Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-007-F2024/it202-api-project-milestone-3-2024-m24/grade/mcp62

Course: IT202-007-F2024
Assigment: [IT202] API Project Milestone 3 2024 m24
Student: Michael P. (mcp62)

Submissions:

Submission Selection

1 Submission [submitted] 12/11/2024 3:20:20 PM

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/viev
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:

Concept/Association: The main concept here is managing LinkedIn profiles and their favorite status. There are two main entities: Users (from Users table) LinkedIn Profiles (from LinkedInProfiles table) Relationship Type: This is a One-to-Many relationship where: One User can have Many LinkedIn Profiles Each LinkedIn Profile belongs to One User

Other considerations: Within the Admin watchlist page the admin is given full control over all profiles and data (able to unassociate profiles from other accounts) This relationship would technically be many to many since more than one person can have the admin role, and all the admins have access to all other profiles. Despite this that is not the intended use case. The main relationship is still One-to-Many.

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 (manually or from API): Manual updates through edit_profile.php only update specific fields while preserving other data (like favorite status) API updates through store_profile.php use ON DUPLICATE KEY UPDATE

which updates existing records while maintaining associations Data Visibility and Reassociation: Users immediately see the new data as there's no caching Data does not need to be reassociated because: The unique constraint on (user_id, linkedin_username) ensures updates modify existing records rather than creating new ones Favorite status and user associations are preserved during updates The modified timestamp automatically updates to show when changes occurred If user is deleted data DOES need to be reassociated.

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

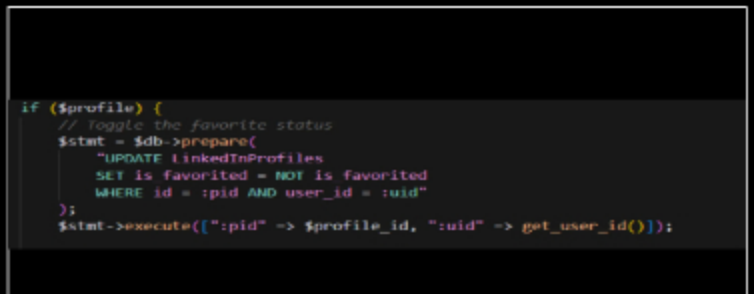
100%

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



```
if ($profile) {
    // Toggle the favorite status
    $stmt = $db->prepare(
        "UPDATE LinkedInProfiles
        SET is_favorited = NOT is_favorited
        WHERE id = :pid AND user_id = :uid"
    );
    $stmt->execute([":pid" => $profile_id, ":uid" => get_user_id()]);
```

Favorite Toggle logic

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:

User clicks favorite button which triggers JavaScript function AJAX request sent to favorite_profile.php Backend verifies: User is logged in Profile exists User owns the profile Toggles favorite status in database Returns success/failure response Frontend updates UI based on response

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

Profiles table

Caption(s) (required) ✓

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

Task Response Prompt

Describe each column/association table

Response:

At the begining of the table their is a User ID which associates each User with one this linkedin Porfiles table, but each user will only see data associated with their profiles ID. The main association column are is_favorited. Each User can associate favorited profiles on their account.

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://github.com/Onervv/mcp62-IT202-007/pull/60

URL
https://github.com/Onervv/mcp62-IT202-007/pul

URL #2
https://mcp62-prod-adbf9d9e4a42.herokuapp.com/Project/linkedin/favorited_profiles.php

URL
https://mcp62-prod-adbf9d9e4a42.herokuapp.co

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

100%

Group: Current User's Association 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

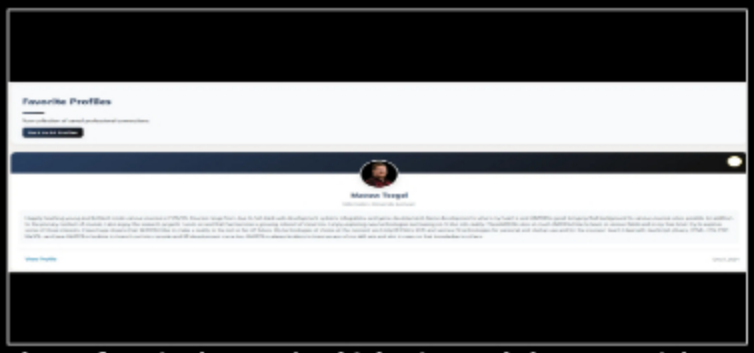
Sub-Task 100%

Group: 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



shows favorited users (multiple I just only have one right now)

Caption(s) (required) ✓

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

Sub-Task 100%

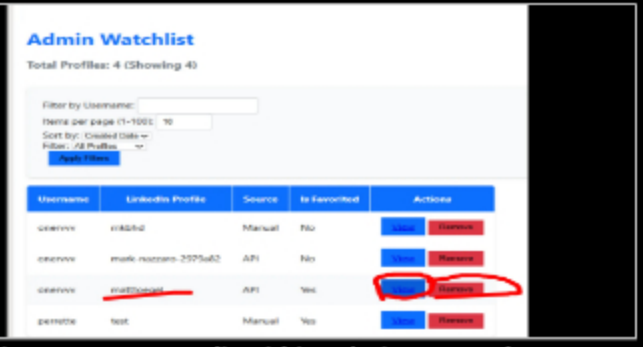
Group: Current User's Association Page
Task #1: Screenshots of this page
Sub Task #2: Show the single view buttons/links, delete button/links, and delete all button/link

Task Screenshots

Gallery Style: 2 Columns

4 2 1

Technically single view



But requirements were split within admin page, where

admin has access to detailed single view and controls

Caption(s) (required) ✓

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

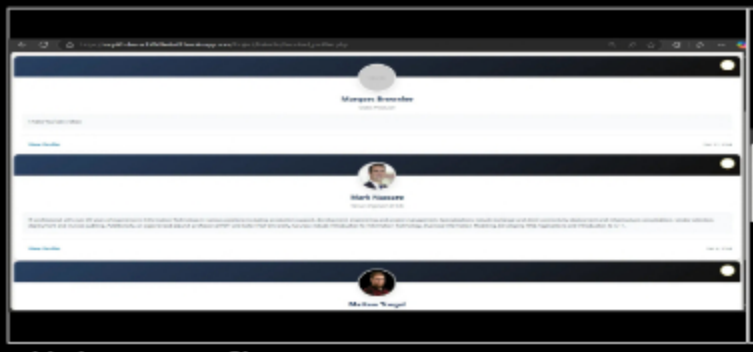
Sub-Task 100%

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



added 2 more profiles

changes reflected in favorites tab notation and admin view watchlist

Caption(s) (required) ✓

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

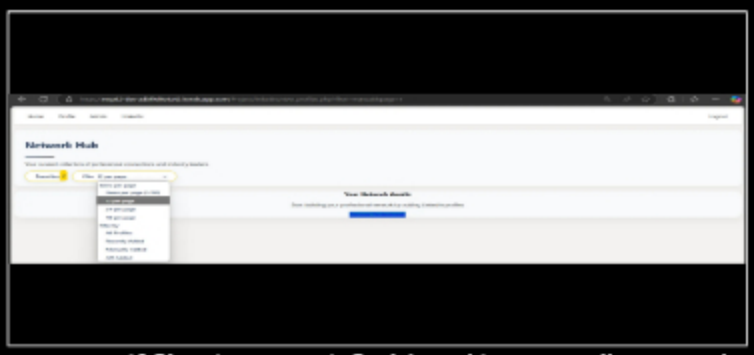
Sub-Task 100%

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



message if filter is not satisfied (used is_manually created to show)

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

```php
// Fetch only favorited profiles for current user
$db = getDB();
$stmt = $db->prepare(
    "SELECT * FROM LinkedInProfiles
    WHERE user_id = :user_id
    AND is_favorited = 1
    ORDER BY created DESC"
); <- #10-15 $stmt = $db->prepare
$stmt->execute([":user_id" => get_user_id()]);
$profiles = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>
```

fetch for user

```php
function is_favorited($profile_id) {
    $db = getDB();
    $stmt = $db->prepare(
        "SELECT 1 FROM UserFavorites
        WHERE user_id = :uid AND profile_id = :pid"
    );
    try {
        $stmt->execute([
            ":uid" => get_user_id(),
            ":pid" => $profile_id
        ]);
        return $stmt->fetchColumn() ? true : false;
    } catch (PDOException $e) {
        error_log(var_export($e->errorInfo, true));
        return false;
    }
} <- #133-149 function is_favorited($profile_id)
?>
```

checking individual favorited status

```php
// Build base query for fetching profiles
$base_query = "SELECT p.*, u.username,
    CASE WHEN p.is_favorited = 1 THEN 'Yes' ELSE 'No' END as is_favorited,
    CASE WHEN p.is_manual = 1 THEN 'Manual' ELSE 'API' END as source
    FROM LinkedInProfiles p
    LEFT JOIN Users u ON p.user_id = u.id";

$params = [];
```

admin view for favorites

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:

The favoriting system works by using a boolean flag (is_favorited) directly in the LinkedInProfiles table. When a user favorites a profile, the system updates this flag using a toggle query that flips the boolean value. The system ensures security by always checking that the user owns the profile through the user_id column before allowing any favorite operations. To display favorites, the system joins the LinkedInProfiles table with the Users table and uses a CASE statement to convert the boolean is_favorited into "Yes" or "No" for display. When filtering for favorites, it adds a WHERE clause checking for is_favorited = 1. The results are then sorted and paginated based on user preferences. This creates a one-to-one relationship where users can only favorite their own profiles, but then again the argument can still be made of other realtionships due to how admin and admin watchlist works

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

```
<?php foreach ($profiles as $profile) : ?>
    <div class="col-md-6 col-lg-4">
        <div class="profile-card">
            <div class="profile-banner">
                <button class="btn favorite-btn active"
                        onclick="toggleFavorite(this, <?php se($profile, 'id'); ?>)"
                        data-favorited="1">
                    <i class="bi bi-star-fill"></i>
```

responsible for outputting each profile and creating toggle Favorite

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

The favorited profiles page displays a user's saved LinkedIn profiles in a responsive grid layout, showing profile details like name, headline, and summary, with options to unfavorite or view full profiles, and includes an empty state message when no favorites exists.

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

this is done in watchlist with view

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

This is done within detailed_view_profile.php which provides more profile information as it is shown from the table itself, This works by preparing and querying the database, then mapping each column of data to the detailed view values.

Sub-Task

100%

Group: Current User's Association Page

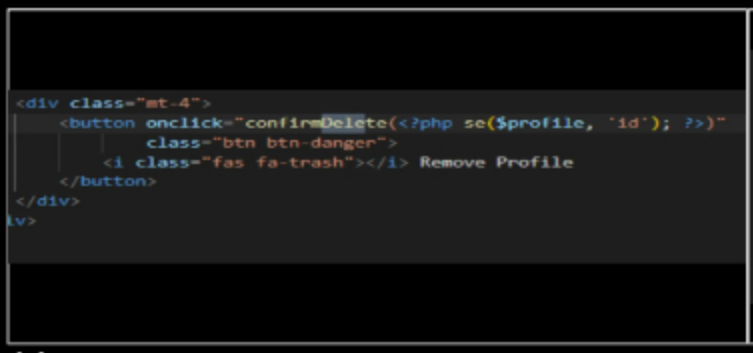
Task #2: Screenshot the code

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

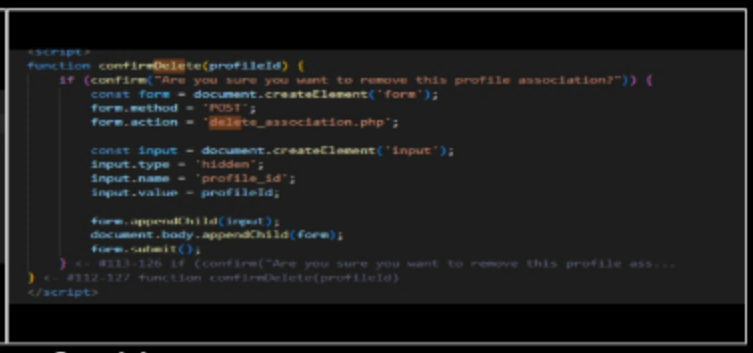
Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
<div class="mt-4">
    <button onclick="confirmDelete(<?php se($profile, 'id'); ?>)"
            class="btn btn-danger">
        <i class="fas fa-trash"></i> Remove Profile
    </button>
</div>
</v>
```



```
<script>
function confirmDelete(profileId) {
    if (confirm("Are you sure you want to remove this profile association?")) {
        const form = document.createElement('form');
        form.method = 'POST';
        form.action = 'delete_association.php';

        const input = document.createElement('input');
        input.type = 'hidden';
        input.name = 'profile_id';
        input.value = profileId;

        form.appendChild(input);
        document.body.appendChild(form);
        form.submit();
    } <- 811-126 if (confirm("Are you sure you want to remove this profile ass...
} <- 812-127 function confirmDelete(profileId)
</script>
```

delete

confirm delete pop up

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

In all use cases of delete, as mentioned in the requirments the user will need to re-associate the data back with them, meaning they will need to recreate the profile. The reasoning for this is due to all delete buttons are set to remove the data entry from the linkein profile table. Removing an entity WITHOUT compeltely removing the table field would be the favorite button as shown before which is associated with each user individually and can be toggled, all a user needs to do in order to get back a favorited profile is just favorite them again.

Sub-Task

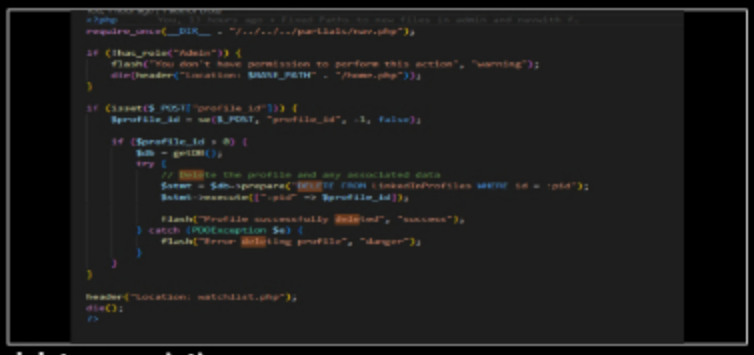
100%

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



delete association

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

Checks to ensure that the user has the admin role, additionally preps the Database by making sure profile_id is > 0. Creates a statment to run and directly deletes based on profile ID

Sub-Task

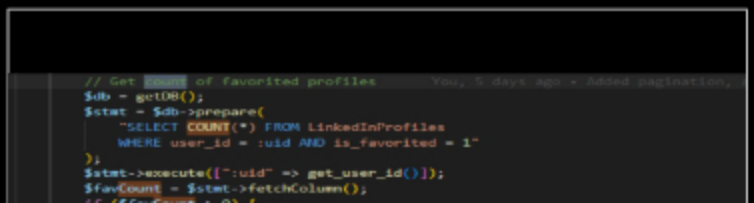
100%

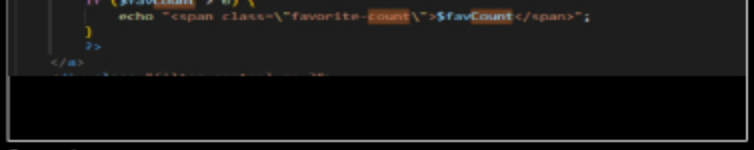
Group: Current User's Association Page
Task #2: Screenshot the code
Sub Task #6: Show the logic related to the count of all associated items to the user (even the ones not shown in the filtered results)

Task Screenshots

Gallery Style: 2 Columns

4 2 1





favorite count

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

Counts how many profiles the current user has favorited by querying the LinkedInProfiles table and displays the count in a span element if it's greater than zero, providing visual feedback about the user's favorite collection size.

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

count based on filter

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

Count of profiles is determined by executing a SQL query that counts the total number of profiles in the LinkedInProfiles table, optionally filtered by username and whether the profile is favorited. The query uses a COUNT(*) function to get the total number of profiles that match the specified criteria. This count is then used to calculate the total number of pages for pagination and is displayed on the page to inform the user of the total number of profiles available based on the current filters.

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

Pagination Logic

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

The watchlist filtering system checks for user defined parameters from the URL, applying default values if none are provided or if they're invalid. It then constructs a SQL query that combines these filters, where the username filter searches for partial matches, the favorite filter shows only favorited profiles, and the results are sorted according to the specified column and order, with pagination applied to limit the number of results shown per page.

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

https://github.com/Onervv/mcp62-IT202-007/pull/49

URL

https://github.com/Onervv/mcp62-IT202-007/pul

URL #2
https://mcp62-prod-adbf9d9e4a42.herokuapp.com/Project/admin/watchlist.php

https://mcp62-prod-adbf9d9e4a42.herokuapp.co

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

Admin watchlist

Caption(s) (required) ✓

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

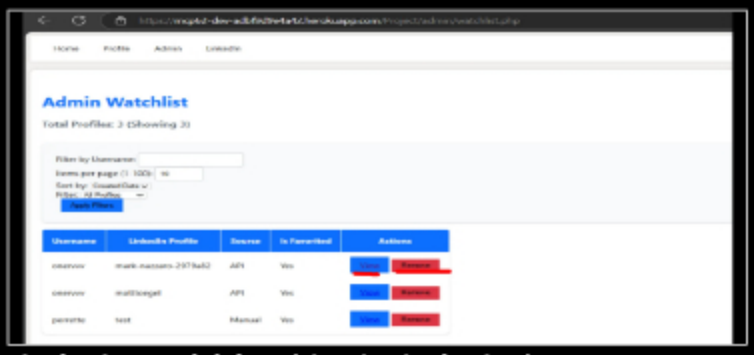
Sub-Task 100%

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



single view and delete (view is single view)

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

Username

clickable

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

different number of shown item count

filter by favorite

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

filter by username to show message

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

base query

main query

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:

It uses CASE statements to format the is_favorited and source fields for display purposes. If a username filter is provided, it appends a WHERE clause to the query to search for profiles with usernames that match the filter. If the "favorited" filter is selected, it further refines the query to include only profiles marked as favorited. The query is extended with an ORDER BY clause to sort the results based on user-selected criteria. It includes LIMIT and OFFSET clauses to implement pagination, ensuring only a subset of results is fetched per page.

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

Displaying information

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:

This form allows users to filter and sort the list of profiles displayed on the page. It includes input fields for filtering by username, setting the number of items per page (with a constraint of 1-100), and selecting sorting criteria (created date, username, or favorited status). Additionally, it provides a dropdown to filter profiles by all or only favorited ones. When the user submits the form, the selected parameters are sent via a GET request, updating the URL and triggering the server-side logic to apply these filters and sorting options to the query that fetches the profiles.

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

for each for view

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

in watchlist.php, the foreach loop iterates over each profile fetched from the database and generates a table row for

each one. Within this loop: View Button: A View button is created for each profile, linking to a detailed view page. The URL includes the profile's ID as a query parameter, allowing the detailed view page to fetch and display the specific profile's information.

Sub-Task

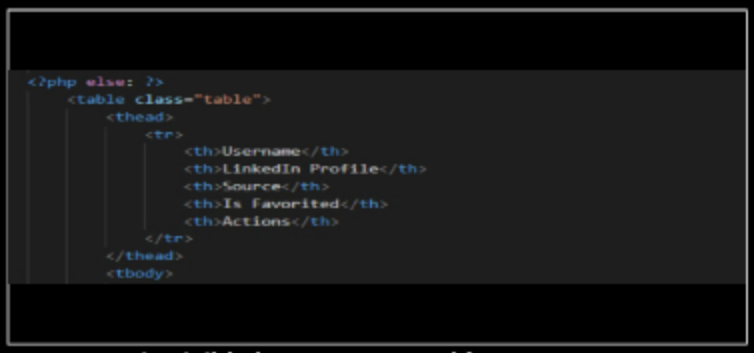
100%

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



```
<?php else: ?>
    <table class="table">
        <thead>
            <tr>
                <th>Username</th>
                <th>LinkedIn Profile</th>
                <th>Source</th>
                <th>Is Favorited</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
```

Username is visible but not traversable

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

User information is shown on admin page but does not send the current user anywhere, the view button mimics this similar feature where it provides a full out look on the specific profile but in the context of my project it doesnt make sense to be able to access other users profiles becasue it would require the user to login with that persons credentials. There is no user-specific data besides the profiles what tthey decided to store. Watchlist.php is a way to combine multiple user data to one place.

Sub-Task

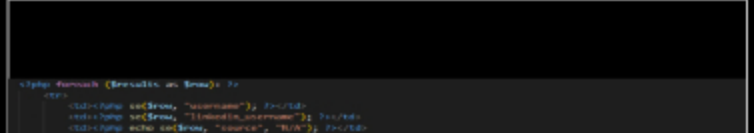
100%

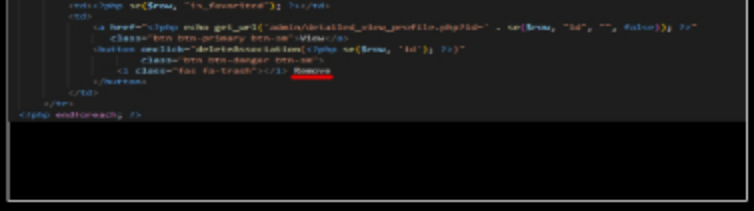
Group: All Users Association Page (likely an admin page)
Task #2: Screenshot the code
Sub Task #5: Each record should have a button/link for delete (this may be an admin-only thing but should be present for the specific role) Note: this is to delete the relationship and not the specific entity

Task Screenshots

Gallery Style: 2 Columns

4 2 1





remove

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

Delete Button: A "Remove" button is also generated for each profile. This button triggers a JavaScript function deleteAssociation() when clicked, passing the profile's ID. The function creates a form dynamically and submits it to a server-side script (delete_association.php) to handle the deletion of the profile association.

Sub-Task

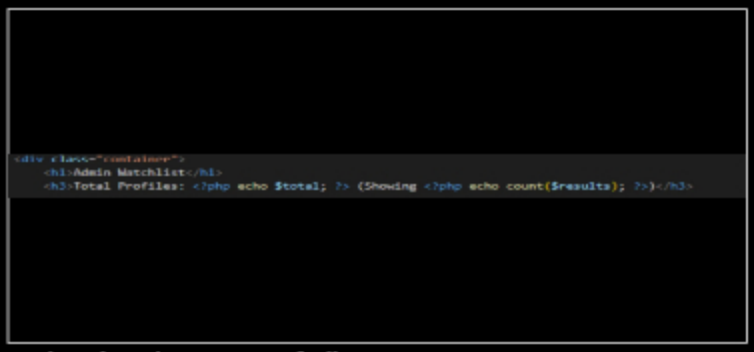
100%

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



code related to count of all content

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

Total count of profiles is calculated based on the filters applied, making sure that the count reflects only those profiles that meet the specified criteria. This logic allows the admin to see how many profiles match their search and filter settings, even if not all are displayed on the current page due to pagination.

The total count of profiles ($total) is dependent on the filters applied. If a username filter is specified, the count query will only count profiles where the username matches the filter criteria.

Favorites.php would be the only file that has a static count

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

filter param

same count code

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

Again, total count of profiles ($total) is dependent on the filters applied.

The $username_filter parameter affects the total count because it's applied to the $count_query When a username filter is provided, only profiles matching that username are included in the total count The $sort and $order parameters don't affect the total count (they only affect how the results are displayed)

Sub-Task

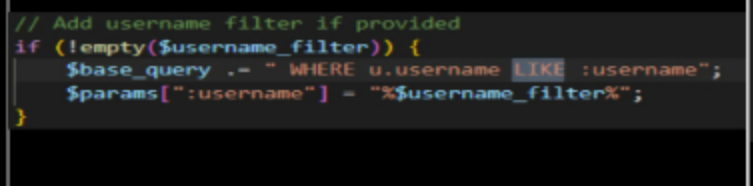
100%

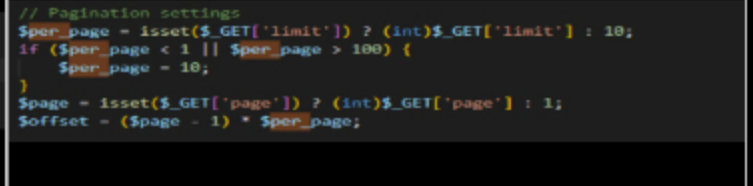
Group: All Users Association Page (likely an admin page)
Task #2: Screenshot the code
Sub Task #8: Show the logic related to filter/sort (should include a partial match for username) (limit should be constrained to 1-100 otherwise default to 10)

Task Screenshots

Gallery Style: 2 Columns

4 2 1





username filter	page view limits and default

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

Task Response Prompt

Explain in concise steps how this logically works
Response:

The partial username filter allows users to search for profiles by entering any part of a username. When a user enters a search term in the username filter input, the application wraps the term with % on both sides. For example, if a user enters "jo", the SQL query uses LIKE '%jo%', which matches usernames containing "jo" anywhere in the string. This filter is applied to both the main query that fetches the profiles and the count query that determines the total number of matching profiles.

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://github.com/Onervv/mcp62-IT202-007/pull/51

URL
https://github.com/Onervv/mcp62-IT202-007/pul

URL #2
https://mcp62-prod-adbf9d9e4a42.herokuapp.com/Project/admin/watchlist.php

URL
https://mcp62-prod-adbf9d9e4a42.herokuapp.co

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.

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

would look similar to entity association page

Caption(s) (required) ✓

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

Sub-Task

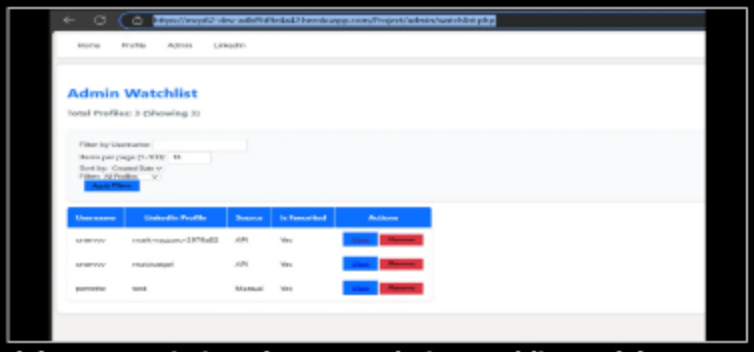
100%

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



delete_association.php uses Admin watchlist to delete profiles

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

closest thing to unassociated to associated count would be favorites, where profile count != favorite count

where a filter too can display no association due to filter but still display number of favorites

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

same "network awaits message" as seen in Admin watchlist

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

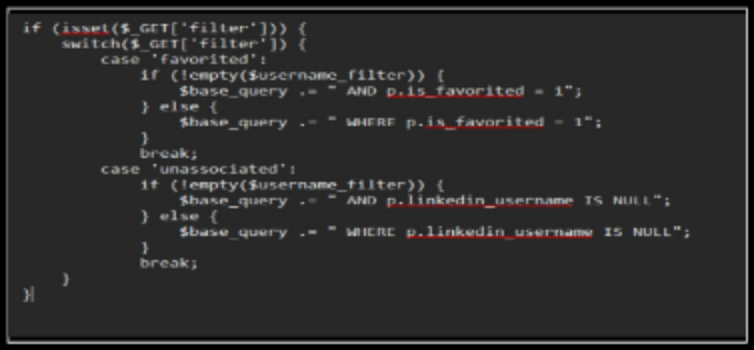
100%

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



```php
if (isset($_GET['filter'])) {
    switch($_GET['filter']) {
        case 'favorited':
            if (!empty($username_filter)) {
                $base_query .= " AND p.is_favorited = 1";
            } else {
                $base_query .= " WHERE p.is_favorited = 1";
            }
            break;
        case 'unassociated':
            if (!empty($username_filter)) {
                $base_query .= " AND p.linkedin_username IS NULL";
            } else {
                $base_query .= " WHERE p.linkedin_username IS NULL";
            }
            break;
    }
}
```

I dont have this feature coded, this is how I would implement it

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 would Add a new filter option for unassociated profiles When selected, it would only show profiles where linkedin_username is NULL The filter would work alongside existing username filters The total count would automatically reflect only unassociated profiles when this filter is active, this is just complicated due to needing to either change the way my delete works project wide or write logic to store profiles even after being deleted.

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

The code would either be fitted in the filter, or it would be its own button that leads to its on view page

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

An unassociation page would display results in a table format similar to the watchlist page, showing information about profiles that don't have LinkedIn connections. The table would include columns for the username, any partial profile data that exists, and an action button to either delete the unassociated profile or attempt to re-associate it with a LinkedIn profile. The page would include similar filtering and pagination features as the watchlist, allowing admins to search through unassociated profiles and limit the number of results per page. The header would show the total count of unassociated profiles and how many are currently being displayed, helping administrators track the scope of unassociated data.

Sub-Task

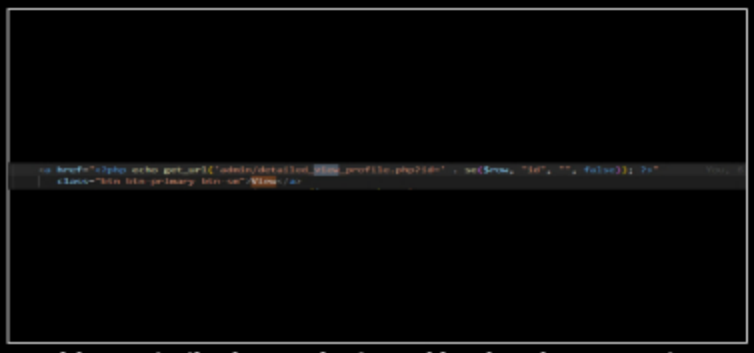
100%

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



would use similar button logic and lead to the same view as watchlist but additionally show unassociated profiles

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

It would work the same as detailed view logically but would need custom logic for potentially being able to reassociate the data back to a user or the original user (this idea would be contingent on if I edited the functionality of every delete button), but would be one way to add delete button because then within the unassociated profile view would be a delete button to remove the profile permanently

Sub-Task

100%

Group: Unassociated Page

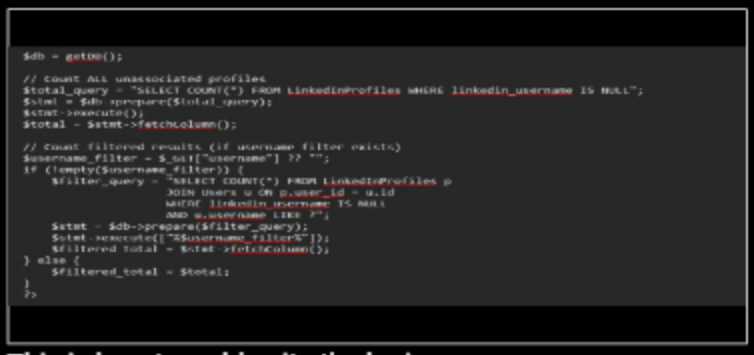
Task #2: Screenshot the code

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



```
$db = getDB();

// Count ALL unassociated profiles
$total_query = "SELECT COUNT(*) FROM LinkedInProfiles WHERE linkedin_username IS NULL";
$stmt = $db->prepare($total_query);
$stmt->execute();
$total = $stmt->fetchColumn();

// Count filtered results (if username filter exists)
$username_filter = $_GET["username"] ?? "";
if (!empty($username_filter)) {
    $filter_query = "SELECT COUNT(*) FROM LinkedInProfiles p
                     JOIN Users u ON p.user_id = u.id
                     WHERE linkedin_username IS NULL
                     AND u.username LIKE ?";
    $stmt = $db->prepare($filter_query);
    $stmt->execute(["%$username_filter%"]);
    $filtered_total = $stmt->fetchColumn();
} else {
    $filtered_total = $total;
}
?>
```

This is how I would write the logic

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Explain in concise steps how this logically works

Response:

The logic for counting unassociated profiles would work in two layers: first, it counts ALL profiles that have no LinkedIn username to get the total number of unassociated profiles in the system. Then, if a filter is applied, it runs a second count query that includes both the unassociated condition AND the filter criteria, giving us the number of profiles that are both unassociated and match the search terms. This allows administrators to see both the overall scope of unassociated profiles and how many match their current search criteria, even if they're only viewing a small subset of the results due to pagination.

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

would be the exact same as this but with UnassociatedLinkedInProfiles and/or WHERE p.user_id IS NULL

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

The current query uses a LEFT JOIN, which keeps ALL LinkedIn profiles and matches them with users where possible. It counts everything, regardless of whether there's a user association or not. To get unassociated profiles specifically, I would modify the query to only count profiles where there is NO user association.

Sub-Task

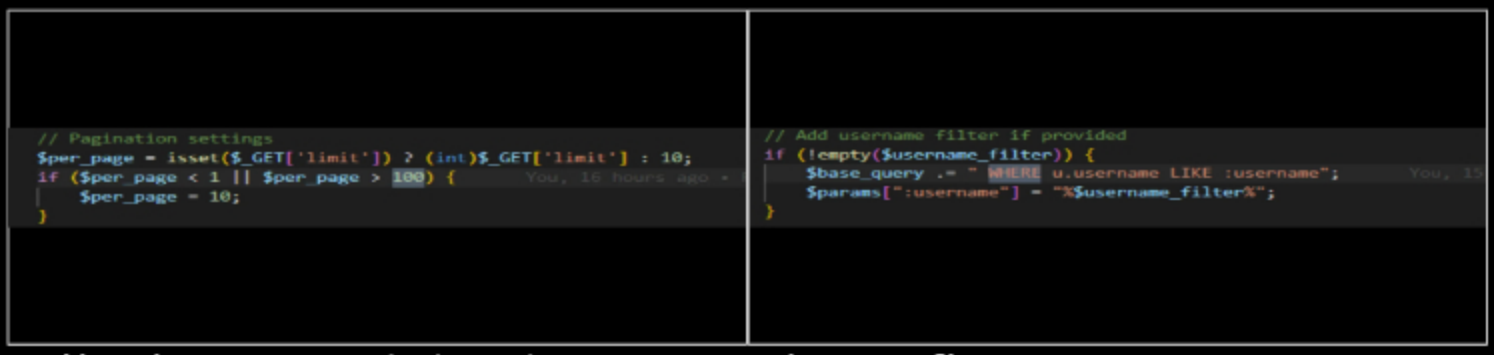
100%

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



Would use the exact same pagination settings and username filter

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

For unassociated profiles, I would need to make sure that the filter only applies to profiles without a user association. The pagination logic would remain the same.

End of Task 2

Task

100%

Group: Unassociated 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://github.com/Onervv/mcp62-IT202-007/pull/51

URL
https://github.com/Onervv/mcp62-IT202-007/pul

URL #2
https://mcp62-prod-adbf9d9e4a42.herokuapp.com/Project/linkedin/view_profiles.php?filter=manual&page=1

URL
https://mcp62-prod-adbf9d9e4a42.herokuapp.co

End of Task 3

End of Group: Unassociated Page

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

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

Association 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

Search results

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

Toggle Assocation added & delete message

Robo chicken is now associated with different account

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

searching

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

Task Response Prompt

Explain in concise steps how this logically works
Response:

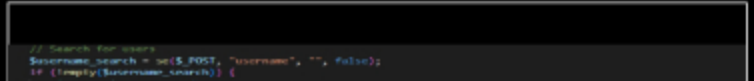
This code works by using SQL's LIKE operator with % wildcards on both sides of the search term (%$search_term%). The % symbols act as wildcards, allowing matches before and after the search term. For example, searching "john" would match "johnny", "johnson", and "bigjohn". This is applied to both LinkedIn profile searches (checking username and names) and user searches (checking usernames).

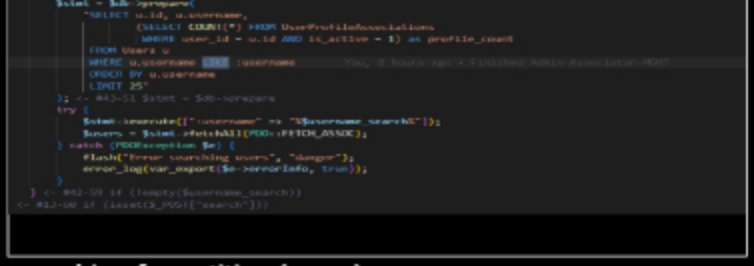
Sub-Task
100%
Group: Admin Association Management (like UserRoles)
Task #2: Screenshots of the code
Sub Task #2: Search form field for finding a partial match of entities

Task Screenshots

Gallery Style: 2 Columns

4 2 1





searching for entities (users)

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

Searches for users whose usernames partially match the search term For each matching user, it also counts their active profile associations using a subquery Results are ordered by username and limited to 25 matches The results are stored in the $users array, which is later used to display the search results in the UI. The code also includes error handling through a try-catch block that will display and log any database errors that occur during the search.

Sub-Task

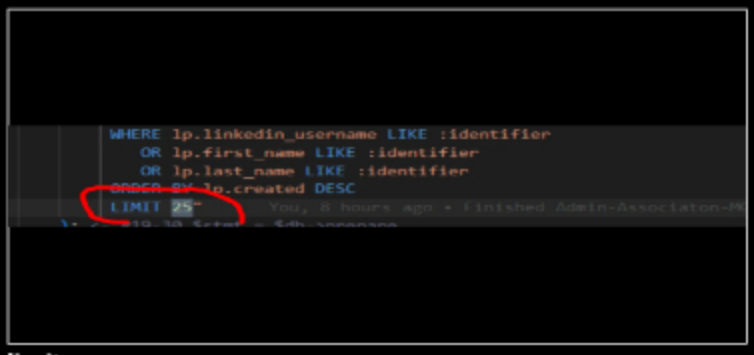
100%

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



limit query

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:

LIMIT 25 in the SQL query acts as a data cap on the result set.

When the database executes the query, it first finds ALL matching records based on the username search criteria It then orders these results alphabetically by username (ORDER BY u.username) The LIMIT 25 clause tells MySQL to

return only the first 25 rows from this ordered set, discarding any additional matches

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

```
<tbody>
    <?php foreach ($profiles as $profile) : ?>
        <tr>
            <td>
                <?php se($profile, "full_name"); ?>
                <small class="text-muted d-block">
                    @<?php se($profile, "linkedin_username"); ?>
                </small>
            </td>
            <td>
                <span class="badge bg-info">
                    <?php se($profile, "association_count"); ?>
                </span>
            </td>
            <td>
                <input type="checkbox"
                       class="form-check-input"
                       name="selected_profiles[]"
                       value="<?php se($profile, 'id'); ?>" />
            </td>
        </tr>
    <?php endforeach; ?>
```

each user code

```
<?php foreach ($users as $user) : ?>
    <tr>
        <td><?php se($user, "username"); ?></td>
        <td>
            <span class="badge bg-info">
                <?php se($user, "profile_count"); ?>
            </span>
        </td>
        <td>
            <input type="checkbox"
                   class="form-check-input"
                   name="selected_users[]"
                   value="<?php se($user, 'id'); ?>" />
        </td>
    </tr>
<?php endforeach; ?>
```

for each checkbox

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

Loops through each user in the $users array from the database query Creates a checkbox input for each user using Bootstrap's form-check-input class Uses selected_users[] as the name (the square brackets allow for multiple selections) Sets the checkbox value to the user's ID When the form is submitted, all checked boxes will be included in the $_POST ['selected_users'] array

Sub-Task

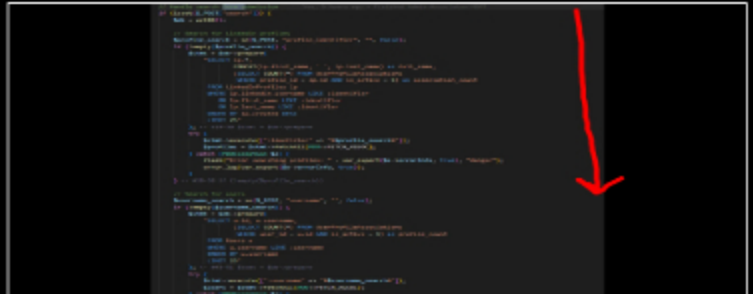
100%

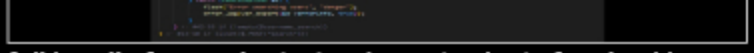
Group: Admin Association Management (like UserRoles)
Task #2: Screenshots of the code
Sub Task #5: Code related to submitting the checkbox lists

Task Screenshots

Gallery Style: 2 Columns

4 2 1





full handle form submission (contains logic for checkboxes to function)

Caption(s) (required) ✓

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

Task Response Prompt

Explain in concise steps how this logically works

Response:

Triggers when the form is submitted with the associate button Retrieves arrays of selected user IDs and profile IDs from POST data Validates that at least one user and one profile are selected Uses a nested loop to process every combination of selected users and profiles For each combination, it executes an UPSERT query that: Inserts a new association if one doesn't exist (is_active = 1) Toggles is_active status if the association already exists Keeps track of successful and failed operations Displays flash messages to inform the user of the results The SQL uses ON DUPLICATE KEY UPDATE to handle both creating new associations and toggling existing ones in a single query.

Sub-Task

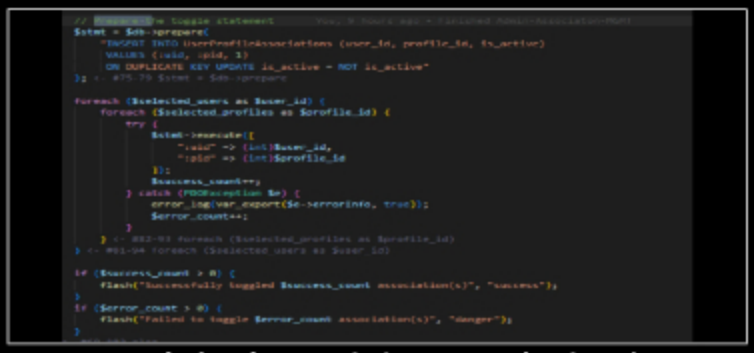
100%

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



prepare toggle (apply association on toggle trigger)

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 association doesn't exist, the INSERT part creates a new record with is_active = 1 If the association already exists, the ON DUPLICATE KEY UPDATE part triggers and flips the is_active value using NOT is_active So new associations are created as active Existing active associations become inactive Existing inactive associations become active I made a nested loops to ensure every selected user is paired with every selected profile Type casting to (int) prevents SQL issues by making sure the IDs are integers

Single query handles both creation and toggling of associations, making it efficient.

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://github.com/Onervv/mcp62-IT202-007/pull/61

URL
https://github.com/Onervv/mcp62-IT202-007/pul

URL #2
https://mcp62-dev-adbf9d9e4a42.herokuapp.com/Project/admin/assign_entities.php

URL
https://mcp62-dev-adbf9d9e4a42.herokuapp.com

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%

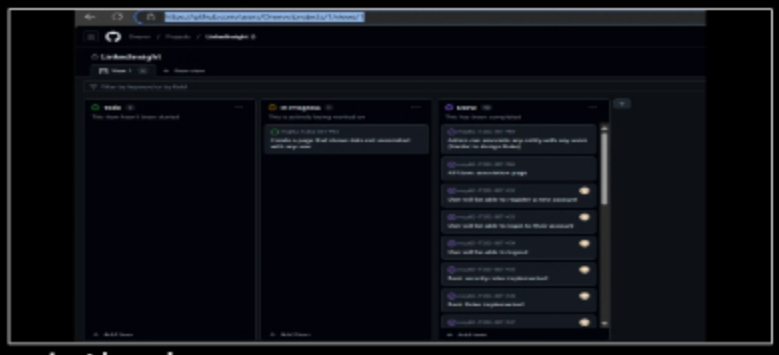
Group: Misc
Task #1: Screenshot of your project board from GitHub (tasks should be in the proper column)
Weight: ~25%
Points: ~0.25

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

URL
https://github.com/users/Onervv/projects/1/view

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:

So the main burn out I had was not fully understanding the power of bootstrap, im using bootsrtap classes for some styling but I mainly wrote custom styling which was ALOT of overcomplicated styling and not needed (like 700 lines of css in this project). I should have used bootraps cdn headers, amd its utilites, and follow a template like my friend do (which ended up coming out looking nicer). Additionally, I also came short on "Create a page that shows data not associated with any user" although I had ideas and some sudo code, the concept never came to light and with time contraints I didnt know how I would effectivly create the feature within my project.

End of Task 3

Task

100%

Group: Misc
Task #4: WakaTime Screenshot
Weight: ~25%
Points: ~0.25

COLLAPSE

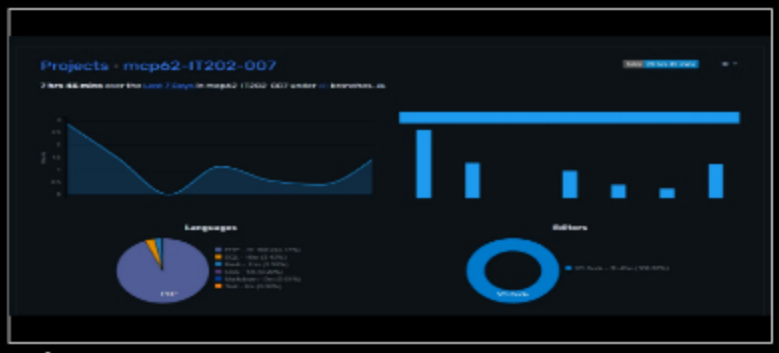
Details:

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



waka

End of Task 4

End of Group: Misc
Task Status: 4/4

End of Assignment