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
Assigment: [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/vie
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

COLLAPSE

Task

100%

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

100%

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

Task Status: 2/2

Group

100%

Group: Handle Data Association
Tasks: 3
Points: 1

COLLAPSE

Task

100%

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

100%

Group: Handle Data Association
Task #1: Screenshots of the code
Sub Task #1: Show the related code

Task Screenshots

Gallery Style: 2 Columns

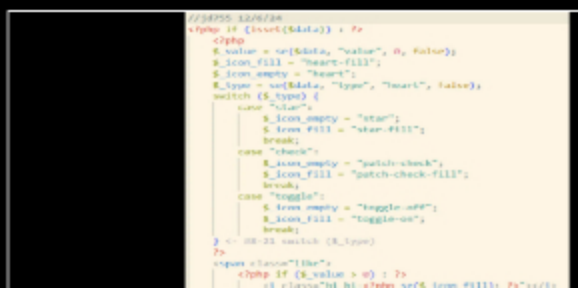
4 2 1

```
//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; ?>
```

Movie cards being created

```
//jd755 12/6/24
<?php render_like(["value" => $data["is_watched"]]); ?>
```

Within movie_card.php, the function render_like() is called.



inside render_like.php

toggle_watched.php

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

100%

Group: Handle Data Association
Task #2: Screenshot of the association table(s)
Weight: ~33%
Points: ~0.33

COLLAPSE

Sub-Task

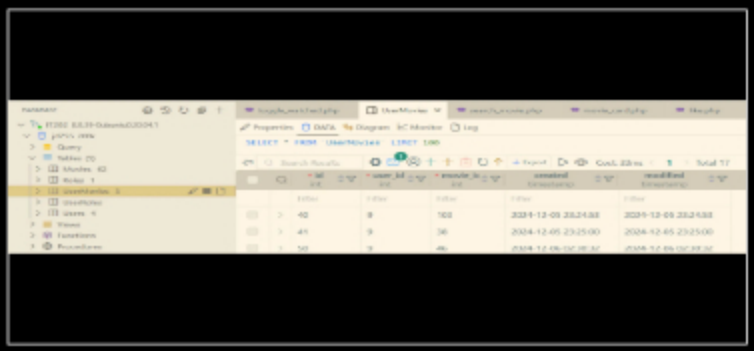
100%

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



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

100%

Group: Handle Data Association
Task #3: Add related links
Weight: ~33%
Points: ~0.33

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/Project/search_movie.php?title=&filter=

URL
https://it202-jd755-prod-028c8689138c.herokuap

URL #2
https://github.com/legitdaylight/jd755-IT202-007/pull/59

URL
https://github.com/legitdaylight/jd755-IT202-007

End of Task 3

End of Group: Handle Data Association
Task Status: 3/3

Group

100%

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

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

100%

Group: Current User's Association Page

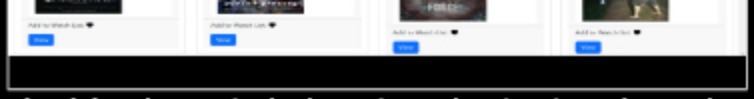
Task #1: Screenshots of this page

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*

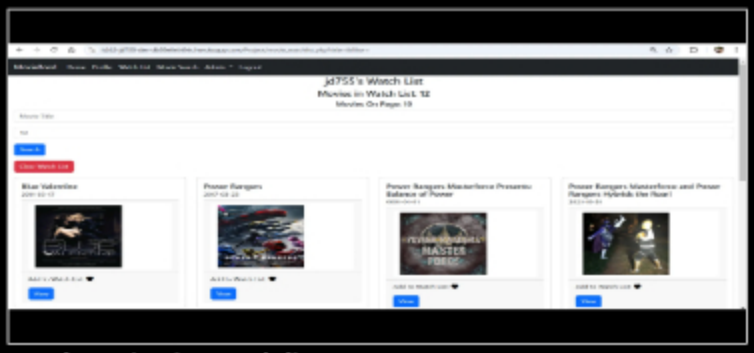
Sub-Task 100%

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 100%

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 doesnt 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)

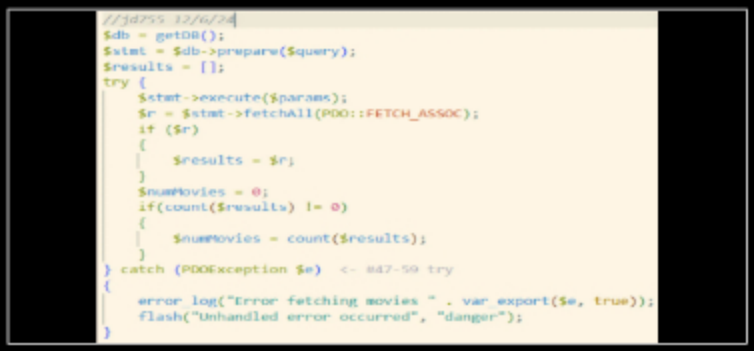
Task Screenshots

Gallery Style: 2 Columns

4 2 1

part 1

part 2



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 data, 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

100%

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 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 users

is_watched, the heart from before is rendered. Finally, at the bottom the view button is created only if users are logged in.

Sub-Task

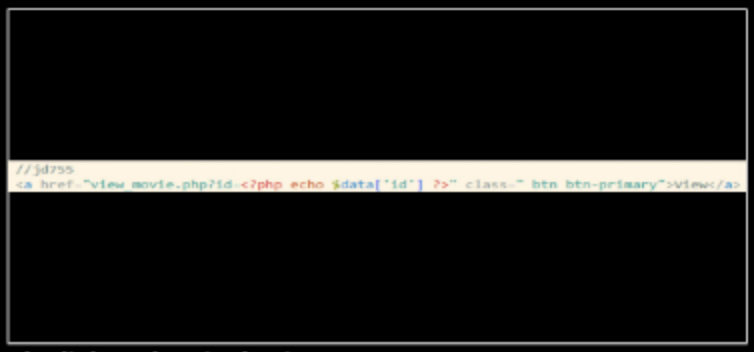
100%

Group: Current User's Association Page
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 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

100%

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1

code that calls toggle_watched

Where $watchedURL is used

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 the user already has an association, the association is deleted.

Sub-Task

100%

Group: Current User's Association Page

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

4 2 1

```
//jd755 12/6/24
if(isset($_POST['user_id']))
{
    try {
        $stmt = $db->prepare("DELETE FROM UserMovies WHERE user_id = :user_id");
        $stmt->execute($params);
        flash("Cleared Watch 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));
    }
} <-- #01-03 if(isset($_POST['user_id']))
```

```
//jd755 12/6/24
<form method="POST">
    <?php render_input(["type" => "hidden", "name" => "user_id", "value"=>get_user_id()]); ?>
    <?php render_button(["text" => "Clear Watch List", "type" => "submit", "color"=>"danger"]); ?>
</form>
```

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 iser_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

100%

Group: Current User's Association Page

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

4 2 1

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

100%

Group: Current User's Association Page

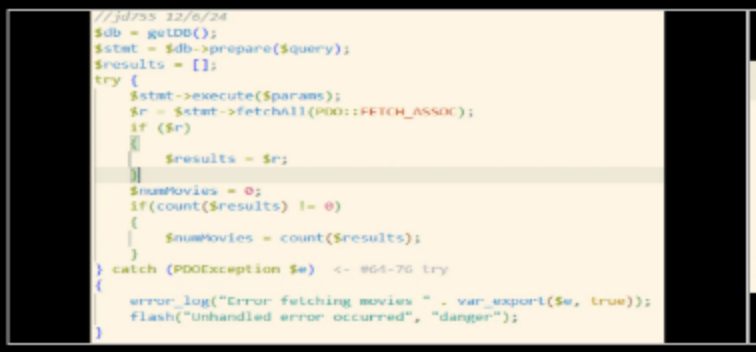
Task #2: Screenshot the code

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)

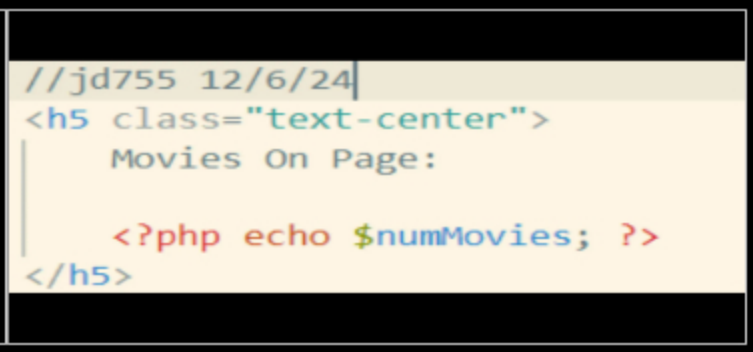
Task Screenshots

Gallery Style: 2 Columns

4 2 1



numMovies being calculated



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%

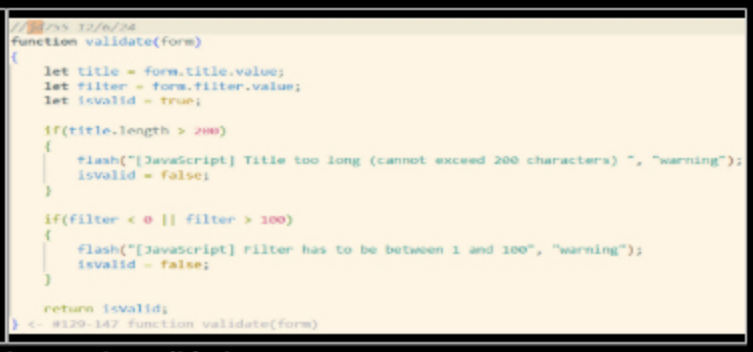
Group: Current User's Association Page
Task #2: Screenshot the code
Sub Task #8: Show the logic related to filter/sort (limit should be constrained to 1-100 otherwise default to 10)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

filter constraints



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 valdiation 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
#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

URL

https://it202-jd755-prod-028c8689138c.herokuapp.com/Project/movie_watchlist.php?title=&filter=

https://it202-jd755-prod-028c8689138c.herokuap

URL #2
https://github.com/legitdaylight/jd755-IT202-007/pull/65

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

End of Task 3

End of Group: Current User's Association Page
Task Status: 3/3

Group

100%

Group: All Users Association Page (likely an admin page)
Tasks: 3
Points: 2

COLLAPSE

Task

100%

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

100%

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

all user associations

Caption(s) (required) ✓

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

Sub-Task 100%

Group: All Users Association Page (likely an admin page)
Task #1: Screenshots of this page
Sub Task #2: Show the single view buttons/links, delete button/links

Task Screenshots

Gallery Style: 2 Columns

4 2 1

buttons on the side

Caption(s) (required) ✓

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

Sub-Task 100%

Group: All Users Association Page (likely an admin page)
Task #1: Screenshots of this page
Sub Task #3: Show the username related to the specific entity and that it's clickable

Task Screenshots

Gallery Style: 2 Columns

4 2 1

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 100%

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 100%

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

specific user

specific title

both title and username

no records

Caption(s) (required) ✓

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

End of Task 1

Task

100%

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

Columns: 1

Sub-Task

100%

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

```php
//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 $e)  <- #87-95 try
{
    error_log("Error fetching movies " . var_export($e, true));
    flash("Unhandled 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

100%

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 necessay 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

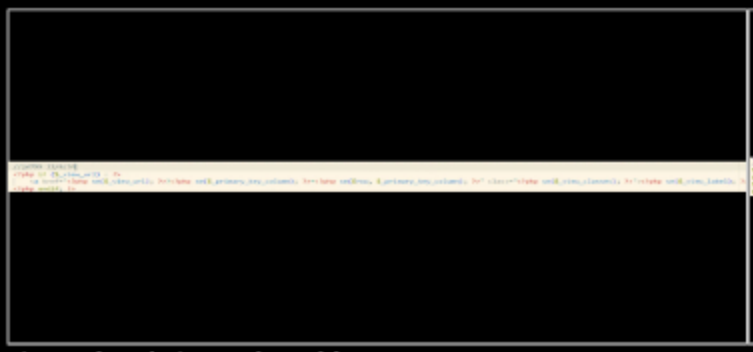
100%

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

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

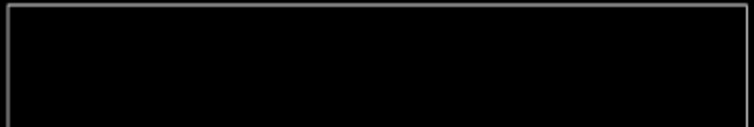
100%

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

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

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 seperate 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

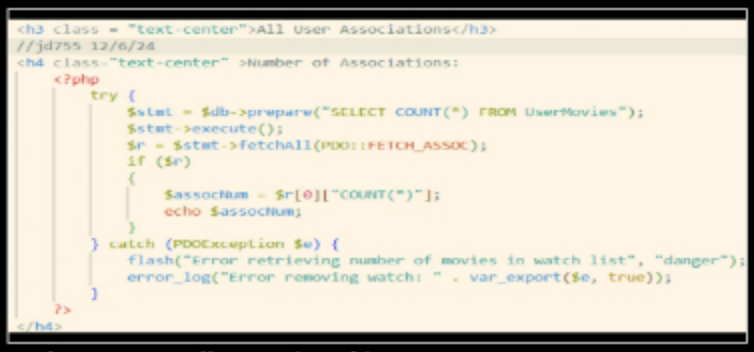
100%

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));
        }
    ?>
</h4>
```

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

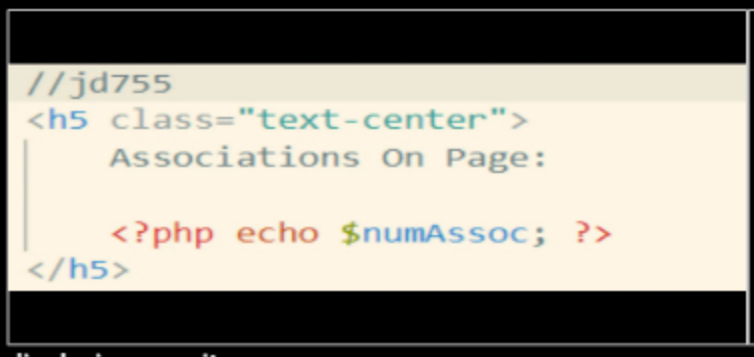
100%

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 $e)  <- #89-97 try
{
    error_log("Error fetching movies " . var_export($e, true));
    flash("Unhandled 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.

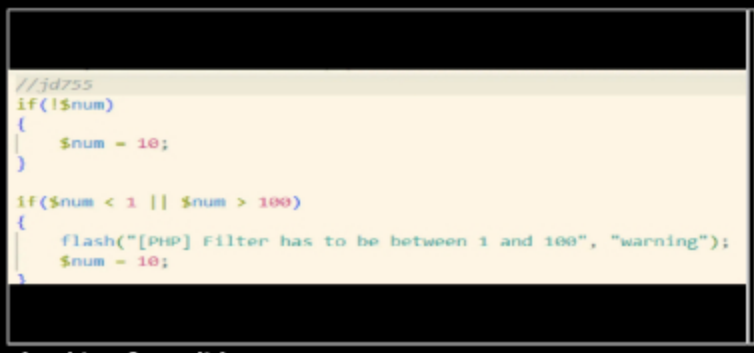
Sub-Task 100%

Group: All Users Association Page (likely an admin page)
Task #2: Screenshot the code
Sub Task #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)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

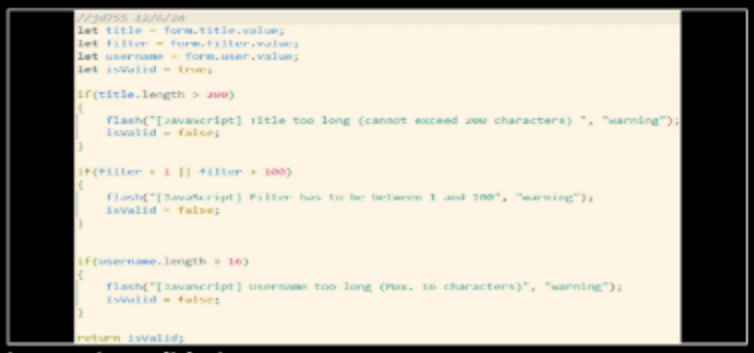


```
//jd755
if(!$num)
{
    $num = 10;
}

if($num < 1 || $num > 100)
{
    flash("[PHP] Filter has to be between 1 and 100", "warning");
    $num = 10;
}
```

checking for valid num

checking for valid title and username



```
//jd755 12/6/24
let title = form.title.value;
let filter = form.filter.value;
let username = form.user.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.

End of Task 2

Task

100%

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/Project/admin/list_assoc.php?title=&user=&filter=

URL
https://it202-jd755-prod-028c8689138c.herokuap

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

100%

Group: Unassociated Page
Tasks: 3
Points: 2

COLLAPSE

Task

100%

Group: Unassociated 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

requirement.

Columns: 1

Sub-Task 100%

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 100%

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 100%

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*

Sub-Task

100%

Group: Unassociated 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

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

COLLAPSE

Details:

Include ucid/date comments for each code screenshot

Columns: 1

Sub-Task

100%

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

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

100%

Group: Unassociated Page

Task #2: Screenshot the code

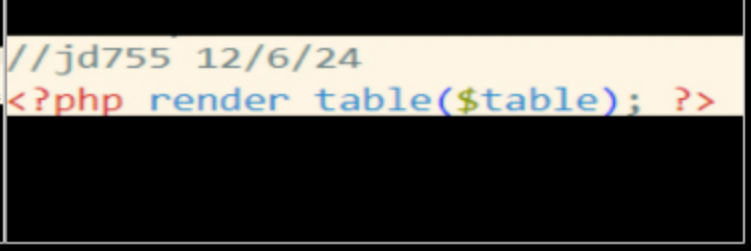
Sub Task #2: Show the code related to the display of the results

Task Screenshots

Gallery Style: 2 Columns

4 2 1

formation of $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

100%

Group: Unassociated Page

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

100%

Group: Unassociated Page

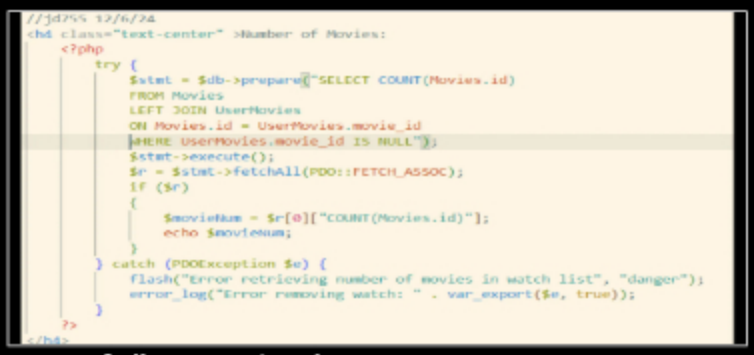
Task #2: Screenshot the code

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
<h4 class="text-center" >Number of Movies:
    <?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)
            {
                $movieNum = $r[0]["COUNT(Movies.id)"];
                echo $movieNum;
            }
        } catch (PDOException $e) {
            flash("Error retrieving number of movies in watch list", "danger");
            error_log("Error removing watch: " . var_export($e, true));
        }
    ?>
</h4>
```

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

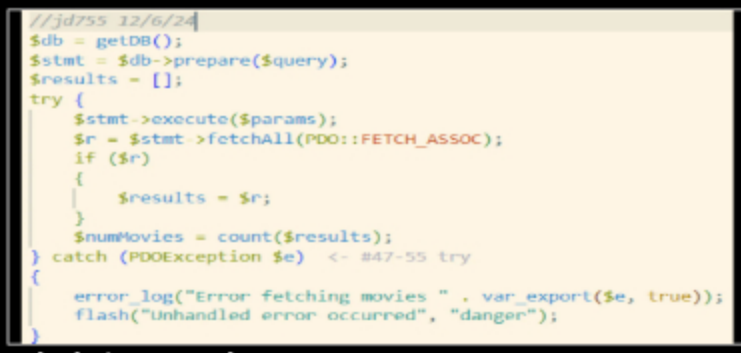
Task #2: Screenshot the code

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

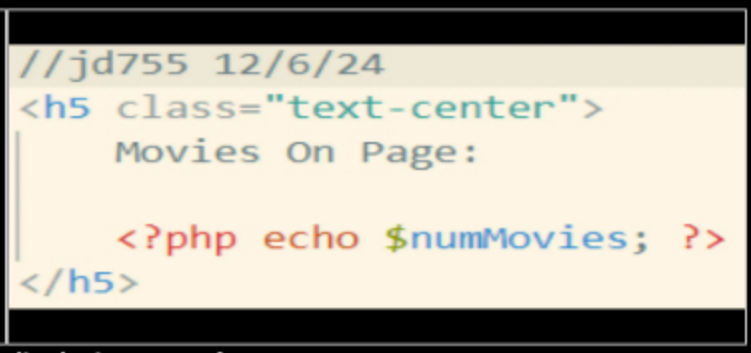
Gallery Style: 2 Columns

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 $e)  <- #47-55 try
{
    error_log("Error fetching movies " . var_export($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:

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

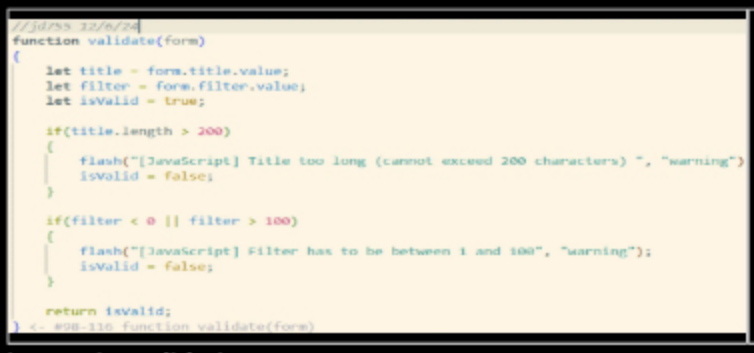
100%

Group: Unassociated Page
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



```
//jd/5% 12/6/24
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;
} <- #98-116 function validate(form)
```

javascript validation

```
//jd/5% 12/6/24
$title = $_GET["title"];
$num=$_GET["filter"];
$params = [];

if(!$num)
{
    $num = 10;
}

if($num < 1 || $num > 100)
{
    flash("[PHP] Filter has to be between 1 and 100", "warning");
    $num = 10;
}
//jd/5% 12/6/24
if(!$title)
{
    $query = "select Movies.Id, Movies.title, Movies.image_url, Movies.release_date
    FROM Movies
    <<<- JOIN UserMovies
    ON Movies.Id = UserMovies.movie_id
    WHERE UserMovies.movie_id IS NULL
    ORDER BY Movies.created DESC LIMIT $num";
}
else
{
    if(strlen($title) > 200)
    {
        flash("[PHP] Title too long (cannot exceed 200 characters) ", "warning");
```

php validation

```
//jd/5% 12/6/24
if(!$title)
{
    $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 $num";
}
else
{
    if(strlen($title) > 200)
    {
        flash("[PHP] Title too long (cannot exceed 200 characters) ", "warning");
        $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 $num";
    }
    else
    {
        $search = sqls_str( "title", "", false);
        $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 AND Movies.title LIKE :title ORD
        $params = ["title" => "%$search%"];
    } <- #97-41 else
```

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
#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/Project/admin/list_not_assoc.php?title=&filter=

URL
https://it202-jd755-prod-028c8689138c.herokuap

URL #2
https://github.com/legitdaylight/jd755-IT202-007/pull/66

URL
https://github.com/legitdaylight/jd755-IT202-007

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%

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

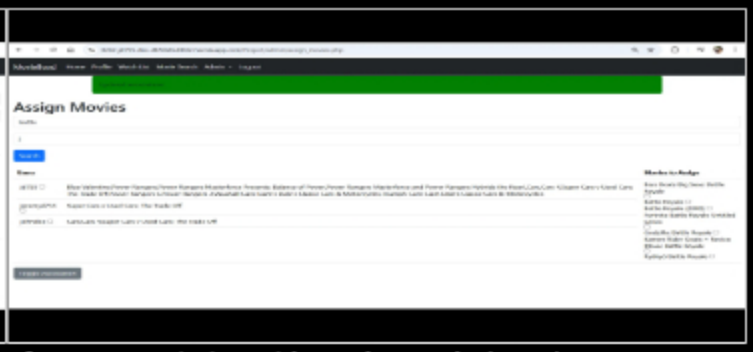
Sub Task #3: Show the result of entities and users being associated and unassociated

Task Screenshots

Gallery Style: 2 Columns

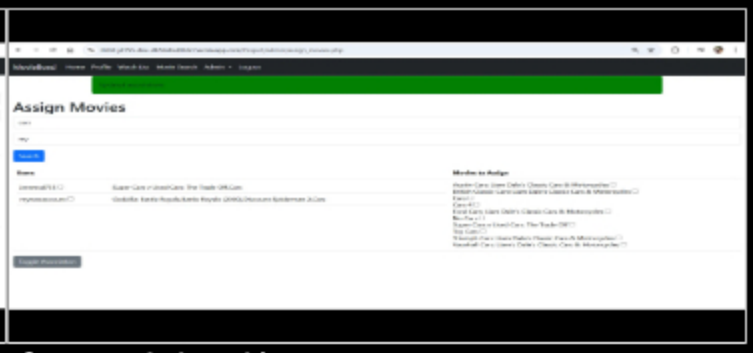
4 2 1

before unassociating with Battle Royale(2000)



after unaccociating 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

Task

100%

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

Columns: 1

Sub-Task

100%

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

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 seleting 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

100%

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

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:

Similary 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

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

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

4 2 1

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 sucessfully 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

100%

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
#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/Project/admin/assign_movies.php

URL

https://it202-jd755-prod-028c8689138c.herokuap

URL #2

https://github.com/legitdaylight/jd755-IT202-007/pull/68

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

100%

Group: Misc
Tasks: 4
Points: 1

COLLAPSE

Task

100%

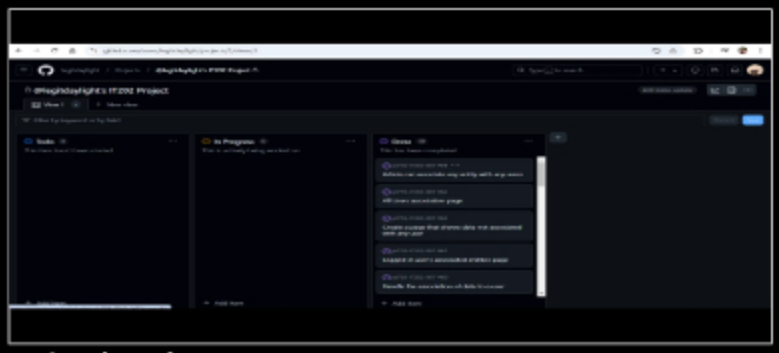
Group: Misc
Task #1: Screenshot of your project board from GitHub (tasks should be in the proper column)
Weight: ~25%
Points: ~0.25

COLLAPSE

Task Screenshots

Gallery Style: 2 Columns

4 2 1



project board

End of Task 1

Task

100%

Group: Misc
Task #2: Provide a direct link to the project board on GitHub
Weight: ~25%
Points: ~0.25

COLLAPSE

Task URLs

URL #1

URL

https://github.com/users/legitdaylight/projects/2/views/1 https://github.com/users/legitdaylight/projects/2

End of Task 2

Task

100%

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 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 of 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

100%

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 1

wakatime part 1

waka time part 2

End of Task 4

End of Group: Misc
Task Status: 4/4

End of Assignment