

# Submission Worksheet

**CLICK TO GRADE**

<https://learn.ethereallab.app/assignment/IT202-007-F2024/it202-api-project-milestone-3-2024-m24/grade/jd755>

Course: IT202-007-F2024

Assignment: [IT202] API Project Milestone 3 2024 m24

Student: Jeremy R. (jd755)

## Submissions:

Submission Selection

1 Submission [submitted] 12/7/2024 12:23:58 AM

## Instructions

[^ COLLAPSE ^](#)

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

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

Branch name: Milestone3

**Group**

Group: API

Tasks: 2

Points: 1

100%

▲ COLLAPSE ▲

### Task



Group: API  
Task #1: Data Related to Users  
Weight: ~50%  
Points: ~0.50

▲ COLLAPSE ▲

### Checklist

\*The checkboxes are for your own tracking

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

## ≡ Task Response Prompt

Response:

The idea is that all the users can pick a movie from the movie search and add it to their watch list. This is a many-many relationship as a user watch list many movies and a movie can be watch listed by many users

End of Task 1

### Task



Group: API  
Task #2: Updating Entities  
Weight: ~50%  
Points: ~0.50

▲ COLLAPSE ▲

### Checklist

\*The checkboxes are for your own tracking

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

## ≡ Task Response Prompt

Response:

When an update occurs, the associated data is updated as well. For example, If I pull in a movie from the API and this movie is already in the database, then the movie's contents are updated. All the associations stay the same as the IDs don't change.

End of Task 2

End of Group: API

## Group

## Group: Handle Data Association

## Tasks: 3

**Points: 1**

^ COLLAPSE ^

## Task

## Group: Handle Data Association

## Task #1: Screenshots of the code

Weight: ~33%

Points: ~0.33

COLLAPSE

## Details:

**Option 1:** Related pages will have a button to do association (like favorites or similar).

**Option 2:** a separate page will be used to associate entities to a user by some other user (like assignment of entities)

**Include ucid/date comments for each code screenshot**

### Sub-Task

### Group: Handle Data Association

### Task #1: Screenshots of the code

#### Sub Task #1: Show the related code

## Task Screenshots

## Gallery Style: 2 Columns

4 2

```
//jd755 12/6/24
<?php foreach ($results as $movie): ?>
|     <div class="col-3">
|         <?php movie_card($movie); ?>
|     </div>
<?php endforeach; ?>
<?php if (empty($results)): ?>
|     No records to show
<?php endif; ?>
```

//jd755 12/6/24

#### **Movie cards being created**

Within movie\_card.php, the function render\_like() is called.

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## ≡ Task Response Prompt

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

Response:

In the search movie menu, each movie card has a heart button underneath and the data of the movie contains a column called 'is\_watched'. When the user clicks this button, the data is sent to movie\_card.php where it checks if the option 'is\_watched' is set. If it is, the data is sent to toggle\_watched.php. Here, it calls the UserMovies database to add in a new association between the signed in user and the movie id. If this fails and if the failure is due to duplicate values, then the association is deleted from the database.

End of Task 1

### Task



Group: Handle Data Association

Task #2: Screenshot of the association table(s)

Weight: ~33%

Points: ~0.33

▲ COLLAPSE ▲

### Sub-Task



Group: Handle Data Association

Task #2: Screenshot of the association table(s)

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

## 🖼 Task Screenshots

Gallery Style: 2 Columns

4      2      1

The screenshot shows the MySQL Workbench interface with the 'UserMovies' table selected. The table structure includes columns: movie\_id, user\_id, is\_watched, created\_timestamp, and modified\_timestamp. The data shows four rows of associations between users and movies.

	movie_id	user_id	is_watched	created_timestamp	modified_timestamp
1	40	9	0	2024-12-05 23:24:00	2024-12-05 23:24:00
2	41	9	0	2024-12-05 23:25:00	2024-12-05 23:25:00
3	42	9	0	2024-12-05 23:25:00	2024-12-05 23:25:00
4	50	9	0	2024-12-06 00:00:00	2024-12-06 00:00:00

UserMovies table

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## ≡ Task Response Prompt

## Describe each column/association table

Response:

**Id:** The Id of the association. **user\_id:** The user\_id of the association. This is a foreign key matching the id column of the Users database. **movie\_id:** The movie\_id of the association. This is a foreign key matching the id column of the Movies database.

End of Task 2

### Task



Group: Handle Data Association

Task #3: Add related links

Weight: ~33%

Points: ~0.33

[▲ COLLAPSE ▲](#)

### Checklist

\*The checkboxes are for your own tracking

#	Details
<input checked="" type="checkbox"/> #1	Include the heroku prod link for the page that creates the association
<input checked="" type="checkbox"/> #2	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

## Task URLs

URL #1

<https://it202-jd755->



[https://it202-jd755-prod-028c8689138c.herokuapp.com/project/search\\_movie.php?title=&filter=](https://it202-jd755-prod-028c8689138c.herokuapp.com/project/search_movie.php?title=&filter=)

[title=&filter=](#)

URL #2

<https://github.com/legitdaylight/jd755-IT202-007/pull/59>



<https://github.com/legitdaylight/jd755-IT202-007>

End of Task 3

End of Group: Handle Data Association

Task Status: 3/3

### Group



Group: Current User's Association Page

Tasks: 3

Points: 2

[▲ COLLAPSE ▲](#)

### Task



Group: Current User's Association Page

Task #1: Screenshots of this page

100%

Weight: ~33%  
 Points: ~0.67

▲ COLLAPSE ▲

**1 Details:**

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

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



Columns: 1

Sub-Task

100%

Group: Current User's Association Page

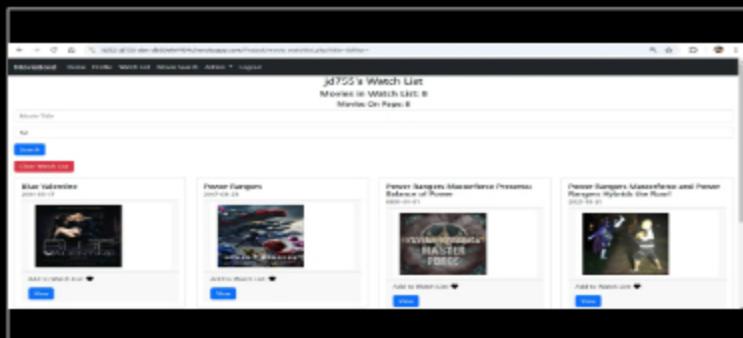
Task #1: Screenshots of this page

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

**☒ Task Screenshots**

Gallery Style: 2 Columns

4 2 1



The watch list page. Each result has its title, release date, and image visible.

**Caption(s) (required) ✓**

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: Current User's Association Page

Task #1: Screenshots of this page

Sub Task #2: Show the single view buttons/links, delete button/links, and delete all button/link

**☒ Task Screenshots**

Gallery Style: 2 Columns

4 2 1



The delete button is the heart icon, the view is underneath, and delete all is underneath the search button.

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

#### Sub-Task

Group: Current User's Association Page

Task #1: Screenshots of this page

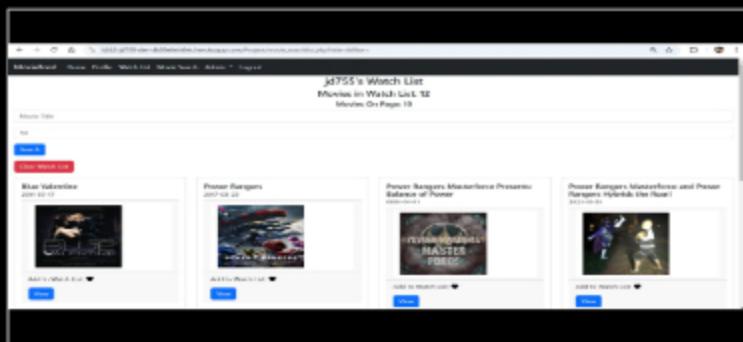
Sub Task #3: Show variations of the number of shown items count and show the count of total number of associated items to the user



## Task Screenshots

Gallery Style: 2 Columns

4      2      1



Total movies in watch list vs current ones on page

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

#### Sub-Task

Group: Current User's Association Page

Task #1: Screenshots of this page

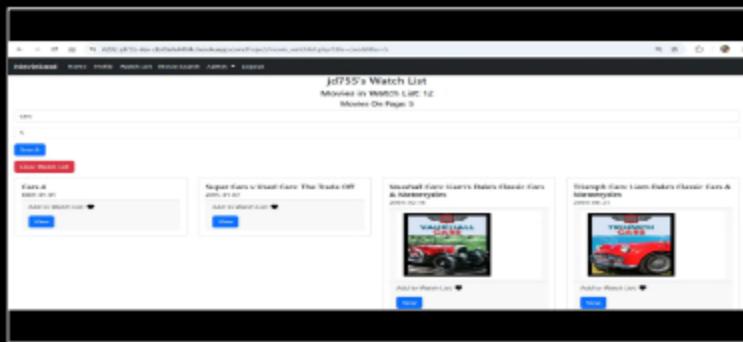
Sub Task #4: Show variations of the filter/sort including no results (proper message should be visible)



## Task Screenshots

Gallery Style: 2 Columns

4      2      1



title: cars number: 5



movie title that doesn't exist

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

## End of Task 1

### Task

100%

Group: Current User's Association Page

Task #2: Screenshot the code

Weight: ~33%

Points: ~0.67

▲ COLLAPSE ▾

#### Details:

Include ucid/date comments for each code screenshot



Columns: 1

#### Sub-Task

100%

Group: Current User's Association Page

Task #2: Screenshot the code

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

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
//jd795_32/6/24
$db = getDB();
$query = "SELECT * FROM userMovies WHERE user_id = :user_id AND movie_id = :movie_id";
$stmt = $db->prepare($query);
$stmt->execute(['user_id' => $user_id, 'movie_id' => $movie_id]);
$movies = $stmt->fetchAll(PDO::FETCH_ASSOC);
if(count($movies) < 1)
{
    $msg = "No movies found for user ID: " . $user_id;
    $error = true;
}
else
{
    $msg = "User ID: " . $user_id . " has " . count($movies) . " movies associated with them.";
```

part 1

```
//jd795_32/6/24
else
{
    if(strlen($title) > 200)
    {
        Flash(["msg" . $title . " too long (cannot exceed 200 characters)", "warning"]);
        $query = "SELECT Movies.id, Movies.title, Movies.image_url, Movies.release_date $assoc_check
        FROM Movies
        INNER JOIN UserMovies
        ON UserMovies.movie_id = Movies.id
        WHERE UserMovies.user_id = :user_id
        ORDER BY Movies.created
        DESC LIMIT $num";
    }
    else
    {
        $search = $_GET["title"];
        $query = "SELECT Movies.id, Movies.title, Movies.image_url, Movies.release_date $assoc_check
        FROM Movies
        INNER JOIN UserMovies
        ON UserMovies.movie_id = Movies.id
        WHERE UserMovies.user_id = :user_id AND title LIKE :title
        ORDER BY Movies.created
        DESC LIMIT $num";
    }
}
$params = [
    "title" => "%$search%",
    "user_id" => $user_id,
    "title" => "%$search%"
```

part 2

```
//jd795_32/6/24
$db = getDB();
$stmt = $db->prepare($query);
$stmt->execute(['user_id' => $user_id, 'movie_id' => $movie_id]);
$movies = $stmt->fetchAll(PDO::FETCH_ASSOC);
if(count($movies) < 1)
{
    $msg = "No movies found for user ID: " . $user_id;
    $error = true;
}
else
{
    $msg = "User ID: " . $user_id . " has " . count($movies) . " movies associated with them.";
```

part 3

Caption(s) (required) ✓

Caption Hint: Describe/highlight what's being shown

= Task Response Prompt

## **=, Task Response Prompt**

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

## Response:

- I start by getting the user id by calling the `get_user_id()` function which gets the id from the session data.
  - I create variable `$assoc_check` which checks if the association between the user and the movie already exists. This is placed into `is_watched`.
  - I make sure `$num` is within bounds
  - If the movie title search is blank, I do the query that grabs the movie id, movie title, movie image url, movie release date, and `is_watched`. This info is obtained by doing an inner join with the `UserMovies` table where the movie ids match and where the user id matches the current user id. I used inner join as I wanted only the matching information.
  - If the title is not blank, I check if it is larger than 200 characters. If it is, the same query as above is created.
  - If the title is valid and not blank, I edit the above query to include partial matching of the inputted title
  - Finally after the query is created, the query is called and saved into variable `$results`

### Sub-Task

Group: Current User's Association Page

#### Task #2: Screenshot the code

**Sub Task #2:** Show the code related to the display of the results

## Task Screenshots

## Gallery Style: 2 Columns

4 2 1

```
//jd755 12/6/24
<?php foreach ($results as $movie): ?>
|     <div class="col-3">
|         <?php movie_card($movie); ?>
|     </div>
<?php endforeach; ?>
```

#### The creation of the cards

## movie card contents

**Caption(s) (required) ✓**

**Caption Hint:** *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

#### **Response:**

- A for each loop is used to loop through every element of the results array.
  - Each element is passed into the movie\_card function.
  - Inside movie\_card, the data contains within the passed in movie is pulled.
  - movie\_card.php itself has html elements that constructs a small card with the information. At the top is the movie title, then the release date. Then I check if the data has an imageURL. If yes, then the image is displayed using an `<img>` tag. If not, this part is skipped. Next, if the user is logged in and if the data has an `is_watched`, the heart from before is rendered. Finally, at the bottom the view button is created only if user

is\_watched, the heart icon before is rendered. Finally, at the bottom the view button is created only if users are logged in.

**Sub-Task**

Group: Current User's Association Page

Task #2: Screenshot the code

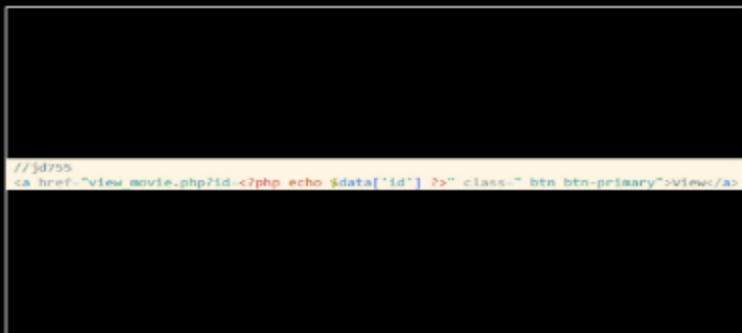
Sub Task #3: Each record should have a button/link for single view

100%

## Task Screenshots

Gallery Style: 2 Columns

4 2 1



The link to the single view

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

Right at the bottom of the movie card is button that takes the user to a single view. view\_movie.php uses a movie id, so I append the current movie's id to the link

**Sub-Task**

Group: Current User's Association Page

Task #2: Screenshot the code

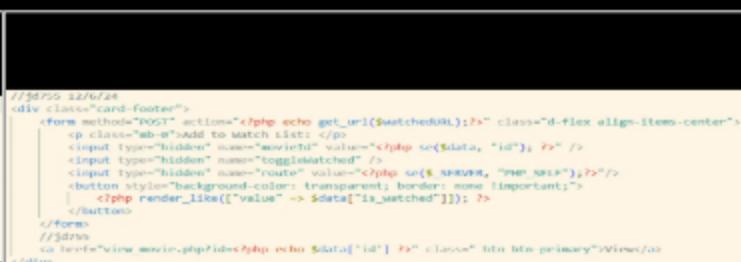
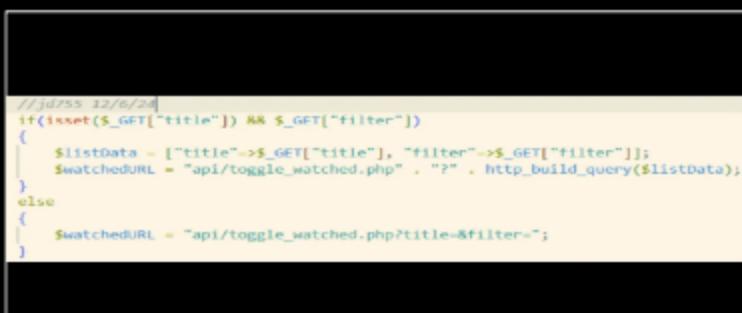
Sub Task #4: Each record should have a button/link for delete (this may be an admin-only thing but should be present for the specific role) Note: this is to delete the relationship and not the specific entity

100%

## Task Screenshots

Gallery Style: 2 Columns

4 2 1



**Caption(s) (required)** ✓Caption Hint: *Describe/highlight what's being shown***Task Response Prompt***Explain in concise steps how this logically works*

Response:

When movie\_card is used, it creates a url to the toggle\_watched page. Later, if the user is logged in, the heart button is rendered. When clicked, callstoggle\_watched where, if the user already has an association, the association is deleted.

**Sub-Task**

Group: Current User's Association Page

100%

Task #2: Screenshot the code

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

**Task Screenshots**

Gallery Style: 2 Columns

The screenshot displays two columns of code and their resulting HTML output. The left column shows PHP code for a database query to delete associations from the UserMovies table where user\_id matches the current user\_id. The right column shows the generated HTML code for a form with a POST method, containing a hidden input for user\_id and a submit button labeled "Clear Match List".

```

//jd755_32/6/20
if(isset($_POST['user_id'])) {
    try {
        $stmt = $db->prepare("DELETE FROM usermovies WHERE user_id = :user_id");
        $stmt->execute(['param']);
        $flash("Cleared Match List", "success");
        die(header("Location: " . get_url("movie_watchlist.php?title=$title&filter=$filter")));
    } catch (PDOException $e) {
        $flash("Error removing item from watch list", "danger");
        error_log("Error removing watch: " . var_export($e, true));
    }
} else {
    jd755_32/6/20
}

```

query that deletes all associations      HTML that handles deletion

**Caption(s) (required)** ✓Caption Hint: *Describe/highlight what's being shown***Task Response Prompt***Explain in concise steps how this logically works*

Response:

Underneath the search boxes is a button that allows the user to clear their watch list. When this button is pressed, it sends to user\_id to post. When this happens, the block of code that checks if user\_id is set in post calls a query that deletes everything from the UserMovie table where the user\_id matches the current user id.

**Sub-Task**

Group: Current User's Association Page

100%

Task #2: Screenshot the code

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

**Task Screenshots**

Gallery Style: 2 Columns

```
//jd755 12/6/24
ch4 class="text-center" movies in Watch List
<?php
$userID = $params['user_id'];
try {
$stmt = $db->prepare("SELECT COUNT(user_id) FROM usermovies WHERE user_id = ?"); //>
$stmt->execute();
$sr = $stmt->fetchall(PDO::FETCH_ASSOC);
if ($sr) {
$movieCount = $sr[0]['COUNT(user_id)'];
echo $movieCount;
}
} catch (PDOException $e) {
flash("Error retrieving number of movies in watch list", "danger");
error_log("Error removing watch: " . var_export($e, true));
}
?>
```

counting all associations

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*

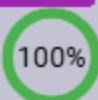
## ≡, Task Response Prompt

*Explain in concise steps how this logically works*

**Response:**

I used a query to count all instances of user\_id inside UserMovie that matches the current user id. This query returns an array. This array contains a subarray that contains the total number of instances. I take this number and echo it.

**Sub-Task**



Group: Current User's Association Page

Task #2: Screenshot the code

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

## ❑ Task Screenshots

Gallery Style: 2 Columns

```
//jd755 12/6/24
$db = gdDB();
$stmt = $db->prepare($query);
$results = [];
try {
$stmt->execute($params);
$sr = $stmt->fetchall(PDO::FETCH_ASSOC);
if ($sr) {
$results = $sr;
}
$numMovies = 0;
if(count($results) != 0)
{
$numMovies = count($results);
}
} catch (PDOException $e) <- #G4-TG try
{
error_log("Error fetching movies " . var_export($e, true));
flash("Unhandled error occurred", "danger");
}
```

numMovies being calculated

```
//jd755 12/6/24
<h5 class="text-center">
    Movies On Page:
    <?php echo $numMovies; ?>
</h5>
```

movies on page being displayed

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*

## ≡, Task Response Prompt

*Explain in concise steps how this logically works*

**Response:**

When the query to get all the movies based on the filters is used, it returns an array. I count the number of elements in this array to determine the number of items on the page. I then later use it in the HTML to display it on the page.

**Sub-Task**

100%

Group: Current User's Association Page

Task #2: Screenshot the code

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

**Task Screenshots**

Gallery Style: 2 Columns

4 2 1

```

function validate()
{
    if(filter < 1 || filter > 100)
    {
        flash("filter has to be between 1 and 100", "warning");
        filter = 10;
    }

    if(title.length > 200)
    {
        flash("[JavaScript] Title too long (cannot exceed 200 characters)", "warning");
        title = '';
    }

    if(filter < 0 || filter > 100)
    {
        flash("[Javascript] filter has to be between 1 and 100", "warning");
        filter = 10;
    }
}

```

filter constraints

```

function validate()
{
    let title = form.title.value;
    let filter = form.filter.value;
    let isValid = true;

    if(title.length > 200)
    {
        flash("[JavaScript] Title too long (cannot exceed 200 characters)", "warning");
        isValid = false;
    }

    if(filter < 0 || filter > 100)
    {
        flash("[Javascript] filter has to be between 1 and 100", "warning");
        isValid = false;
    }

    return isValid;
}

```

java script validation

**Caption(s) (required) ✓**Caption Hint: *Describe/highlight what's being shown***Task Response Prompt***Explain in concise steps how this logically works*

Response:

If the \$num variable is blank, it is defaulted to 10. If \$num is less than 1 or greater than 100, it is also defaulted to 10. I also if title is not greater than 200 characters. I also included the java script validation for this as well.

End of Task 2

**Task**

100%

Group: Current User's Association Page

Task #3: Add related links

Weight: ~33%

Points: ~0.67

▲ COLLAPSE ▲

**Checklist**

\*The checkboxes are for your own tracking

#	Details
<input checked="" type="checkbox"/> #1	Include the heroku prod link for the page that creates the association
<input checked="" type="checkbox"/> #2	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

**Task URLs**

URL #1

UR

<https://it202-jd755->

[https://it202-jd755-prod-028c8689138c.herokuapp.com/project/movie\\_watchlist.php?title=&filter=](https://it202-jd755-prod-028c8689138c.herokuapp.com/project/movie_watchlist.php?title=&filter=)

URL #2

<https://github.com/legitdaylight/jd755-IT202-007/pull/65>

URL

<https://github.com/legitdaylight/jd755-IT202-007>

## End of Task 3

### End of Group: Current User's Association Page

Task Status: 3/3

#### Group



Group: All Users Association Page (likely an admin page)

Tasks: 3

Points: 2

▲ COLLAPSE ▾

#### Task



Group: All Users Association Page (likely an admin page)

Task #1: Screenshots of this page

Weight: ~33%

Points: ~0.67

▲ COLLAPSE ▾

#### Details:

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

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



Columns: 1

#### Sub-Task



Group: All Users Association Page (likely an admin page)

Task #1: Screenshots of this page

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

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

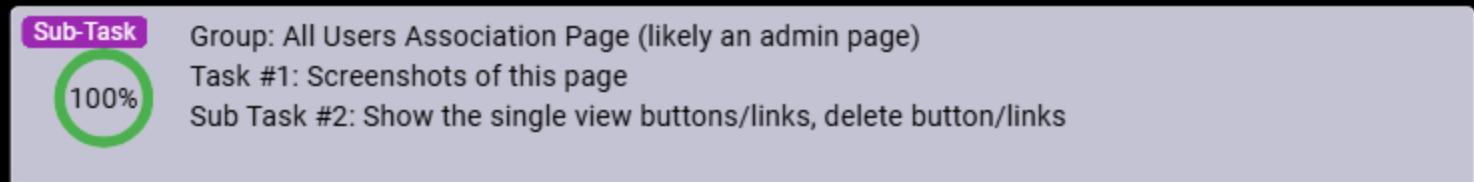


12	1.27	Power Ratings: Methodology Review: Behavior of Power	1	2011	1	<a href="#">View</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
13	1.40	Methodology Review: Model Metrics and Model Metrics Impact on Model	1	2011	1	<a href="#">View</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
14	1.22	Power Ratings: Model Metrics and Model Metrics Impact on Model	1	2011	1	<a href="#">View</a>	<a href="#">Edit</a>	<a href="#">Delete</a>
15	1.18	Model Metrics II	1	2011	1	<a href="#">View</a>	<a href="#">Edit</a>	<a href="#">Delete</a>

### **all user associations**

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*



## Task Screenshots

## Gallery Style: 2 Columns

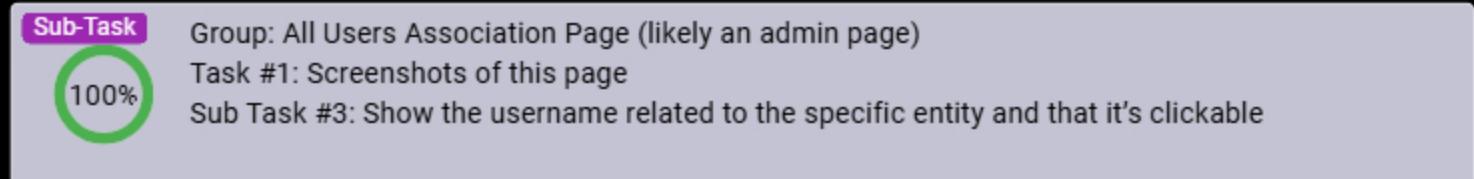
4 2 1

User ID	First Name	Last Name	Age	Gender	Occupation	Action
1	John	Doe	30	M	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
2	Jane	Doe	28	F	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
3	Mike	Smith	40	M	Project Manager	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
4	Alice	Johnson	35	F	Marketing Specialist	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
5	Bob	Smith	45	M	Product Manager	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
6	Charlie	Johnson	32	M	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
7	David	Smith	42	M	Project Manager	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
8	Eve	Johnson	38	F	Marketing Specialist	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
9	Frank	Smith	50	M	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
10	Grace	Johnson	48	F	Project Manager	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
11	Henry	Smith	35	M	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
12	Ivy	Johnson	40	F	Marketing Specialist	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
13	Jack	Smith	55	M	Product Manager	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
14	Lily	Johnson	30	F	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
15	Mike	Smith	40	M	Project Manager	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
16	Nancy	Johnson	35	F	Marketing Specialist	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
17	Oliver	Smith	50	M	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
18	Peter	Johnson	45	M	Project Manager	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
19	Quinn	Smith	32	F	Marketing Specialist	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
20	Ryan	Johnson	42	M	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
21	Samantha	Smith	38	F	Project Manager	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
22	Tommy	Johnson	55	M	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
23	Ulysses	Smith	40	M	Project Manager	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
24	Vivian	Johnson	35	F	Marketing Specialist	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
25	Wendy	Smith	50	M	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
26	Xavier	Johnson	42	M	Project Manager	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
27	Yvonne	Smith	38	F	Marketing Specialist	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>
28	Zachary	Johnson	55	M	Software Engineer	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">View Details</a>

**buttons on the side**

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*



## Task Screenshots

## Gallery Style: 2 Columns

All User Associations									
Number of Associations is: 21									
Associations On Page: 10									
Select Field									
Username									
<b>Latest Movies</b>									
Rank	Movie	Year	Rating	Genre	Director	Actors	Plot	Reviews	Score
12	Elan Reborn	2020	7.5	Science Fiction	1	John Doe	Elan Reborn is a gripping science fiction thriller that explores the consequences of a breakthrough in artificial intelligence.	5	8.2
13	Wander Angels	2020	8.0	Science Fiction	1	John Doe	Wander Angels is a thought-provoking science fiction film that challenges our assumptions about reality and the nature of existence.	5	8.5
14	Peter Parker: Homecoming	2017	7.2	Action, Adventure	1	Tom Holland	Peter Parker: Homecoming is a fun-filled action-adventure film that follows the young Peter Parker as he tries to balance his双重生活.	5	7.8
15	Spider-Man: Far From Home	2019	7.8	Action, Adventure	1	Tom Holland	Spider-Man: Far From Home is a thrilling action-adventure film that continues the Spider-Man franchise with a bang.	5	8.0
16	Peter Parker: Homecoming	2017	7.2	Action, Adventure	1	Tom Holland	Peter Parker: Homecoming is a fun-filled action-adventure film that follows the young Peter Parker as he tries to balance his双重生活.	5	7.8
17	Peter Parker: Homecoming II	2019	7.8	Action, Adventure	1	Tom Holland	Peter Parker: Homecoming II is a thrilling action-adventure film that continues the Spider-Man franchise with a bang.	5	8.0
18	Peter Parker: Homecoming III	2021	8.0	Action, Adventure	1	Tom Holland	Peter Parker: Homecoming III is a gripping action-adventure film that sets the stage for the future of the Spider-Man franchise.	5	8.5

I was not able to get the username clickable, so I have a button on the side that links to the user profile

**Caption(s) (required)**

**Caption Hint:** *Describe/highlight what's being shown*

### Sub-Task

Group: All Users Association Page (likely an admin page)



## Task #1: Screenshots of this page

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

## Task Screenshots

## Gallery Style: 2 Columns

4 2 1

### **total associations and associations different**

**another difference**

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*

### Sub-Task

Group: All Users Association Page (likely an admin page)



### Task #1: Screenshots of this page

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

## Task Screenshots

## Gallery Style: 2 Columns

4 2 1

All User Associations						
Member of Associations: 21 Associations On Page: 9						
Latest Movies						
assoc_id	id	title	is_critic	username	assoc_type	Actions
101	100	Unrateable Items	0	unrateable	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
102	101	Horror Movie	0	horror	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
103	102	Indian Movies	0	indian	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
104	103	Korean Movies	0	korean	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
105	104	Korean Movies 2	0	korean2	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
106	105	Korean Movies and Indian Movies Hybrid	0	koreanindian	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
107	106	Korean Movies	0	korean3	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
108	107	Horror Movie 2	0	horror2	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
109	108	Indian Movies 2	0	indian2	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
110	109	Unrateable Items 2	0	unrateable2	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
111	110	Horror Movie 3	0	horror3	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
112	111	Korean Movies 3	0	korean3	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
113	112	Korean Movies and Indian Movies Hybrid 2	0	koreanindian2	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>
114	113	Korean Movies 4	0	korean4	1	<a href="#">View Details</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Details</a>

### specific user

### specific title



The screenshot shows two side-by-side browser windows. Both windows have a header bar with tabs, back/forward buttons, and a search bar. The left window's title bar includes the URL 'http://127.0.0.1:8000/accounts/all\_user\_associations/'. The right window's title bar includes the URL 'http://127.0.0.1:8000/accounts/jhi\_all\_user\_associations/'. Both windows show a 'User Associations' table with columns: user\_id, id, and name. The left window has 29 rows, and the right window has 17 rows. Below the table, both windows feature a red-bordered box labeled 'Latest Movies' containing a table with columns: movie\_id, id, and name. The left window has 1 row, and the right window has 1 row.

## both title and username

no records

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*

End of Task 1

## Task

## Group: All Users Association Page (likely an admin page)

## Task #2: Screenshot the code

**Weight: ~33%**

Points: ~0.67



^ COLLAPSE ^

## ⓘ Details:

**Include ucid/date comments for each code screenshot**

6

Columns: 1

### Sub-Task

Group: All Users Association Page (likely an admin page)

## Task #2: Screenshot the code

**Sub Task #1:** Show the code related to fetching all associations (including the query).

## Task Screenshots

## Gallery Style: 2 Columns

4 2 1

## part 1

part 2

```
//jdj55 12/6/24]
$db = getDB();
$stmt = $db->prepare($query);
$results = [];
$numAssoc = 0;
try {
    $stmt->execute($params);
    $r = $stmt->fetchAll(PDO::FETCH_ASSOC);
    if ($r)
    {
        $results = $r;
    }
    $numAssoc = count($results);
} catch (PDOException $e) <- #87-95 try
{
    error_log("Error fetching movies " . var_export($e, true));
    flash("Unhanlded error occurred", "danger");
}
```

part 3

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*

## Task Response Prompt

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

### **Response:**

I create a query that selects the id from user\_movies, the id from movies, the title from movies, is\_api from movies, and the username from Users. I also create a variable called assoc\_count that gets the number of time a movie appears in the UserMovies table. I do two inner joins. The first is with UserMovies that matches where Movies.id is equal to UserMovies.id. The second inner join is with Users where the UserMovies.user\_id matches with User.id. These joins are necessary as I need data from all of these tables and only the matching data.

### Sub-Task

Group: All Users Association Page (likely an admin page)

## Task #2: Screenshot the code

Sub Task #2: Show the code related to the display of the results

## ▲ Task Screenshots

## Gallery Style: 2 Columns

4 2 1

The creation of the table variable, the delete URL, ignore\_column, and filterData.

### Where render\_table is called

The updated table.php. Couldn't fit the entire line

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*

## Task Response Prompt

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

Response\*

Similarly to what was used before in the previous milestone, I used the render\_table() function to create the table with the data taken from the database. However, I added in functionality to support the profile redirection. Now, table supports "profile\_url" when filling in the parameters. This works similarly to the other buttons. I also updated the delete button to allow for the deletion of associations. This was necessary as, previously, the delete button only supported deleting movies. I checked if an association is to be deleted if an association column is contained within the passed in data.

**Sub-Task**

Group: All Users Association Page (likely an admin page)

Task #2: Screenshot the code

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

100%

## Task Screenshots

Gallery Style: 2 Columns

4

2

1



view url code in render table

view url being passed into the table array

**Caption(s) (required) ✓**

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

The view functions works similarly to how it did in milestone 2. The data that is passed into render function contains the id of the movie to be viewed. render table constructs the url and redirects the user to the view page.

**Sub-Task**

Group: All Users Association Page (likely an admin page)

Task #2: Screenshot the code

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

100%

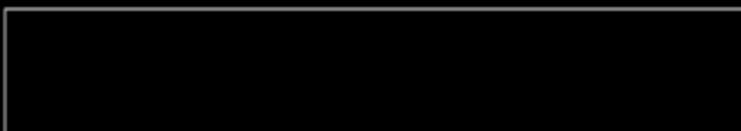
## Task Screenshots

Gallery Style: 2 Columns

4

2

1



added in profile url functionality

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

## ≡, Task Response Prompt

*Explain in concise steps how this logically works*

Response:

As stated before, I couldn't figure out how to get a clickable username using the render\_table() function. I wanted to use this function as I liked how it looked and how it compacted all the information vs. the cards. To get around this, I added in view profile functionality as explained above.

### Sub-Task

100%

Group: All Users Association Page (likely an admin page)

Task #2: Screenshot the code

Sub Task #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

## ☒ Task Screenshots

Gallery Style: 2 Columns

4

2

1



Delete url

```
if($assoc_id != null){  
    $sql = "SELECT * FROM UserMovies WHERE assoc_id = ?";  
    $stmt = $conn->prepare($sql);  
    $stmt->execute([$assoc_id]);  
    $assoc = $stmt->fetch();  
    $assoc_id = $assoc['assoc_id'];  
    $user_id = $assoc['user_id'];  
    $movie_id = $assoc['movie_id'];  
  
    if($assoc_id != null){  
        $sql = "DELETE FROM UserMovies WHERE assoc_id = ?";  
        $stmt = $conn->prepare($sql);  
        $stmt->execute([$assoc_id]);  
        $conn->commit();  
        echo "Successfully deleted association: user_id: " . $user_id . ", movie_id: " . $movie_id . "  
        die(header("location: " . get_url('associations')));  
    }  
}  
  
else{  
    echo "No association found";  
    die(header("location: " . get_url('associations')));  
}
```

delete\_assoc.php

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

## ≡, Task Response Prompt

*Explain in concise steps how this logically works*

Response:

As explained above, I added in functionality to the render\_table code to allow for the deletion of associations. This is done by checking if the passed in data has a "assoc\_id". If yes, then the delete button will be used for the deletion of the association. As separate delete\_assoc.php file is used to delete the association from the database. This file gets the id of the association and, using an SQL query, deletes it from the UserMovies table.

**Sub-Task**

100%

Group: All Users Association Page (likely an admin page)

Task #2: Screenshot the code

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

**Task Screenshots**

Gallery Style: 2 Columns

4      2      1

```
<h3 class = "text-center">All User Associations</h3>
//jd755 12/6/24
<h4 class="text-center" >Number of Associations:
<?php
try {
    $stmt = $db->prepare("SELECT COUNT(*) FROM UserMovies");
    $stmt->execute();
    $r = $stmt->fetchall(PDO::FETCH_ASSOC);
    if ($r) {
        $assocNum = $r[0]["COUNT(*)"];
        echo $assocNum;
    }
} catch (PDOException $e) {
    flash("Error retrieving number of movies in watch list", "danger");
    error_log("Error removing watch: " . var_export($e, true));
}
?>
c/bd>
```

code to count all associated items

**Caption(s) (required)** ✓Caption Hint: *Describe/highlight what's being shown***Task Response Prompt***Explain in concise steps how this logically works*

Response:

I use an sql query to count the number of items inside the UserMovies table. I then echo this number after extracting it from the resulting array.

**Sub-Task**

100%

Group: All Users Association Page (likely an admin page)

Task #2: Screenshot the code

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

**Task Screenshots**

Gallery Style: 2 Columns

4      2      1

```
//jd755
<h5 class="text-center">
    Associations On Page:
<?php echo $numAssoc; ?>
</h5>
```

displaying num items on page

```
//jd755 12/6/24
$db = getDB();
$stmt = $db->prepare($query);
$results = [];
$numAssoc = 0;
try {
    $stmt->execute($params);
    $r = $stmt->fetchall(PDO::FETCH_ASSOC);
    if ($r)
        [
            $results = $r;
        ]
    $numAssoc = count($results);
} catch (PDOException $e) {
    error_log("Error fetching movies " . var_export($e, true));
    flash("Unhandle error occurred", "danger");
}
```

get num of items

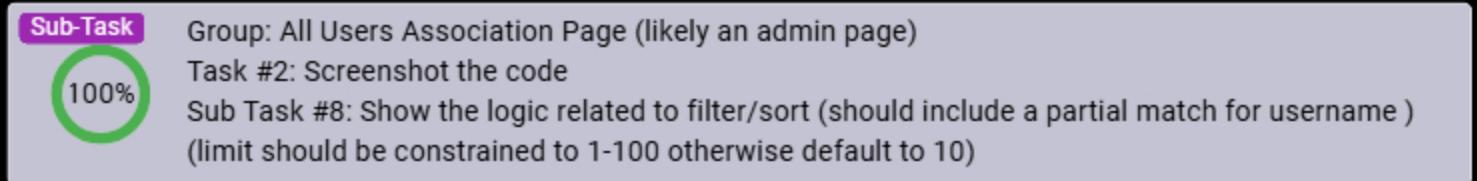
**Caption(s) (required)** ✓Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

### **Response:**

I calculate the number of items simply by counting the elements in the resulting array after doing the sql query. I then display this number in the HTML.



## Task Screenshots

## Gallery Style: 2 Columns

4 2 1

checking for valid num

checking for valid title and username

```
function validateForm() {
    let title = form.title.value;
    let filter = form.filter.value;
    let username = form.username.value;
    let isValid = true;

    if(title.length > 200) {
        flash("JavaScript: title too long (cannot exceed 200 characters)", "warning");
        isValid = false;
    }

    if(filter < 1 || filter > 100) {
        flash("JavaScript: Filter has to be between 1 and 100", "warning");
        isValid = false;
    }

    if(username.length > 16) {
        flash("JavaScript: username too long (max. 16 characters)", "warning");
        isValid = false;
    }

    return isValid;
}
```

## javascript validation

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

### **Response:**

I check if \$num falls within the range of 1-100. I then check if title is greater than 200 characters. If it is, it displays a warning to the user and does a default query. I then check if the username is greater than 16 characters. Once again, send an error if this is the case. I also included the appropriate javascript validations.

**Task**

Group: All Users Association Page (likely an admin page)

Task #3: Add related links

Weight: ~33%

Points: ~0.67

▲ COLLAPSE ▲

**Checklist**

\*The checkboxes are for your own tracking

#	Details
#1	Include the heroku prod link for the page that creates the association
#2	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

**Task URLs****URL #1**

[https://it202-jd755-prod-028c8689138c.herokuapp.com/admin/list\\_assoc.php?title=&user=&filter=](https://it202-jd755-prod-028c8689138c.herokuapp.com/admin/list_assoc.php?title=&user=&filter=)

URL

<https://it202-jd755-prod-028c8689138c.herokuapp.com/>

**URL #2**

<https://github.com/legitdaylight/jd755-IT202-007/pull/67>

URL

<https://github.com/legitdaylight/jd755-IT202-007>

**End of Task 3****End of Group: All Users Association Page (likely an admin page)****Task Status:** 3/3**Group**

Group: Unassociated Page

Tasks: 3

Points: 2

▲ COLLAPSE ▲

**Task**

Group: Unassociated Page

Task #1: Screenshots of this page

Weight: ~33%

Points: ~0.67

▲ COLLAPSE ▲

**i Details:**

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

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



Columns: 1

**Sub-Task**

Group: Unassociated Page

Task #1: Screenshots of this page

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

**Task Screenshots**

Gallery Style: 2 Columns

4 2 1

main page

**Caption(s) (required)** ✓Caption Hint: *Describe/highlight what's being shown***Sub-Task**

Group: Unassociated Page

Task #1: Screenshots of this page

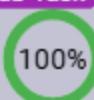
Sub Task #2: Show the single view buttons/links

**Task Screenshots**

Gallery Style: 2 Columns

4 2 1

single view on the side

**Caption(s) (required)** ✓Caption Hint: *Describe/highlight what's being shown***Sub-Task**

Group: Unassociated Page

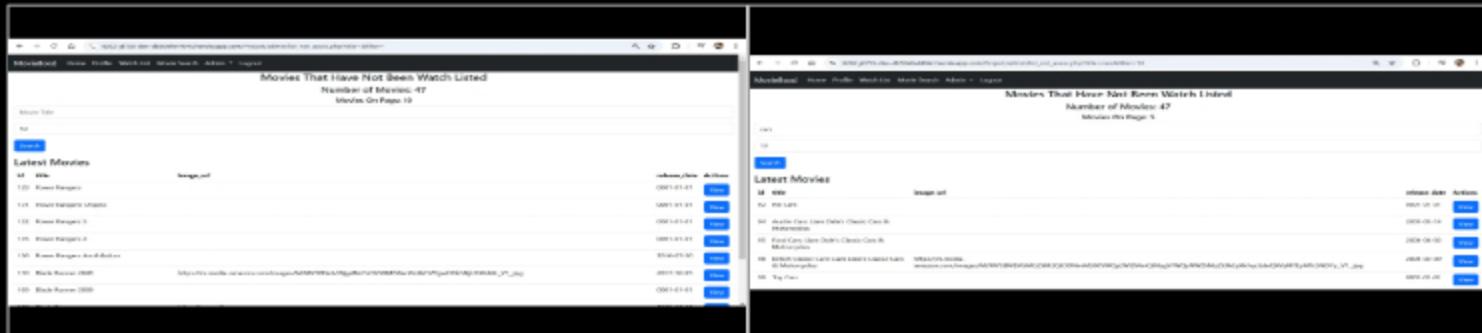
Task #1: Screenshots of this page

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

## Task Screenshots

## Gallery Style: 2 Columns

4 2 1



### number of movies vs number shown

another example

**Caption(s) (required) ✓**

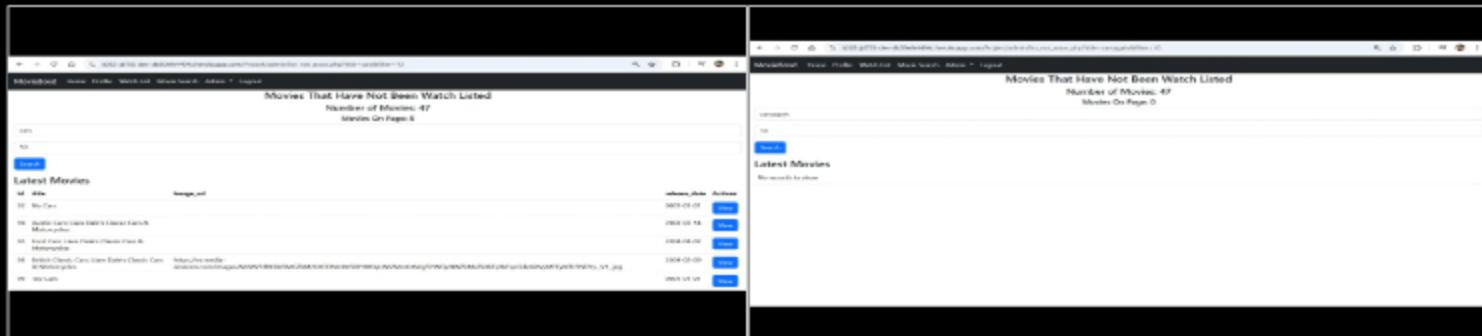
**Caption Hint:** *Describe/highlight what's being shown*

<b>Sub-Task</b>	Group: Unassociated Page
100%	Task #1: Screenshots of this page
	Sub Task #4: Show variations of the filter/sort including no results (proper message should be visible)

## Task Screenshots

## Gallery Style: 2 Columns

4 2 1



## example of filter

no records

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*

End of Task 1

## Task

 100%	Group: Unassociated Page
	Task #2: Screenshot the code
	Weight: ~33%
	Points: ~0.67

**i** Details:

Include ucid/date comments for each code screenshot



Columns: 1

**Sub-Task**



Group: Unassociated Page

Task #2: Screenshot the code

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

## Task Screenshots

Gallery Style: 2 Columns

4                    2                    1

```
SELECT * FROM movies AS m
LEFT JOIN UserMovies AS um ON m.id = um.movie_id
WHERE um.movie_id IS NULL;
```

code to fetch unassociated items

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

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

Response:

I selected the id, title, image url, and release date from Movies while doing a left join UserMovies. I checked where UserMovies.movie\_id is null and where UserMovies.movie\_id is equal to movie id. The left join is necessary as I wanted all the data from the Movies table. The filter logic works the same as before, as in I check for the validity of num and title.

**Sub-Task**



Group: Unassociated Page

Task #2: Screenshot the code

Sub Task #2: Show the code related to the display of the results

## Task Screenshots

Gallery Style: 2 Columns

4                    2                    1

```
//admin/admin.php  
$filterData = ["title" => $title, "filter" => $num];  
$deleteURL = "admin/delete_movie.php?<?php http_build_query($filterData);  
$table = [{"data" => $results, "title" => "Latest Movies", "view_url" => get_url("view_movie.php")}]
```

formation of \$table

//jd755 12/6/24

```
<?php render_table($table); ?>
```

rendering the table

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

I used the render\_table() function as before, but this time I only passed in the view\_url into \$table. Later in the HTML, I call the render\_table function.

Sub-Task

Group: Unassociated Page

100%

Task #2: Screenshot the code

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

## Task Screenshots

Gallery Style: 2 Columns

4      2      1



the view url

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

I passed in the view\_movie url to the \$table array in order for the view button to appear on the result table.

Sub-Task

Group: Unassociated Page

100%

Task #2: Screenshot the code

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

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
//jd755 12/6/24
<?php
try {
    $stmt = $db->prepare("SELECT COUNT(Movies.id)
    FROM Movies
    LEFT JOIN UserMovies
    ON Movies.id = UserMovies.movie_id
    WHERE UserMovies.movie_id IS NULL");
    $stmt->execute();
    $r = $stmt->fetchAll(PDO::FETCH_ASSOC);
    if ($r) {
        $numMovies = $r[0]["COUNT(Movies.id)"];
        echo $numMovies;
    }
} catch (PDOException $e) {
    flash("Error retrieving number of movies in watch list", "danger");
    error_log("Error removing watch: " . var_export($e, true));
}
?>
</body>
```

count of all unassociated

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

I used COUNT() on the Movie.id to get the number of movie ids that are't in the UserMovies list. I used a left join based on UserMovies.movie\_id is equal to Movies.id. THen I check where UserMovies.movie\_id is equal to NULL. This would return all of the movies whos movie id in UserMovies are null.

Sub-Task

100%

Group: Unassociated Page

Task #2: Screenshot the code

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

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
//jd755 12/6/24
$db = getDB();
$stmt = $db->prepare($query);
$results = [];
try {
    $stmt->execute($params);
    $r = $stmt->fetchAll(PDO::FETCH_ASSOC);
    if ($r) {
        $results = $r;
    }
    $numMovies = count($results);
} catch (PDOException $e) {
    error_log("Error fetching movies " . var_export($e, true));
    flash("Unhandled error occurred", "danger");
}
?>
```

calculating num elements

```
//jd755 12/6/24
<h5 class="text-center">
    Movies On Page:
    <?php echo $numMovies; ?>
</h5>
```

displaying num elements

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

Response.

The logic here works the same as it does in the other pages. I get the length of the resulting array and I display that number in the HTML.

Sub-Task

Group: Unassociated Page

100%

Task #2: Screenshot the code

Sub Task #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)

## Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
//july 22/6/24
function validate(form)
{
    let title = form.title.value;
    let filter = form.filter.value;
    let isValid = true;

    if(title.length > 200)
    {
        flash("[JavaScript] Title too long (cannot exceed 200 characters)", "warning");
        isValid = false;
    }

    if(filter < 0 || filter > 100)
    {
        flash("[JavaScript] Filter has to be between 1 and 100", "warning");
        isValid = false;
    }

    return isValid;
}

// july 22/6/24
function validateForm()
{
    let title = document.getElementById("title").value;
    let filter = document.getElementById("filter").value;
    let isValid = true;

    if(title.length > 200)
    {
        flash("[JavaScript] Title too long (cannot exceed 200 characters)", "warning");
        isValid = false;
    }

    if(filter < 0 || filter > 100)
    {
        flash("[JavaScript] Filter has to be between 1 and 100", "warning");
        isValid = false;
    }

    return isValid;
}
```

javascript validation

```
//july 22/6/24
function validate()
{
    let title = document.getElementById("title").value;
    let filter = document.getElementById("filter").value;
    let params = [];

    if(filter === "")
    {
        flash("Filter is empty", "warning");
        filter = 10;
    }

    if(filter < 1 || filter > 100)
    {
        flash("Filter has to be between 1 and 100", "warning");
        filter = 10;
    }

    if(filter > 0)
    {
        params.push(`filter=${filter}`);
    }

    if(title.length > 200)
    {
        flash("Title too long (cannot exceed 200 characters)", "warning");
    }
}

//july 22/6/24
let query = `SELECT movies.id, movies.title, movies.image_url, movies.release_date
FROM movies
LEFT JOIN usermovies
ON movies.id = usermovies.movie_id
WHERE usermovies.movie_id IS NULL
ORDER BY movies.created DESC LIMIT 100`;

let params = [];

if(filter.length > 0)
{
    query += ` WHERE movies.title LIKE "%${filter}%"`;
    params.push(`filter=${filter}`);
}

query += ` ORDER BY movies.title ASC`;

let result = await fetch(query, {
    method: "GET",
    headers: {
        "Content-Type": "application/json"
    },
    params: params
}).then(res => res.json())
    .catch(error => console.error(error));

document.querySelector("#movies").innerHTML = result.map(movie => `
    

\${movie.title}



Release Date: ${movie.release_date}


`).join(`
```

`);

document.querySelectorAll(".image").forEach(img =&gt; img.addEventListener("click", async () =&gt; {
 const movieId = img.getAttribute("data-movie-id");
 const movie = result.find(m =&gt; m.id === movieId);
 const url = `/movie/\${movieId}`;
 window.location.href = url;
}))

php validation

```
//july 22/6/24
let query = `SELECT movies.id, movies.title, movies.image_url, movies.release_date
FROM movies
LEFT JOIN usermovies
ON movies.id = usermovies.movie_id
WHERE usermovies.movie_id IS NULL
ORDER BY movies.created DESC LIMIT 100`;

let params = [];

if(filter.length > 0)
{
    query += ` WHERE movies.title LIKE "%${filter}%"`;
    params.push(`filter=${filter}`);
}

query += ` ORDER BY movies.title ASC`;

let result = await fetch(query, {
    method: "GET",
    headers: {
        "Content-Type": "application/json"
    },
    params: params
}).then(res => res.json())
    .catch(error => console.error(error));

document.querySelector("#movies").innerHTML = result.map(movie => `
    

\${movie.title}



Release Date: ${movie.release_date}


`).join(`
```

`);

document.querySelectorAll(".image").forEach(img =&gt; img.addEventListener("click", async () =&gt; {
 const movieId = img.getAttribute("data-movie-id");
 const movie = result.find(m =&gt; m.id === movieId);
 const url = `/movie/\${movieId}`;
 window.location.href = url;
}))

partial match of title

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works*

Response:

I check if the num is within the valid range. If its not, it gets defaulted to 10. I also check if title is valid by seeing if it is less than or equal to 200 characters. I use a partial match of title in the sql query if the title is valid by using the LIKE keyword in SQL.

End of Task 2

Task

Group: Unassociated Page

Task #3: Add related links

100%

Weight: ~33%  
 Points: ~0.67

▲ COLLAPSE ▲

**Checklist**

\*The checkboxes are for your own tracking

#	Details
<input checked="" type="checkbox"/> #1	Include the heroku prod link for the page that creates the association
<input checked="" type="checkbox"/> #2	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

**Task URLs****URL #1**

[https://it202-jd755-prod-028c8689138c.herokuapp.com/admin/list\\_not\\_assoc.php?  
 title=&filter=](https://it202-jd755-prod-028c8689138c.herokuapp.com/admin/list_not_assoc.php?title=&filter=)

URL

[https://it202-jd755-prod-028c8689138c.herokuapp.com/admin/list\\_not\\_assoc.php?  
 title=&filter=](https://it202-jd755-prod-028c8689138c.herokuapp.com/admin/list_not_assoc.php?title=&filter=)

**URL #2**

<https://github.com/legitdaylight/jd755-IT202-007/pull/66>

URL

<https://github.com/legitdaylight/jd755-IT202-007/pull/66>

**End of Task 3****End of Group: Unassociated Page**

Task Status: 3/3

**Group**

100%

Group: Admin Association Management (like UserRoles)

Tasks: 3

Points: 1

▲ COLLAPSE ▲

**Task**

100%

Group: Admin Association Management (like UserRoles)

Task #1: Screenshots of the page

Weight: ~33%

Points: ~0.33

▲ COLLAPSE ▲

**Details:**

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

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



Columns: 1

**Sub-Task**

Group: Admin Association Management (like UserRoles)

100%

- Group: Admin Association Management (like UserRoles)  
Task #1: Screenshots of the page  
Sub Task #1: Show the search form with valid data

## Task Screenshots

Gallery Style: 2 Columns

4

2

1



main page

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

- Group: Admin Association Management (like UserRoles)  
Task #1: Screenshots of the page  
Sub Task #2: Show the results of the search

## Task Screenshots

Gallery Style: 2 Columns

4

2

1



result of search

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

100%

- Group: Admin Association Management (like UserRoles)  
Task #1: Screenshots of the page  
Sub Task #3: Show the result of entities and users being associated and unassociated

## Task Screenshots

Gallery Style: 2 Columns

**before unassociating with Battle Royale(2000)**

after unassociating with Battle Royale (2000)

**before associating with Cars**

after associating with cars

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*

End of Task 1



## Group: Admin Association Management (like UserRoles)

## Task #2: Screenshots of the code

**Weight: ~33%**

Points: ~0.33

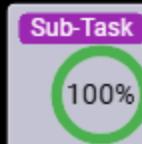
COLLAPSE

## Details:

**Include ucid/date comments for each code screenshot**

1

Columns: 1



Group: Admin Association Management (like UserRoles)

## Task #2: Screenshots of the code

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

## Task Screenshots

## Gallery Style: 2 Columns

4

2

1

```
    <?php $this->loadTemplate("header"); ?>
    <?php $this->loadTemplate("header"); ?>
    <?php $this->loadTemplate("header"); ?>
```

search form for username in the middle

sql query to find partial match of usernames at the top

**Caption(s) (required)** ✓

**Caption Hint:** *Describe/highlight what's being shown*

## Task Response Prompt

*Explain in concise steps how this logically works.*

### **Response:**

I call render input to create an input of type search. The inputted data of this input is sent to the \$username variable to be used in the sql query. The sql query works by selecting the username and id of each matched username. I also did a inner query that concatenates all of the movies associated with that user by doing a join with the UserMovies table from the Movie table and selecting all the data that match the movie ids in both tables. This result is marked as movies. After, I do a partial match with username by using the inputted username variable.

### Sub-Task

Group: Admin Association Management (like UserRoles)

## Task #2: Screenshots of the code

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

## Task Screenshots

## Gallery Style: 2 Columns

4

2

1

```

CREATE PROCEDURE [dbo].[sp_update_member]
@id int,
@username nvarchar(50),
@password nvarchar(50),
@name nvarchar(50),
@mobile nvarchar(15),
@email nvarchar(50),
@gender char(1),
@status bit
AS
BEGIN
    -- Check if user exists
    IF EXISTS (SELECT * FROM members WHERE id = @id)
        BEGIN
            UPDATE members SET
                username = @username,
                password = @password,
                name = @name,
                mobile = @mobile,
                email = @email,
                gender = @gender,
                status = @status
        END
    ELSE
        BEGIN
            INSERT INTO members (id, username, password, name, mobile, email, gender, status)
            VALUES (@id, @username, @password, @name, @mobile, @email, @gender, @status)
        END
    END
END

CREATE PROCEDURE [dbo].[sp_get_member_by_id]
@id int
AS
BEGIN
    SELECT * FROM members WHERE id = @id
END

CREATE PROCEDURE [dbo].[sp_get_member_by_name]
@name nvarchar(50)
AS
BEGIN
    SELECT * FROM members WHERE name = @name
END

CREATE PROCEDURE [dbo].[sp_get_member_by_email]
@email nvarchar(50)
AS
BEGIN
    SELECT * FROM members WHERE email = @email
END

CREATE PROCEDURE [dbo].[sp_get_member_by_mobile]
@mobile nvarchar(15)
AS
BEGIN
    SELECT * FROM members WHERE mobile = @mobile
END

CREATE PROCEDURE [dbo].[sp_get_member_by_status]
@status bit
AS
BEGIN
    SELECT * FROM members WHERE status = @status
END

CREATE PROCEDURE [dbo].[sp_get_member_by_gender]
@gender char(1)
AS
BEGIN
    SELECT * FROM members WHERE gender = @gender
END

CREATE PROCEDURE [dbo].[sp_get_member_by_email_and_password]
@email nvarchar(50),
@password nvarchar(50)
AS
BEGIN
    SELECT * FROM members WHERE email = @email AND password = @password
END

CREATE PROCEDURE [dbo].[sp_get_member_by_name_and_password]
@name nvarchar(50),
@password nvarchar(50)
AS
BEGIN
    SELECT * FROM members WHERE name = @name AND password = @password
END

CREATE PROCEDURE [dbo].[sp_get_member_by_email_and_name]
@email nvarchar(50),
@name nvarchar(50)
AS
BEGIN
    SELECT * FROM members WHERE email = @email AND name = @name
END

CREATE PROCEDURE [dbo].[sp_get_member_by_email_and_mobile]
@email nvarchar(50),
@mobile nvarchar(15)
AS
BEGIN
    SELECT * FROM members WHERE email = @email AND mobile = @mobile
END

CREATE PROCEDURE [dbo].[sp_get_member_by_name_and_mobile]
@name nvarchar(50),
@mobile nvarchar(15)
AS
BEGIN
    SELECT * FROM members WHERE name = @name AND mobile = @mobile
END

CREATE PROCEDURE [dbo].[sp_get_member_by_email_and_name_and_mobile]
@email nvarchar(50),
@name nvarchar(50),
@mobile nvarchar(15)
AS
BEGIN
    SELECT * FROM members WHERE email = @email AND name = @name AND mobile = @mobile
END

```

**form for movie title**

SQL query for selecting movies at the bottom

**Caption(s) (required) ✓**

**Caption Hint:** *Describe/highlight what's being shown*

### Task Response Prompt

*Explain in concise steps how this logically works*

### **Response:**

Similar to above, but for the movie titles. The text is sent to the \$movie variable. This sql query works by getting the

title (renamed to name) and the id from the Movies table where the title matches the partial match of the inputted title.

**Sub-Task**

100%

Group: Admin Association Management (like UserRoles)

Task #2: Screenshots of the code

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

## Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
SELECT user_id, username
FROM (
    SELECT user_id, username
    FROM users
    WHERE user_id <= 25
) AS user_results
ORDER BY user_id ASC
LIMIT 25;
```

```
SELECT movie_id, title
FROM (
    SELECT movie_id, title
    FROM movies
    WHERE title LIKE '%$%' OR title LIKE '$%'
) AS movie_results
ORDER BY movie_id ASC
LIMIT 25;
```

Limits seen in the sql queries

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

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

Response:

In both of the sql queries for the users and the movies, I add a LIMIT to limit the results to 25 entities.

**Sub-Task**

100%

Group: Admin Association Management (like UserRoles)

Task #2: Screenshots of the code

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

## Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
<table class="table">
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| <input |

```

checkbox being created

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## ≡, Task Response Prompt

*Explain in concise steps how this logically works*

Response:

The checkbox code for both username and movies work the same. Every entity in the resulting arrays for both queries (\$users for the username, \$movies for the movies) are iterated through. For each of these element, a label is created with its respective data (id, username/title) and a checkbox input is used underneath to link with the label.

**Sub-Task**

100%

Group: Admin Association Management (like UserRoles)

Task #2: Screenshots of the code

Sub Task #5: Code related to submitting the checkbox lists

## ■ Task Screenshots

Gallery Style: 2 Columns

4      2      1



submit button at the bottom

**Caption(s) (required)** ✓

Caption Hint: *Describe/highlight what's being shown*

## ≡, Task Response Prompt

*Explain in concise steps how this logically works*

Response:

Both checkbox are within the same </form> block. At the end is a submit button that sends the data contained within the checkbox (all the users selected and all the movies selected) and sends it to POST.

**Sub-Task**

100%

Group: Admin Association Management (like UserRoles)

Task #2: Screenshots of the code

Sub Task #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)

## ■ Task Screenshots

Gallery Style: 2 Columns

```

private void associateMovieWithUser(int userId, int movieId) {
    User user = User.findById(userId);
    Movie movie = Movie.findById(movieId);
    UserMovie userMovie = UserMovie.create(user, movie);
    UserMovie.save();
}

private void unassociateMovieWithUser(int userId, int movieId) {
    User user = User.findById(userId);
    Movie movie = Movie.findById(movieId);
    UserMovie userMovie = UserMovie.create(user, movie);
    UserMovie.delete();
}

private void associateUserWithMovie(int userId, int movieId) {
    User user = User.findById(userId);
    Movie movie = Movie.findById(movieId);
    UserMovie userMovie = UserMovie.create(user, movie);
    UserMovie.save();
}

private void unassociateUserWithMovie(int userId, int movieId) {
    User user = User.findById(userId);
    Movie movie = Movie.findById(movieId);
    UserMovie userMovie = UserMovie.create(user, movie);
    UserMovie.delete();
}

private void associateUserWithMovie(int userId, int movieId) {
    User user = User.findById(userId);
    Movie movie = Movie.findById(movieId);
    UserMovie userMovie = UserMovie.create(user, movie);
    UserMovie.save();
}

private void unassociateUserWithMovie(int userId, int movieId) {
    User user = User.findById(userId);
    Movie movie = Movie.findById(movieId);
    UserMovie userMovie = UserMovie.create(user, movie);
    UserMovie.delete();
}

```

code to update submission

### Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

## Task Response Prompt

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

Response:

If the data sent from the checkboxes are set in the POST (users and movieList), a double for each loop is used to add/remove the associations between the user and the movie. First, the association is attempted to be created between the user and the movie. An SQL statement to add in the combination of the user id and movie id to the UserMovie table is called. If this passes, then the association was successfully created. If it fails, then the type of the error is checked. If the error is due to a duplicate, then a different sql query is called that deletes from UserMovies the data that has both the movie id and the user id.

## End of Task 2

### Task



Group: Admin Association Management (like UserRoles)

Task #3: Add related links

Weight: ~33%

Points: ~0.33

[▲ COLLAPSE ▲](#)

### Checklist

\*The checkboxes are for your own tracking

#	Details
<input checked="" type="checkbox"/> #1	Include the heroku prod link for the page that creates the association
<input checked="" type="checkbox"/> #2	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

## Task URLs

URL #1

<https://it202-jd755->

prod-028c8689138c.herokuapp.com/project/admin/assign\_movies.php

URL #2

[https://github.com/legitdaylight/jd755-IT202-007](https://github.com/legitdaylight/jd755-IT202-007/pull/68)

URL

<https://it202-jd755-prod-028c8689138c.herokuapp.com/>

URL

<https://github.com/legitdaylight/jd755-IT202-007>

## End of Task 3

End of Group: Admin Association Management (like UserRoles)

Task Status: 3/3

### Group

Group: Misc

Tasks: 4

Points: 1

100%

▲ COLLAPSE ▲

### Task

Group: Misc

Task #1: Screenshot of your project board from GitHub (tasks should be in the proper column)

Weight: ~25%

Points: ~0.25

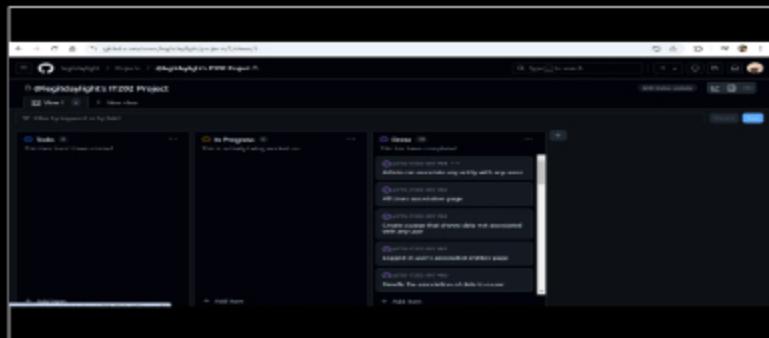
100%

▲ COLLAPSE ▲

## ▣ Task Screenshots

Gallery Style: 2 Columns

4                    2                    1



project board

## End of Task 1

### Task

Group: Misc

Task #2: Provide a direct link to the project board on GitHub

Weight: ~25%

Points: ~0.25

100%

▲ COLLAPSE ▲

## 🔗 Task URLs

URL #1

URL

End of Task 2

## Task

Group: Misc

**Task #3: Talk about any issues or learnings during this assignment**

**Weight:** ~25%

Points: ~0.25

COLLAPSE

## Task Response Prompt

## Response:

Although this milestone was easier than the second, I did run into a few roadblocks. Notably, almost all of the user associations table feature gave me problems. I couldn't figure out how to create a link to a user profile, so I had to settle with creating a button off to the side. I also had trouble having a delete button for the association and a view button for the entity. After getting stuck on that for a while, I did find a solution by altering the table.php code.

End of Task 3

## Task

### Group: Misc

#### Task #4: WakaTime Screenshot

**Weight: ~25%**

Points: ~0.25

COLLAPSE

**● 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: 2 Columns

4

2



**End of Task 4**

**End of Group: Misc**

**Task Status: 4/4**

**End of Assignment**