #3 - Points: 1

Text: Explain how the page works

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Provide the high-level steps of how the filter/sorting works and how data is displayed
☐ #2	1	Summarize what you're showing on the screen
☐ #3	1	Mention your design/style choice
☐ #4	1	Mention which users can interact with the view, edit, and delete links

Response:

The user can limit the number of records displayed by filling in the form field. The value is stored in a variable and used in the DB Select statement when listing records. The number entered must be between 0 and 100.

The user can also click on each column to sort alphabetically or by date.
The songs are displayed in an HTML table with columns titile, artist, and creation date. There are also buttons to edit or delete each record, which can only be used by an admin. Clicking the edit button redirects to edit.php, and clicking delete shows a confirmation dialog, and deletes the record from the DB via a POST request, if the user confirms.

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-rr42-prod-dc703459f313.herokuapp.com/Project/admin/list.php

URL #2
https://github.com/rRahmanM6/rr42-it202-008/pull/65

COLLAPSE

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

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the details page

Checklist *The checkboxes are for your own tracking

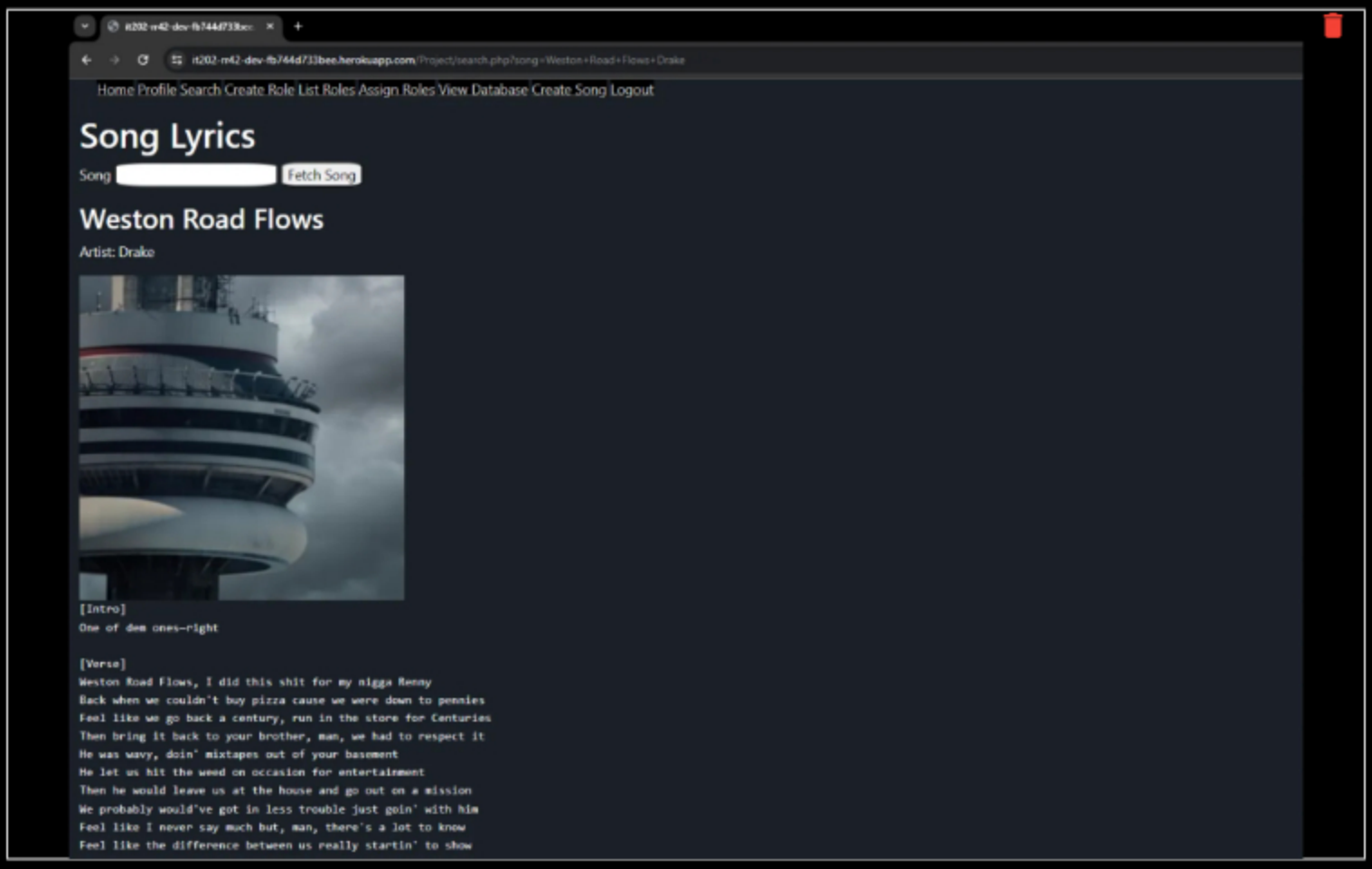
#	Points	Details

☐ #1	1	Entity should be fetch by id (via the url)
☐ #2	1	A missing id should redirect back to the list page with an applicable message
☐ #3	1	Design/Style should be considered (i.e., bootstrap, custom css, etc)
☐ #4	1	Data shown should be more detailed/inclusive than the summary view
☐ #5	1	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)
☐ #6	1	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)
☐ #7	1	Make sure the heroku dev url is visible in the address bar
☐ #8	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Clicking on a title from the list page links the user to the relevant search page with the lyrics.

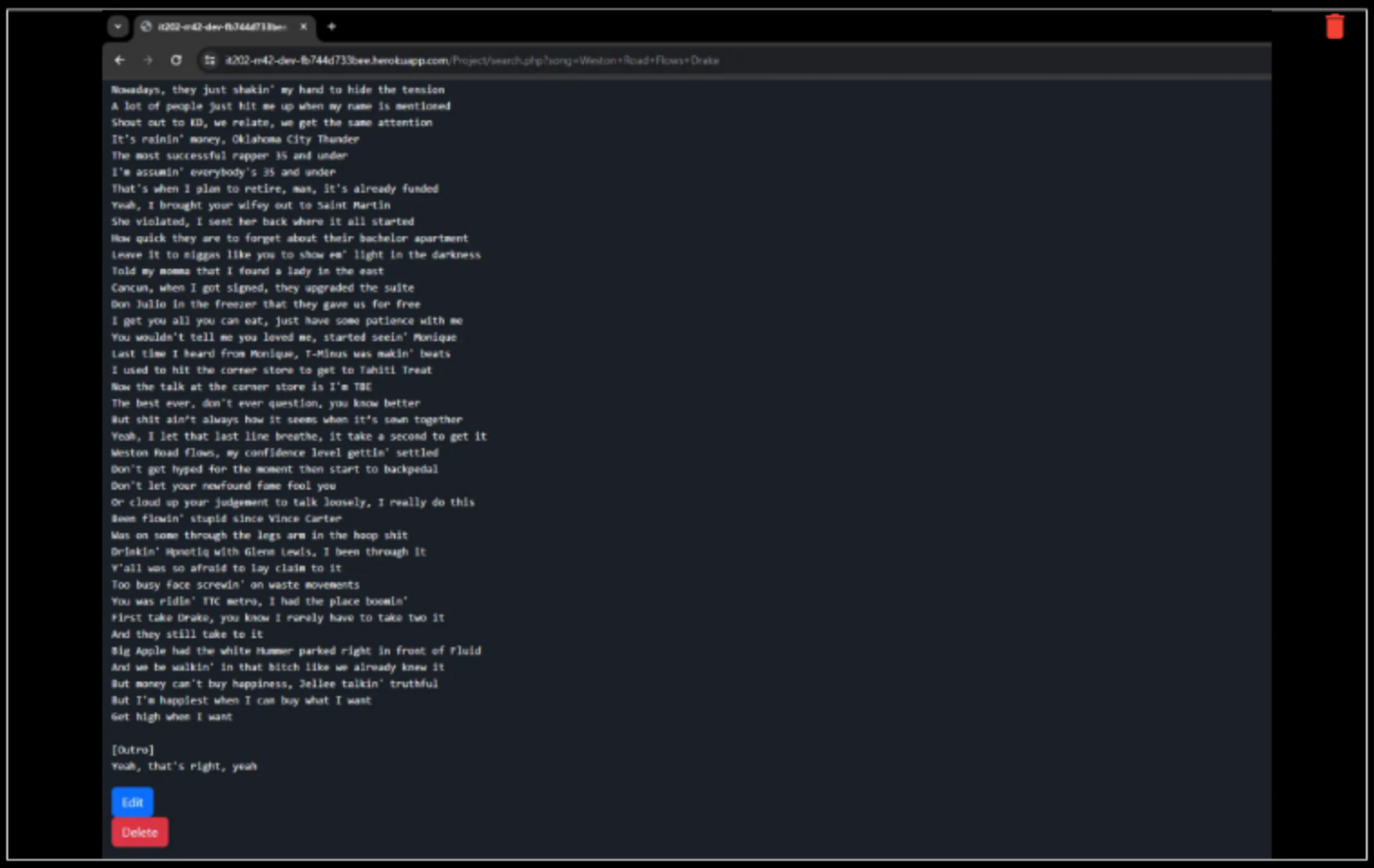
Checklist Items (3)

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

#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)

#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)

respective role)



Edit and delete buttons at the bottom of the page only visible to admins.

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the details page code

Checklist

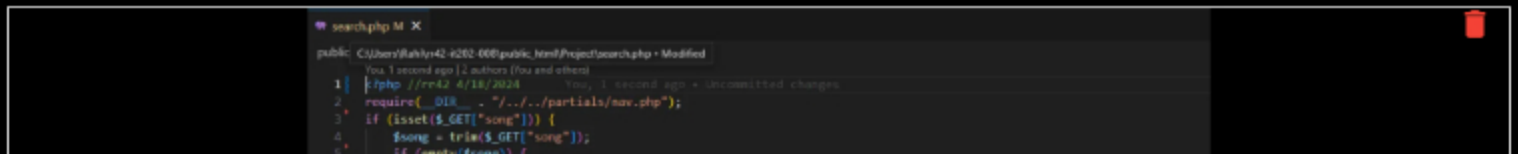
*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Show how id is fetched
☐ #2	1	Show the DB query to get the record
☐ #3	1	Show the code related to presenting the data and showing the links
☐ #4	1	Include ucid/date comments for each code screenshot
☐ #5	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```php
        Flash("Please enter a song name", "danger");
        die(header("Location: " . $_SERVER["PHP_SELF"]));
    }
    $data = ["query" => $song];
    $endpoint = "https://lyrics-api3.p.rapidapi.com/search.php";
    $isRapidAPI = true;
    $rapidAPIHost = "lyrics-api3.p.rapidapi.com";
    $result = get($endpoint, "SONG_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);
    } else {
        $result = [];
    }
    if (!empty($result)) {
        $db = getDB();
        $firstSong = $result[0];
        $songName = $firstSong['name'];
        $artist = $firstSong['artist'];
        $image = $firstSong['image'];
        $songId = $firstSong['id'];
        $stmt = $db->prepare("SELECT * FROM SONGS WHERE label = ?");
        $stmt->execute([$songId]);
        $song = $stmt->fetch();
        if (!$song) {
            $query = "INSERT INTO SONGS (label, title, artist, image, lyrics, is_api) VALUES (?, ?, ?, ?, '', 1)";
            $stmt = $db->prepare($query);
            $stmt->execute([$songId, $songName, $artist, $image]);

            $lyricsEndpoint = "https://lyrics-api3.p.rapidapi.com/lyrics.php?id=$songId";
            $lyricsResult = get($lyricsEndpoint, "SONG_API_KEY", [], true, "lyrics-api3.p.rapidapi.com");

            if (se($lyricsResult, "status", 400, false) == 200 && isset($lyricsResult["response"])) {
                $lyricsData = json_decode($lyricsResult["response"], true);
                if (isset($lyricsData['lyrics'])) {
                    $lyrics = $lyricsData['lyrics'];
                    $query = "UPDATE SONGS SET lyrics = ? WHERE label = ?";
                    $stmt = $db->prepare($query);
                    $stmt->execute([$lyrics, $songId]);
                }
            }
        }
    }
}
?>
```

search.php

Checklist Items (0)

```php
<div class="container-fluid"> <!--#2 4/18/2024-->
    <h1>Song Lyrics</h1>
    <form>
        <div>
            <label>Song</label>
            <input name="song" />
            <input type="submit" value="Fetch Song" />
        </div>
    </form>
    <div class="row">
        <?php if (isset($result) && !empty($result)) : ?>
            <?php
            $firstSong = $result[0];
            $songName = $firstSong['name'];
            $artist = $firstSong['artist'];
            $image = $firstSong['image'];
            $firstSongId = $firstSong['id'];
            $db = getDB();
            $stmt = $db->prepare("SELECT * FROM SONGS WHERE label = ?");
            $stmt->execute([$firstSongId]);
            $song = $stmt->fetch();

            if ($song) {
                $lyrics = $song['lyrics'];
            } else {
                $lyricsEndpoint = "https://lyrics-api3.p.rapidapi.com/lyrics.php?id=$firstSongId";
                $lyricsResult = get($lyricsEndpoint, "SONG_API_KEY", [], true, "lyrics-api3.p.rapidapi.com");

                if (se($lyricsResult, "status", 400, false) == 200 && isset($lyricsResult["response"])) {
                    $lyricsData = json_decode($lyricsResult["response"], true);
                    if (isset($lyricsData['lyrics'])) {
                        $lyrics = $lyricsData['lyrics'];
                        $query = "INSERT INTO SONGS (label, title, artist, image, lyrics, is_api) VALUES (?, ?, ?, ?, ?, 1)";
                        $stmt = $db->prepare($query);
                        $stmt->execute([$firstSongId, $songName, $artist, $image, $lyrics]);
                    } else {
                        $lyrics = "Lyrics not available.";
                    }
                } else {
                    $lyrics = "Failed to fetch lyrics.";
                }
            }
            echo "<h2>$songName</h2>";
            echo "<p>Artist: $artist</p>";
            echo "<img src='$image' alt='Song Image' style='max-width: 400px; max-height: 400px;'>";
            echo "<pre>$lyrics</pre>";
            if (has_role("Admin")) {
                $editSongId = $song['id'];
                echo "<div class='admin-actions'>";
                echo "<a href='edit.php?id=$editSongId' class='btn btn-primary'>Edit</a>";
                echo "<form method='POST' class='delete-form'>";
                echo "<input type='hidden' name='delete_id' value='$firstSongId'>";
                echo "<button type='submit' class='btn btn-danger' onclick='return confirm(\"Are you sure you want to delete this song?\")'>Delete</button>";
                echo "</form>";
                echo "</div>";
```

search.php (continued)

Checklist Items (0)

Task #3 - Points: 1

COLLAPSE

Task #3 - Points: 1

Text: Explain how the page works

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Provide the high-level steps for handling the DB lookup and presenting the data

Response:

From the list page, clicking on the title of the song redirects to search.php with the song query filled in.

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-rr42-prod-dc703459f313.herokuapp.com/Project/search.php

COLLAPSE

Edit Data Page (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the edit page

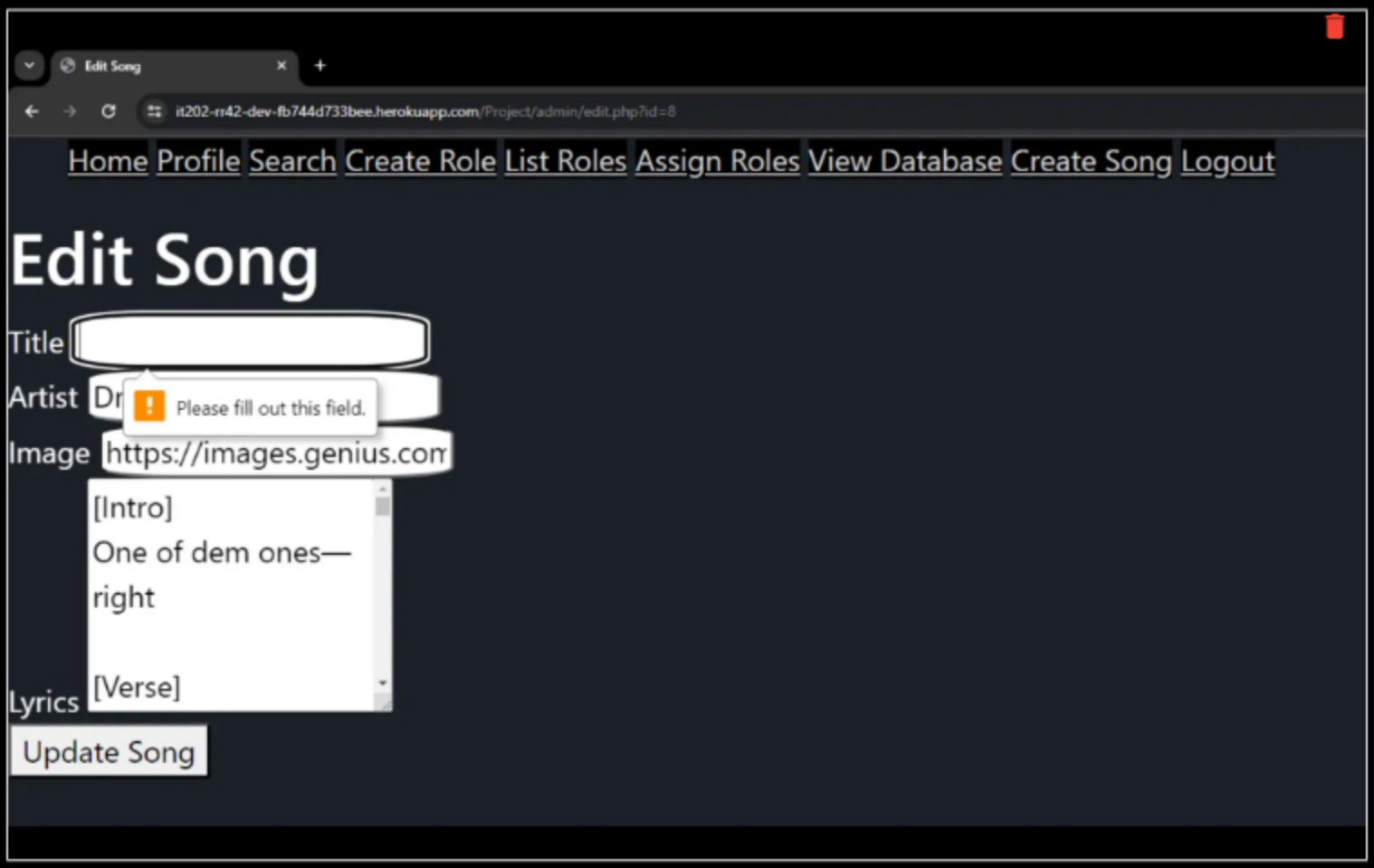
Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Show before and after screenshots of data you'll edit
☐ #2	1	Show examples of validation messages
☐ #3	1	Show an example of successful edit messages
☐ #4	1	Design/Style should be considered (i.e., bootstrap, custom css, etc)
☐ #5	1	Make sure the heroku dev url is visible in the address bar
☐ #6	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

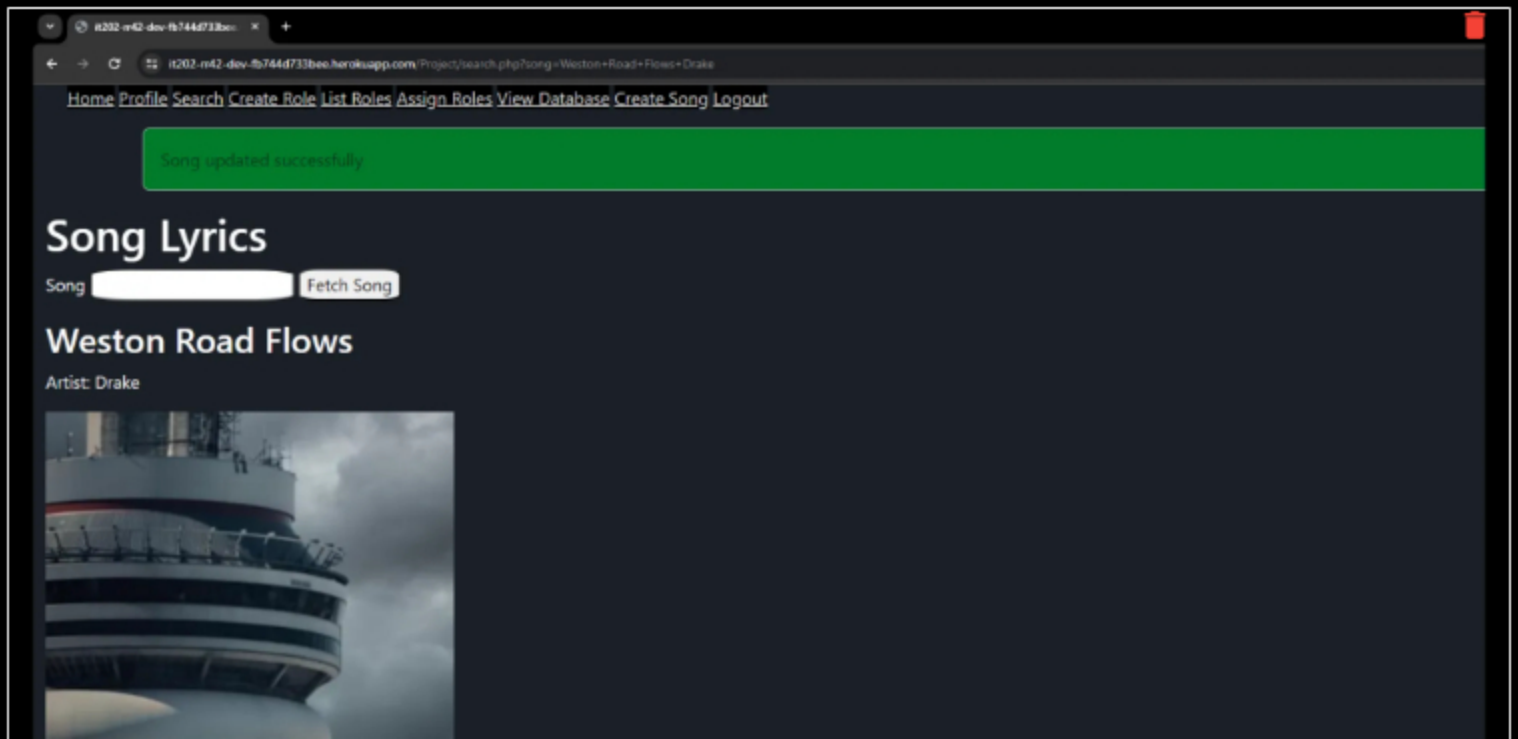


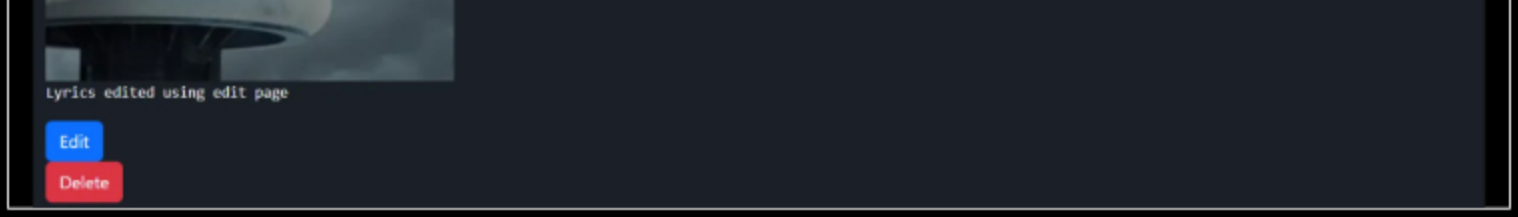
edit page with empty title form.

Checklist Items (2)

#1 Show before and after screenshots of data you'll edit

#2 Show examples of validation messages





after editing lyrics of song

Checklist Items (2)

#1 Show before and after screenshots of data you'll edit

#3 Show an example of successful edit messages

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of edit page code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Form should have correct data types for each property being requested
☐ #2	1	Form should have correct validation for each field (HTML, JS, and PHP)
☐ #3	1	Successful edit should have a user-friendly message
☐ #4	1	Any errors should have user-friendly messages
☐ #5	1	Include any other rules like role guards and login checks
☐ #6	1	Include ucid/date comments for each code screenshot
☐ #7	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

edit.php

public_html > Project > admin > edit.php > html

```php
<?php //rr42 4/19/2024
require(__DIR__ . "/../../../partials/nav.php");
if (!has_role("Admin")) {
    flash("You don't have permission to view this page", "warning");
    die(header("Location: $BASE_PATH" . "/home.php"));
}
if (!isset($_GET['id']) || !is_numeric($_GET['id'])) {
    flash("Invalid song ID", "danger");
    die(header("Location: $BASE_PATH" . "/list.php"));
}
$songId = $_GET['id'];
$db = getDB();
$stmt = $db->prepare("SELECT * FROM SONGS WHERE id = ?");
$stmt->execute([$songId]);
$song = $stmt->fetch(PDO::FETCH_ASSOC);
if (!$song) {
    flash("Song not found", "danger");
    die(header("Location: $BASE_PATH" . "/list.php"));
}
if ($_SERVER["REQUEST_METHOD"] == "POST") {
```

```php
    $title = trim($_POST["title"]);
    $artist = trim($_POST["artist"]);
    $image = trim($_POST["image"]);
    $lyrics = trim($_POST["lyrics"]);

    $query = "UPDATE SONGS SET title = ?, artist = ?, image = ?, lyrics = ? WHERE id = ?";
    $stmt = $db->prepare($query);
    $stmt->execute([$title, $artist, $image, $lyrics, $songId]);

    flash("Song updated successfully", "success");
    header("Location: list.php");
}
?>
```

edit.php

Checklist Items (0)

```php
<!DOCTYPE html> <!--rr42 4/19/2024!-->
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Song</title>
</head>

<body>
    <h1>Edit Song</h1>
    <form method="POST">
        <div>
            <label>Title</label>
            <input type="text" name="title" value="<?php echo htmlspecialchars($song['title']); ?>" required>
        </div>
        <div>
            <label>Artist</label>
            <input type="text" name="artist" value="<?php echo htmlspecialchars($song['artist']); ?>" required>
        </div>
        <div>
            <label>Image</label>
            <input type="text" name="image" value="<?php echo htmlspecialchars($song['image']); ?>" required>
        </div>
        <div>
            <label>Lyrics</label>
            <textarea name="lyrics" rows="5" required><?php echo htmlspecialchars($song['lyrics']); ?></textarea>
        </div>
        <button type="submit">Update Song</button>
    </form>
</body>

</html>
```

edit.php (continued)

Checklist Items (0)

COLLAPSE

Task #3 - Points: 1

Text: Screenshot of records from DB

Checklist

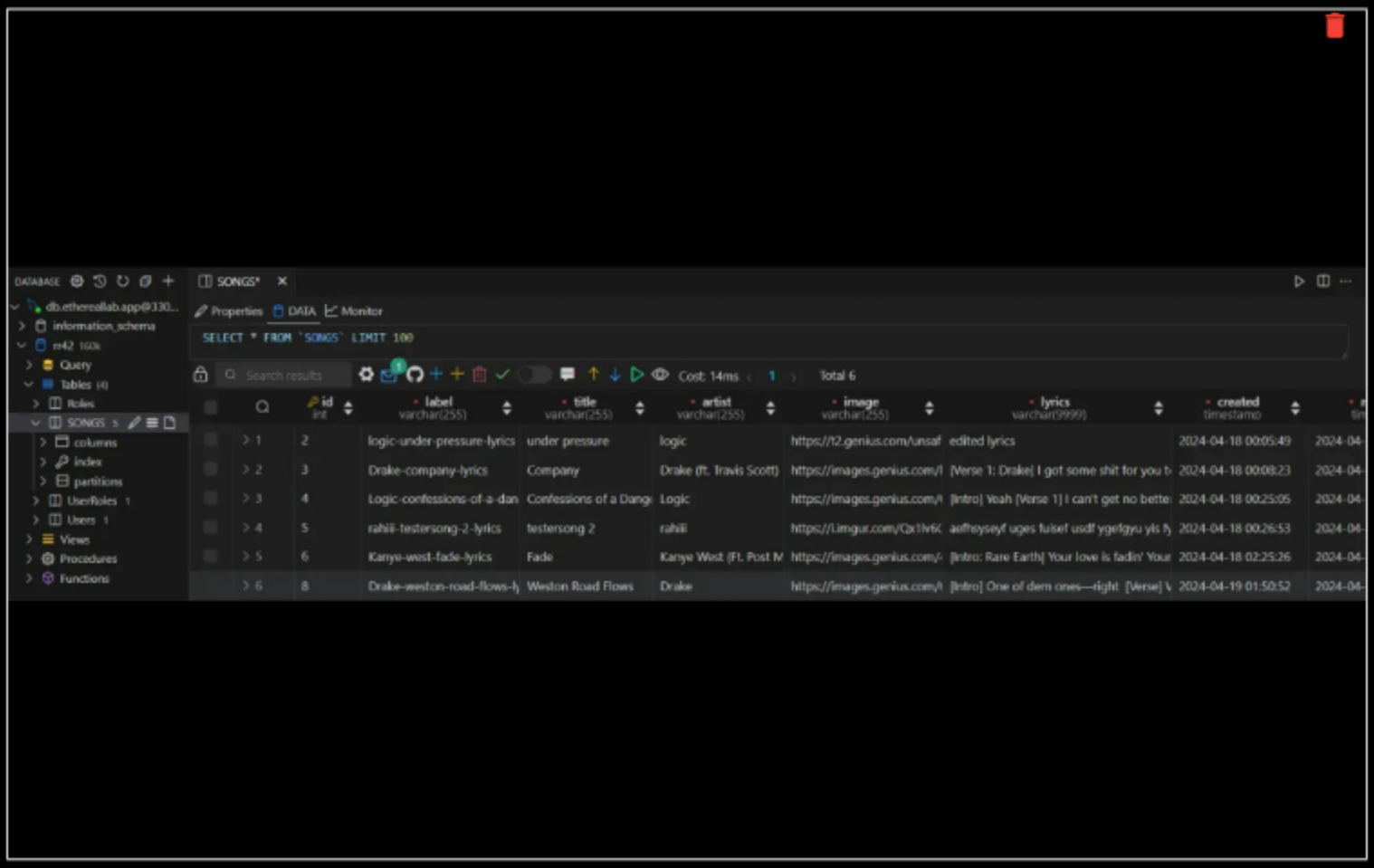
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show a before and after screenshot of the record
#2	1	Note what differs
#3	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

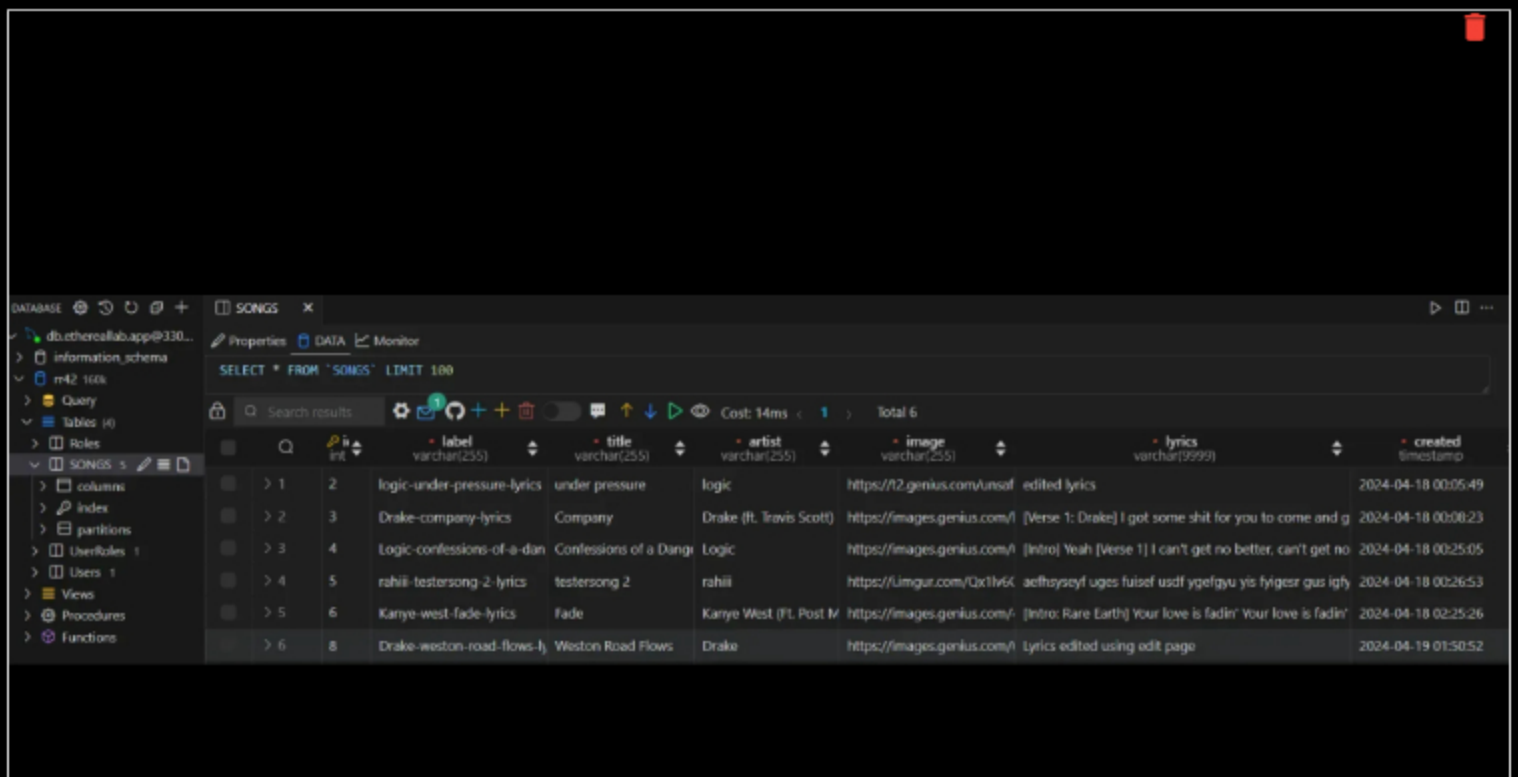
Small Medium Large



Before editing last record

Checklist Items (1)

#1 Show a before and after screenshot of the record



After edit of last entry. Lyrics data different

Checklist Items (2)

#1 Show a before and after screenshot of the record

#2 Note what differs

COLLAPSE

Task #4 - Points: 1

Text: Explain the process

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	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
☐ #2	1	Briefly describe the validations for the applicable fields

Response:

The edit.php page is accessed using a specific song id. The corresponding song data is fetched from the DB and the information is used to fill in the edit form. When the user submits the form, the values in the form fields are used to create an SQL UPDATE query. No form field can be left blank or a validation message will appear on the empty form field.

COLLAPSE

Task #5 - 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-rr42-prod-dc703459f313.herokuapp.com/Project/admin/edit.php?id=8

URL #2

https://github.com/rRahmanM6/rr42-it202-008/pull/66

Delete Handling (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots related to delete

Checklist

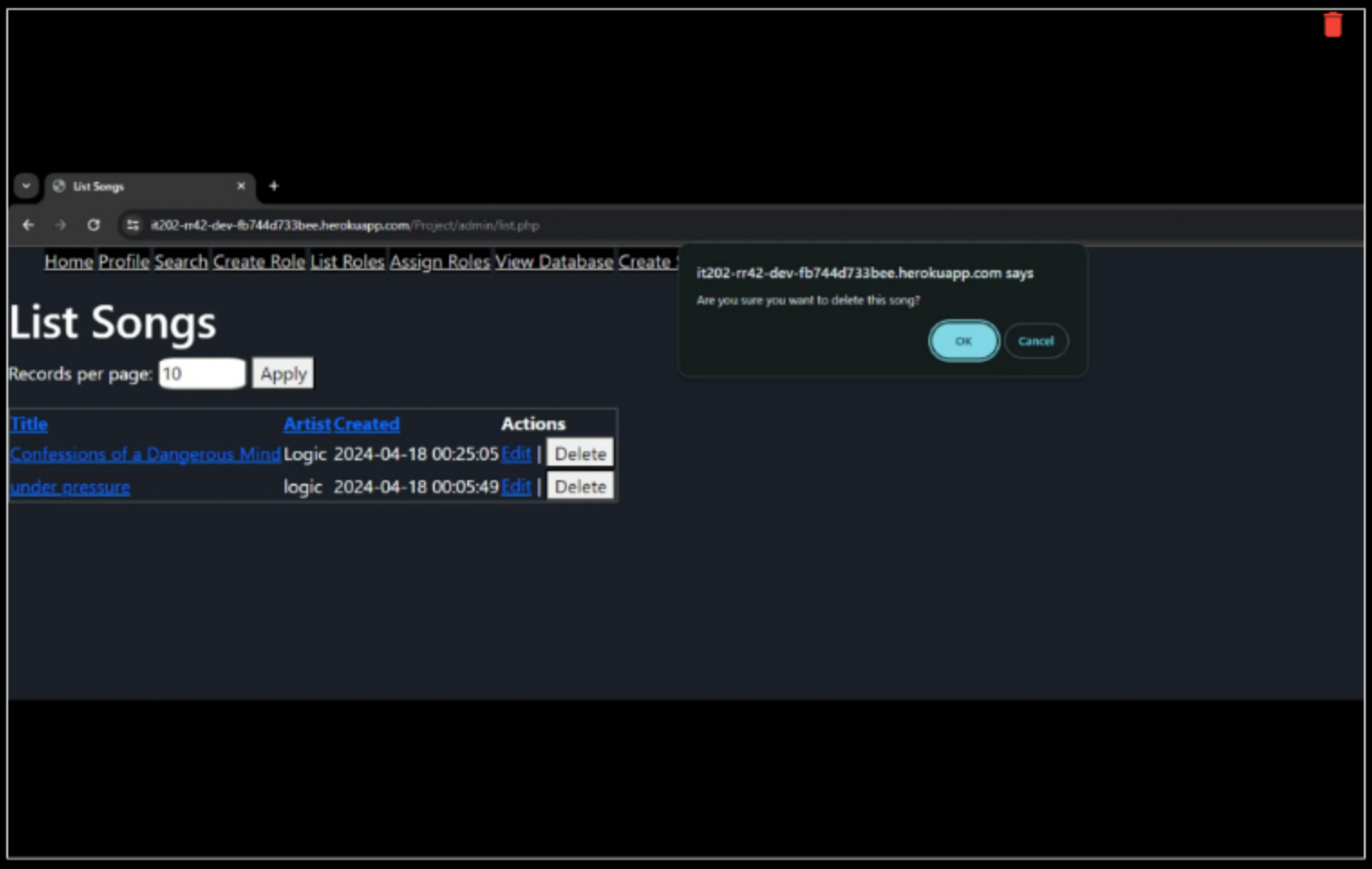
*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Show the success message of a delete
☐ #2	1	Show any error messages of a failed delete (like id not being passed)
☐ #3	1	Show the code related to the delete processing

Task Screenshots:

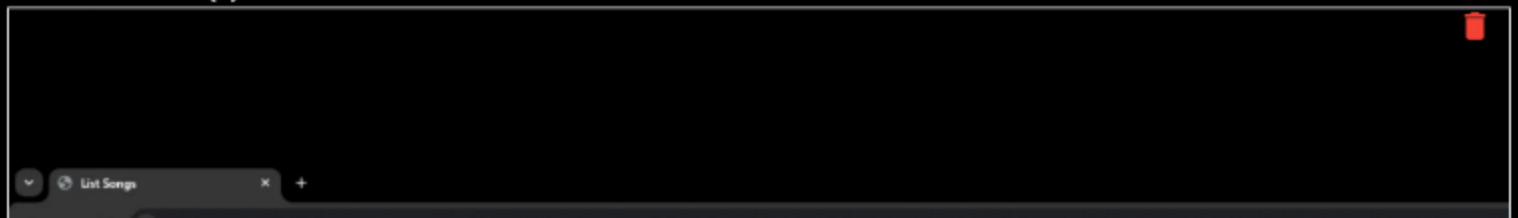
Gallery Style: Large View

Small Medium Large



delete confirmation

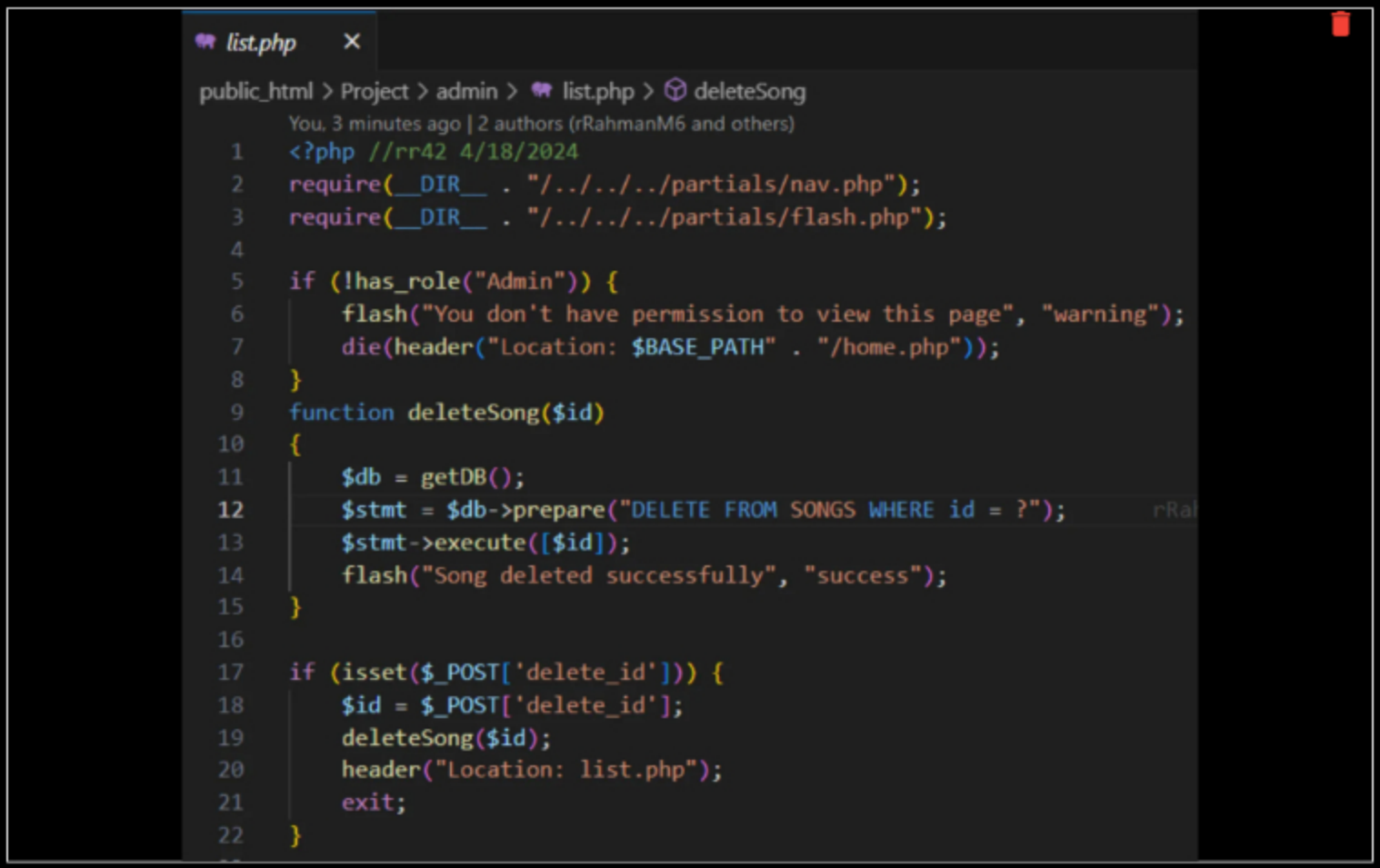
Checklist Items (0)



successful delete

Checklist Items (1)

#1 Show the success message of a delete



delete function in list.php

Checklist Items (1)

#3 Show the code related to the delete processing

```php
<?php foreach ($songs as $song) : ?><!--rr42 4/18/2024!-->
    <tr>
        <td><a href="../search.php?song=<?php echo urlencode($song['title'] . ' ' . $song['artist']); ?>"><?php echo $song['title']; ?></a></td>
        <td><?php echo $song['artist']; ?></td>
        <td><?php echo $song['created']; ?></td>
        <td>
            <a href="edit.php?id=<?php echo $song['id']; ?>">Edit</a> |
            <form method="POST" style="display: inline;">
                <input type="hidden" name="delete_id" value="<?php echo $song['id']; ?>">
                <button type="submit" onclick="return confirm('Are you sure you want to delete this song?')">Delete</button>
            </form>
        </td>
    </tr>
<?php endforeach; ?>
```

delete button in list.php

Checklist Items (1)

#3 Show the code related to the delete processing

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the data

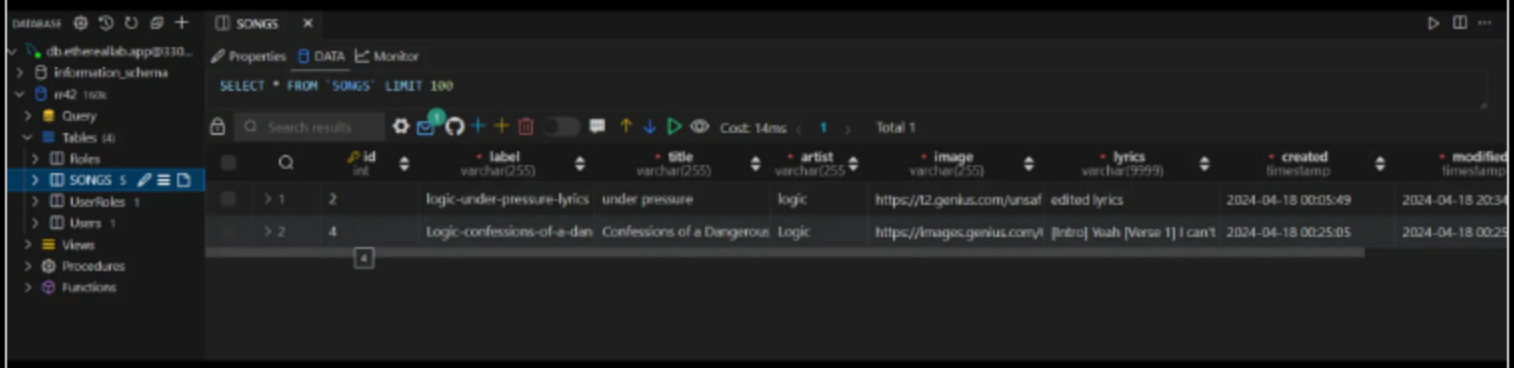
Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Show a before and after screenshot of the DB data
☐ #2	1	Note what changed (i.e., record removed or soft delete value changed)
☐ #3	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



before deleting last record

Checklist Items (1)

#1 Show a before and after screenshot of the DB data

COLLAPSE

Task #3 - Points: 1

Text: Explain the delete logic

Checklist

*The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Is it a soft or hard delete
☐ #2	1	Are there any necessary roles or restrictions? (can only delete their data, can only be done by admin, etc)
☐ #3	1	Provide the high-level steps for handling the DB lookup and handling the delete

Response:

A hard delete is done meaning the record is fully deleted from the database with no backups in place. Only administrators have the ability to delete a record.

An SQL DELETE statement is used to delete a song from the database. The delete statement deletes record where the id matches the song the user wishes to delete.

COLLAPSE

Task #4 - 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/rRahmanM6/rr42-it202-008/pull/66

Misc (1 pt.)

COLLAPSE

COLLAPSE

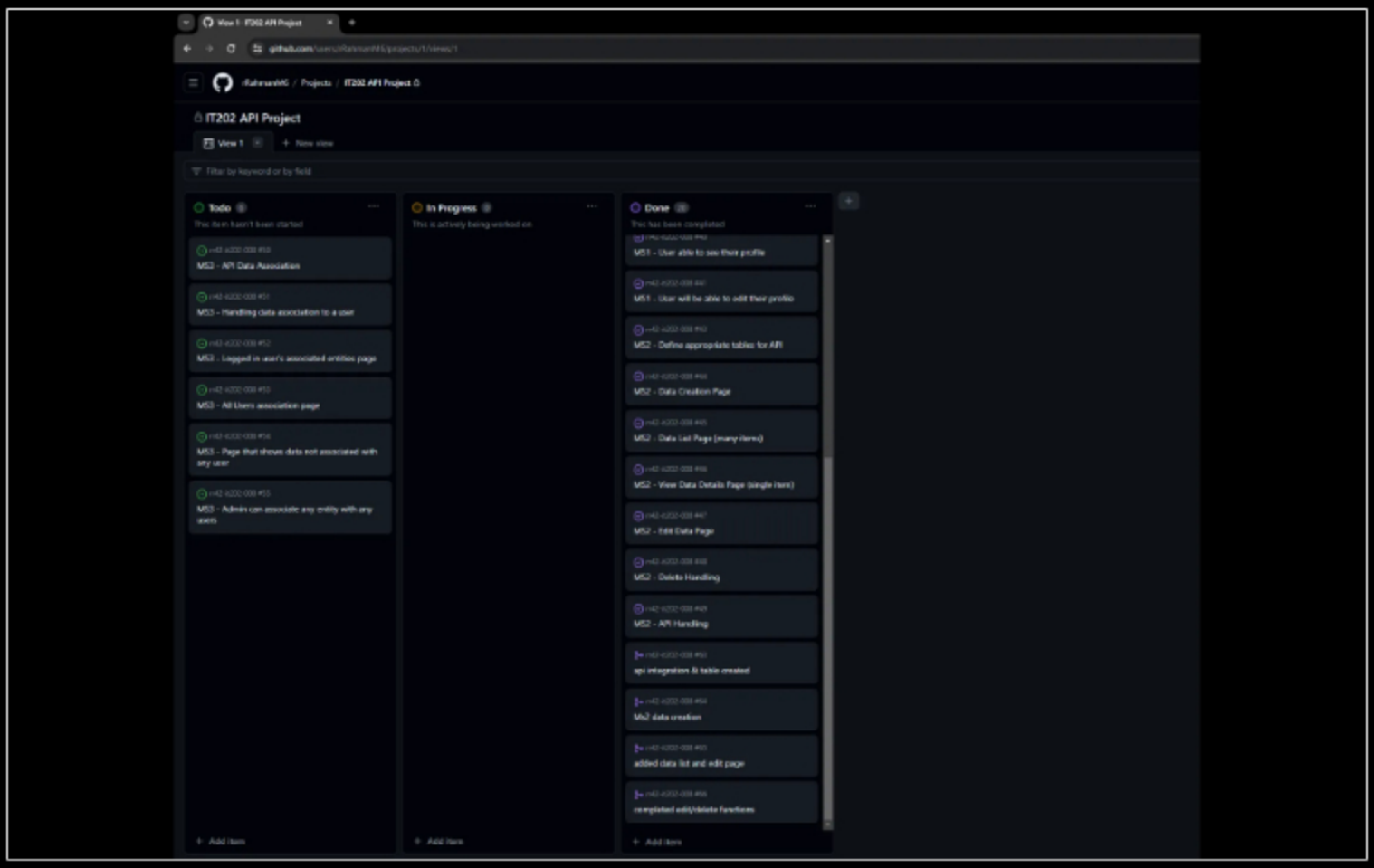
Task #1 - Points: 1

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



project board

COLLAPSE

Task #2 - Points: 1

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

URL #1

https://github.com/users/rRahmanM6/projects/1/views/1

COLLAPSE

Task #3 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

I will say the most difficult part of this project so far has been figuring out how to fetch and work with the API data. I had to do a lot of testing and trial & error to get it to work how I wanted just because of the way my API returns data. After working through that, most other features were relatively straightforward.

COLLAPSE

Task #4 - Points: 1

Text: WakaTime Screenshot

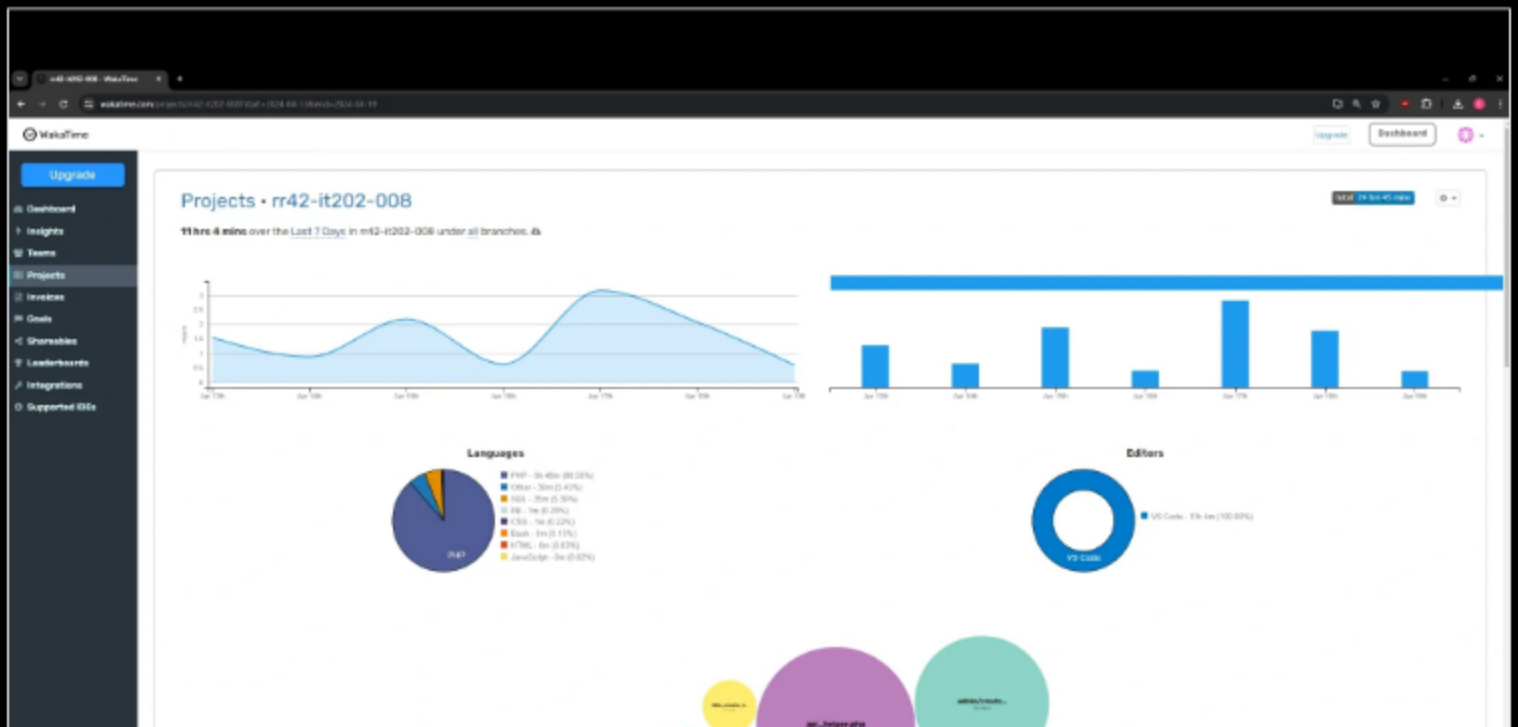
Details:

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



wakatime

End of Assignment