Course: IT202-008-S2025
Assignment: IT202 Milestone 2
Student: Gori H. (gs658)
Status: Submitted | Worksheet Progress: 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Grading Link: https://learn.ethereallab.app/assignment/v3/IT202-008-S2025/it202-milestone-2/grading/gs658

Instructions

1. Refer to Milestone2 of this doc: https://docs.google.com/document/d/1XE96a8DQ52Vp49XACBDTNCq0xYDt3kF29cO88EWVwfo/view
2. Ensure you read all instructions and objectives before starting.
3. Ensure you've gone through each lesson related to this Milestone
4. Switch to the Milestone2 branch
    1. `git checkout Milestone2` (ensure proper starting branch)
    2. `git pull origin Milestone2` (ensure history is up to date)
5. Fill out the below worksheet
    - Ensure there's a comment with your UCID, date, and brief summary of the snippet in each screenshot
    - Ensure proper styling is applied to each page
    - Ensure there are no visible technical errors; only user-friendly messages are allowed
6. Once finished, click "Submit and Export"
7. Locally add the generated PDF to a folder of your choosing inside your repository folder and move it to Github
    1. `git add .`
    2. `git commit -m "adding PDF"`
    3. `git push origin Milestone2`
    4. On Github merge the pull request from `Milestone2` to `dev`
    5. On Github create a pull request from `dev` to `prod` and immediately merge. (This will trigger the `prod` deploy to make the heroku prod links work)
8. Upload the same PDF to Canvas
9. Sync Local
10. `git checkout dev`
11. `git pull origin dev`

100%

Section #1: (2 pts.) Data

100%

Task #1 (0.50 pts.) - API Data Table

Combo Task:

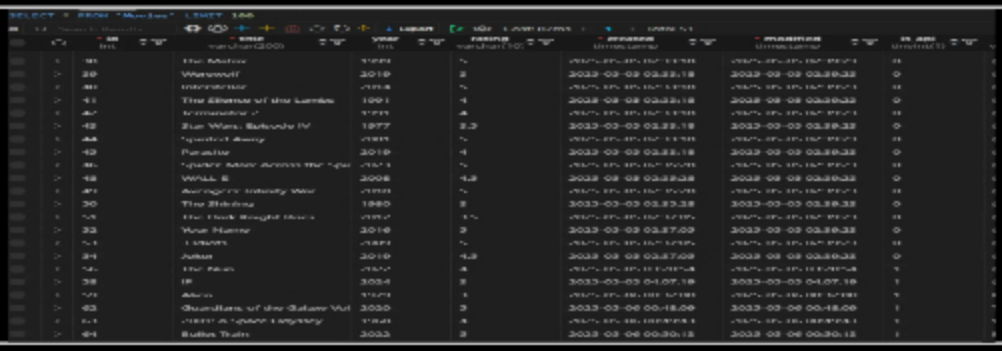
Weight: *25%*
Objective: *API Data Table*

Image Prompt

Weight: *34%*

Details:

- Show the table(s) structure of the table made to hold your API
- There should be at least 3 properties/fields beyond `id`, `created`, `modified`, and respective `is_api` or `api_id` columns
- Logical types and constraints should be set
- Show at least a few records from each API table (samples should include both API fetched data and custom entries noted by an `api_id` or `is_api column)



movie table structure

Saved: 5/7/2025 11:35:21 PM

Url Prompt

Weight: *33%*

Details:

- Add a direct link to your sql file from Github (should end in `.sql`)

URL #1
https://github.com/goribanu/gs658-IT202-3008/Milestone2/public_html/project/sql/006_create_table_movies.sql

https://github.com/goribanu/gs658

Saved: 5/7/2025 11:35:21 PM

Text Prompt

Weight: *33%*

Details:

- Briefly note what each field/property represents

Your Response:

Id is a unique identifier for each movie (primary key), title is the movie's name, year is when the movie was released, rating is rating of the movie (out of 5), is_api indicates whether the movie data came from the API (1) or was added manually (0), and plot gives a brief description of what the movie is about.

Saved: 5/7/2025 11:35:21 PM

100%

Task #2 (0.50 pts.) - Data Transformation

Combo Task:

Weight: *25%*
Objective: *Data Transformation*

Image Prompt

Weight: *40%*

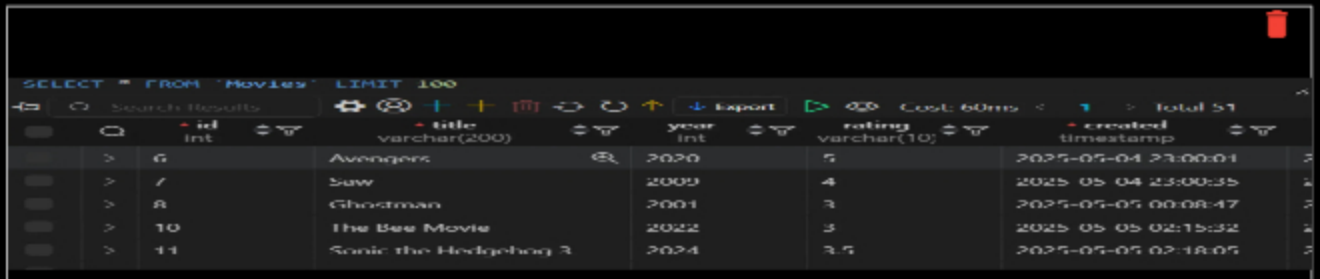
Details:

- Show the code that handles the data transformation/conversion (from the API)
- Show a Heroku dev example of API getting fetched and loaded into the DB

```
        error_log("OMDB Response: " . var_export($results, true));
    /* Handles data transformation, checks if the API call was successful.
    It decodes JSON string into a PHP associative array using json_decode
    and safely pulls the "Plot" field, else says not available. This data
    is then used to insert into Movies table.*/
        if (isset($results, "status", 400, false) == 200 && isset($results["response"])) {
            $results = json_decode($results["response"], true);
            $is_api = 1;
            $plot = $results["Plot"] ?? "Plot not available";
        } else {
            $results = [];
        }
    }
```

code that handles data transformation/conversion from API

api gets fetched



loaded into db

Saved: 5/7/2025 11:59:28 PM

Url Prompt

Weight: *20%*

Details:

- Include Heroku prod link to API fetch page
- Include pull request link for this feature

URL #1
https://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/movie-search.php

URL
https://gs658-it202-008-prod-9640

Saved: 5/7/2025 11:59:28 PM

Text Prompt

Weight: *40%*

Details:

- Will you be using all the data or just a subset?
- Do you need to convert any data or extract pieces of a property's value?
- Briefly explain the transformation/extraction logic to get the API data to the shape your application intends to use

Your Response:

I will be using just a subset of the data from my API. I will be using the movie title, year, and plot. I didn't need to convert too much data, I did extract specific fields from the JSON response to display onto my pages. The JSON response was decoded and looped through the Search array.

Saved: 5/7/2025 11:59:28 PM

100%

Task #3 (0.50 pts.) - API Duplicates

Text Prompt

Weight: *25%*
Objective: *API Duplicates*

Details:

- Consider how duplicate entries are handled
  - What happens if you get the same result from the API for an entity that exists but has been unmodified?
  - What happens if you get the same result from the API for an entity that exists but has been modified?

Your Response:

If I get a duplicate movie from the API that already exists and hasn't been modified, my website prevents adding it to avoid any duplicate entities. If the API returns a version that has been modified (ex. a different title or rating), it is added as a new entity.

Saved: 5/8/2025 12:02:51 AM

100%

Task #4 (0.50 pts.) - API Fetching

Text Prompt

Weight: *25%*
Objective: *API Fetching*

Details:

- Consider how your application will trigger this data fetch
  - (i.e., periodic via cron job, poor man's cron job, on app start/awake, etc)
  - (i.e., User or Admin triggered via a specific page/action)
  - (i.e., Lazy loaded during searches for stale/missing content)
  - What happens if you get the same result from the API for an entity that exists but has been modified?

Your Response:

My application will trigger this fetch data when a user searches for a specific movie title manually on the search page. This means it's a form of lazy loading since data is pulled only when needed. If the API returns a modified version of a movie that already exists, it will be added again.

Saved: 5/8/2025 12:06:17 AM

Section #2: (1.5 pts.) Data Creation

Task #1 (0.60 pts.) - Validations

Combo Task:

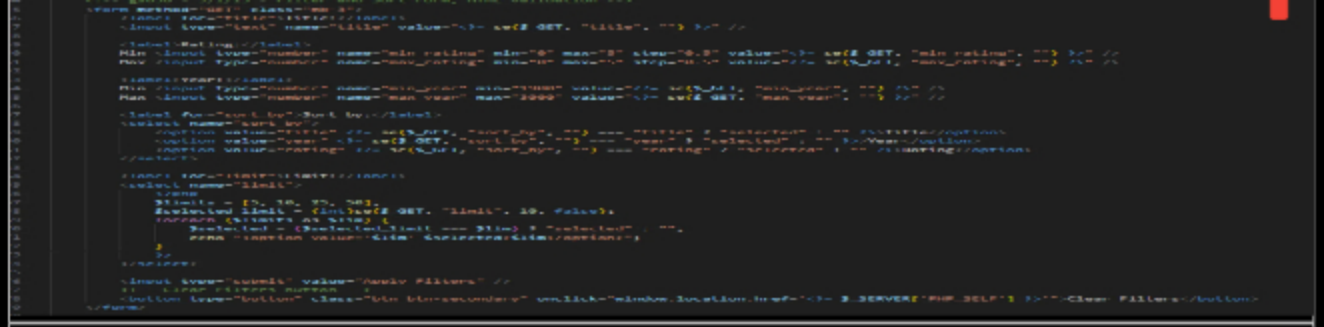
Weight: *40%*
Objective: *Validations*

Image Prompt

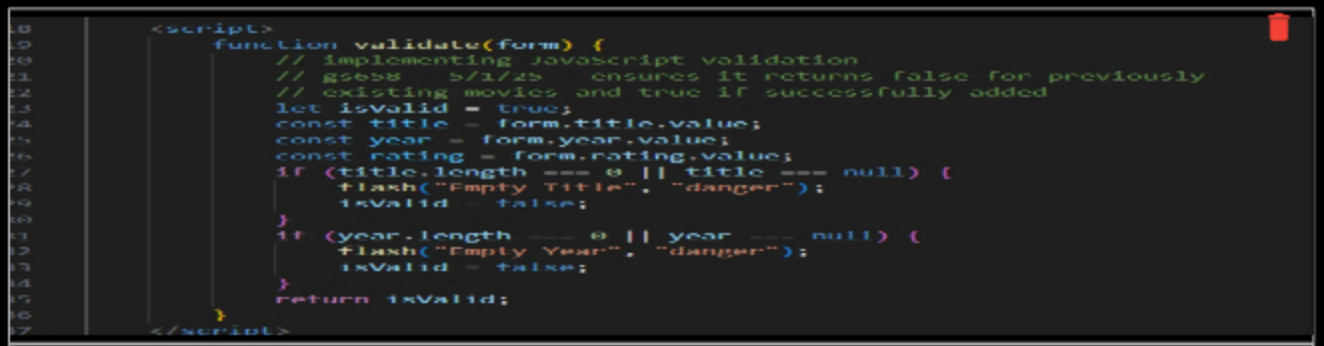
Weight: *50%*

Details:

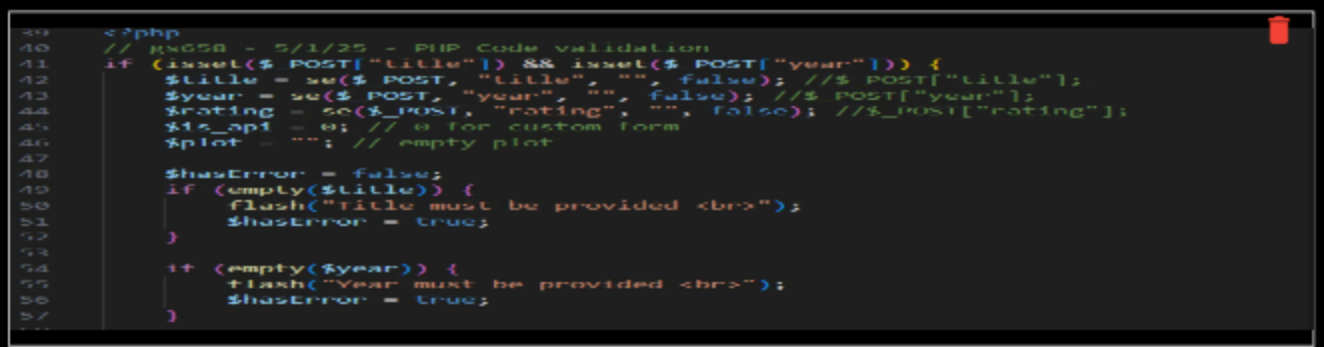
- Show the html validations (code)
- Show the javascript validations (code)
- Show the php validations (code)



showing HTML validation



showing JS validation



showing php validation

Saved: 5/8/2025 12:23:55 AM

Text Prompt

Weight: *50%*

Details:

- Briefly explain the html validations used
- Briefly explain the javascript validation logic
- Briefly explain the php validation logic

Your Response:

HTML - I used basic HTML validations like setting type="number" for numeric inputs. I added min, max, and step to make sure the movie ratings and years stay within a valid range.

Javascript - The validate() function checks if the title and year fields are empty. If they are, it flashes an error & prevents the form from submitting. It also ensures it returns false for previously existing movies and true if successfully added.

PHP - The validation code checks if the required fields like movie title and year are there (not empty). This makes sure that invalid data doesn't get into the database.

100%

Task #2 (0.60 pts.) - Examples

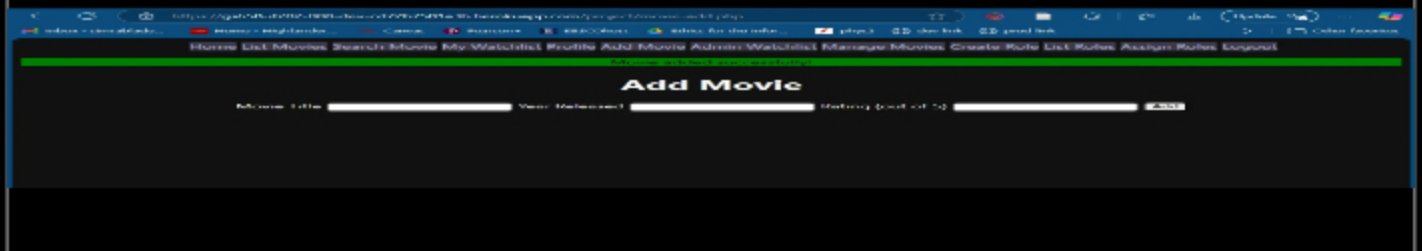
Image Prompt

Weight: *40%*
Objective: *Examples*

Details:

- Show a successful creation
- Demonstrate various validation errors
- Ensure heroku url is visible

adding movie



successfully added movie

100%

Task #3 (0.30 pts.) - Links

Url Prompt

Weight: *20%*
Objective: *Links*

Details:

- Include the heroku prod link to this page
- Include pull request link for this feature

URL #1
https://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/movie-add.php

URL
https://gs658-it202-008-prod-9640930

Saved: 5/8/2025 12:28:32 AM

100%

Section #3: (1.5 pts.) Data List Page (Many Items)

100%

Task #1 (0.60 pts.) - Code

Combo Task:

Weight: *40%*
Objective: *Code*

Image Prompt

Weight: *50%*

Weight: 50%

Details:

- Show the code that handles the filter/sort logic
- Show the code that generates the list/grid output
- Styling must be applied

code that handles filter/sort logic

code that generates list/grid output

```css
/* gs658 - 5/1/24 - Applies styling for filter/sort pages */
input {
    border-radius: 5px;
}

body {
    text-align: center;
    color: white;
    background-color: #121212;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}
```

styling for filter/sort

```css
/* gs658    5/1/25    Applies styling for movie list table */
table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
    background-color: #1a1a1a;
    color: #fff;
}

table th,
table td {
    padding: 12px 15px;
    border: 1px solid #444;
    text-align: left;
}
```

styling for movie list/grid layout

Saved: 5/8/2025 12:42:40 AM

Text Prompt

Weight: *50%*

Details:

- Briefly explain how the filter/sort logic works and formulates the results
- Briefly explain how the list/grid output is generated
- Briefly note the styling choices

100%

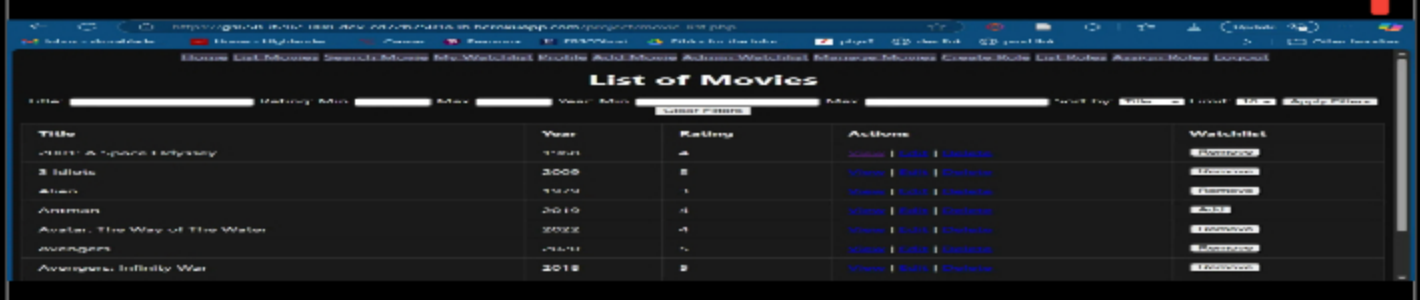
Task #2 (0.60 pts.) - Examples

Image Prompt

Weight: *40%*
Objective: *Examples*

Details:

- Show a few variations of filter/sort applied (including no records)
- Each list item should be a summary
- Each list item should have the following links
  - A link to a single view of the specific entity (i.e., a details page)
  - A link to delete this entity (this may be an admin-only functionality, but it should be present for the respective role)
  - A link to edit this entity (this may be an admin-only functionality, but it should be present for the respective role)
- Ensure heroku url is visible



main page with links to view, edit, and delete pages

rating min 4 max 5 filter applied, limit 10 movies

100%

Task #3 (0.30 pts.) - Links

Url Prompt

Weight: *20%*
Objective: *Links*

Details:

- Include the heroku prod link to this page
- Include pull request link for this feature

URL #1
https://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/movie-list.php

URL
https://gs658-it202-008-prod-9640930

Saved: 5/8/2025 12:42:59 AM

100%

Section #4: (1.5 pts.) View Details Page (Single Item)

Task #1 (0.60 pts.) - Code

Combo Task:

Weight: *40%*
Objective: *Code*

Image Prompt

Weight: *50%*

Details:

- Show the code that handles loading of the record (and how the id retrieved for usage)
  - Include the code that handles invalid/missing ids
- Show the code that generates the entity output
- Styling must be applied

```php
    // Check if the movie id is provided
    $movie_id = isset($_GET['id']) ? (int)$_GET['id'] : null;
    if (!$movie_id) {
        flash("Movie ID is required", "danger");
        redirect("movie-list.php");
        exit;
    }

    $db = getDB();
    $query = "SELECT * FROM Movies WHERE id = :id";
    $params = [":id" => $movie_id];

    // Prepare and execute the query
    $stmt = $db->prepare($query);
    $stmt->bindValue(":id", $movie_id, PDO::PARAM_INT);
```

code that handles loading of the record/how the id is retrieved for usage

```php
try {
    $stmt->execute();
    $movie = $stmt->fetch(PDO::FETCH_ASSOC);
    // handles invalid / missing IDs
    if (!$movie) {
        flash("Movie not found", "danger");
        redirect("movie-list.php");
        exit;
    }
} catch (Exception $e) {
    flash("Error fetching movie details: " . $e->getMessage(), "danger");
    redirect("movie-list.php");
    exit;
}
?>
```

code handling problematic ids

```php
<div class="container-fluid">
    <h1>Movie Details</h1>
    <!-- generated movie output -->
    <p><strong>Title:</strong> <?= htmlspecialchars($movie['title']) ?></p>
    <p><strong>Year:</strong> <?= htmlspecialchars($movie['year']) ?></p>
    <p><strong>Rating:</strong> <?= htmlspecialchars($movie['rating']) ?></p>
    <p><strong>Plot:</strong> <?= htmlspecialchars($movie['plot'] ?? 'Plot not available') ?></p>

    <!-- Action buttons to go back -->
    <a href="movie-list.php" class="btn btn-secondary">Back to Movie List</a>
</div>
```

code that generates movie outputs

```
body {
    text-align: center;
    color: white;
    background color: #121212;
    font family: 'segoe UI', tahoma, geneva, verdana, sans serif;
}
```

css styling

```
p {
    margin-top: 10px;
    color: #ccc;
}
```

more css styling

Saved: 5/8/2025 1:02:53 AM

100%

Task #2 (0.60 pts.) - Examples

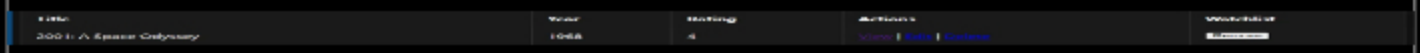
Image Prompt

Weight: *40%*
Objective: *Examples*

Details:

- Show a few examples of different entities
- Each entity item should have more details than the summary view (if possible)
- Each entity should have the following links
  - A link to edit the single entity (this may be an admin-only thing, but it should be present for the respective role)
  - A link to delete the single entity (this may be an admin-only thing, but it should be present for the respective role)

entity #1 view page



entity 1 view/edit/delete page

entity #2 view page

100%

Task #3 (0.30 pts.) - Links

Url Prompt

Weight: *20%*
Objective: *Links*

Details:

- Include the heroku prod link to this page

- Include the heroku prod link to this page
- Include pull request link for this feature

URL #1
https://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/movie-view.php

URL
https://gs658-it202-008-prod-9640930

Saved: 5/8/2025 1:04:12 AM

Section #5: (1.5 pts.) Edit Page

Task #1 (0.60 pts.) - Code

Combo Task:

Weight: *40%*
Objective: *Code*

Image Prompt

Weight: *50%*

Details:

- Show the code that handles loading of the record (and how the id retrieved for usage)
  - Include the code that handles invalid/missing ids
- Show the code that generates the form with the correct data types and populates it
- Show the html, javascript, and php validation (should be similar to create)
- Styling must be applied

```php
// check if the ID parameter is provided
if (!isset($_GET['id'])) {
    flash("Movie ID is required.", "danger");
    redirect("movies_list.php"); // Redirect to the movie list page if ID is n
    die();
}

$movie_id = $_GET['id'];

$db = getDB();

// Fetch the current movie data from the database
$query = "SELECT * FROM Movies WHERE id = :id LIMIT 1";
$stmt = $db->prepare($query);
$stmt->bindValue(':id', $movie_id, PDO::PARAM_INT);
$stmt->execute();
$movie = $stmt->fetch(PDO::FETCH_ASSOC);

if (!$movie) {
    flash("Movie not found.", "danger");
    redirect("movies_list.php"); // Redirect if movie is not found
    die();
}
```

code that handles loading of the record/how the

code that handles loading of the record/how the

```php
    // checks invalid / missing ids
    if (!$movie) {
        flash("Movie not found.", "danger");
        redirect("movies_list.php"); // Redirect if movie is not found
        die();
    }
```

checks for invalid/missing ID

code that generates the form with the correct data types & populates it

HTML validation

```php
    // Handling form submission for updating the movie details
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $title = trim($_POST['title']);
        $year = intval($_POST['year']);
        $rating = floatval($_POST['rating']);

        // Validate inputs
        if (empty($title) || empty($year) || empty($rating)) {
            flash("Title, Year, and Rating are required fields.", "danger");
        } else {
            // Update the movie in the database
            $update_query = "UPDATE Movies SET title = :title, year = :year, ratin
            $update_stmt = $db->prepare($update_query);
            $update_stmt->bindValue(':title', $title, PDO::PARAM_STR);
            $update_stmt->bindValue(':year', $year, PDO::PARAM_INT);
            $update_stmt->bindValue(':rating', $rating, PDO::PARAM_STR);
            $update_stmt->bindValue(':id', $movie_id, PDO::PARAM_INT);

            try {
```

PHP validation

```
body {
    text-align: center;
    color: white;
    background-color: #121212;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}
```

css styling

100%

Task #2 (0.60 pts.) - Examples

Image Prompt

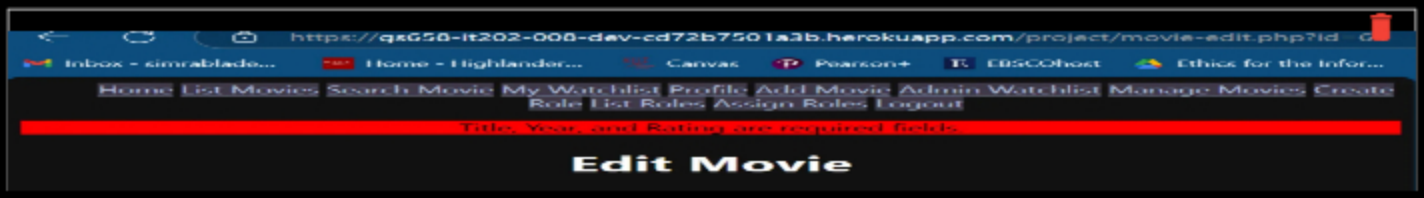
Weight: *40%*
Objective: *Examples*

Details:

- Show a before and after of updating an entity (clearly caption)
- Successful update should have an appropriate message shown
- The updated data should be shown in the form
- Any errors should have user-friendly messages shown
- Show some examples of validation errors (html, javascript, php)

antman view page before

antman view page after with successful message



100%

Task #3 (0.30 pts.) - Links

Url Prompt

Weight: *20%*
Objective: *Links*

Details:

- Include the heroku prod link to this page
- Include pull request link for this feature

URL #1
https://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/movie-edit.php

URL
https://gs658-it202-008-prod-9640930

Saved: 5/8/2025 1:04:26 AM

100%

Section #6: (1.5 pts.) Delete Logic

100%

Task #1 (0.60 pts.) - Code

Combo Task:

Weight: *40%*
Objective: *Code*

Image Prompt

Weight: *50%*

Details:

- Show the code that handles deletion of the record (and how the id retrieved for usage)
  - Include the code that handles invalid/missing ids
- Handle any necessary roles/permissions (i.e., it's not likely that anyone can delete any entity they choose)
- Show the redirect logic that goes back to the previous page (where the route came from), if it was a list page, it should maintain the filter/sort criteria

```php
$movie_id = $_GET['id']; // retrieves the movie ID from the query string

$db = getDB();

// Fetch the movie data to ensure it exists
$query = "SELECT * FROM Movies WHERE id = :id LIMIT 1";
$stmt = $db->prepare($query);
$stmt->bindValue(':id', $movie_id, PDO::PARAM_INT);
$stmt->execute();
$movie = $stmt->fetch(PDO::FETCH_ASSOC);

if (!$movie) {
    flash("Movie not found.", "danger");
    redirect("movie-list.php"); // Redirect if movie is not found
    die();
}

// Delete the movie from the database
$query = "DELETE FROM Movies WHERE id = :id";
$stmt = $db->prepare($query);
$stmt->bindValue(':id', $movie_id, PDO::PARAM_INT);
```

code that handles deletion of the record/how the id is retrieved for usage

```php
// Checks if the ID parameter is provided
if (!isset($_GET['id'])) {
    flash("Movie ID is required.", "danger");
    redirect("movie-list.php"); // Redirect to the movie list page if ID is m
    die();
}

$movie_id = $_GET['id']; // retrieves the movie ID from the query string

$db = getDB();

// Fetch the movie data to ensure it exists
$query = "SELECT * FROM Movies WHERE id = :id LIMIT 1";
$stmt = $db->prepare($query);
$stmt->bindValue(':id', $movie_id, PDO::PARAM_INT);
$stmt->execute();
$movie = $stmt->fetch(PDO::FETCH_ASSOC);

if (!$movie) {
    flash("Movie not found.", "danger");
    redirect("movie-list.php"); // Redirect if movie is not found
    die();
}
```

code that handles invalid/missing ids

```php
try {
    $stmt->execute();
    flash("Movie deleted successfully.", "success");
    redirect("movie-list.php"); // Redirect to the movie list after successful
} catch (Exception $e) {
    flash("Error deleting movie: " . $e->getMessage(), "danger");
    redirect("movie_list.php"); // Redirect back to the movie list on error
}
?>
```

redirect logic that goes back to the previous page

Text Prompt

Weight: *50%*

Details:

- Briefly explain how the deletion logic works
- Is it a soft or hard delete?
- Note any permissions/rules applied.
- Briefly explain how the redirect maintains the previous route and any filter/sort data

Your Response:

100%

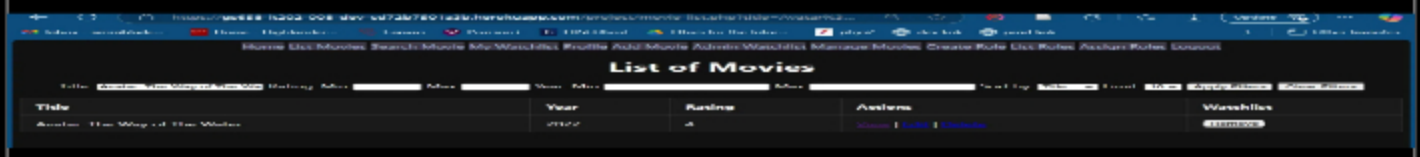
Task #2 (0.60 pts.) - Examples

Image Prompt

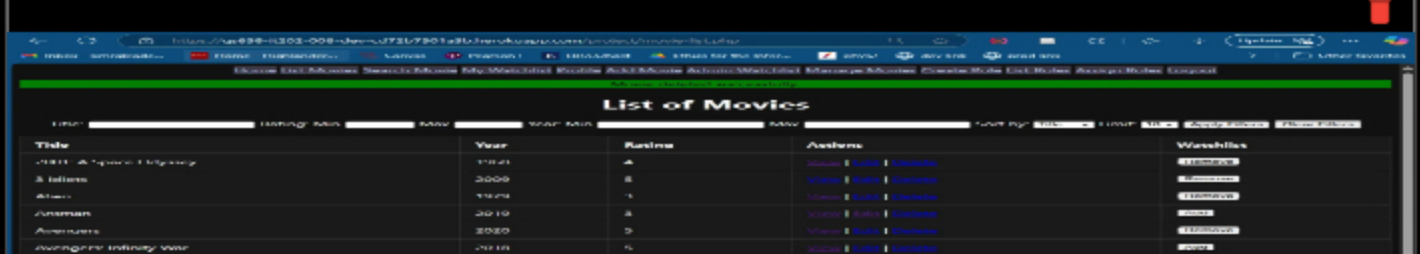
Weight: *40%*
Objective: *Examples*

Details:

- Show a successful delete message
- Show a before and after database view of the record being deleted (clearly caption)
- Successful update should have an appropriate message shown
- The updated data should be shown in the form
- Any errors should have user-friendly messages shown



movie list page view before avatar was deleted



100%

Task #3 (0.30 pts.) - Links

Url Prompt

Weight: *20%*
Objective: *Links*

Details:

- Include the heroku prod link to this page (even though it's not a full page, it's more like how logout works)
- Include pull request link for this feature

URL #1
https://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/movie-delete.php

URL
https://gs658-it202-008-prod-9640930

Saved: 5/8/2025 1:04:33 AM

100%

Section #7: (0.5 pts.) Misc

100%

Task #1 (0.17 pts.) - Github Details

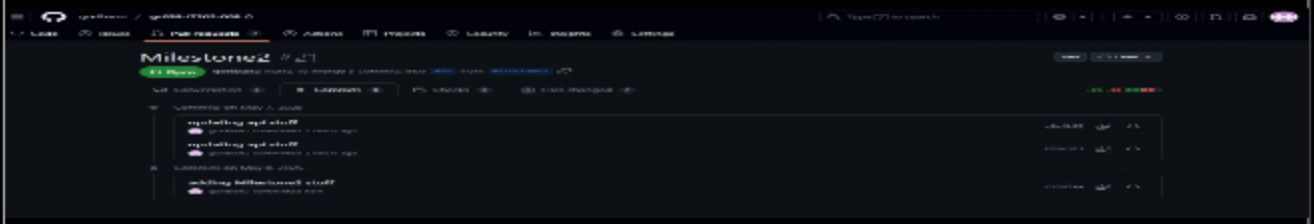
Combo Task:

Weight: *33.33%*
Objective: *Github Details*

Image Prompt

Weight: *60%*

Details:
From the Commits tab of the Pull Request screenshot the commit history



screenshot of commit history

Saved: 5/8/2025 1:49:15 AM

Url Prompt

Weight: *40%*

Details:
Include the link to the Pull Request (should end in `/pull/#`)

URL #1
https://github.com/goribanu/gs658-IT202-001/1

URL
https://github.com/goribanu/gs658

Saved: 5/8/2025 1:49:15 AM

100%

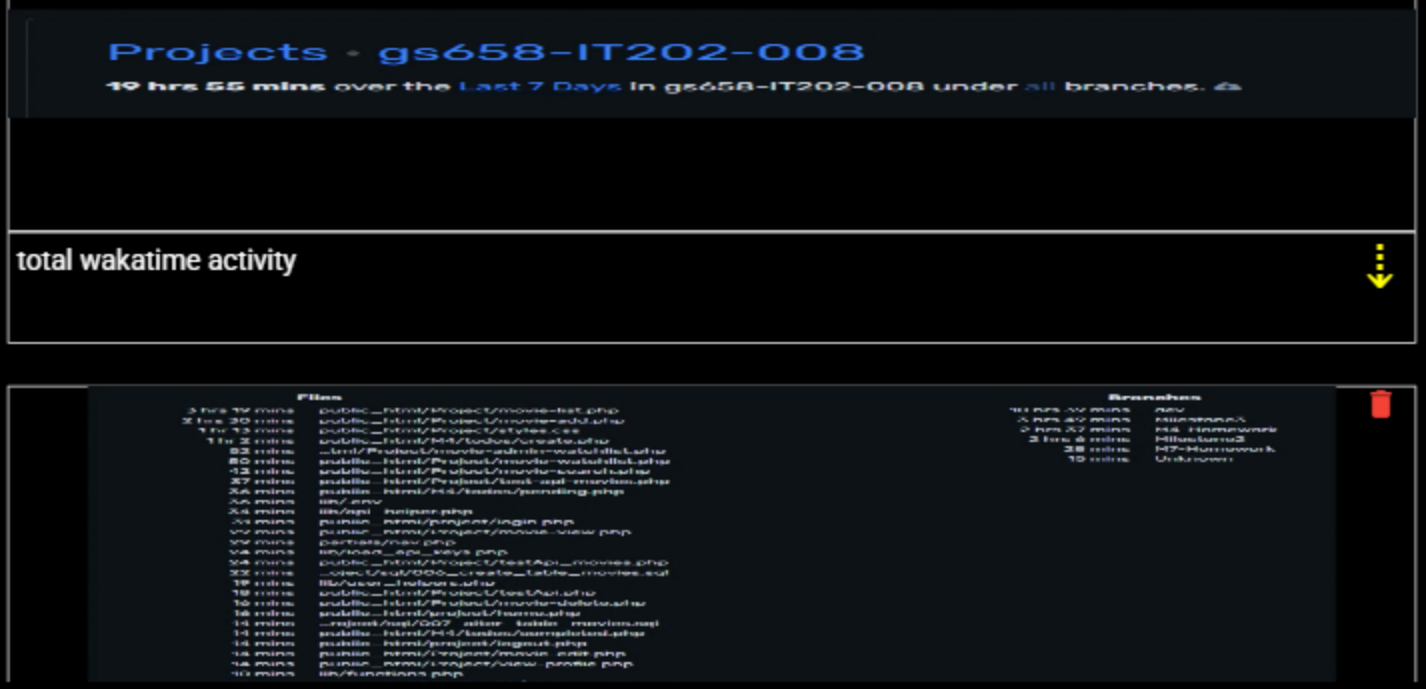
Task #2 (0.17 pts.) - WakaTime - Activity

Image Prompt

Weight: *33.33%*
Objective: *WakaTime - Activity*

Details:

- Visit the WakaTime.com Dashboard
- Click `Projects` and find your repository
- Capture the overall time at the top that includes the repository name
- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary



total wakatime activity

Task #3 (0.17 pts.) - Reflection

Weight: *33.33%*
Objective: *Reflection*

Sub-Tasks:

Task #1 (0.33 pts.) - What did you learn?

Text Prompt

Weight: *33.33%*
Objective: *What did you learn?*

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

I learned how to delete records from my movie database while checking for their IDs and user permissions. I also got more comfortable with handling redirects back to specific pages, like from the edit page back to the movie view page, and understanding the differences between soft and hard deletes.

Saved: 5/8/2025 1:52:00 AM

Task #2 (0.33 pts.) - What was the easiest part of the assignr

Text Prompt

Weight: *33.33%*
Objective: *What was the easiest part of the assignment?*

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

The easiest part of this assignment was the redirects because it was just one line of code and very easy to figure out how to do (clear concept). It was also easy to update nav.php with the certain restrictions (per user role) and I understood all the ideas clearly.

Saved: 5/8/2025 1:53:13 AM

Task #3 (0.33 pts.) - What was the hardest part of the assignı

Text Prompt

Weight: *33.33%*
Objective: *What was the hardest part of the assignment?*

Details:
Briefly answer the question (at least a few decent sentences)

Your Response:

The hardest part of this assignment was submitting all the screenshots. There were a lot of requirements, a lot of sections, and a lot of validations for each part. It was difficult to manage each one, and if an error occurred, it took a lot of time to debug and figure out what and where exactly the issue was.

Saved: 5/8/2025 1:54:23 AM