Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-api-project-milestone-3-2024/grade/na569

IT202-008-S2024 - [IT202] API Project Milestone 3 2024

1.
Submissions:
2.
Submission Selection
3.
1 Submission [active] 4/27/2024 12:46:24 PM
4.
5.
6.
Instructions
7.
8.
COLLAPSE
9.
10.

Implement the Milestone 3 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 Milestone3 branch
Create a pull request from Milestone3 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: Milestone3

Tasks: 26 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

#	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:

My website has a "watch list" that users can put movies on. This is a many to many relationship. Multiple users can add the same movie to their watch list, and a single user can have multiple movies on their watch list.

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:

When an update occurs to the movie details in the database, it also updates the movie information on the user's watch list. The users see the new data.

COLLAPSE

Handle Data Association (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the code

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	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)
#2	1	Include ucid/date comments for each code screenshot
#3	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```php
<td>
    <a href="movie_details.php?id=<?php echo $row['id']; ?>" class="btn btn-secondary">View Details</a>
    <a href="edit_movie.php?id=<?php echo $row['id']; ?>" class="btn btn-secondary">Edit</a>
    <a href="delete_movie.php?id=<?php echo $row['id']; ?>" class="btn btn-secondary">Delete</a>
    <a href="add_to_watchlist.php?id=<?php echo $row['id']; ?>" class="btn btn-secondary">Add to WatchList</a>
    //na569, 4/29/24
</td>
```

I added a button called Add to Watchlist as one of my actions on the list_movies page. The button links the add_to_watchlist page that adds the movie to the user's watchlist.

Checklist Items (3)

#1 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)

#2 Include ucid/date comments for each code screenshot

#3 Clearly caption screenshots

^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshot of the association table(s)

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the table(s) you made to handle the associations
#2	1	Should have some example data
#3	1	Make sure the heroku dev url is visible in the address bar
#4	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
SELECT * FROM `UserMovies` LIMIT 100
```

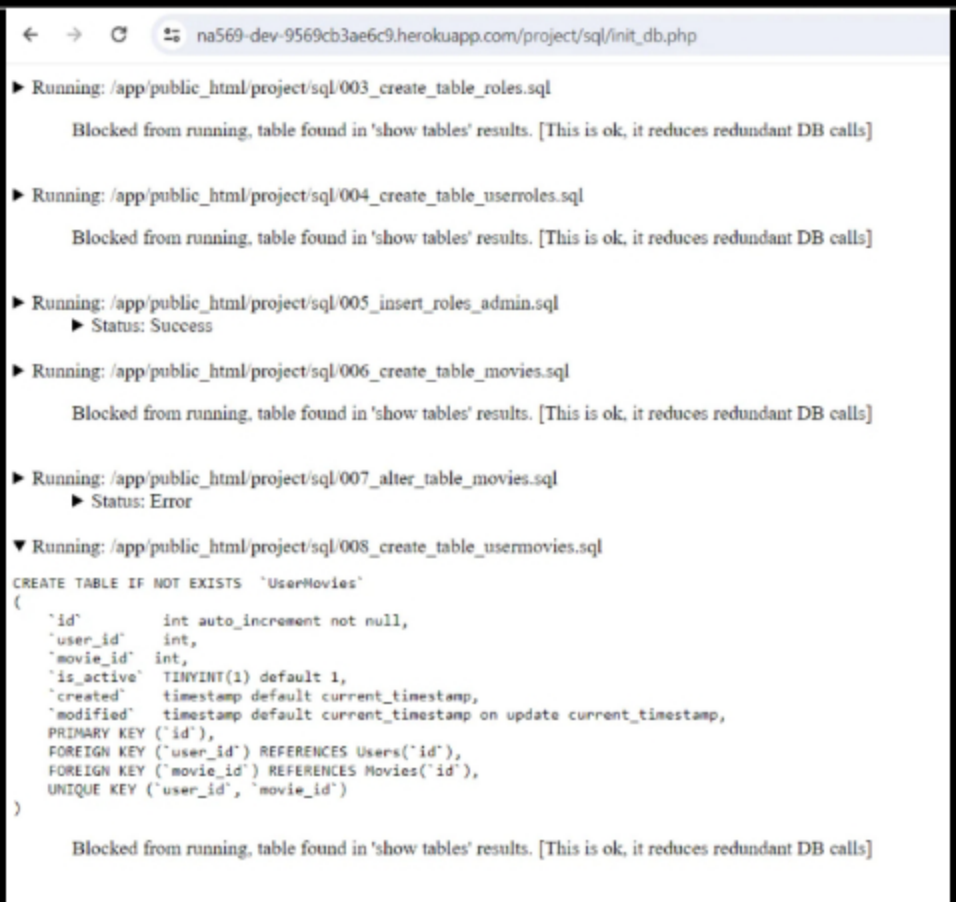
Search results | Cost: 74ms | 1 | Total 7

	id int	user_id int	movie_id int	is_active tinyint(1)	created timestamp	modified timestamp
1	38	5	397	1	2024-04-30 04:08:48	2024-04-30 04:08:48
2	39	5	398	1	2024-04-30 04:08:49	2024-04-30 04:08:49
3	40	5	399	1	2024-04-30 04:08:50	2024-04-30 04:08:50
4	41	5	109	1	2024-04-30 04:08:52	2024-04-30 04:08:52
5	52	1	795	1	2024-04-30 14:23:41	2024-04-30 14:23:41
6	53	1	789	1	2024-04-30 14:23:43	2024-04-30 14:23:43
7	56	1	398	1	2024-04-30 16:07:01	2024-04-30 16:07:01

table on VScode

Checklist Items (1)

#2 Should have some example data



Init complete, used approximately 4 db calls.

init_db.php on heroku dev of association table

Checklist Items (2)

#1 Show the table(s) you made to handle the associations

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

COLLAPSE

Task #3 - Points: 1

Text: Explain solution

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Mention which option your application handles regarding association
#2	1	Describe each column/association table

Response:

I went with Option 1 to add the logic for association on an already existing page. I allow users to put movies on their watchlist. The watchlist contains information regarding the movies in the UserMovies table that keeps track of the associations. It contains details about the movies such as title, genre, released data, synopsis, and buttons that link to actions.

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 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://na569-prod-74aadc581478.herokuapp.com/project/list_movies.php

URL #2

https://github.com/nafisa37/na569-it202-008/pull/64

Current User's Association Page (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of this page

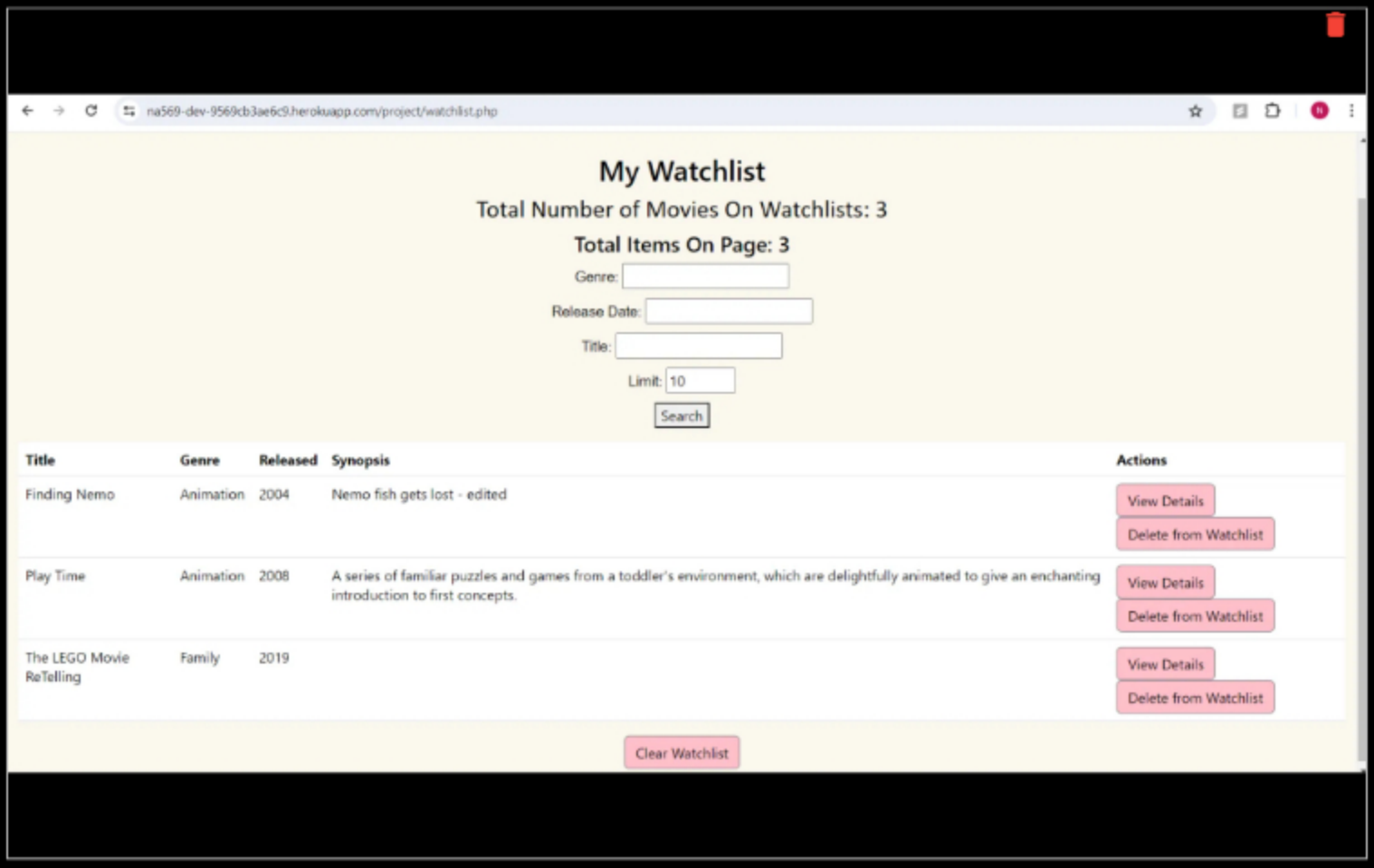
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the summary of the results with relevant information per entity
#2	1	Show the single view buttons/links, delete button/links, and delete all button/link
#3	1	Show variations of the number of shown items count and show the count of total number of associated items to the user
#4	1	Show variations of the filter/sort including no results (proper message should be visible)
#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

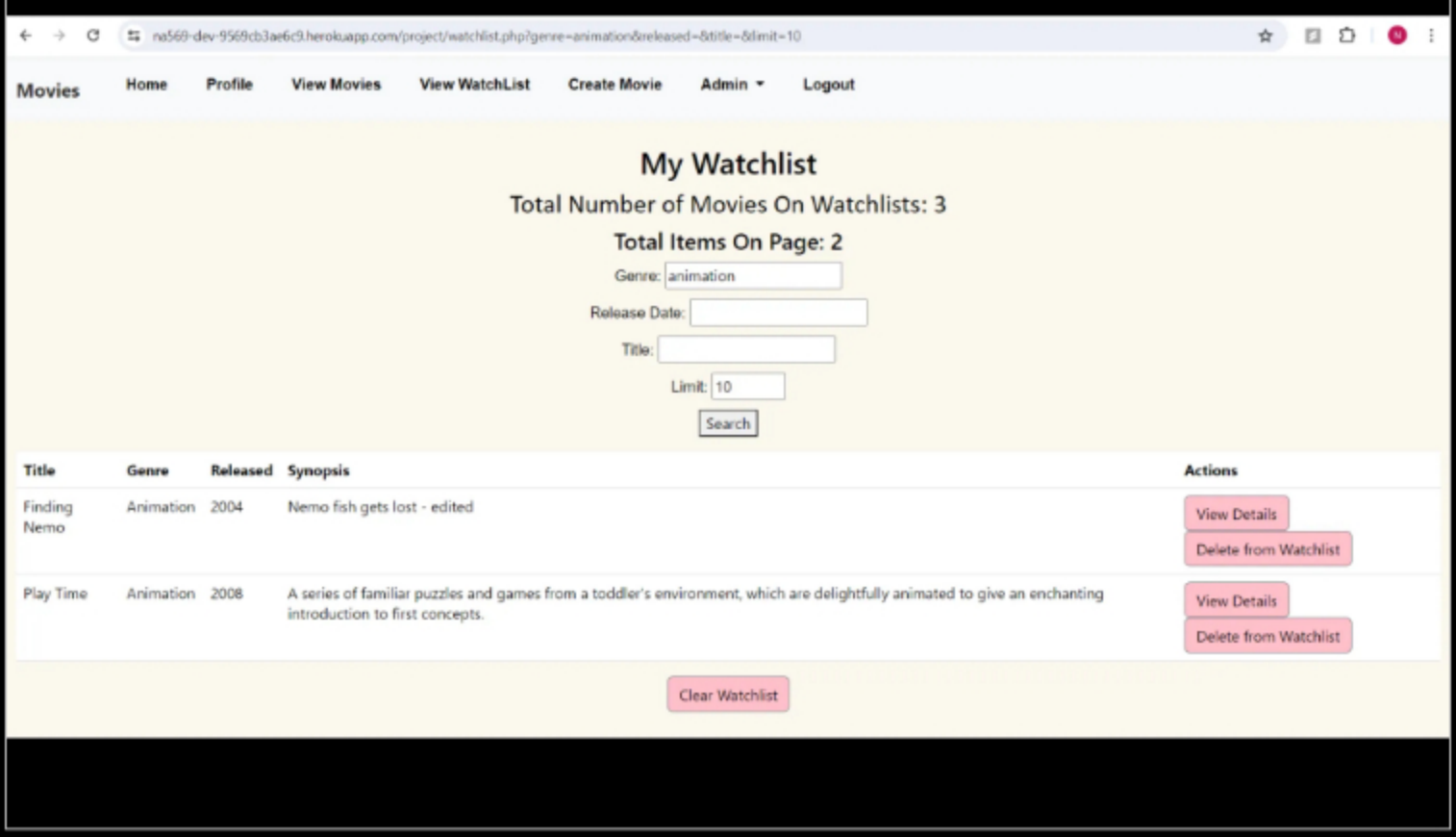
Small Medium Large



shows summary of results with relevant information about each entity, contains the view details and delete association buttons

Checklist Items (1)

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

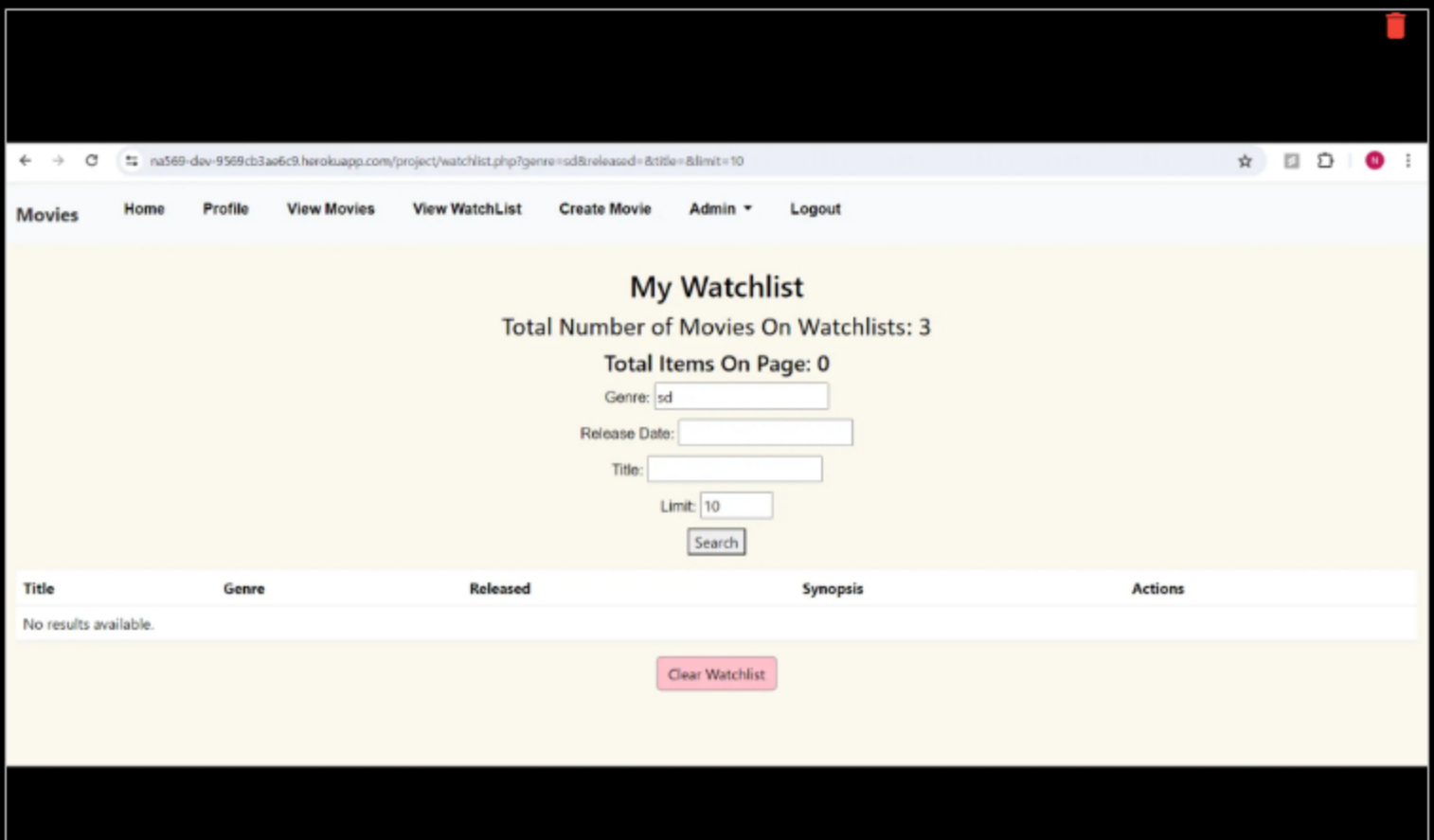


shows variations of the visible results count and total number of movies on watchlist count, also includes a filter on the genre

Checklist Items (2)

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

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



no results available when filtering/sorting

Checklist Items (1)

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

COLLAPSE

Task #2 - Points: 1

Text: Screenshot the code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the code related to fetching the user's associations (including the query)
#2	1	Show the code related to the display of the results
#3	1	Each record should have a button/link for single view
#4	1	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
#5	1	Show the logic for deleting all associations for the user (this may be admin-only but should be present for the specific role)
#6	1	Show the logic related to the count of all associated items to the user (even the ones not shown in the filtered results)
#7	1	Show the logic related to the count of the items on the page (this value should change based on the filter applied)
#8	1	Show the logic related to filter/sort (limit should be constrained to 1-100 otherwise default to 10)
#9	1	Include ucid/date comments for each code screenshot
#10	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```php
$genre = isset($_GET['genre']) ? $_GET['genre'] : '';
$title = isset($_GET['title']) ? $_GET['title'] : '';
$released = isset($_GET['released']) ? $_GET['released'] : '';
$limit = isset($_GET['limit']) ? intval($_GET['limit']) : 10; // Default to 10
$limit = max(1, min($limit, 100));

$query = "SELECT m.id, m.title AS Title, m.genre AS Genre, m.released AS Released, m.synopsis AS Synopsis
          FROM UserMovies um
          JOIN Movies m ON um.movie_id = m.id
          WHERE um.user_id = :user_id";
$params = [":user_id" => $user_id];
if (!empty($genre)) {
    $query .= " AND m.genre LIKE :genre";
    $params[':genre'] = "%$genre%";
}
if (!empty($title)) {
    $query .= " AND m.title LIKE :title";
```

```php
    $params[':title'] = "%$title%";
}
if (!empty($released)) {
    $query .= " AND m.released LIKE :released";
    $params[':released'] = "%$released%";
}
$query .= " ORDER BY m.created DESC LIMIT $limit";

// Execute the query, na569, 4/30/24
$db = getDB();
$stmt = $db->prepare($query);
try {
    $stmt->execute($params);
    $results = $stmt->fetchAll();
} catch (PDOException $e) {
    error_log("Error fetching movies: " . $e->getMessage());
    flash("An error occurred while fetching movies", "danger");
}
```

code for fetching user's watchlist movies

Checklist Items (1)

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

```php
<table class="table">
    <thead>
        <tr>
            <th>Title</th>
            <th>Genre</th>
            <th>Released</th>
            <th>Synopsis</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        <?php if (empty($results)) : ?>
            <tr>
                <td colspan="5">No results available.</td>
            </tr>
        <?php else : ?>
            <?php foreach ($results as $movie) : ?>
                <tr>
                    <td><?php echo htmlspecialchars($movie['Title']); ?></td>
                    <td><?php echo htmlspecialchars($movie['Genre']); ?></td>
                    <td><?php echo htmlspecialchars($movie['Released']); ?></td>
                    <td><?php echo htmlspecialchars($movie['Synopsis']); ?></td>
                    <td>
                        <a href="movie_details.php?id=<?php echo $movie['id']; ?>" class="btn btn-secondary">View Details</a>
                        <a href="delete_association.php?id=<?php echo $movie['id']; ?>" class="btn btn-secondary">Delete from W
                    </td>
                </tr>
            <?php endforeach; ?>
        <?php endif; ?>
    </tbody>
</table>
</div> //na569, 4.30.24
```

code showing results of the watchlist, links to buttons to view details and delete relationship

Checklist Items (3)

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

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

#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
//na569, 4.30.24
$user_id = get_user_id();

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["clear_watchlist"])) {
    $db = getDB();
    $query = "DELETE FROM UserMovies WHERE user_id = :user_id";
    $stmt = $db->prepare($query);
    try {
        $stmt->execute([":user_id" => $user_id]);
        flash("Watchlist cleared successfully", "success");
        die(header("Location: " . get_url("watchlist.php")));
        exit;
    } catch (PDOException $e) {
        error_log("Error clearing watchlist: " . $e->getMessage());        You, 17
        flash("An error occurred while clearing watchlist", "danger");
    }
} <- #13-27 if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["cle...
```

deletes all movies off watchlist

Checklist Items (1)

#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
// Execute the query, na569, 4/30/24
$db = getDB();
$stmt = $db->prepare($query);
try {
    $stmt->execute($params);
    $results = $stmt->fetchAll();
} catch (PDOException $e) {
    error_log("Error fetching movies: " . $e->getMessage());
    flash("An error occurred while fetching movies", "danger");
}

$query = "SELECT COUNT(*) AS total_count FROM UserMovies WHERE user_id = :user_id";
$params = [":user_id" => $user_id];

// Execute the query to fetch the count, na569, 4.30.24
$db = getDB();
try {       You, 17 hours ago • going through most of the DataAssociation bulle…
    $stmt = $db->prepare($query);
    $stmt->execute($params);
    $row = $stmt->fetch(PDO::FETCH_ASSOC);
    $totalItemsCount = $row['total_count'];
} catch (PDOException $e) {
    error_log("Error fetching total count: " . $e->getMessage());
    flash("An error occurred while fetching total count", "danger");
    $totalItemsCount = 0;
} <- #75-79 catch (PDOException $e)

?>

<div style="text-align: center;">
<div class="container-fluid">
    <h2>My Watchlist</h2>
        <h3> Total Number of Movies On Watchlists: <?php echo $totalItemsCount; ?></h3>
        <h4>Total Items On Page: <?php echo count($results); ?></h4>
    <form method="GET">
```

shows logic and queries to get count of all items on page and all items that are available that are associated to user

Checklist Items (2)

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

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

```
$genre = isset($_GET['genre']) ? $_GET['genre'] : '';
$title = isset($_GET['title']) ? $_GET['title'] : '';
$released = isset($_GET['released']) ? $_GET['released'] : '';
$limit = isset($_GET['limit']) ? intval($_GET['limit']) : 10; // Default to 10
$limit = max(1, min($limit, 100));

$query = "SELECT m.id, m.title AS Title, m.genre AS Genre, m.released AS Released, m.synopsis AS Synopsis
          FROM UserMovies um
          JOIN Movies m ON um.movie_id = m.id
          WHERE um.user_id = :user_id";
$params = [":user_id" => $user_id];
if (!empty($genre)) {
    $query .= " AND m.genre LIKE :genre";
    $params[':genre'] = "%$genre%";
}
if (!empty($title)) {
    $query .= " AND m.title LIKE :title";
    $params[':title'] = "%$title%";
}
if (!empty($released)) {
    $query .= " AND m.released LIKE :released";
    $params[':released'] = "%$released%";
}
$query .= " ORDER BY m.created DESC LIMIT $limit";

// Execute the query, na569, 4/30/24
```

shows logic to filter/sort through results

Checklist Items (1)

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

COLLAPSE

Task #3 - Points: 1

Text: Explain the solution

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Mention how you determine the result list (include the association logic and filters)
#2	1	Mention the logic behind the two counts (total items and visible result count)
#3	1	Mention the logic for single delete and delete all associations for the specific user

Response:

The result list is determined by creating a query that fetches associated data from the UserMovies table. It joins the

The result list is determined by creating a query that fetches associated data from the UserMovies table. It joins the UserMovies table with the Movies table by the movie_id column to fetch the details about the movies added to the user's watchlist. The total items count is created by an SQL query that counts the number of associations that are in the UserMovies for the user. The visible result count comes from the number of the rows that are returned from the query after the filters are applied. When the user clicks Delete From Watchlist button, it links to the delete_association page that deletes the row containing that movie ID from the UserMovies table. When the user clicks the Clear Watchlist button, it triggers an SQL query that deletes all the associations from the UserMovies table.

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 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://na569-prod-74aadc581478.herokuapp.com/project/watchlist.php

URL #2

https://github.com/nafisa37/na569-it202-008/pull/65

COLLAPSE

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

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of this page

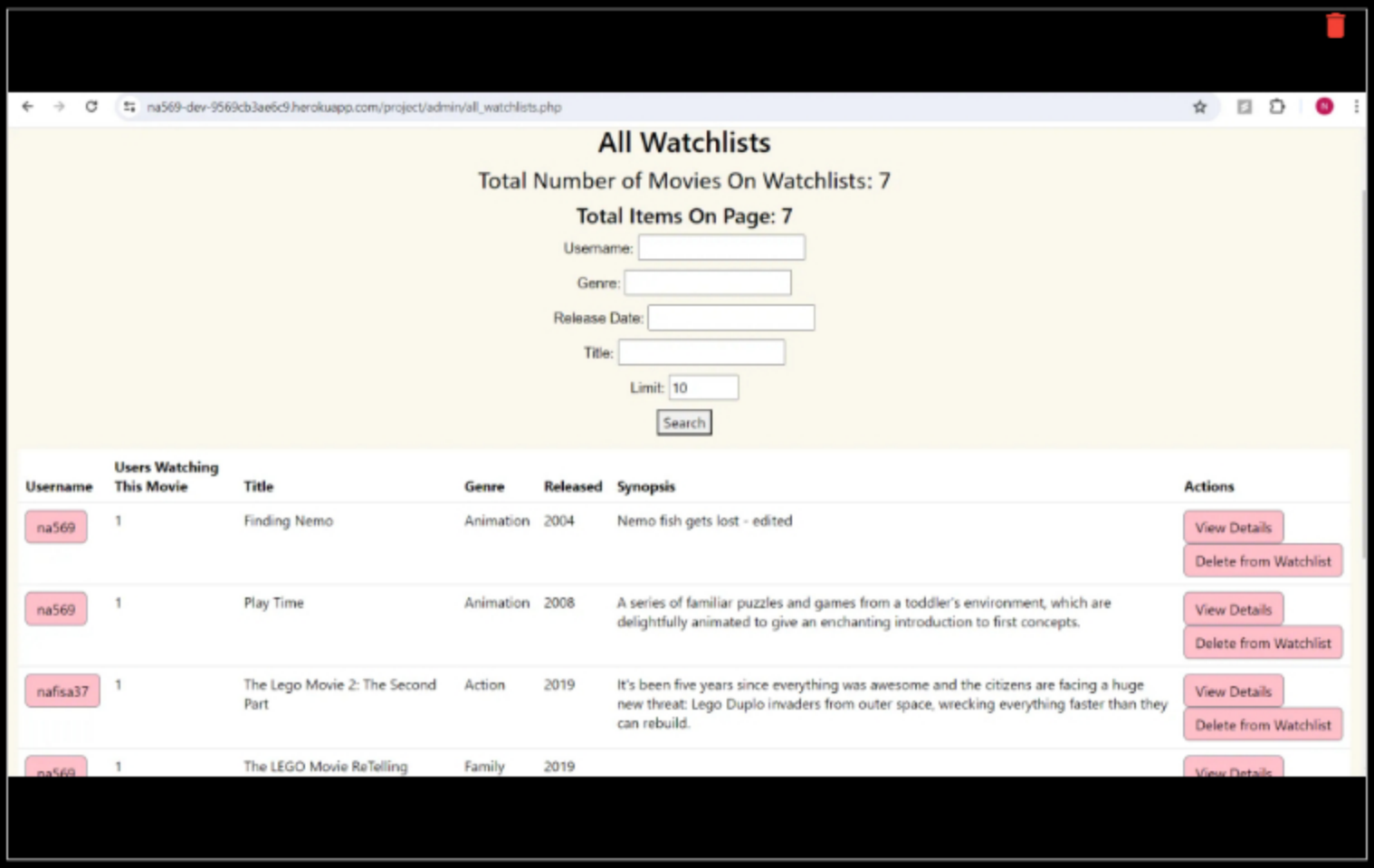
Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Show the summary of the results with relevant information per entity
☐ #2	1	Show the single view buttons/links, delete button/links
☐ #3	1	Show the username related to the specific entity and that it's clickable
☐ #4	1	Show variations of the number of shown items count and show the count of total number of associated items
☐ #5	1	Show variations of the filter/sort including no results (proper message should be visible)
☐ #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



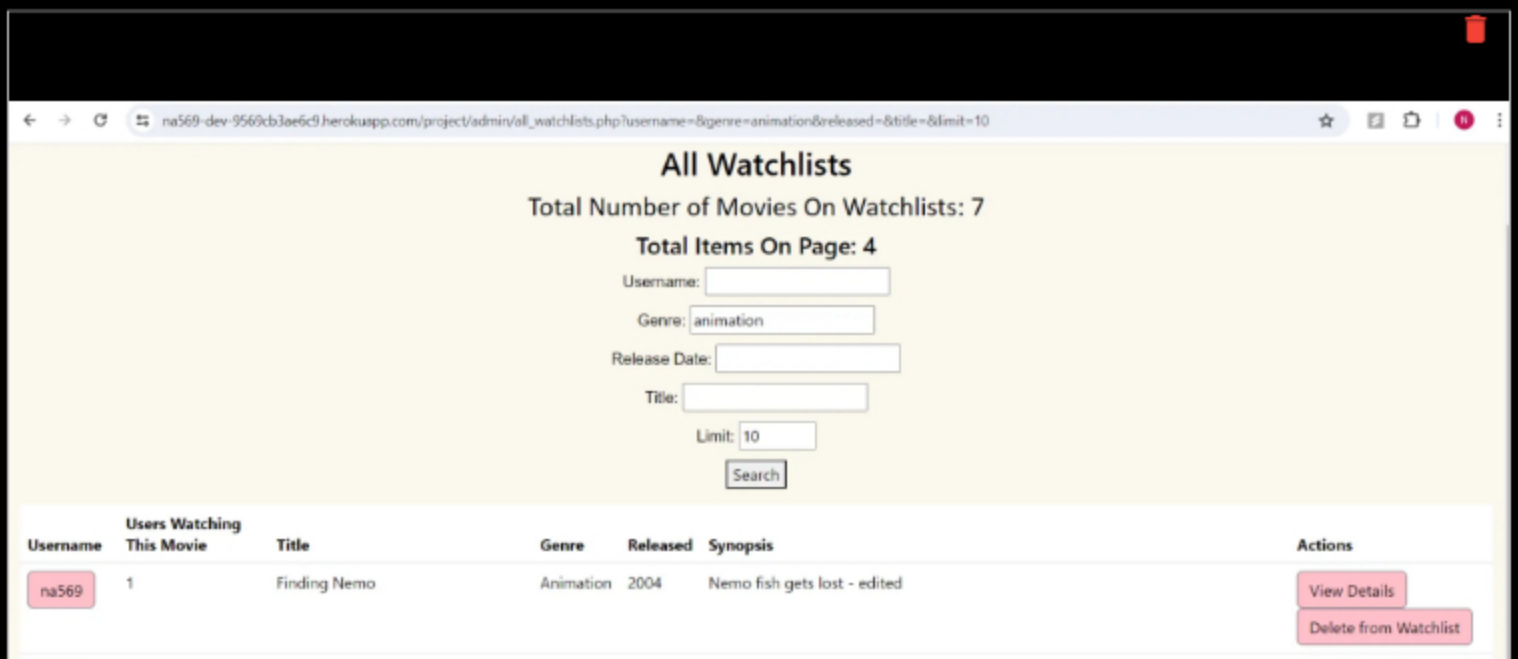
showcases summary of results, contains link to view details and delete association, clickable button for username that leads to username's profile page

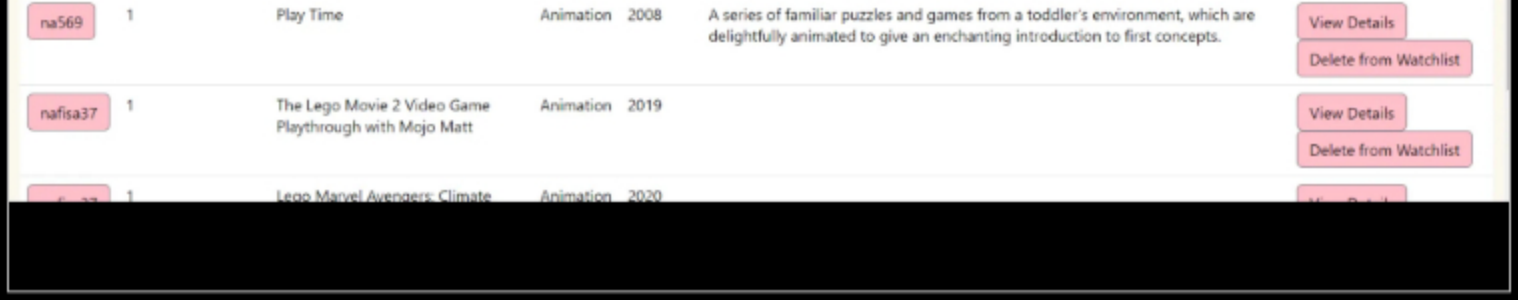
Checklist Items (3)

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

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

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



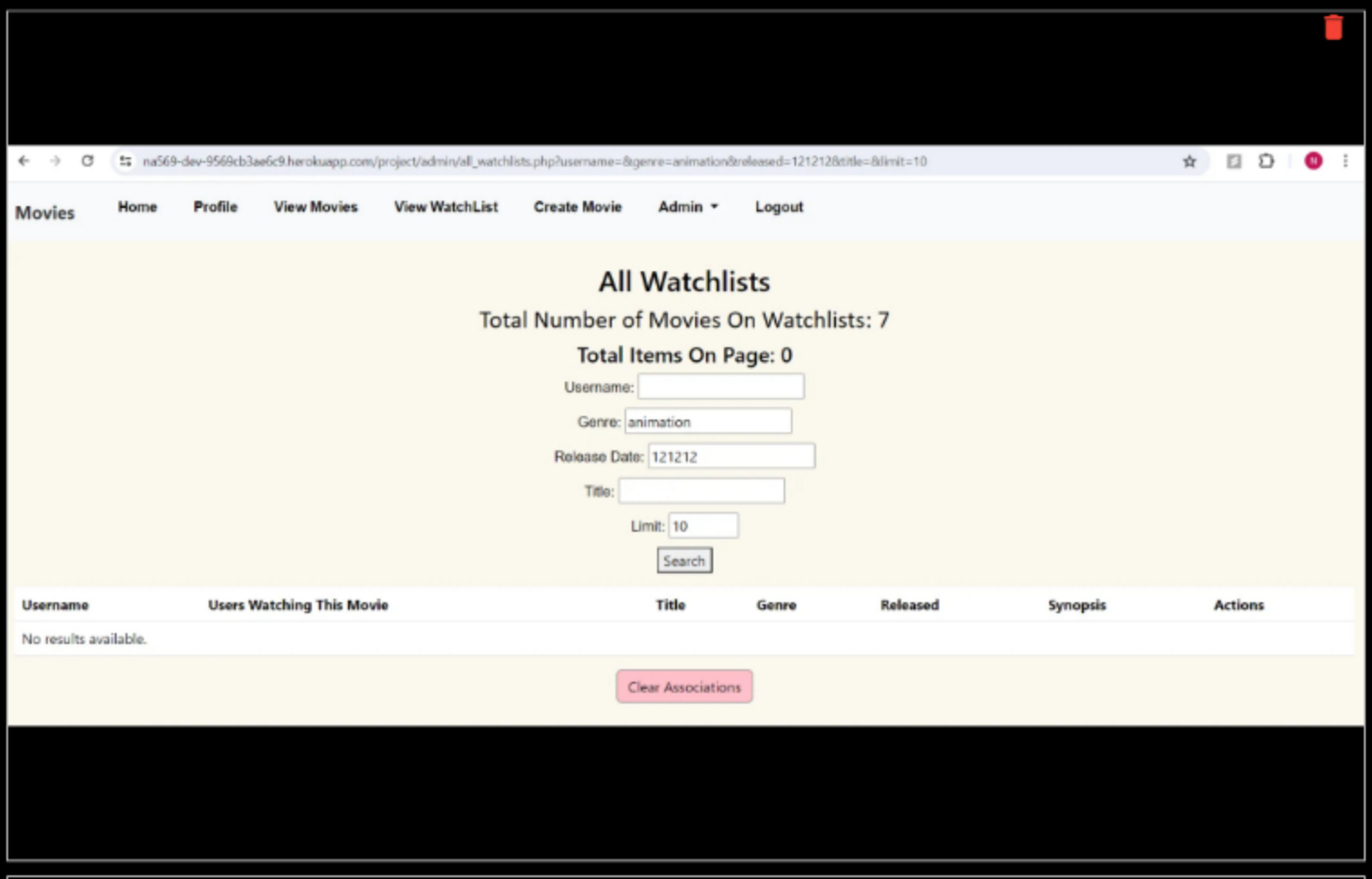


shows variation of the total number of movies on watchlist and total items showcased on page with genre filtered as well

Checklist Items (2)

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

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



no results available when filtering through results

Checklist Items (1)

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

COLLAPSE

Task #2 - Points: 1

Text: Screenshot the code

Checklist *The checkboxes are for your own tracking

Checklist — The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the code related to fetching all associations (including the query)
#2	1	Show the code related to the display of the results
#3	1	Each record should have a button/link for single view
#4	1	Each record should have a username field that is clickable to go to the user's profile page
#5	1	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
#6	1	Show the logic related to the count of all associated items (even the ones not shown in the filtered results)
#7	1	Show the logic related to the count of the items on the page (this value should change based on the filter applied)
#8	1	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)
#9	1	Include ucid/date comments for each code screenshot
#10	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```php
// Execute the query, na569, 4.30.24
$db = getDB();
$stmt = $db->prepare($query);
try {      You, 10 hours ago • finishing up all users association
    $stmt->execute($params);
    $results = $stmt->fetchAll();
} catch (PDOException $e) {
    error_log("Error fetching movies: " . $e->getMessage());
    flash("An error occurred while fetching movies", "danger");
}
```

fetches the associations

Checklist Items (1)

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

```
<table class= table >
    <thead>
        <tr>
            <th>Username</th>
            <th>Users Watching This Movie</th>
            <th>Title</th>
            <th>Genre</th>
            <th>Released</th>
            <th>Synopsis</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        <?php if (empty($results)) : ?>
            <tr>
                <td colspan="7">No results available.</td>
            </tr>
        <?php else : ?>
            <?php foreach ($results as $movie) : ?>
                <tr>
                    <td>
                        <a href="../profile.php" class="btn btn-secondary"><?php echo htmlspecialchars($movie['Username']);
                    </td>
                    <td><?php echo $movie['Watchlist_Count']; ?></td>
                    <td><?php echo htmlspecialchars($movie['Title']); ?></td>
                    <td><?php echo htmlspecialchars($movie['Genre']); ?></td>
                    <td><?php echo htmlspecialchars($movie['Released']); ?></td>
                    <td><?php echo htmlspecialchars($movie['Synopsis']); ?></td>
                    <td>
                        <a href="../movie_details.php?id=<?php echo $movie['id']; ?>" class="btn btn-secondary">View Detail
                        <a href="../delete_association.php?id=<?php echo $movie['id']; ?>" class="btn btn-secondary">Delete
                    </td>
                </tr>
            <?php endforeach; ?> //na569, 4.30.24
        <?php endif; ?>
    </tbody>
</table>
```

showcases table in which data is displayed, contains link to view details and delete association

Checklist Items (3)

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

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

#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

```
// Execute the query, na569, 4.30.24
$db = getDB();
$stmt = $db->prepare($query);
try {
    $stmt->execute($params);
    $results = $stmt->fetchAll();
} catch (PDOException $e) {
    error_log("Error fetching movies: " . $e->getMessage());
    flash("An error occurred while fetching movies", "danger");
}

$totalItemsQuery = "SELECT COUNT(*) AS total_items FROM UserMovies";
$stmt = $db->prepare($totalItemsQuery);
$stmt->execute();
$row = $stmt->fetch(PDO::FETCH_ASSOC);
$totalItemsCount = $row['total_items'];
```

```php
?>

<div style="text-align: center;">
    <div class="container-fluid">
        <h2>All Watchlists</h2>
        <h3> Total Number of Movies On Watchlists: <?php echo $totalItemsCount; ?></h3>
        <h4>Total Items On Page: <?php echo count($results); ?></h4>
```

shows logic related to count of number of items on page and number of associations in general

Checklist Items (2)

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

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

```php
$username = isset($_GET['username']) ? $_GET['username'] : '';
$genre = isset($_GET['genre']) ? $_GET['genre'] : '';
$title = isset($_GET['title']) ? $_GET['title'] : '';
$released = isset($_GET['released']) ? $_GET['released'] : '';
$limit = isset($_GET['limit']) ? intval($_GET['limit']) : 10; // Default to 10
$limit = max(1, min($limit, 100));

$query = "SELECT m.id, m.title AS Title, m.genre AS Genre, m.released AS Released, m.synopsis AS Synopsis,
                 COUNT(DISTINCT um.user_id) AS Watchlist_Count, u.username AS Username
          FROM UserMovies um
          JOIN Movies m ON um.movie_id = m.id
          JOIN Users u ON um.user_id = u.id
          WHERE 1=1";
$params = [];
if (!empty($username)) {
    $query .= " AND u.username LIKE :username";
    $params[':username'] = "%$username%";
}
if (!empty($genre)) {
    $query .= " AND m.genre LIKE :genre";
    $params[':genre'] = "%$genre%";
}
if (!empty($title)) {
    $query .= " AND m.title LIKE :title";
    $params[':title'] = "%$title%";
}
if (!empty($released)) {
    $query .= " AND m.released LIKE :released";
    $params[':released'] = "%$released%";
}
$query .= " GROUP BY m.id, m.title, m.genre, m.released, m.synopsis, u.id, u.username
            ORDER BY m.created DESC LIMIT $limit";

// Execute the query, na569, 4.30.24      You, 2 minutes ago • Uncommitted changes
```

filter/sort logic now including username

Checklist Items (1)

#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)

```php
<?php
require_once(__DIR__ . "/../../partials/nav.php");
//is_logged_in(true);
$user_id = -1;
try {
    $user_id = (int)se($_GET, "id", -1, false);
} catch (Exception $e) {
    //we know it's a data format issue
}
if ($user_id < 1) {
```

```php
    $user_id = get_user_id(); //get our id if we're logged in //ma569, 4.30.24
}
$is_me = $user_id == get_user_id();
$is_edit = isset($_GET["edit"]);
?>
<?php
if ($is_me && $is_edit && isset($_POST["save"])) {
    $email = se($_POST, "email", null, false);
    $username = se($_POST, "username", null, false);
    $hasError = false;
    //sanitize
    $email = sanitize_email($email);
    //validate
    if (!is_valid_email($email)) {
        flash("Invalid email address", "danger");
        $hasError = true;
    }
    if (!is_valid_username($username)) {
        flash("Username must only contain 3-16 characters a-z, 0-9, _, or -", "danger");
        $hasError = true;
    }
    if (!$hasError) {
```

edit to profile page that allows username to be clickable

Checklist Items (1)

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

COLLAPSE

Task #3 - Points: 1

Text: Explain the solution

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Mention how you determine the result list (include the association logic and filters)
#2	1	Mention the logic behind the two counts (total items and visible result count)
#3	1	Mention the logic for single delete
#4	1	Mention the logic for handling the username requirements

Response:

The result list query joins the UserMovies table with the Movies and Users table to fetch all the necessary information. If a username is provided, the SQL filters through the movies based on the username of the user who added it to their watchlist. You can also filter based on genre, title, released date. The total items count is retrieved by a COUNT query on the UserMovies table. The visible result count is retrieved from counting the number of rows returned by the query. When the user clicks Delete from Watchlist, it calls delete_association, which contains logic to delete the association from the UserMovies table based on the movie ID. $_GET['username'] is used to check if there is a username. If there is, the SQL query filters through the movies by the username.

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 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://na569-prod-74aadc581478.herokuapp.com/project/admin/all_watchlists.php

URL #2
https://github.com/nafisa37/na569-it202-008/pull/65

Unassociated Page (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of this page

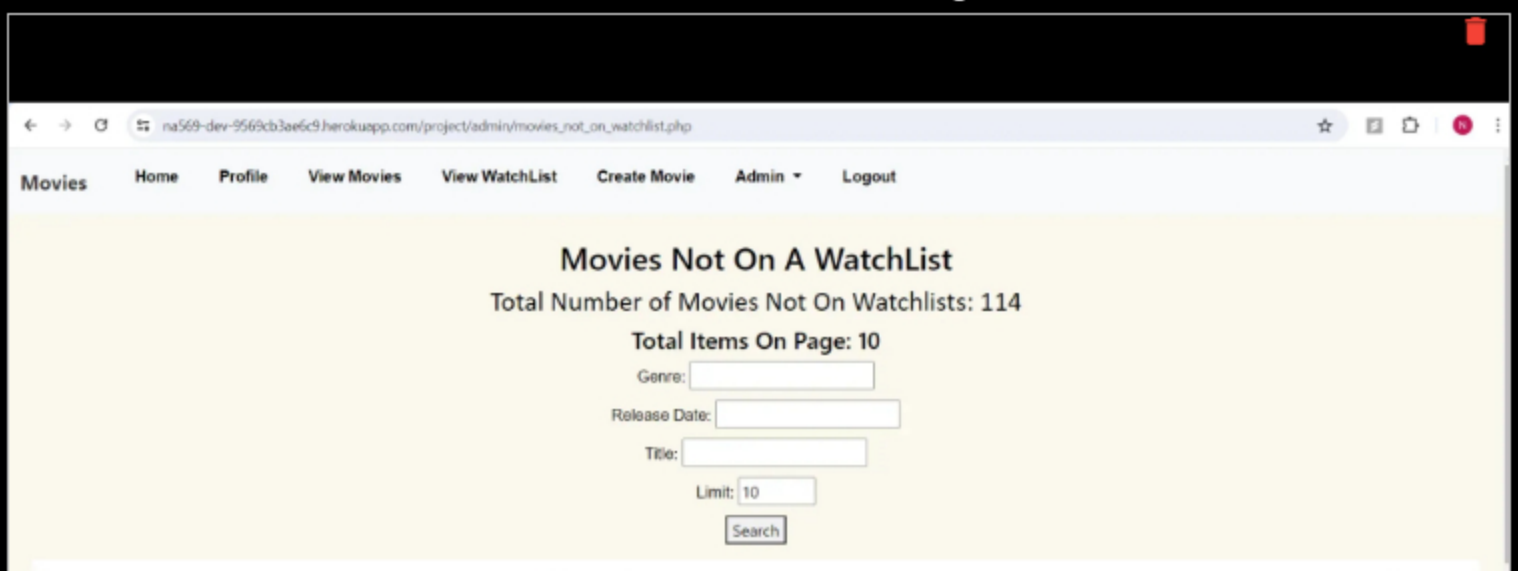
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the summary of the results with relevant information per entity
#2	1	Show the single view buttons/links
#3	1	Show variations of the number of shown items count and show the count of total number of associated items
#4	1	Show variations of the filter/sort including no results (proper message should be visible)
#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



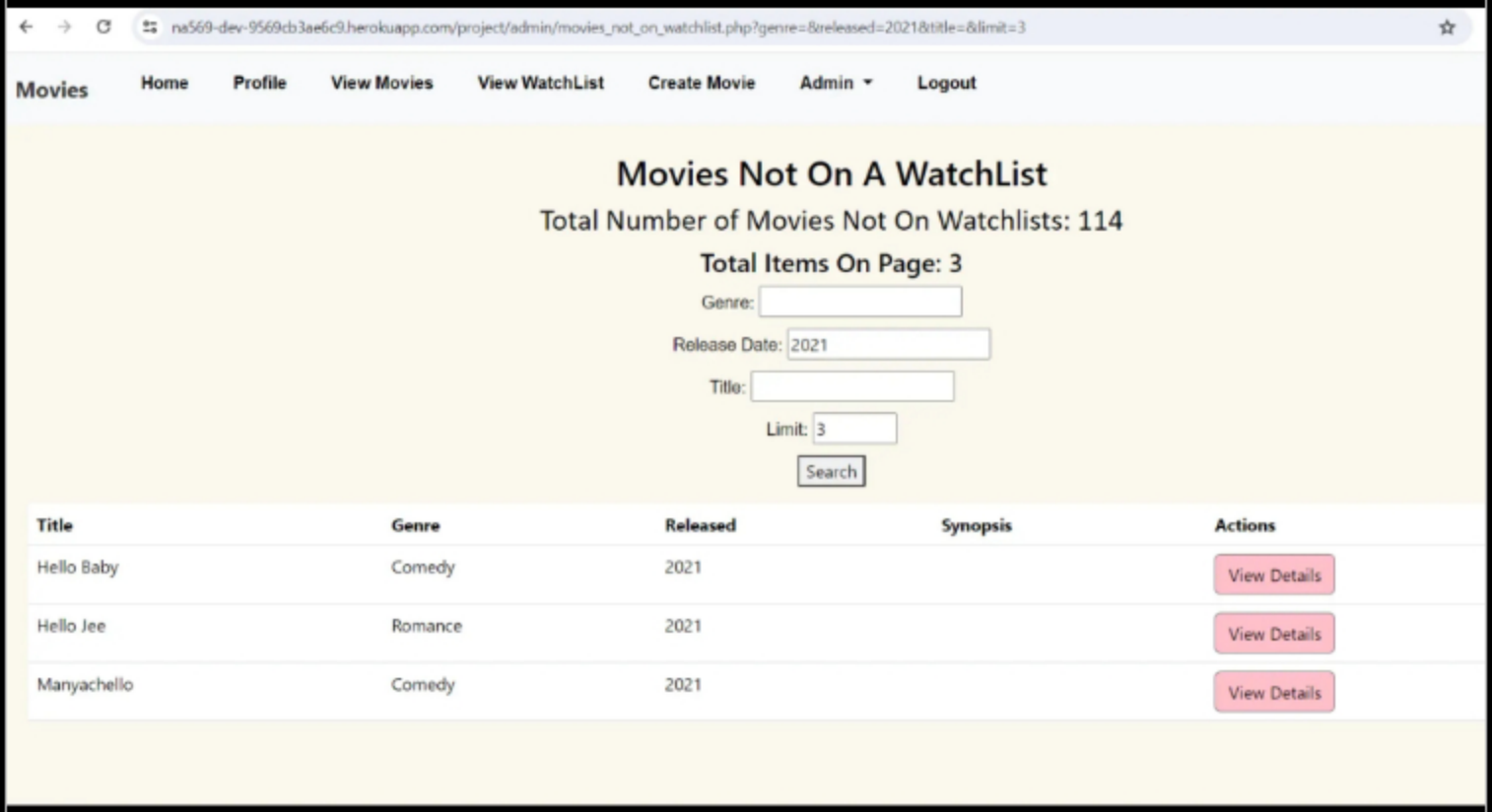
Title	Genre	Released	Synopsis	Actions
Portrait of a Bookstore as an Old Man	Documentary	2003	In 1951, George Whitman opened a bookshop-commune in Paris. George, 92, still runs his "den of anarchists disguised as a bookstore," offering free, dirty beds to poor literati, cutting his ...	View Details
Testing	Testing	2004	Testing	View Details
Soldier Boy	Drama	2019		View Details
Harry Potter	Fantasy	1999	harry potter goes on an adventure	View

showcases view on movies not on a watchlist page and contains single view button

Checklist Items (2)

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

#2 Show the single view buttons/links

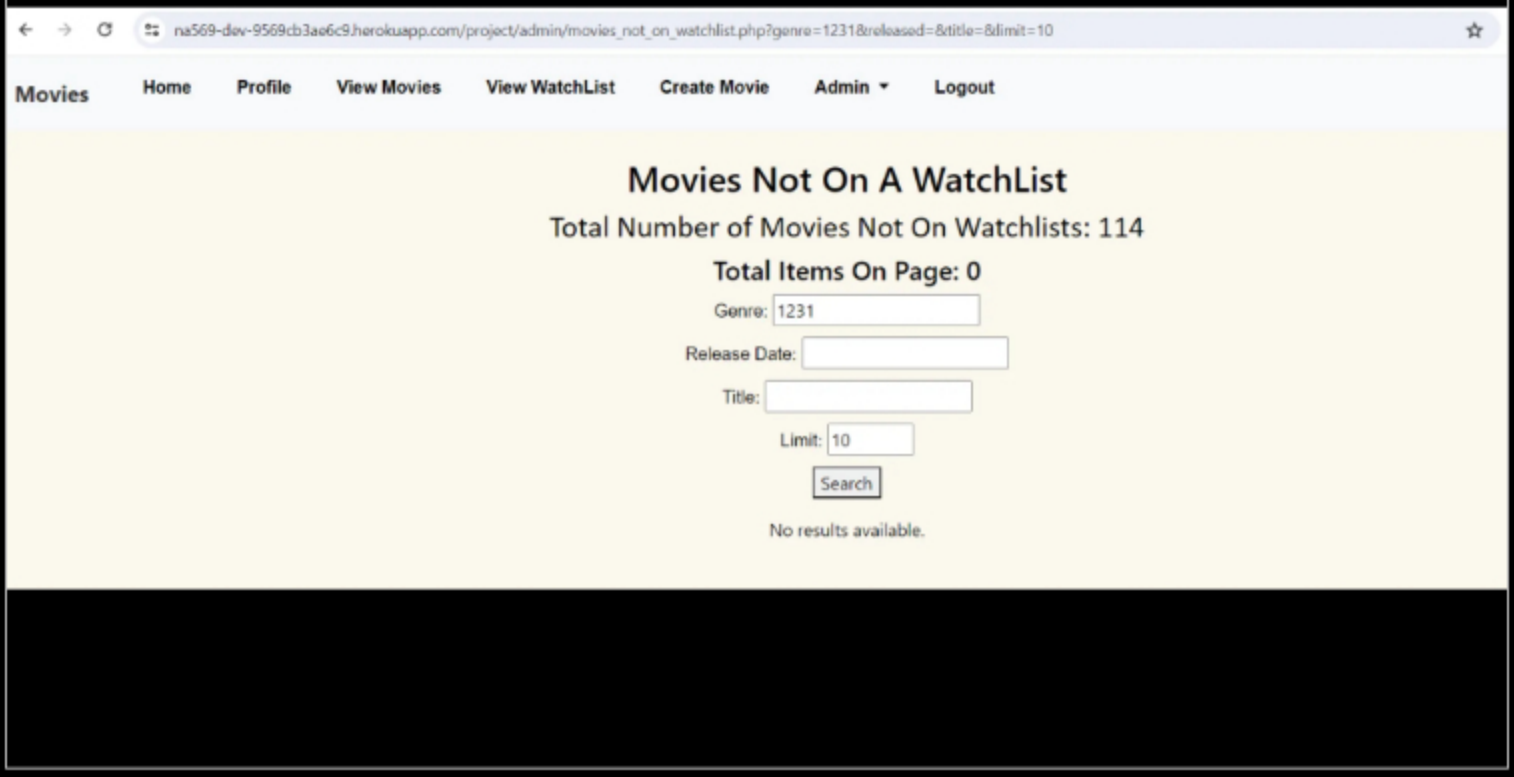


shows total number of movies not on watchlist and total items on page based on filtered result, also showcases filter based on release date

Checklist Items (2)

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

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



no results available when filtering

Checklist Items (1)

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

COLLAPSE

Task #2 - Points: 1

Text: Screenshot the code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the code related to fetching all unassociated entities (including the query)
#2	1	Show the code related to the display of the results
#3	1	Each record should have a button/link for single view
#4	1	Show the logic related to the count of all unassociated items (even the ones not shown in the filtered results)
#5	1	Show the logic related to the count of the items on the page (this value should change based on the filter applied)
#6	1	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)
#7	1	Include ucid/date comments for each code screenshot
#8	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```php
$genre = isset($_GET['genre']) ? $_GET['genre'] : '';
$title = isset($_GET['title']) ? $_GET['title'] : '';
$released = isset($_GET['released']) ? $_GET['released'] : '';
$limit = isset($_GET['limit']) ? intval($_GET['limit']) : 10; // Default to 10
$limit = max(1, min($limit, 100));

$query = "SELECT id, title AS Title, genre as Genre, released as Released, synopsis as Synopsis FROM `Movies`
            WHERE id NOT IN (SELECT movie_id FROM UserMovies)";
$params = [];
if (!empty($genre)) {
    $query .= " AND genre LIKE :genre";
    $params[':genre'] = "%$genre%";
}
if (!empty($title)) {
    $query .= " AND title LIKE :title";
    $params[':title'] = "%$title%";
}
if (!empty($released)) {
    $query .= " AND released LIKE :released";
    $params[':released'] = "%$released%";
}
$query .= " ORDER BY created DESC LIMIT $limit";

$db = getDB();
$stmt = $db->prepare($query);
try {
    $stmt->execute($params);
    $results = $stmt->fetchAll();
} catch (PDOException $e) {
    error_log("Error fetching movies " . var_export($e, true));
    flash("Unhandled error occurred", "danger");
}
//na569, 4.30.24
```

showcases query for fetching all movies not in watchlist and logic for filter/sort

Checklist Items (2)

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

#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
<?php //na569, 4.24.24
$table = ["data" => $results, "title" => "Search Movies", "ignored_columns" => ["id"], "edit_url" => get_url("../edit_movie.php")];
?>
<?php if (empty($results)) : ?>
    <div>No results available.</div>
<?php else : ?>
    <div class="container-fluid">
        <table class="table">
            <thead>
                <tr>
                    <th>Title</th>
                    <th>Genre</th>
                    <th>Released</th>
                    <th>Synopsis</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach ($results as $row) : ?>
                    <tr>
                        <td><?php echo htmlspecialchars($row['Title']); ?></td>
                        <td><?php echo htmlspecialchars($row['Genre']); ?></td>
                        <td><?php echo htmlspecialchars($row['Released']); ?></td>
                        <td><?php echo htmlspecialchars($row['Synopsis']); ?></td>
                        <td>
                            <a href="../movie_details.php?id=<?php echo $row['id']; ?>" class="btn btn-secondary">View Details</a>
                        </td>
```

```
                    </tr>
                <?php endforeach; ?>
            </tbody>
        </table>
    </div>
<?php endif; ?>
```

code that showcases table displaying results, showcases link to view details

Checklist Items (1)

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

```
//$query = "SELECT COUNT(*) AS total_count FROM UserMovies WHERE user_id = :user_id";
$query = "SELECT COUNT(*) AS total_count FROM Movies WHERE id NOT IN (SELECT DISTINCT movie_id FROM UserMovies )";
//$params = [":user_id" => get_user_id()];

// Execute the query to fetch the count
$db = getDB();
try {
    $stmt = $db->prepare($query);
    $stmt->execute();
    $row = $stmt->fetch(PDO::FETCH_ASSOC);
    $total_count = $row['total_count'];
} catch (PDOException $e) {
    error_log("Error fetching total count: " . $e->getMessage());
    flash("An error occurred", "danger");
    $total_count = 0;
} <- #82-86 catch (PDOException $e)
?>
<!-- Search Form -->
<div style="text-align: center;">
<div class="container-fluid">
    <h2>Movies Not On A WatchList</h2>
        <h3> Total Number of Movies On Watchlists: <?php echo $total_count; ?></h3>
        <h4>Total Items On Page: <?php echo count($results); ?></h4>
```

shows logic for all unassociated movies only on the page and how many in total

Checklist Items (0)

COLLAPSE

Task #3 - Points: 1

Text: Explain the solution

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Mention how you determine the result list (include the unassociated logic and filters)
#2	1	Mention the logic behind the two counts (total items and visible result count)

Response:

The SQL query selects movie details from the Movies table where the movie ID is not in the UserMovies table. There

are also filters on the title, genre, and released date. The logic for the two counts is similar to the previous pages. The total items count is found by counting the number of rows in the Movies table where the movie ID is not in the UserMovies table. The visible result count displays the number of rows after applying filters/sort.

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 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://na569-prod-74aadc581478.herokuapp.com/project/admin/movies_not_on_watchlist.php

URL #2

https://github.com/nafisa37/na569-it202-008/pull/66

COLLAPSE

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

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the page

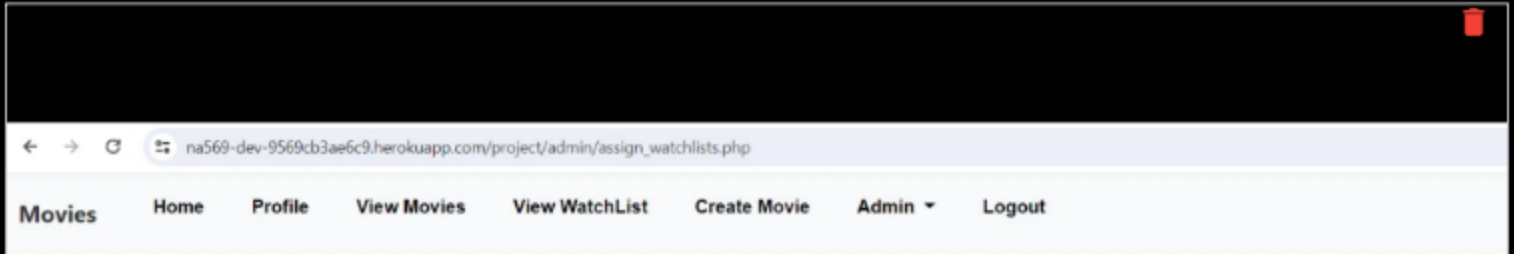
Checklist *The checkboxes are for your own tracking

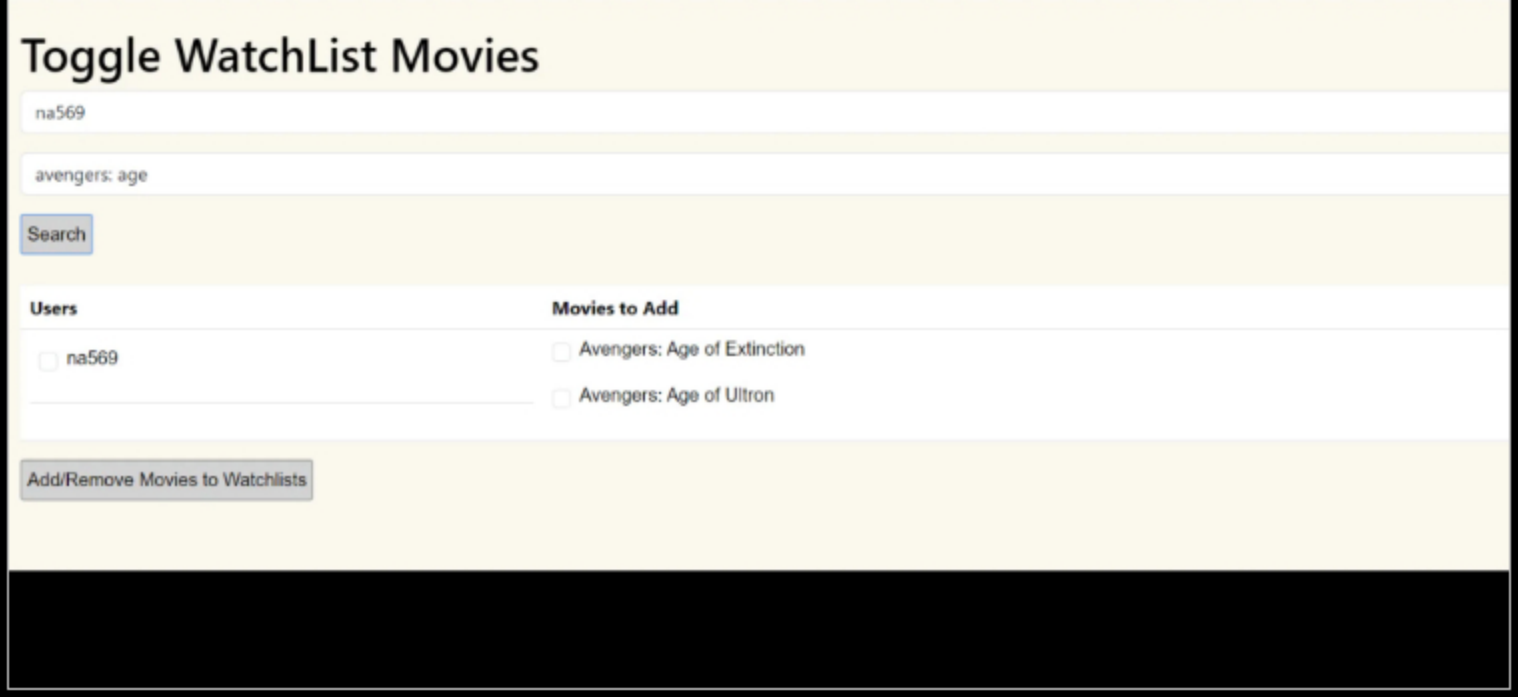
#	Points	Details
#1	1	Show the search form with valid data
#2	1	Show the results of the search
#3	1	Show the result of entities and users being associated and unassociated
#4	1	Make sure the heroku dev url is visible in the address bar
#5	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



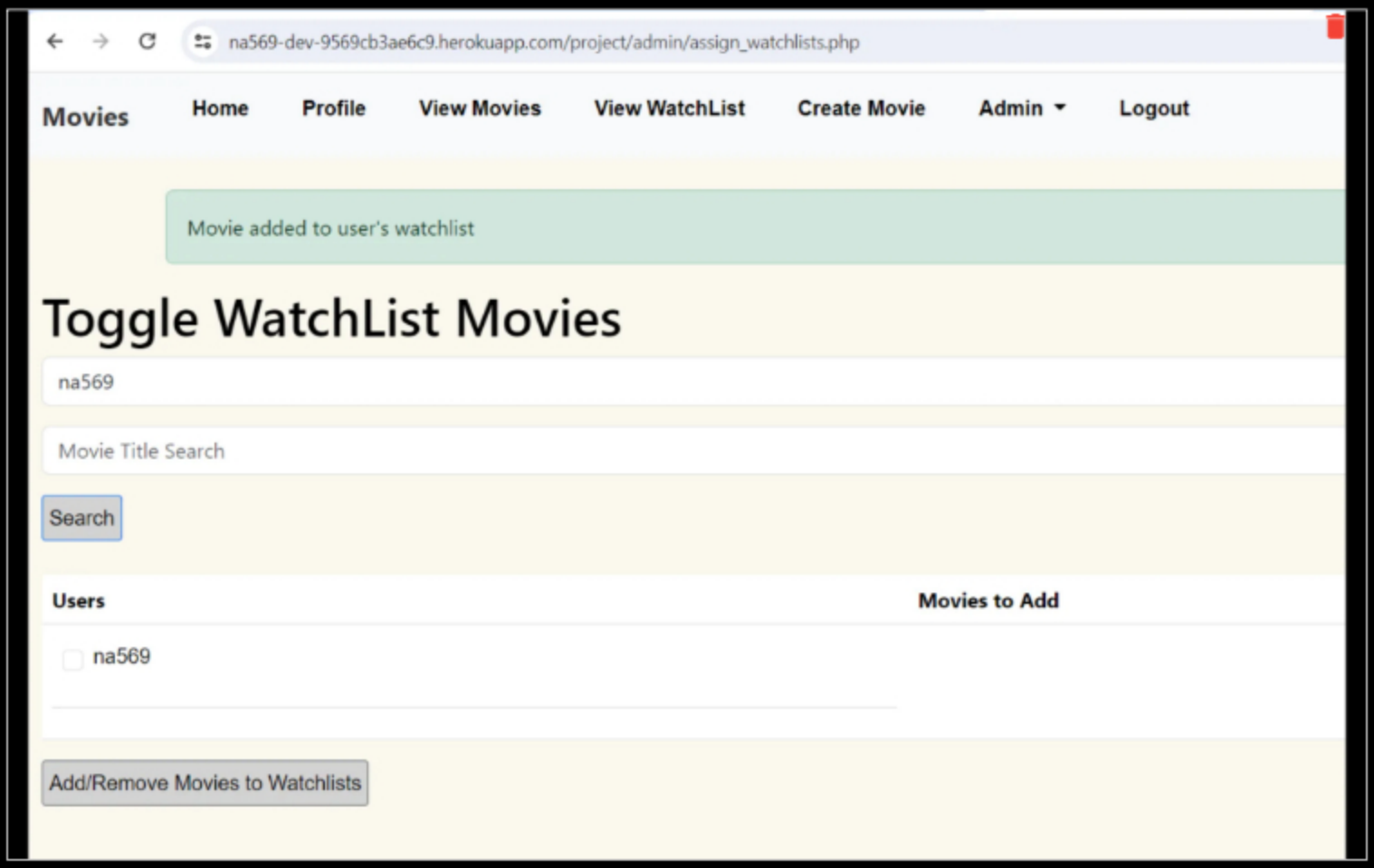


result of searching for username na569 and movies with the words "Avengres: age" in them

Checklist Items (2)

#1 Show the search form with valid data

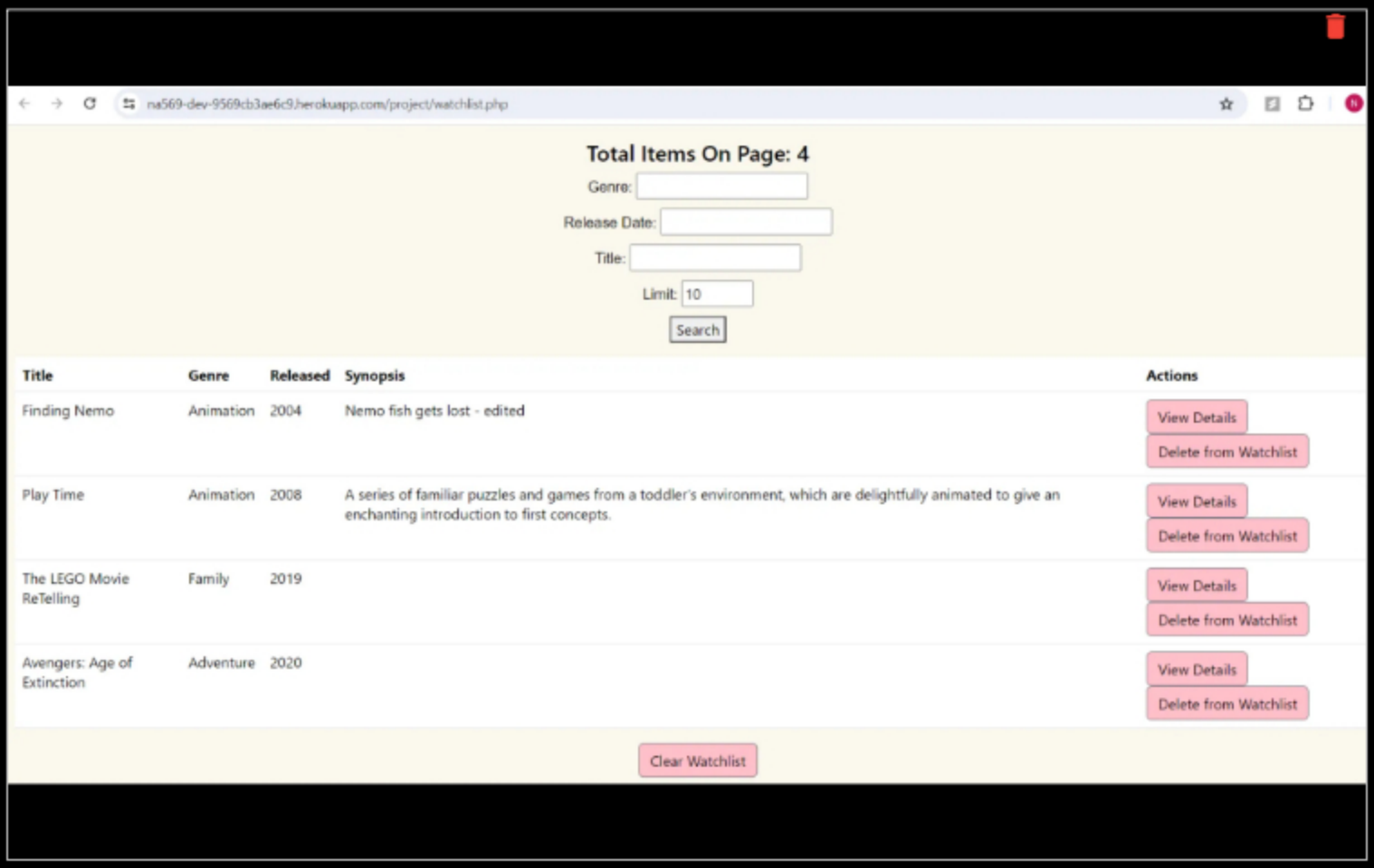
#2 Show the results of the search



added avengers: age of extinction to watchlist

Checklist Items (1)

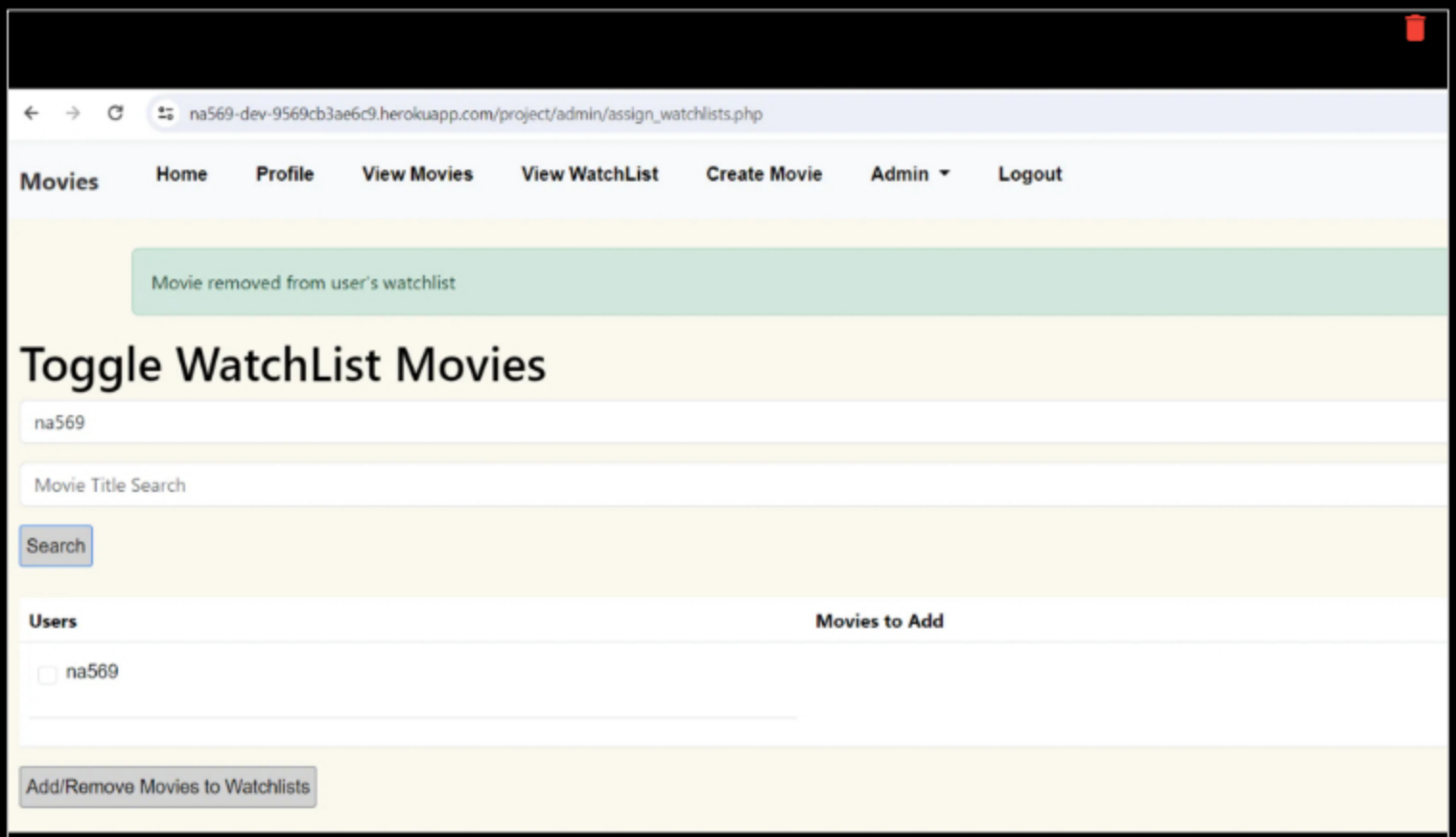
#3 Show the result of entities and users being associated and unassociated



proof of new avengers movie being added to user's watchlist

Checklist Items (1)

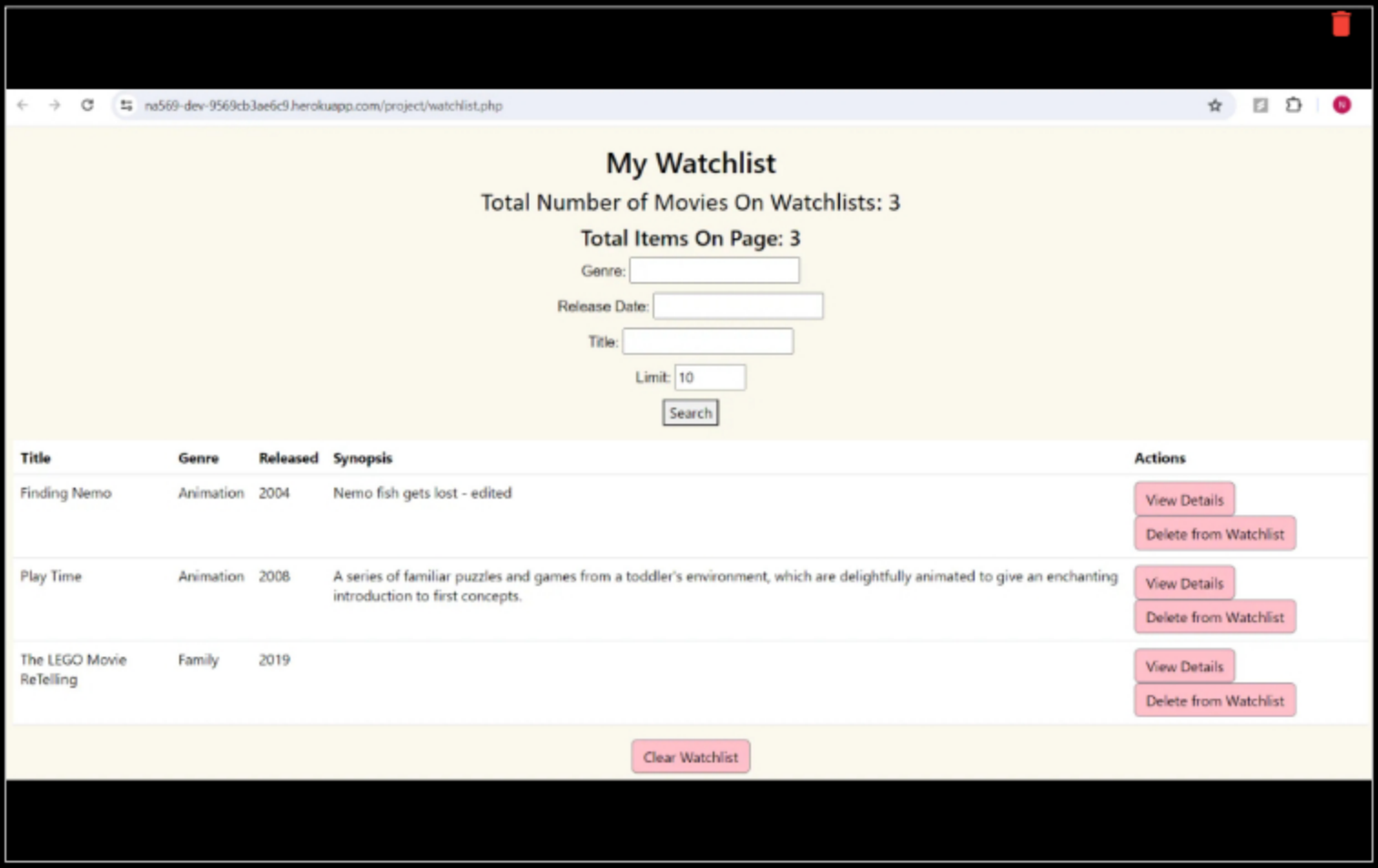
#3 Show the result of entities and users being associated and unassociated



same movie being removed from watchlist

Checklist Items (1)

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



proof of movie being removed off watchlist

Checklist Items (2)

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

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

^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshots of the code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Search form field for finding a partial match of usernames
#2	1	Search form field for finding a partial match of entities
#3	1	Code related to getting a max of 25 results for each field (i.e., 25 users and 25 entities limit)

#4	1	Code that generates the checkboxes next to each list (users and entities)
#5	1	Code related to submitting the checkbox lists
#6	1	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)
#7	1	Include ucid/date comments for each code screenshot
#8	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```php
// Search for user by username, na569, 4.30.24
$users = [];
$username = "";
if (isset($_POST["username"])) {
    $username = se($_POST, "username", "", false);
    if (!empty($username)) {
        $db = getDB();
        $stmt = $db->prepare("SELECT id, username FROM Users WHERE username LIKE :username LIMIT 25");
        try {
            $stmt->execute([":username" => "%$username%"]);
            $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
            if ($results) {
                $users = $results;
            }
        } catch (PDOException $e) {
            flash(var_export($e->errorInfo, true), "danger");
        }
    } else {
        flash("Username must not be empty", "warning");
    }
} <- #59-76 if (isset($_POST["username"]))

//Search for movie by title
$movies = [];
$title = "";
if (isset($_POST["title"])) {
    $title = se($_POST, "title", "", false);
    if (!empty($title)) {
        $db = getDB();
        $stmt = $db->prepare("SELECT id, title FROM Movies WHERE title LIKE :title LIMIT 25");
        try {
            $stmt->execute([":title" => "%$title%"]);
            $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
            if ($results) {
                $movies = $results;
            }
        } catch (PDOException $e) {
```

code for finding max of 25 results of users and titles

Checklist Items (1)

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

```
class="container-fluid">
h1>Toggle WatchList Movies</h1>
```

```
form method="POST"> //na569, 4.30.24
    <?php render_input(["type" => "search", "name" => "username", "placeholder" => "Username Search", "value" => $username]); ?>
    <?php render_input(["type" => "search", "name" => "title", "placeholder" => "Movie Title Search", "value" => $title]); ?>
    <?php render_button(["text" => "Search", "type" => "submit"]); ?>
/form>
```

form field for partial match of username and title

Checklist Items (2)

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

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

```
                                <td>
                                    <?php render_input(["type" => "checkbox", "id" => "user_" . se($user, 'id', "", false), "name" =>
                                </td>
                            </tr>
                        <?php endforeach; ?>
                    </table>
                </td>
                <td>
                    <?php foreach ($movies as $movie) : ?>
                        <div>
                            <?php render_input(["type" => "checkbox", "id" => "movie_" . se($movie, 'id', "", false), "name" => "movie
                        </div>
                    <?php endforeach; ?>
                </td>
            </tr>
        </tbody>
    </table>
    <?php render_button(["text" => "Add/Remove Movies to Watchlists", "type" => "submit", "color" => "secondary"]); ?>
 orm>
```

code that showcases checkboxes next to users and movie titles, and button to submit the lists

Checklist Items (2)

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

#5 Code related to submitting the checkbox lists

```php
// Attempt to apply associations, na569, 4.30.24
if (isset($_POST["users"]) && isset($_POST["movies"])) {
    $user_ids = $_POST["users"];
    $movie_ids = $_POST["movies"];
    if (empty($user_ids) || empty($movie_ids)) {
        flash("Both users and movies need to be selected", "warning");
    } else {
        $db = getDB();
        foreach ($user_ids as $uid) {
            foreach ($movie_ids as $mid) {
                // Check if movie in watchlist
                $stmt = $db->prepare("SELECT COUNT(*) FROM UserMovies WHERE user_id = :uid AND movie_id = :mid");
                $stmt->execute([":uid" => $uid, ":mid" => $mid]);
                $count = $stmt->fetchColumn();

                if ($count > 0) {
                    // If movie already in watchlist, delete it
                    $stmt = $db->prepare("DELETE FROM UserMovies WHERE user_id = :uid AND movie_id = :mid");
                    $stmt->execute([":uid" => $uid, ":mid" => $mid]);
                    flash("Movie removed from user's watchlist", "success");
                } else {
                    // If movie not in watchlist, insert it
                    $stmt = $db->prepare("INSERT INTO UserMovies (user_id, movie_id) VALUES (:uid, :mid)");
                    $stmt->execute([":uid" => $uid, ":mid" => $mid]);
                    flash("Movie added to user's watchlist", "success");
                } <- #30-35 else
            } <- #19-36 foreach ($movie_ids as $mid)
        } <- #18-37 foreach ($user_ids as $uid)
    } <- #16-38 else
} <- #11-39 if (isset($_POST["users"]) && isset($_POST["movies"]))
```

code related to applying association depending on if data is associated or not already

Checklist Items (1)

#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)

^COLLAPSE^

Task #3 - Points: 1

Text: Explain the solution

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Describe the steps for the search and how it works for users and entities
#2	1	Mention how you built the UI for this
#3	1	Describe the steps for the associate/unassociate logic for the combination of users and entities

Response:

I heavily relied on the UserRoles (assign_roles) page that we created earlier to help me create this page. I borrowed most of the code from the page. The user and movie searches are created very similarly, the code get the user/title from the POST request. It then created an SQL query that searches for the user/title based on the input. The results are then stored in an array. For the associate/unassociate button, first I checked to see if the movie is alreayd in the user's wathlist by counting the number of occurences it has in the UserMovies table. If the count is more than 0, the association already exists and it deletes it using an SQL query. If the count is 0, the association does not exist and it creates it using another SQL query.

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 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://na569-prod-74aadc581478.herokuapp.com/project/admin/assign_watchlists.php

URL #2

https://github.com/nafisa37/na569-it202-008/pull/66

COLLAPSE

Misc (1 pt.)

COLLAPSE

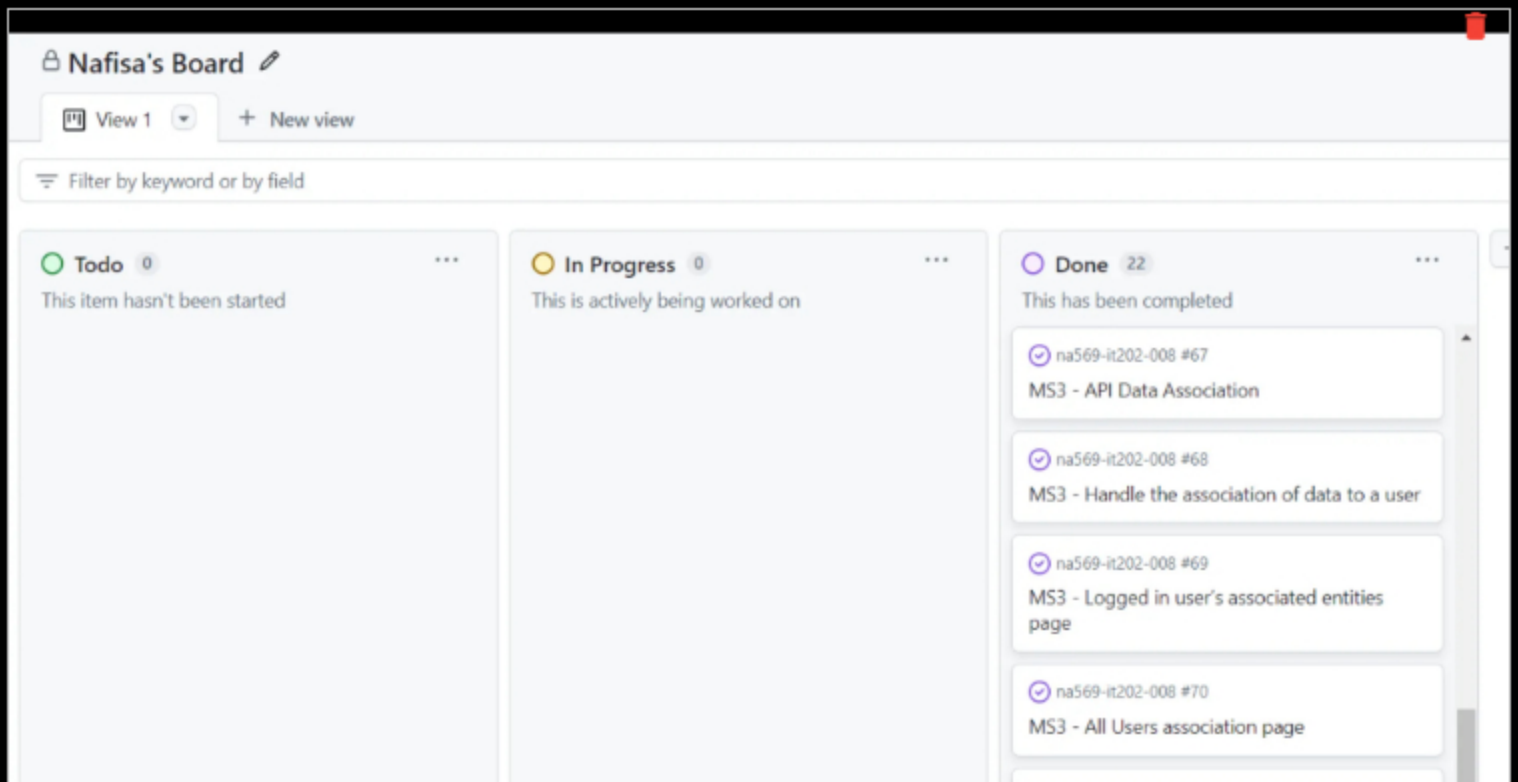
Task #1 - Points: 1

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

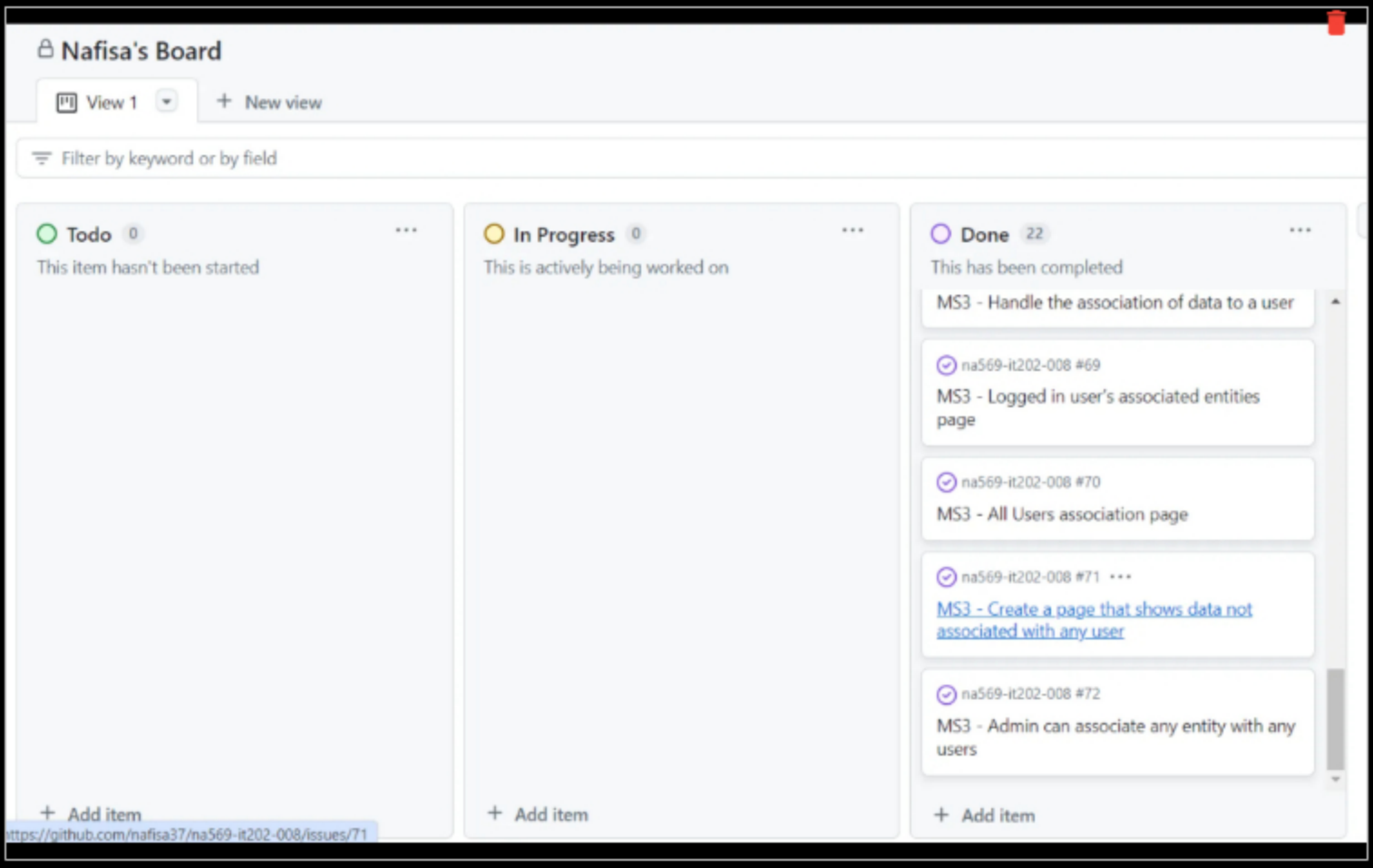
Task Screenshots:

Gallery Style: Large View

Small Medium Large



first screenshot showcasing project board



second screenshot showcasing project board

COLLAPSE

Task #2 - Points: 1

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

URL #1

https://github.com/users/nafisa37/projects/1/views/1

COLLAPSE

Task #3 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

This Milestone was probably the easiest for me of all the previous Milestones. I feel a lot more comfortable in my PHP skills and I focused on reusing as much code as possible from my previous pages created for this project. One of the hardest pages for me was the admin page listing all associations for all users. My SQL queries took me some time to figure out because they got pretty complicated, but I'm taking a database class right now that helped me figure most

of them out.

^COLLAPSE^

Task #4 - Points: 1

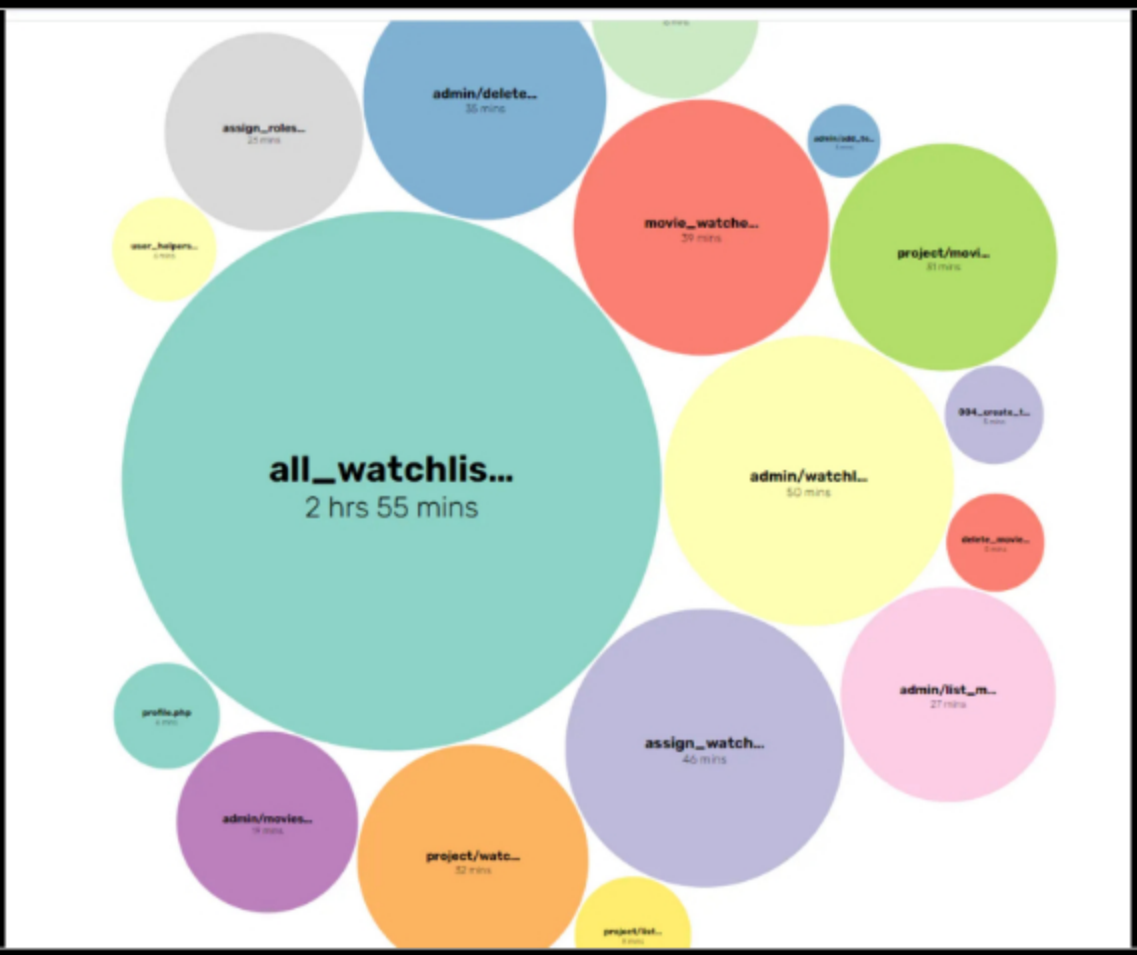
Text: WakaTime Screenshot

Details:

Task Screenshots:

Gallery Style: Large View

Small Medium Large



wakatime screenshot

End of Assignment