Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-api-project-milestone-2-2024/grade/na569

IT202-008-S2024 - [IT202] API Project Milestone 2 2024

Submissions:

Submission Selection

1 Submission [active] 4/24/2024 12:55:40 PM

Instructions

COLLAPSE

Implement the Milestone 2 features from the project's proposal document: https://docs.google.com/document/d/1XE96a8DQ52Vp49XACBDTNCq0xYDt3kF29cO88E
Make sure you add your ucid/date as code comments where code changes are done
All code changes should reach the Milestone2 branch
Create a pull request from Milestone2 to dev and keep it open until you get the output PDF from this assignment.
Gather the evidence of feature completion based on the below tasks.
Once finished, get the output PDF and copy/move it to your repository folder on your local machine.
Run the necessary git add, commit, and push steps to move it to GitHub
Complete the pull request that was opened earlier
Create and merge a pull request from dev to prod
Upload the same output PDF to Canvas

Branch name: Milestone2

Tasks: 29 Points: 10.00

Define Appropriate Tables for Data (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of Table SQL

COLLAPSE

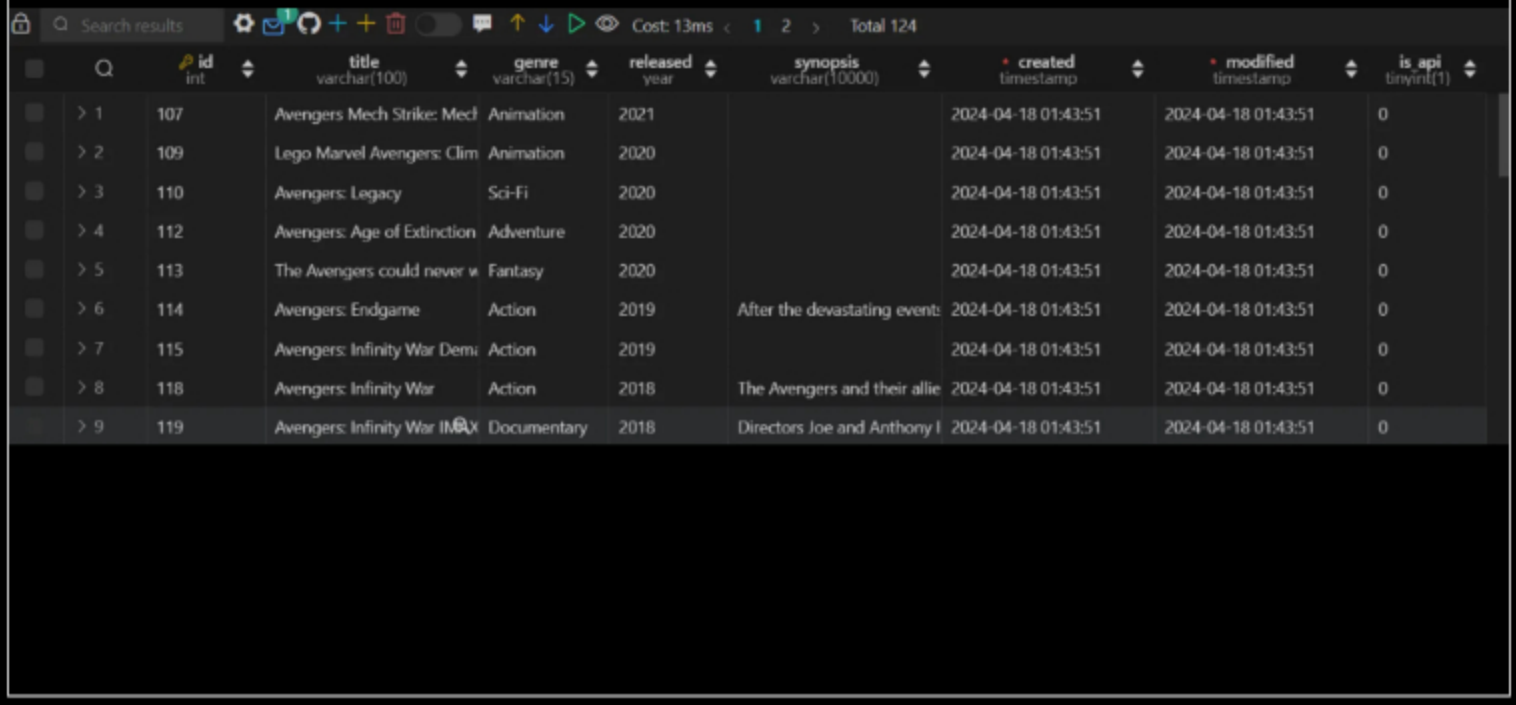
Checklist *The checkbox are for your own tracking

#	Points	Details
#1	1	Table(s) should have the 3 core columns we'll always be using (id, created, modified) plus additional columns for the incoming API data
#2	1	Columns should be logical and thought out (not valid to have a single field of JSON data or similar
#3	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



	id int	title varchar(100)	genre varchar(15)	released year	synopsis varchar(10000)	created timestamp	modified timestamp	is_api tinyint(1)
1	107	Avengers Mech Strike: Mecl	Animation	2021		2024-04-18 01:43:51	2024-04-18 01:43:51	0
2	109	Lego Marvel Avengers: Clim	Animation	2020		2024-04-18 01:43:51	2024-04-18 01:43:51	0
3	110	Avengers: Legacy	Sci-Fi	2020		2024-04-18 01:43:51	2024-04-18 01:43:51	0
4	112	Avengers: Age of Extinction	Adventure	2020		2024-04-18 01:43:51	2024-04-18 01:43:51	0
5	113	The Avengers could never v	Fantasy	2020		2024-04-18 01:43:51	2024-04-18 01:43:51	0
6	114	Avengers: Endgame	Action	2019	After the devastating event	2024-04-18 01:43:51	2024-04-18 01:43:51	0
7	115	Avengers: Infinity War Dem	Action	2019		2024-04-18 01:43:51	2024-04-18 01:43:51	0
8	118	Avengers: Infinity War	Action	2018	The Avengers and their allie	2024-04-18 01:43:51	2024-04-18 01:43:51	0
9	119	Avengers: Infinity War IMAX	Documentary	2018	Directors Joe and Anthony I	2024-04-18 01:43:51	2024-04-18 01:43:51	0

screenshot of SQL database with all columns and some information inputted

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Explain the design

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Note the different fields and their purpose
#2	1	Note if you needed to normalize the data into separate tables or if one table fit your needs

Response:

I have a column with a unique ID for each movie in the database. Title column has the movie title, genre column has the movie's genre, released column has the release date of the movie, synopsis column has the description of the movie if provided by the API. Created and modified columns showcase the timestamp for when data was inputted. is_api column showcases whether the row's movie information comes from manual data or API-generated data. I used one table to fit my needs.

COLLAPSE

Task #3 - Points: 1

Text: Add the pull request link for the branch related to this feature

Details:

Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature.

URL #1

https://github.com/nafisa37/na569-it202-008/pull/46

URL #2

https://github.com/nafisa37/na569-it202-008/pull/45

Data Creation Page (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the creation page

Checklist

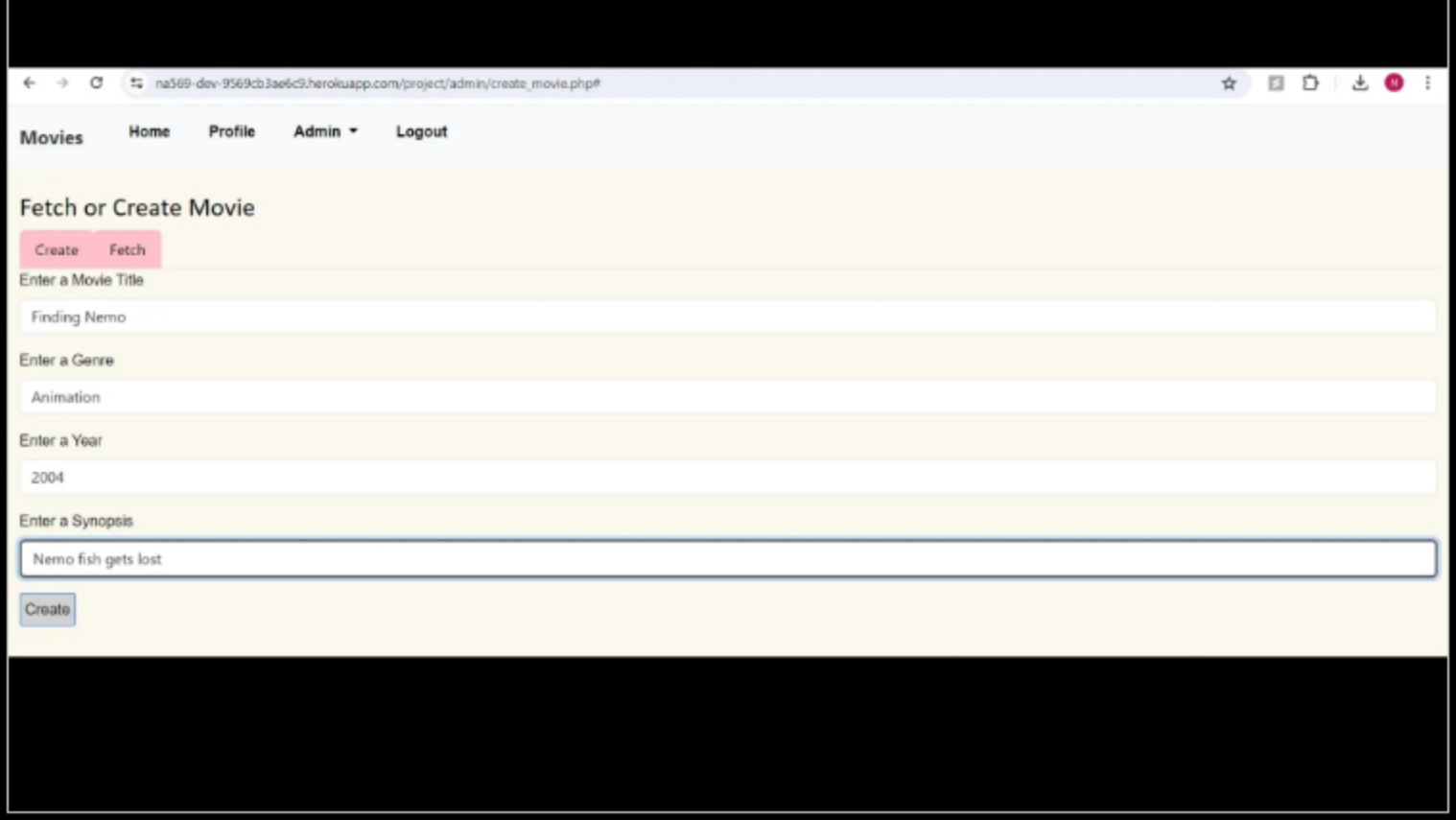
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show potentially valid data filled in for the custom creation page
#2	1	Show how the API data is fetched for API data (must be server-side)
#3	1	Show examples of validation messages
#4	1	Show an example of successful creation message
#5	1	Design/Style should be considered (i.e., bootstrap, custom css, etc)
#6	1	Make sure the heroku dev url is visible in the address bar
#7	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

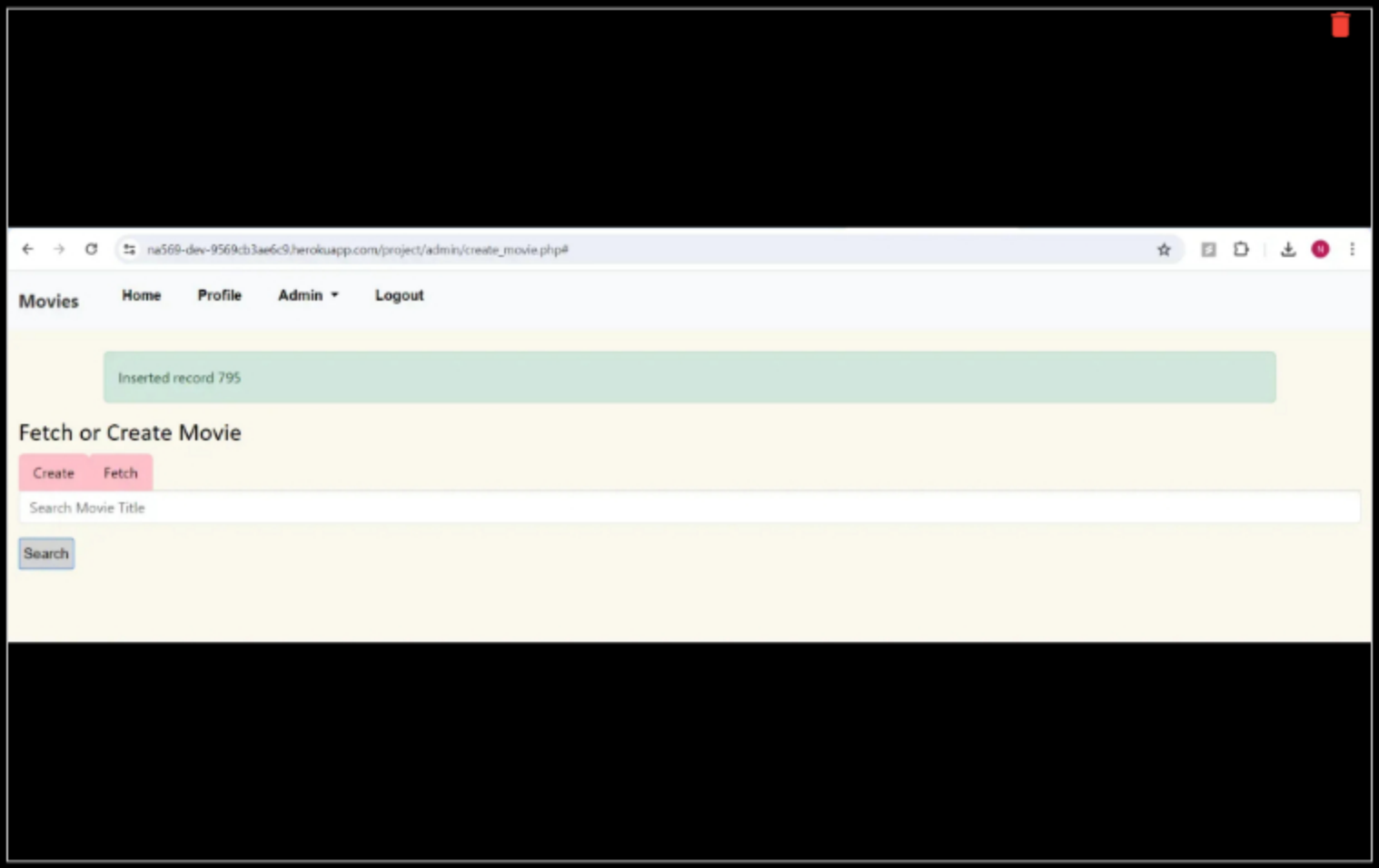
Small Medium Large



shows fields filled in with valid manual data creation

Checklist Items (1)

#1 Show potentially valid data filled in for the custom creation page



showcases success message when inputting manual data into database and when inputting API data

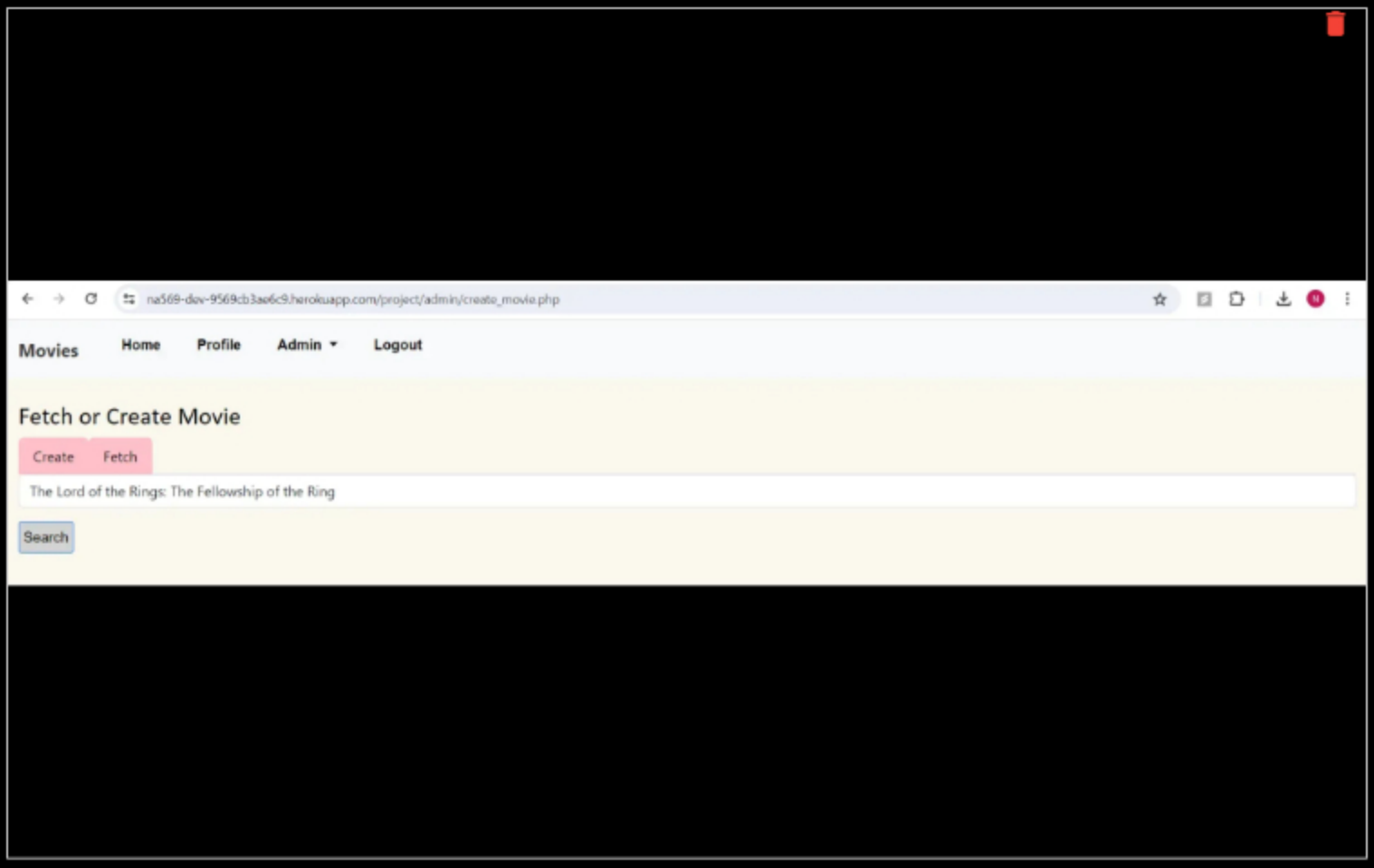
Checklist Items (4)

#4 Show an example of successful creation message

#5 Design/Style should be considered (i.e., bootstrap, custom css, etc)

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

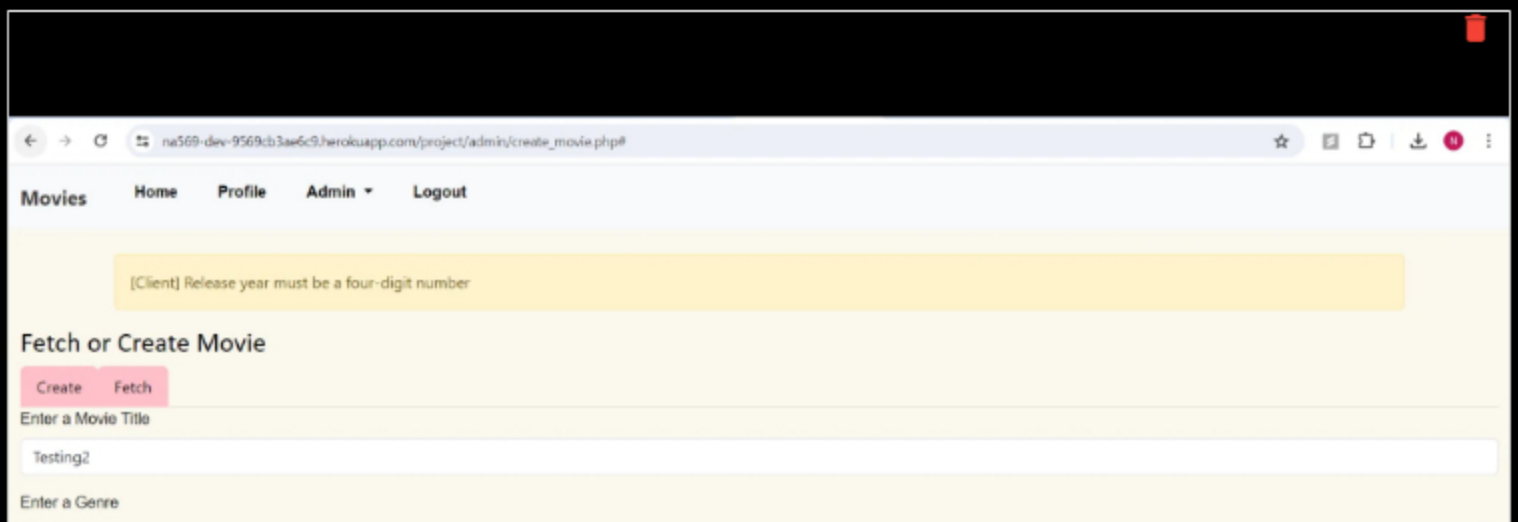
#7 Clearly caption screenshots



showcases data being filled into fetch movie from API

Checklist Items (1)

#2 Show how the API data is fetched for API data (must be server-side)



shows example of validation message when inputting incorrect data fields in creation form

Checklist Items (1)

#3 Show examples of validation messages

^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshots of creation page code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Form should have correct data types for each property being requested
#2	1	Form should have correct validation for each field (HTML, JS, and PHP)
#3	1	Successful creation should have a user-friendly message
#4	1	Any errors should have user-friendly messages
#5	1	Include the form/process for fetching API data
#6	1	Include some indicator between custom data and API data
#7	1	Include any other rules like role guards and login checks
#8	1	Include ucid/date comments for each code screenshot
#9	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
/na569, 4.24.24
<div class="container-fluid">
    <h3>Fetch or Create Movie</h3>
    <ul class="nav nav-tabs">
        <li class="nav-item">
            <a class="nav-link bg-success" href="#" onclick="switchTab('fetch')">Create</a>
        </li>
        <li class="nav-item">
            <a class="nav-link bg-success" href="#" onclick="switchTab('create')">Fetch</a>
        </li>
    </ul>
```

```php
<div id="fetch" class="tab-target">      You, last week • API integration, with database issues
    <form method="POST">
        <?php render_input(["type" => "search", "name" => "title", "placeholder" => "Search Movie Title", "rules" => ["required" => "required"]]); ?>
        <?php render_input(["type" => "hidden", "name" => "action", "value" => "fetch"]); ?>
        <?php render_button(["text" => "Search", "type" => "submit",]); ?>
    </form>
</div>
<div id="create" style="display: none;" class="tab-target">
    <form onsubmit="return validate(this)" method="POST">

        <?php render_input(["type" => "text", "name" => "title", "placeholder" => "Movie Title", "label" => "Enter a Movie Title", "rules" => ["required" => "required"
        <?php render_input(["type" => "text", "name" => "genre", "placeholder" => "Genre", "label" => "Enter a Genre", "rules" => ["required" => "required"]]); ?>
        <?php render_input(["type" => "number", "name" => "released", "placeholder" => "Release Date", "label" => "Enter a Year", "rules" => ["required" => "required"]
        <?php render_input(["type" => "text", "name" => "synopsis", "placeholder" => "Synopsis", "label" => "Enter a Synopsis", "rules" => ["required" => "required"]])

        <?php render_input(["type" => "hidden", "name" => "action", "value" => "create"]); ?>
        <?php render_button(["text" => "Search", "type" => "submit", "text" => "Create"]); ?>
    </form>
</div>
</div>
```

form that contains correct data types for each property, with HTML validation

Checklist Items (1)

#1 Form should have correct data types for each property being requested

```javascript
<script>
    function validate(form) {
        let title = form.title.value;
        let genre = form.genre.value;
        let released = form.released.value;
        let synopsis = form.synopsis.value;
        let isValid = true;

        // Regex to match four digits for the release year
        let validYear = /^\d{4}$/;

        if (title.length < 1) {
            flash("[Client] Title cannot be empty", "warning");
            isValid = false;
        }
        if (genre.length < 1) {
            flash("[Client] Genre cannot be empty", "warning");
            isValid = false;
        }
        if (released.length < 1) {
            flash("[Client] Release year cannot be empty", "warning");
            isValid = false;
        }
        if (!validYear.test(released)) {
            flash("[Client] Release year must be a four-digit number", "warning");
            isValid = false;
        }
        if (synopsis.length < 1) {
            flash("[Client] Synopsis cannot be empty", "warning");
            isValid = false;
        }

        return isValid;
    } <- #123-157 function validate(form)
</script> //na569, 4.24.24
```

javascript validation, with error messages shown

Checklist Items (1)

#2 Form should have correct validation for each field (HTML, JS, and PHP)

```php
</script> //na569, 4.24.24      You, 2 hours ago • Uncommitted changes
<?php
if (isset($_POST["save"])) {
    $title = se($_POST, "title", null, false);
    $genre = se($_POST, "genre", null, false);
```

```php
$released = se($_POST, "released", null, false);
$synopsis = se($_POST, "synopsis", null, false);
$hasError = false;

if (empty($title)) {
    flash("Title is required", "danger");
    $hasError = true;
}

if (empty($genre)) {
    flash("Genre is required", "danger");
    $hasError = true;
}

if (empty($released) || !is_numeric($released) || strlen($released) !== 4) {
    flash("Invalid release year", "danger");
    $hasError = true;
}

if (empty($synopsis)) {
    flash("Synopsis is required", "danger");
    $hasError = true;
}

if (!$hasError) {
    $db = getDB();
    $stmt = $db->prepare("INSERT INTO Movies (title, genre, released, synopsis) VALUES (:title, :genre, :released, :synopsis)");
    try {
        $stmt->execute([":title" => $title, ":genre" => $genre, ":released" => $released, ":synopsis" => $synopsis]);
        flash("Successfully recorded!", "success");
    } catch (PDOException $e) {
```

php validation, with success and error flash messages shown

Checklist Items (1)

#2 Form should have correct validation for each field (HTML, JS, and PHP)

```php
function fetch_movie($movie) //na569, 4.24.24        You, 2 days ago • Uncommitted changes
{
    $result = [];

    $data = ["function" => "MOVIE_DETAILS", "title" => $movie, "datatype" => "json"];

    $endpoint = "https://ott-details.p.rapidapi.com/search";
    $isRapidAPI = true;
    $rapidAPIHost = "ott-details.p.rapidapi.com";

    $result = get($endpoint, "MOVIE_API_KEY", $data, $isRapidAPI, $rapidAPIHost);

    error_log("Response: " . var_export($result, true));
    if (se($result, "status", 400, false) == 200 && isset($result["response"])) {
        $result = json_decode($result["response"], true);

        if (isset($result["movie_results"])) {
            $result = $result["movie_results"];
        }
    } else {
        $result = [];
    }

    if (isset($result["results"])) {
        $movies = $result["results"];

        foreach ($movies as $index => $movie) {
            foreach ($movie as $key => $value) {
                if (!in_array($key, ['genre', 'title', 'synopsis', 'released'])) {
                    unset($movies[$index][$key]);
                } elseif ($key === 'genre' && is_array($value)) {
                    $movies[$index][$key] = $value[0];
                }
                if (!isset($movies[$index]['synopsis'])) {
                    $movies[$index]['synopsis'] = '';
                }
            } <- #30-39 foreach ($movie as $key => $value)
```

function for fetching API data that is called on create_movie page

Checklist Items (1)

#5 Include the form/process for fetching API data

```php
//TODO handle movie fetch //na569, 4.24.24      You, 2 hours ago • Uncommitted changes
if (isset($_POST["action"])) {
    $action = $_POST["action"];
    $movie =  strtoupper(se($_POST, "title", "", false));
    $quote = [];
    $uniqueMovie = [];
    if ($movie) {
        if ($action === "fetch") {
            $result = fetch_movie($movie);
            error_log("Data from API" . var_export($result, true));
            if ($result) {
                foreach ($result as $movie) {
                    $title = $movie['title'];
                    if (!in_array($title, $uniqueMovie)) {
                        $quote = $movie;
                        //$quote["is_api"] =
                        $uniqueMovie[] = $title;
                    } <- #28-32 if (!in_array($title, $uniqueMovie))
                } <- #26-33 foreach ($result as $movie)
            } <- #25-34 if ($result)
        } else if ($action === "create") {
            foreach ($_POST as $k => $v) {
                if (!in_array($k, ["title", "genre", "released", "synopsis"])) {
                    unset($_POST[$k]);
                }
                $quote = $_POST;
                $quote["is_api"] = 0;
                error_log("Cleaned up POST: " . var_export($quote, true));
            } <- #36-43 foreach ($_POST as $k => $v)
        } <- #35-44 else if ($action === "create")
    } else {
        flash("You must provide a movie", "warning");
    }
```

showcases indicator between custom and API data

Checklist Items (2)

#5 Include the form/process for fetching API data

#6 Include some indicator between custom data and API data

```php
...
<?php
//note we need to go up 1 more directory
require(__DIR__ . "/../../../partials/nav.php");
//require_once(__DIR__ . "/../../../lib/movie_api.php");

//na569, 4.24.24
if (!has_role("Admin")) {
    flash("You don't have permission to view this page", "warning");
    die(header("Location: $BASE_PATH" . "/home.php"));
}
?>
```

```php
<?php
```

role admin check

Checklist Items (1)

#7 Include any other rules like role guards and login checks

COLLAPSE

Task #3 - Points: 1

Text: Screenshot of records from DB

Checklist

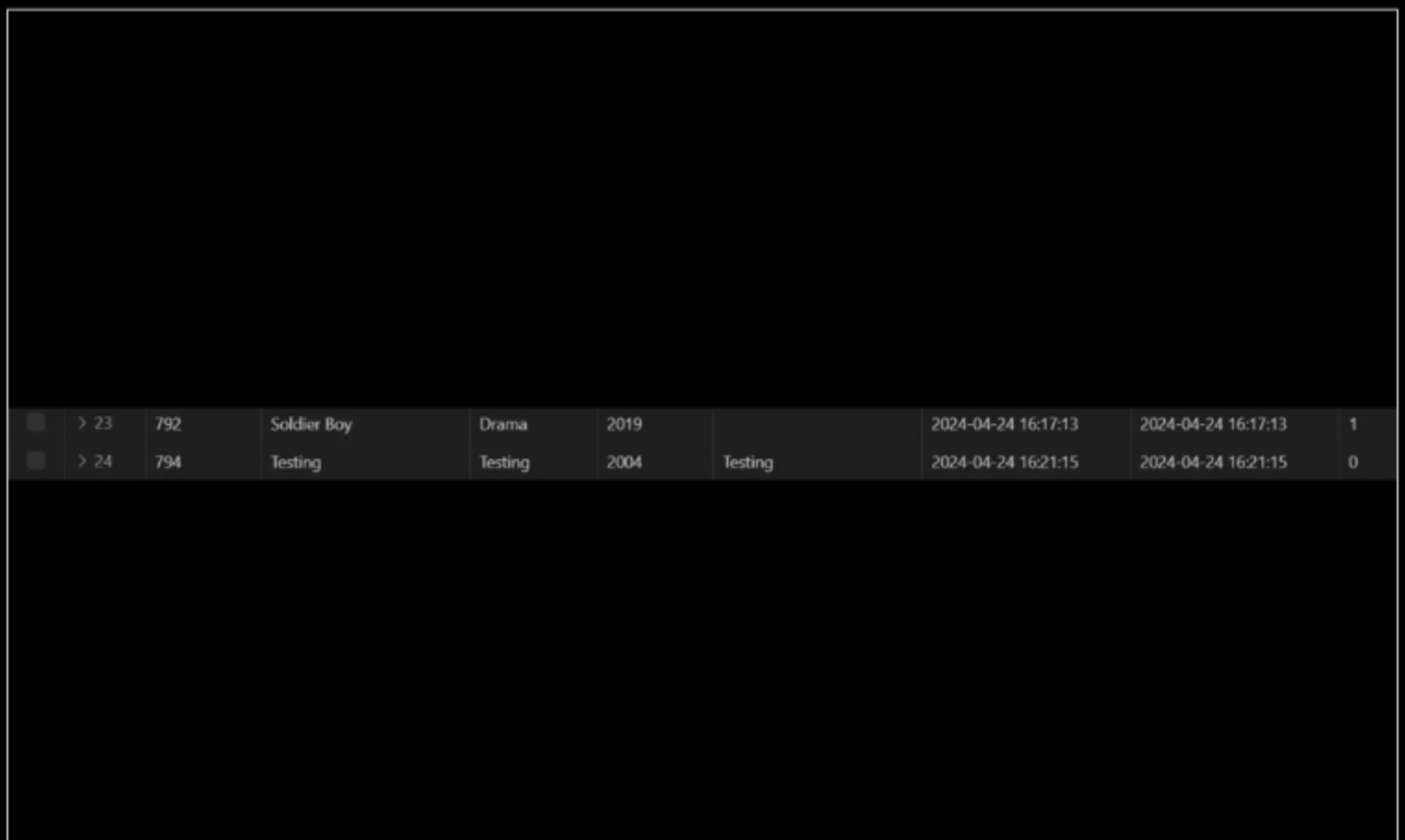
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show at least one record fetched from the API
#2	1	Show at least one record created via the creation form
#3	1	Note what differs
#4	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Solder Boy is a record fetched from the API, and Testing is a manual record I created using the creation form. The last column is the is_api column. 0 indicates that the record is not from an API, while 1 indicates that it is.

Checklist Items (4)

#1 Show at least one record fetched from the API

#2 Show at least one record created via the creation form

#3 Note what differs

#4 Clearly caption screenshots

COLLAPSE

Task #4 - Points: 1

Text: Explain the process

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Provide a high-level step-by-step of how fetching API data works and gets added to your DB (include how duplicates are handled)
#2	1	Provide a high-level step-by-step of how creating custom data works and gets added to your DB (include how duplicates are handled)
#3	1	Briefly describe the validations for the applicable fields
#4	1	Describe how duplicate data is handled

Response:

The user submits a form with a movie title that fetches details related to that movie title from the API, including title, genre, release year, and synopsis. Before adding the movie to the database, the code checks to that it does not already exist in the database to avoid duplicates. If a duplicate is detected, the code notifies the user that a movie with that title already exists. For manual data, it is the same process except the user submits a form with the title, genre, release year, and synopsis instead of just the title. For validations, the title, genre, and synopsis must not be empty. The release year also must be a four digit number.

COLLAPSE

Task #5 - Points: 1

Text: Add related links

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Include the heroku prod link for this page
		Add the pull request link for the branch related to this feature Note: the link should end

#2	1	with /pull/#. Same pull request shouldn't be used for each feature

URL #1

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

URL #2

https://github.com/nafisa37/na569-it202-008/pull/47

Data List Page (many entities) (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the list page

Checklist

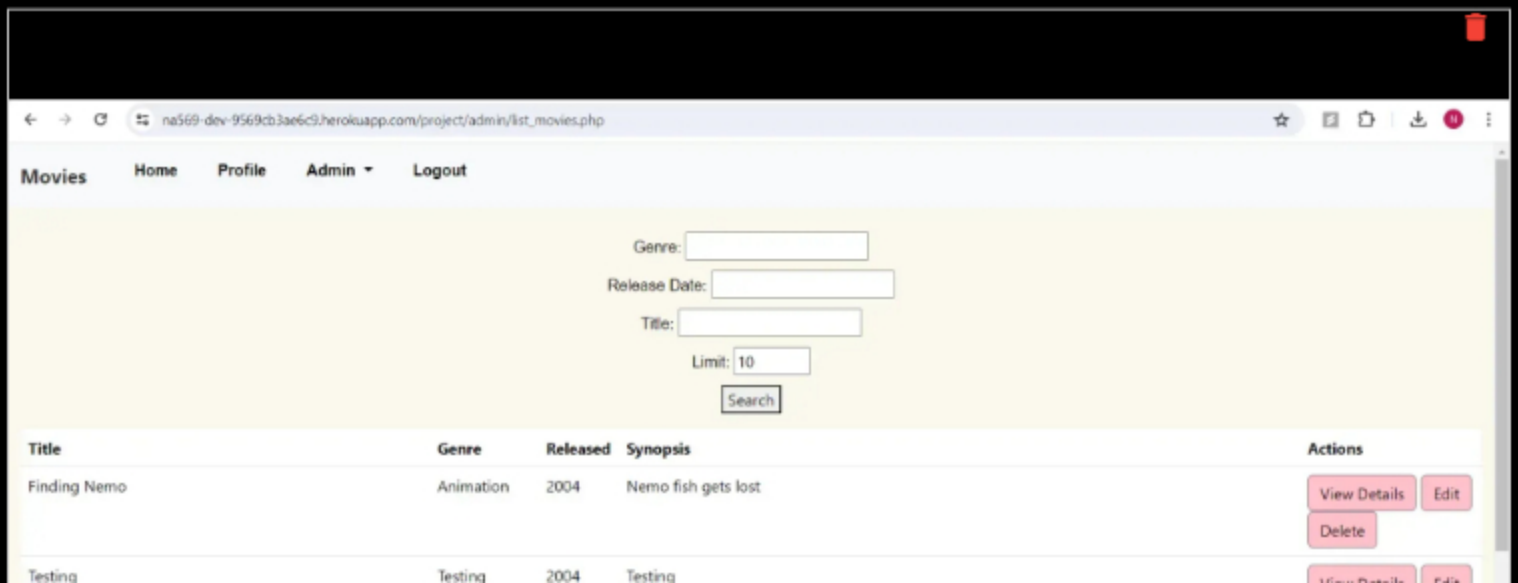
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the page of your entities listed (have a reasonable number shown)
#2	1	Show the filter/sort form based on your data and the required limit field
#3	1	Demonstrate a few varied filters/sorts
#4	1	Demonstrate a filter that doesn't have any records (should show an appropriate message)
#5	1	Each list item should have a link of single view (i.e., details), edit, and delete (some of which may only be visible to admin users)
#6	1	Each list item should have a summary of the entity (likely won't be the entire entity data)
#7	1	Design/Style should be considered (i.e., bootstrap, custom css, etc)
#8	1	Make sure the heroku dev url is visible in the address bar
#9	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



screenshot of listed movies page, showcases links to single view, edit, and delete movies pages and summary of entity

Checklist Items (5)

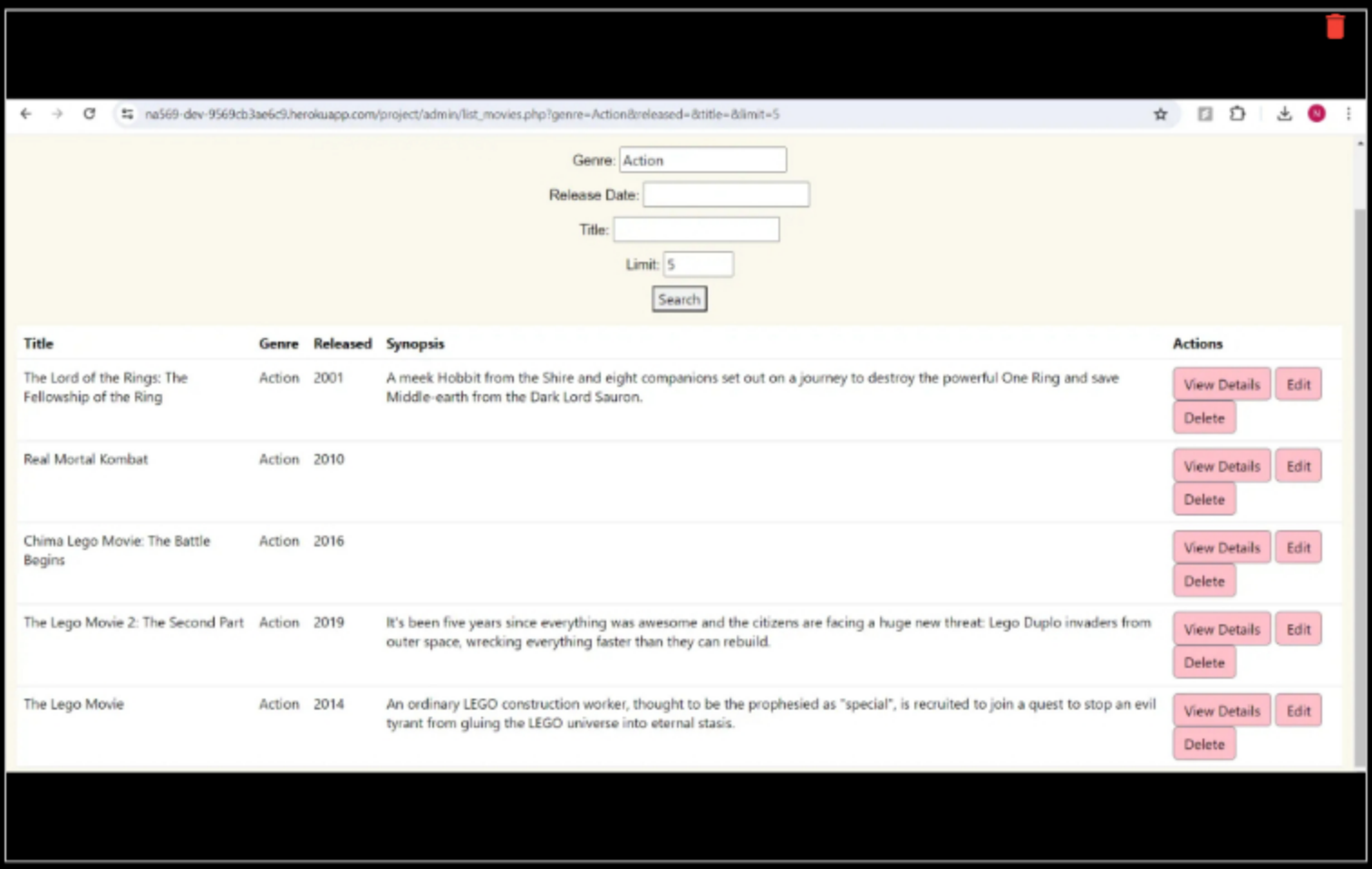
#1 Show the page of your entities listed (have a reasonable number shown)

#5 Each list item should have a link of single view (i.e., details), edit, and delete (some of which may only be visible to admin users)

#6 Each list item should have a summary of the entity (likely won't be the entire entity data)

#7 Design/Style should be considered (i.e., bootstrap, custom css, etc)

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

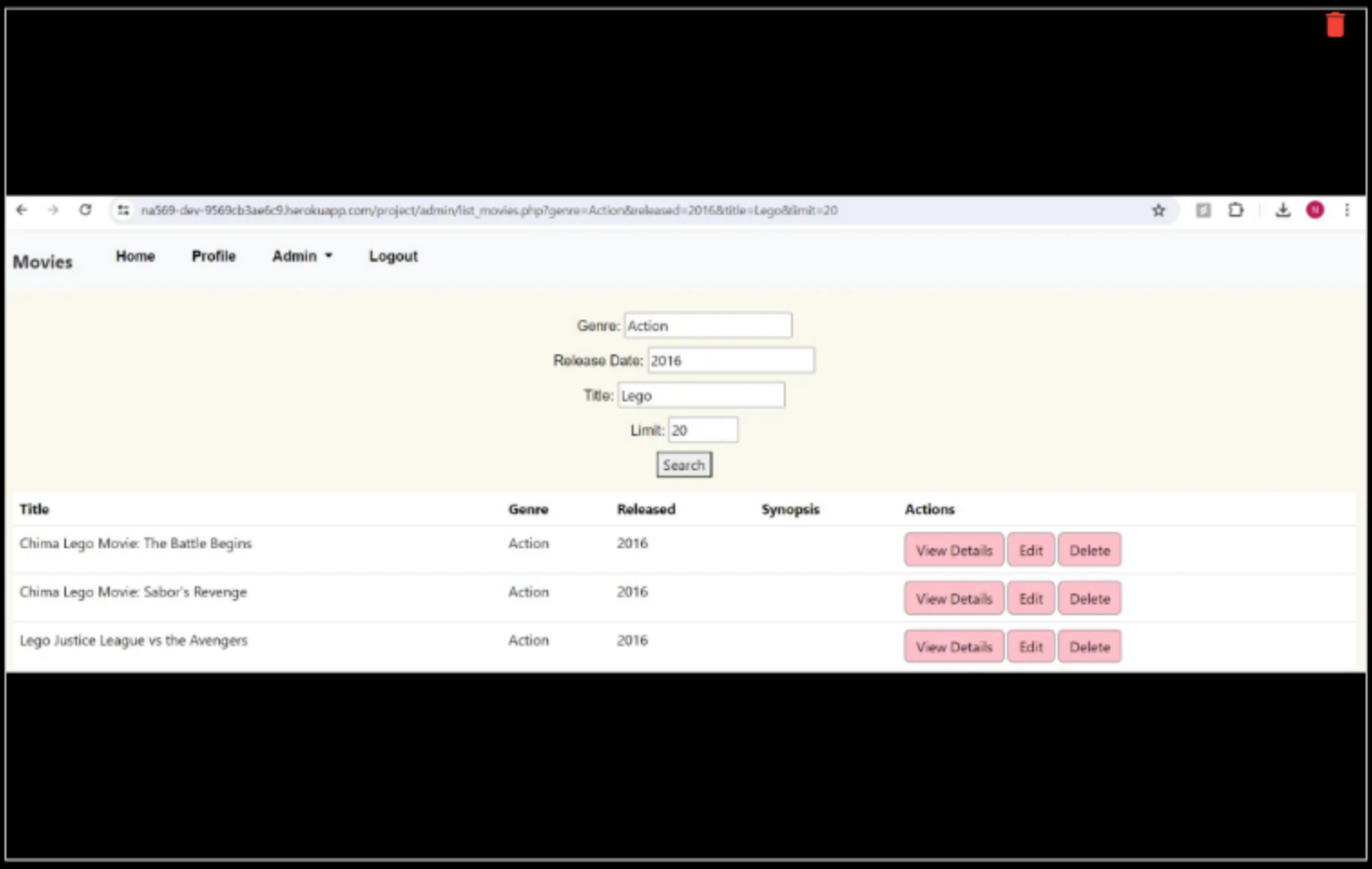


screenshot of filter for 5 action movies applied

Checklist Items (2)

#2 Show the filter/sort form based on your data and the required limit field

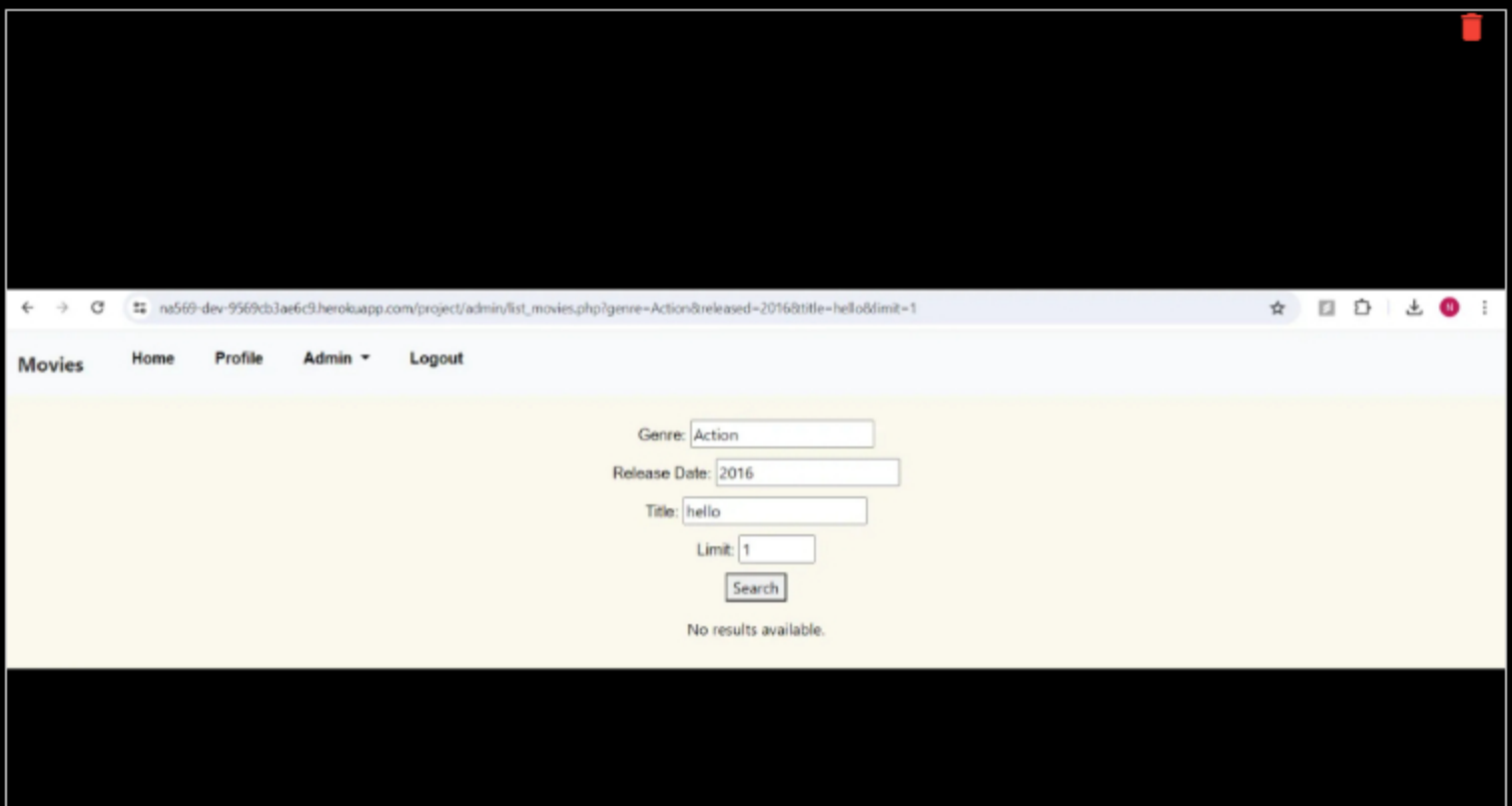
#3 Demonstrate a few varied filters/sorts



another example of a filtered movie list with filtered genre, release date, title, and limit

Checklist Items (1)

#3 Demonstrate a few varied filters/sorts



filter that does not have any results

Checklist Items (1)

#4 Demonstrate a filter that doesn't have any records (should show an appropriate message)

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the list page code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the filter/sort form generation
#2	1	Show the DB query and how the filter/sort is handled (including the restriction on the limit field)
#3	1	Show how the output is generated and displayed
#4	1	Show any restrictions like role guard or login checks
#5	1	Include ucid/date comments for each code screenshot
#6	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

ic_html > project > admin > list_movies.php > ...

```
//na569, 4.24.24      You, 29 seconds ago • Uncommitted changes  <- #62-67 if (isset($_GET['limit']))
?>
<!-- Search Form -->
<div style="text-align: center;">
<form method="GET">
    <div>
        <label for="genre">Genre:</label>
        <input type="text" id="genre" name="genre" value="<?php echo htmlspecialchars($genre); ?>">
    </div>
    <div>
        <label for="released">Release Date:</label>
        <input type="text" id="released" name="released" value="<?php echo htmlspecialchars($released); ?>">
    </div>
    <div>
        <label for="title">Title:</label>
        <input type="text" id="title" name="title" value="<?php echo htmlspecialchars($title); ?>">
    </div>
    <div>
        <label for="limit">Limit:</label>
        <input type="number" id="limit" name="limit" value="<?php echo $limit; ?>" min="1" max="100">
    </div>
    <button type="submit">Search</button>
</form>

<?php
$table = ["data" => $results, "title" => "Search Movies", "ignored_columns" => ["id"], "edit_url" => get_url("admin/edit_movie.php")];
?>
<?php if (empty($results)) : ?>
    <div>No results available.</div>
<?php else : ?>
    <div class="container-fluid">
```

```
        <table class="table">
            <thead>
                <tr>
                    <th>Title</th>
                    <th>Genre</th>
                    <th>Released</th>
```

showcases the filter sort form generation

Checklist Items (1)

#1 Show the filter/sort form generation

```php
$genre = isset($_GET['genre']) ? $_GET['genre'] : '';
$title = isset($_GET['title']) ? $_GET['title'] : '';
$released = isset($_GET['released']) ? $_GET['released'] : '';
$limit = isset($_GET['limit']) ? intval($_GET['limit']) : 10; // Default to 10
$limit = max(1, min($limit, 100));
      You, last week • API integration, with database issues
$query = "SELECT id, title AS Title, genre as Genre, released as Released, synopsis as Synopsis FROM `Movies` WHERE 1=1";
$params = [];
if (!empty($genre)) {
    $query .= " AND genre LIKE :genre";
    $params[':genre'] = "%$genre%";
}
if (!empty($title)) {
    $query .= " AND title LIKE :title";
    $params[':title'] = "%$title%";
}
if (!empty($released)) {
    $query .= " AND released LIKE :released";
    $params[':released'] = "%$released%";
}
$query .= " ORDER BY created DESC LIMIT $limit";

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

showcases DB query and how filter/sort is handled

Checklist Items (1)

#2 Show the DB query and how the filter/sort is handled (including the restriction on the limit field)

public_html > project > admin > list_movies.php > div

```
71  <div style="text-align: center;">
92  <?php //na569, 4.24.24      You, 3 minutes ago • Uncommitted changes
93  $table = ["data" => $results, "title" => "Search Movies", "ignored_columns" => ["id"], "edit_url" => get_url("admin/edit_movie.php")];
94  ?>
95  <?php if (empty($results)) : ?>
96      <div>No results available.</div>
97  <?php else : ?>
98      <div class="container-fluid">
99          <table class="table">
100             <thead>
101                 <tr>
102                     <th>Title</th>
103                     <th>Genre</th>
104                     <th>Released</th>
105                     <th>Synopsis</th>
106                     <th>Actions</th>
107                 </tr>
108             </thead>
109             <tbody>
110                 <?php foreach ($results as $row) : ?>
111                     <tr>
```

```
                        <td><?php echo htmlspecialchars($row['Title']); ?></td>
                        <td><?php echo htmlspecialchars($row['Genre']); ?></td>
                        <td><?php echo htmlspecialchars($row['Released']); ?></td>
                        <td><?php echo htmlspecialchars($row['Synopsis']); ?></td>
                        <td>
                            <a href="movie_details.php?id=<?php echo $row['id']; ?>" class="btn btn-secondary">View Details</a>
                            <a href="edit_movie.php?id=<?php echo $row['id']; ?>" class="btn btn-secondary">Edit</a>
                            <a href="delete_movie.php?id=<?php echo $row['id']; ?>" class="btn btn-secondary">Delete</a>
                        </td>
                    </tr>
                <?php endforeach; ?>
            </tbody>
        </table>
    </div>
<?php endif; ?>
```

shows how the output is generated and displayed

Checklist Items (1)

#3 Show how the output is generated and displayed

```
<script>
//js validation
    let limit = form.limit.value;
    let isValid = true;

    function validate() {
        var limit = document.getElementById('limit').value;
        if (limit < 1 || limit > 100 || isNaN(limit)) { // Check if limit is not between 1 and 100 or not a number
            flash("[Client] Limit must be a number between 1 and 100", "warning");
            isValid = false;
        }

        return isValid;
    } <- #49-57 function validate()
</script>

<?php
//php validation
if (isset($_GET['limit'])) {
    $limit = intval($_GET['limit']);
    if ($limit < 1 || $limit > 100) {
        flash("Limit must be a number between 1 and 100", "warning");
    }
} <- #62-67 if (isset($_GET['limit']))
//na569, 4.24.24
```

showcases php and js validation

Checklist Items (1)

#4 Show any restrictions like role guard or login checks

```
<?php
//note we need to go up 1 more directory
```

```php
require(__DIR__ . "/../../../partials/nav.php");
//na569, 4.24.24
if (!has_role("Admin")) {
    flash("You don't have permission to view this page", "warning");
    die(header("Location: $BASE_PATH" . "/home.php"));
}
```

only admin has access to list pages

Checklist Items (1)

#4 Show any restrictions like role guard or login checks

COLLAPSE

Task #3 - Points: 1

Text: Explain how the page works

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Provide the high-level steps of how the filter/sorting works and how data is displayed
#2	1	Summarize what you're showing on the screen
#3	1	Mention your design/style choice
#4	1	Mention which users can interact with the view, edit, and delete links

Response:

The user fills out the search form with the genre, release date, title, and limit . The server creates an SQL query that filters the movies in the database based on the user's parameters and using the LIKE clause. The ORDER BY clause also sorts the movies so that the most recent movies appear first. The results are displayed in table format. Admin users can view, edit, and delete the links. The buttons are pink to fit the neutral theme of the website.

COLLAPSE

Task #4 - Points: 1

Text: Add related links

Checklist *The checkboxes are for your own tracking

#	Points	Details

#1	1	Include the heroku prod link for this page
#2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1

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

URL #2

https://github.com/nafisa37/na569-it202-008/pull/48

View Details Page (single entity) (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the details page

Checklist

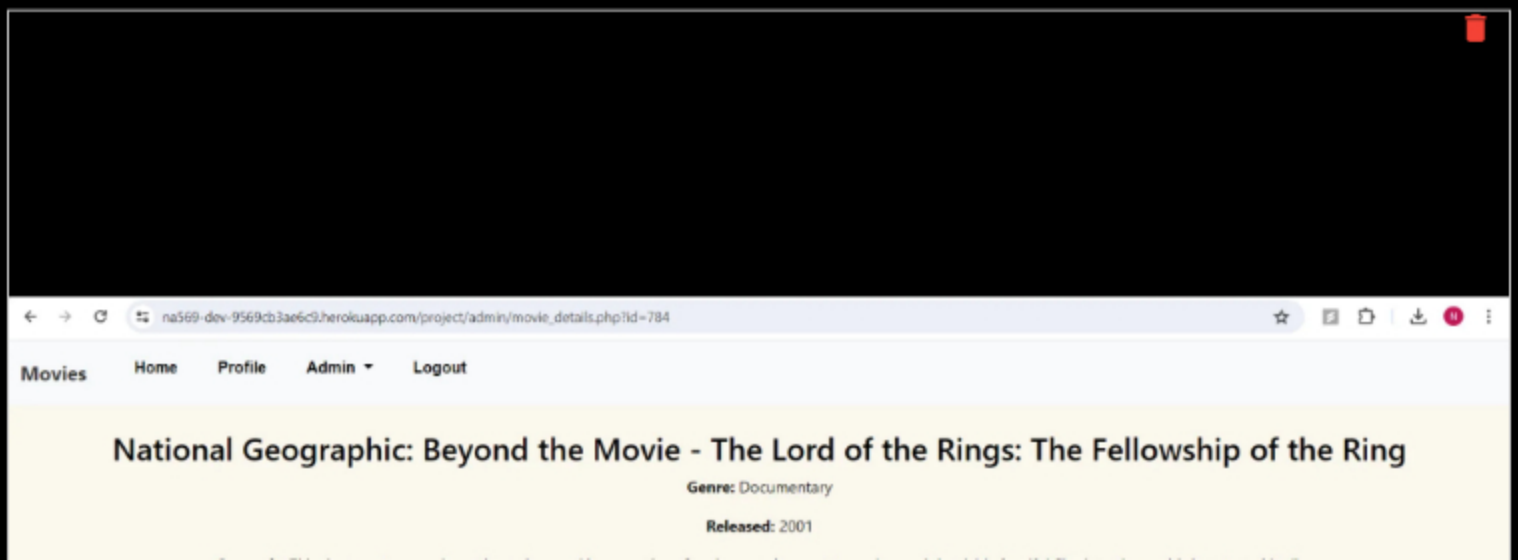
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Entity should be fetch by id (via the url)
#2	1	A missing id should redirect back to the list page with an applicable message
#3	1	Design/Style should be considered (i.e., bootstrap, custom css, etc)
#4	1	Data shown should be more detailed/inclusive than the summary view
#5	1	There should be a link to edit the entity (this may be an admin-only thing, but it should be present for the respective role)
#6	1	There should be a link to delete the entity (this may be an admin-only thing, but it should be present for the respective role)
#7	1	Make sure the heroku dev url is visible in the address bar
#8	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Synopsis: This documentary explores through provoking questions for viewers who want to go beyond the richly fanciful film into the world that started it all.

Edit Movie | Delete Movie

showcases single entity list view with entity fetched by ID in URL and links to edit and delete the entity

Checklist Items (7)

#1 Entity should be fetch by id (via the url)

#3 Design/Style should be considered (i.e., bootstrap, custom css, etc)

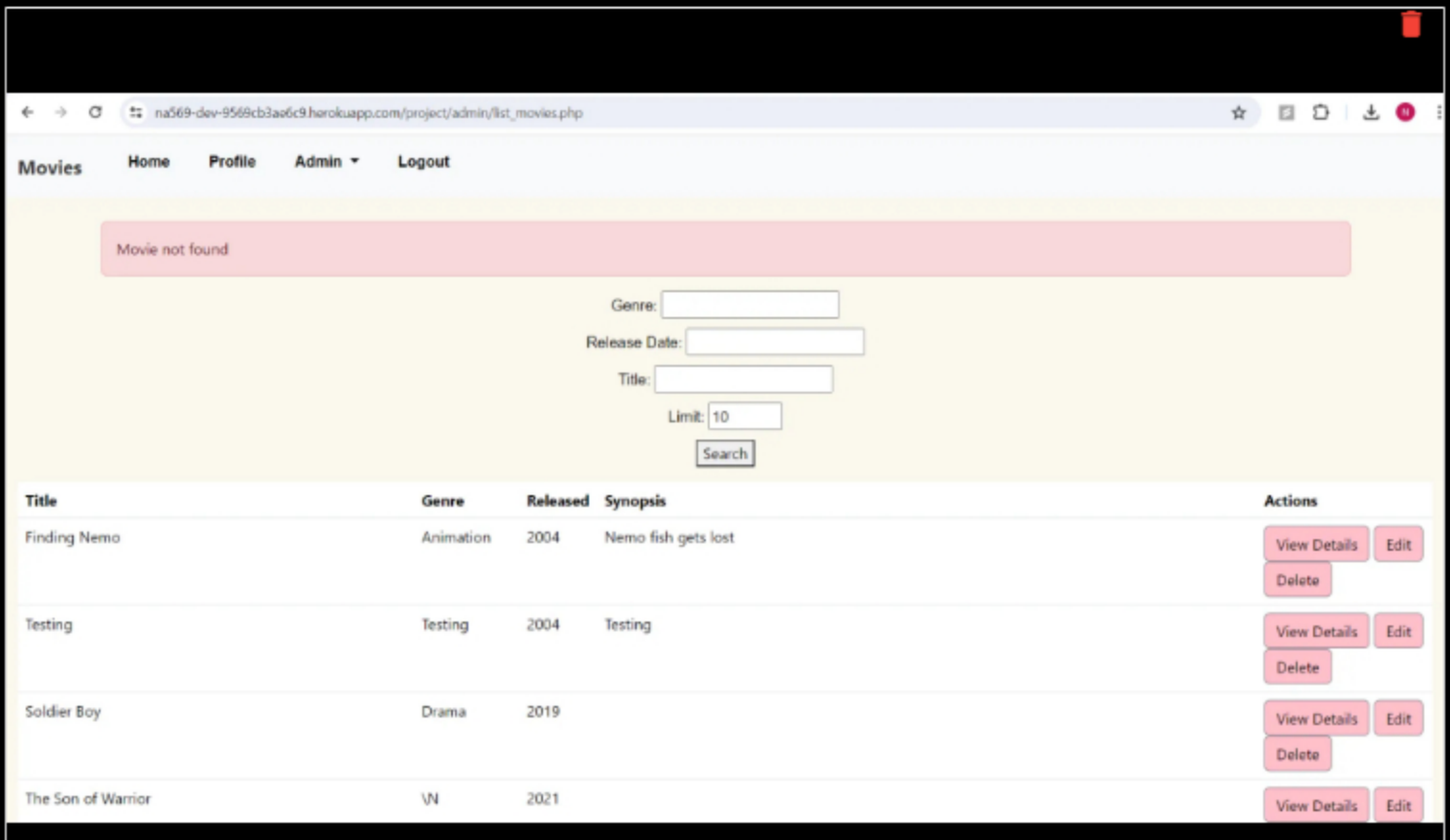
#4 Data shown should be more detailed/inclusive than the summary view

#5 There should be a link to edit the entity (this may be an admin-only thing, but it should be present for the respective role)

#6 There should be a link to delete the entity (this may be an admin-only thing, but it should be present for the respective role)

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

#8 Clearly caption screenshots



invalid ID redirects back to list page with Movie not found message

Checklist Items (1)

#2 A missing id should redirect back to the list page with an applicable message

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the details page code

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Show how id is fetched
#2	1	Show the DB query to get the record
#3	1	Show the code related to presenting the data and showing the links
#4	1	Include ucid/date comments for each code screenshot
#5	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```php
<?php
require_once(__DIR__ . "/../../../partials/nav.php");
is_logged_in(true);

//require(__DIR__ . "/../../../lib/db.php");

//na569,4.24.24       You, 2 days ago • Uncommitted changes
$movieDetails = null;
$movieId = isset($_GET['id']) ? intval($_GET['id']) : null;

// Validate the movie ID
if ($movieId === null || $movieId <= 0) {
    flash("Invalid movie ID", "danger");
    header("Location: list_movies.php");
    exit();
} else {

    $db = getDB();
    $stmt = $db->prepare("SELECT * FROM Movies WHERE id = :id");
    $stmt->bindParam(':id', $movieId);
    $stmt->execute();

    $movieDetails = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($movieDetails === false) {
        flash("Movie not found", "danger");
        header("Location: list_movies.php");
        exit();
    } <- #25-29 if ($movieDetails === false)
} <- #16-30 else
```

shows how ID is fetched and how DB query is used to get the record

Checklist Items (1)

#1 Show how id is fetched

```php
//na569, 4.24.24
echo '<div style="text-align: center;">';
echo "<h2>{$movieDetails['title']}</h2>";
echo "<p><strong>Genre:</strong> {$movieDetails['genre']} </p>";
echo "<p><strong>Released:</strong> {$movieDetails['released']}</p>";
echo "<p><strong>Synopsis:</strong> {$movieDetails['synopsis']}</p>";

echo '<a href="edit_movie.php?id=' . $movieId . '" class="btn btn-secondary" style="margin-left: 10px;">Edit Movie</a>';
echo '<a href="delete_movie.php?id=' . $movieId . '" class="btn btn-secondary" style="margin-left: 10px;">Delete Movie</a>';
echo '</div>';
?>
```

shows how code relating to how data is presented and the links to edit and delete movies

Checklist Items (1)

#3 Show the code related to presenting the data and showing the links

COLLAPSE

Task #3 - Points: 1

Text: Explain how the page works

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Provide the high-level steps for handling the DB lookup and presenting the data

Response:

First, we check to make sure a valid movie ID is in the URL. If the ID is not valid, it displays a flash message stating so and redirect the user back to the moves list page. We use an SQL query to select the columns from the Movies table where id matches the given movie ID. The movie details are then displayed in a format using HTML and put inside elements to center the text alignment.

^COLLAPSE^

Task #4 - Points: 1

Text: Add related links

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Include the heroku prod link for this page
#2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1

https://na569-prod-74aadc581478.herokuapp.com/project/admin/movie_details.php?id=789

URL #2

https://github.com/nafisa37/na569-it202-008/pull/48

URL #3

https://github.com/nafisa37/na569-it202-008/pull/51

Edit Data Page (2 pts.)

^COLLAPSE^

Task #1 - Points: 1

Text: Screenshots of the edit page

Checklist

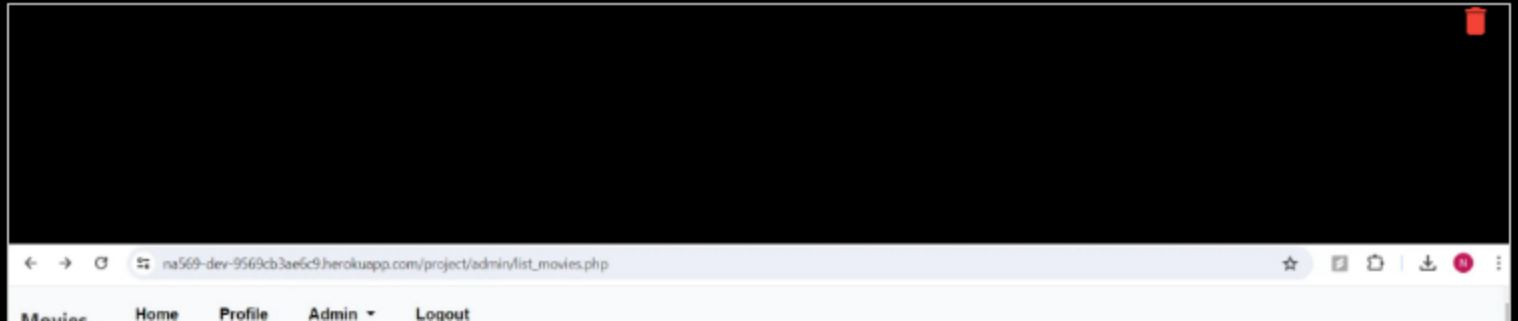
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show before and after screenshots of data you'll edit
#2	1	Show examples of validation messages
#3	1	Show an example of successful edit messages
#4	1	Design/Style should be considered (i.e., bootstrap, custom css, etc)
#5	1	Make sure the heroku dev url is visible in the address bar
#6	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Movies

Genre:

Release Date:

Title:

Limit: 10

Search

Title	Genre	Released	Synopsis	Actions
Finding Nemo	Animation	2004	Nemo fish gets lost	View Details Edit Delete

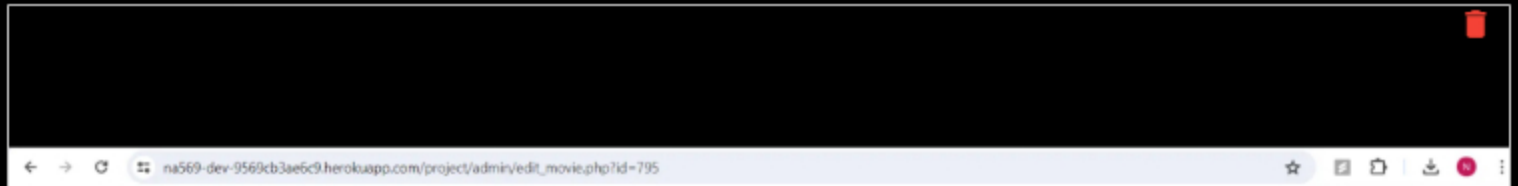
before photo of edited movie

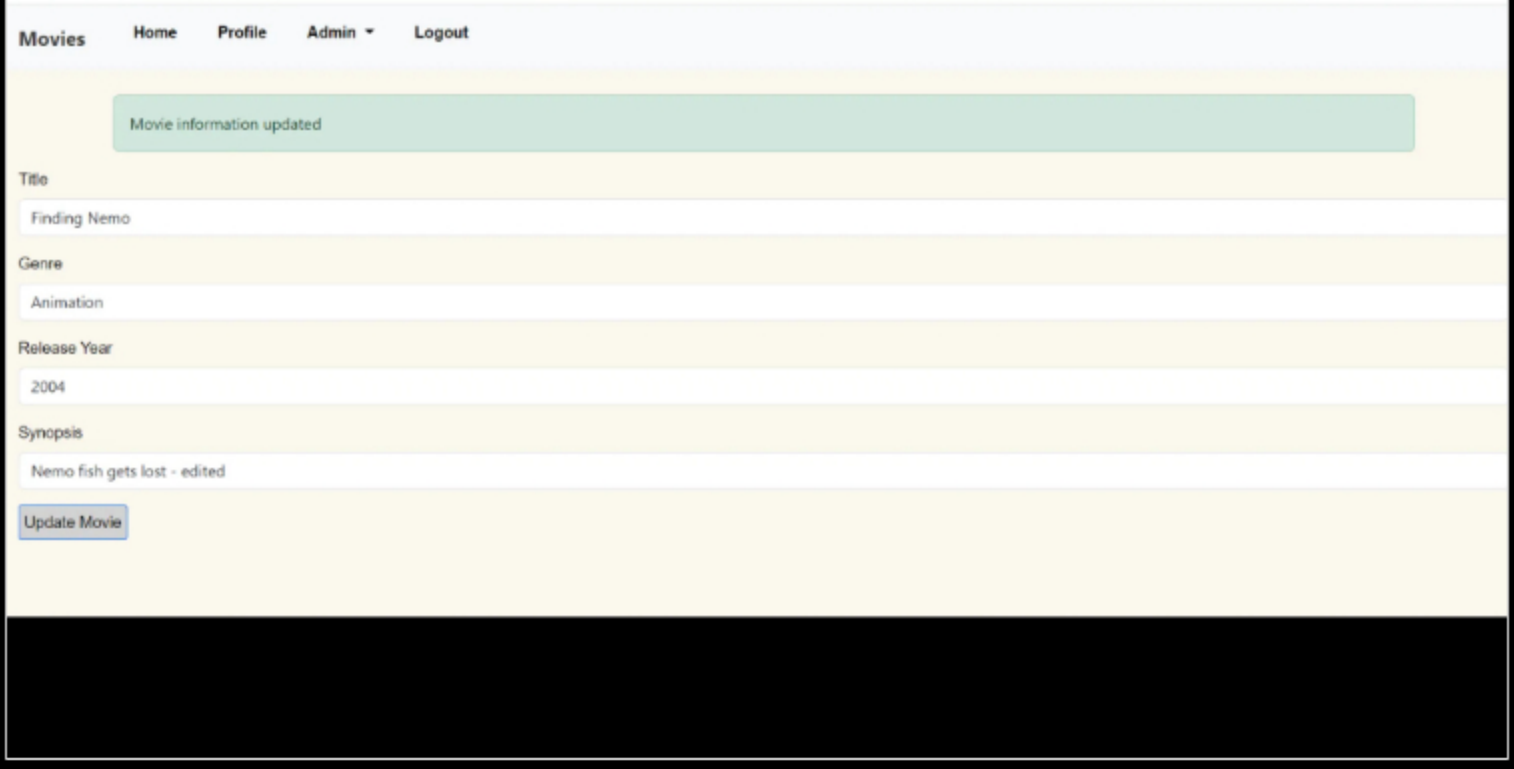
Checklist Items (1)

#1 Show before and after screenshots of data you'll edit

screenshot of edit page that showcases information pre filled in

Checklist Items (0)

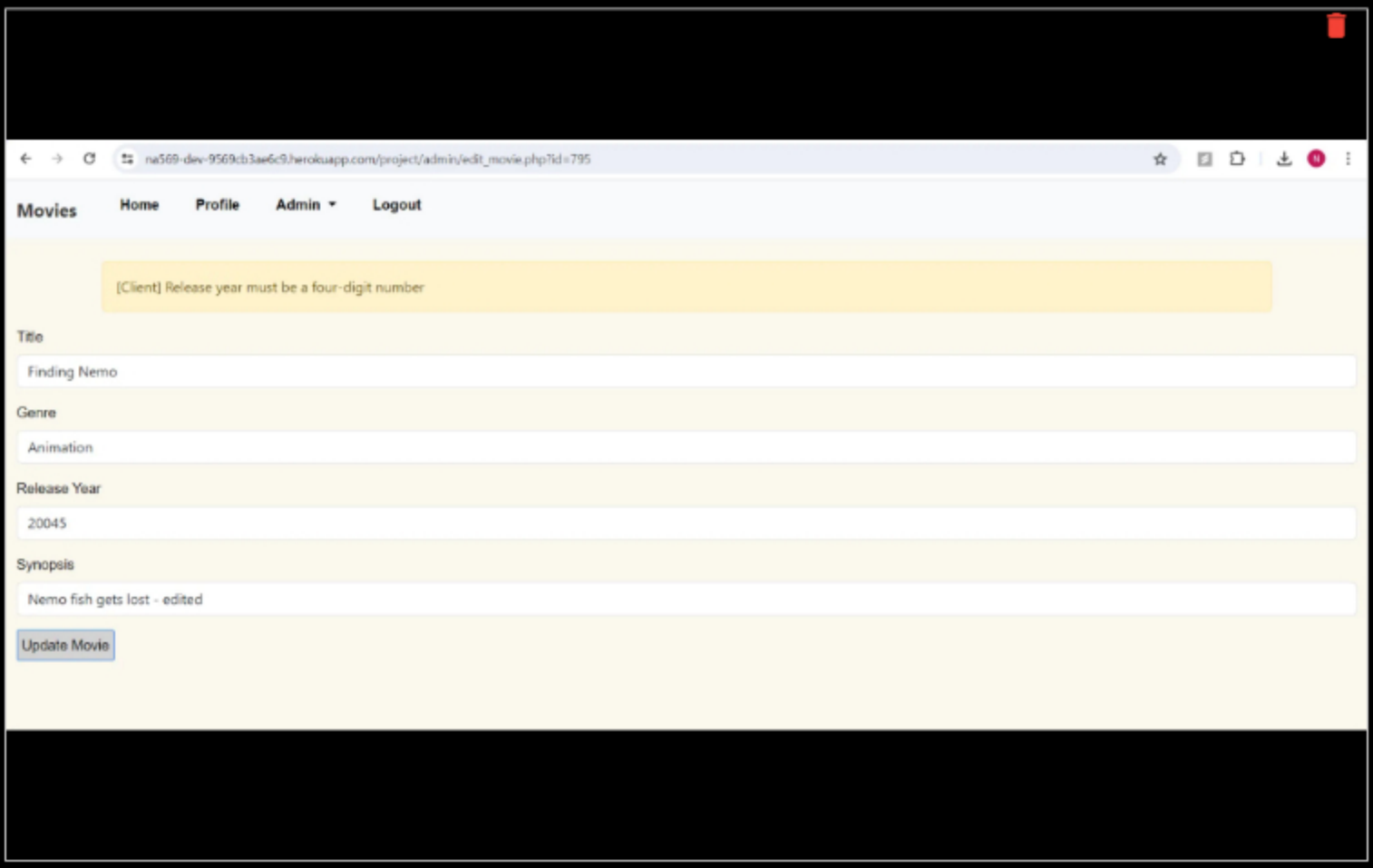




picture of edit to page with success edit message

Checklist Items (1)

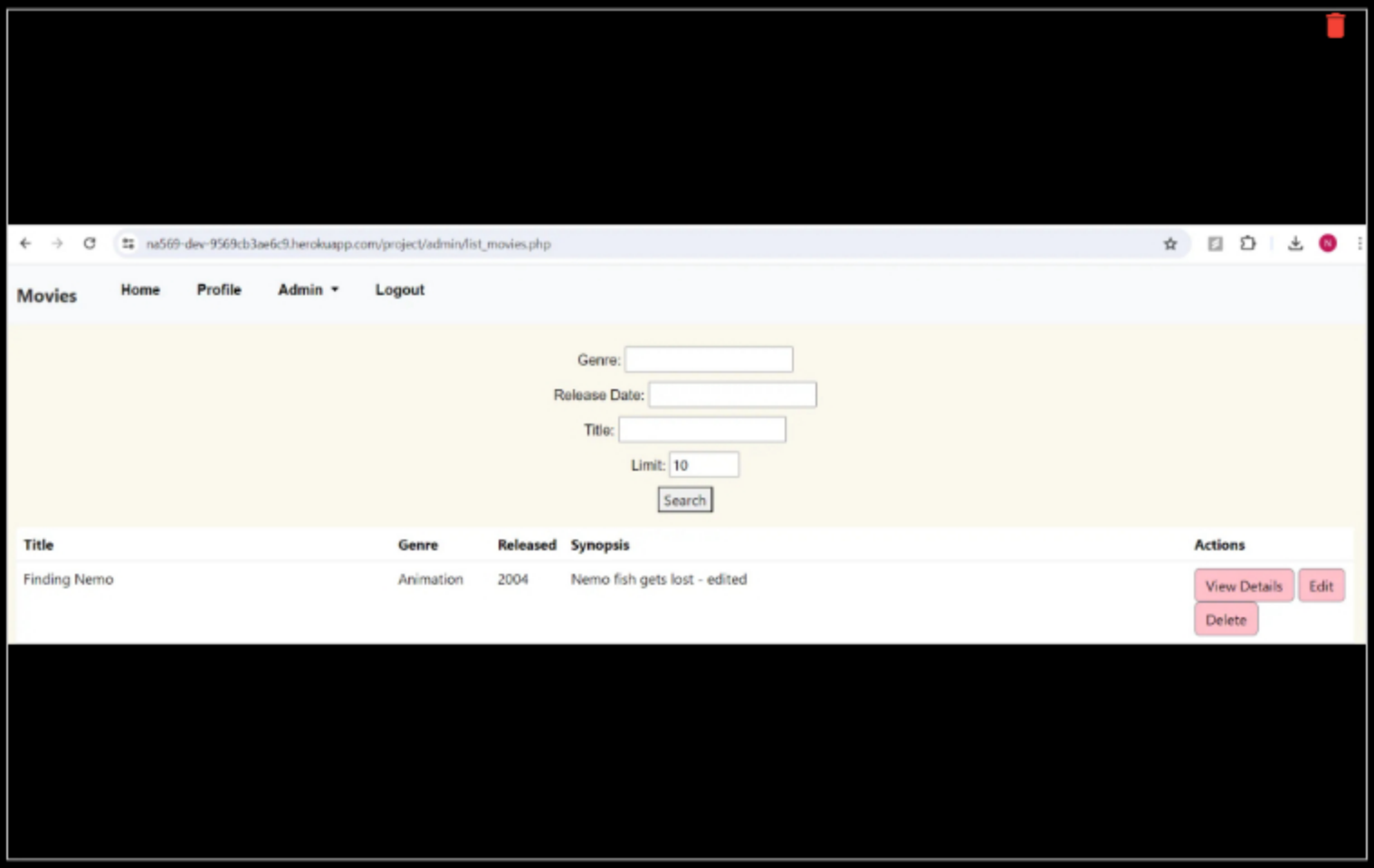
#1 Show before and after screenshots of data you'll edit



example of validation message when movie year is not a valid year

Checklist Items (1)

#2 Show examples of validation messages



after screenshot of edited movie

Checklist Items (1)

#1 Show before and after screenshots of data you'll edit

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of edit page code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Form should have correct data types for each property being requested
#2	1	Form should have correct validation for each field (HTML, JS, and PHP)
#3	1	Successful edit should have a user-friendly message
#4	1	Any errors should have user-friendly messages
#5	1	Include any other rules like role guards and login checks
#6	1	Include ucid/date comments for each code screenshot
#7	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
?>
//na569, 4.24.24
<div class="container-fluid">
    <form method="POST" onsubmit="return validate(this);">
        <?php render_input(["type" => "text", "id" => "title", "name" => "title", "label" => "Title", "value" => $title, "rules" => ["required" => true]]); ?>
        <?php render_input(["type" => "text", "id" => "genre", "name" => "genre", "label" => "Genre", "value" => $genre, "rules" => ["required" => true]]); ?>
        <?php render_input(["type" => "number", "id" => "released", "name" => "released", "label" => "Release Year", "value" => $released, "rules" => ["required" => true]
        <?php render_input(["type" => "text", "id" => "synopsis", "name" => "synopsis", "label" => "Synopsis", "value" => $synopsis, "rules" => ["required" => true]]); ?>
        <?php render_input(["type" => "hidden", "name" => "save"]); ?>
        <?php render_button(["text" => "Update Movie", "type" => "submit"]); ?>
    </form>
</div>
```

form with the correct data types printed. HTML validation

Checklist Items (1)

#1 Form should have correct data types for each property being requested

```
//na569,4.24.24
<script>
    function validate(form) {
        //console.log("hello");

        let released = form.released.value;
        let isValid = true;

        // Regex to match four digits for the release year
        let validYear = /^\d{4}$/;

        if (!validYear.test(released)) {
            flash("[Client] Release year must be a four-digit number", "warning");
            isValid = false;
        }

        return isValid;
    } <- #29-44 function validate(form)
</script>
```

js validation

Checklist Items (1)

#2 Form should have correct validation for each field (HTML, JS, and PHP)

```php
<?php //na569, 4.24.24
if (isset($_POST["save"])) {
    $title = se($_POST, "title", null, false);
    $genre = se($_POST, "genre", null, false);
    $released = se($_POST, "released", null, false);
    $synopsis = se($_POST, "synopsis", null, false);
    $hasError = false;

    if (!preg_match('/^\d{4}$/', $released)) {
        flash("Release year must be a four-digit number", "warning");
        $hasError = true;
    }

    if (!$hasError) {
        $params = [
            ":title" => $title,
            ":genre" => $genre,
            ":released" => $released,
            ":synopsis" => $synopsis,
            ":id" => $movie_id
        ]; <- #61-67 $params =

        $db = getDB();
        $stmt = $db->prepare("UPDATE Movies SET title = :title, genre = :genre, released = :released, synopsis = :synopsis WHERE id = :id");

        try {
            $stmt->execute($params);
            flash("Movie information updated", "success");
        } catch (PDOException $e) {
            flash("An error occurred while updating movie information", "danger");
        }
    } <- #60-78 if (!$hasError)
} <- #48-79 if (isset($_POST["save"]))
?>
```

php validation, showcases success and unsuccessful flash messages

Checklist Items (1)

#2 Form should have correct validation for each field (HTML, JS, and PHP)

```php
<?php
require_once(__DIR__ . "/../../../partials/nav.php");
is_logged_in(true);
?>
//na569, 4.24.24
```

<?php

log-in check

Checklist Items (1)

#5 Include any other rules like role guards and login checks

COLLAPSE

Task #3 - Points: 1

Text: Screenshot of records from DB

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show a before and after screenshot of the record
#2	1	Note what differs
#3	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

Small Medium Large

before screenshot of record

Checklist Items (3)

#1 Show a before and after screenshot of the record

#2 Note what differs

#3 Clearly caption screenshots

after screenshot of record

Checklist Items (3)

#1 Show a before and after screenshot of the record

#2 Note what differs

#3 Clearly caption screenshots

COLLAPSE

Task #4 - Points: 1

Text: Explain the process

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Provide a high-level step-by-step of how the fetching of the record, populating the form, and the update works and gets changed in your DB
#2	1	Briefly describe the validations for the applicable fields

Response:

When the page is loaded with the movie ID in the URL, the movie details are fetched from the database and stored in variables. The variables then pre-fill the form fields in the HTML form. Javascript and PHP validation is used to make sure that the release year is a valid four digit number. When the form is submitted with no validation errors, the form data is used in an SQL update statement that updates the information in the database.

COLLAPSE

Task #5 - Points: 1

Text: Add related links

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Include the heroku prod link for this page
#2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1

https://na569-prod-74aadc581478.herokuapp.com/project/admin/edit_movie.php?id=789

URL #2

https://github.com/nafisa37/na569-it202-008/pull/52

Delete Handling (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots related to delete

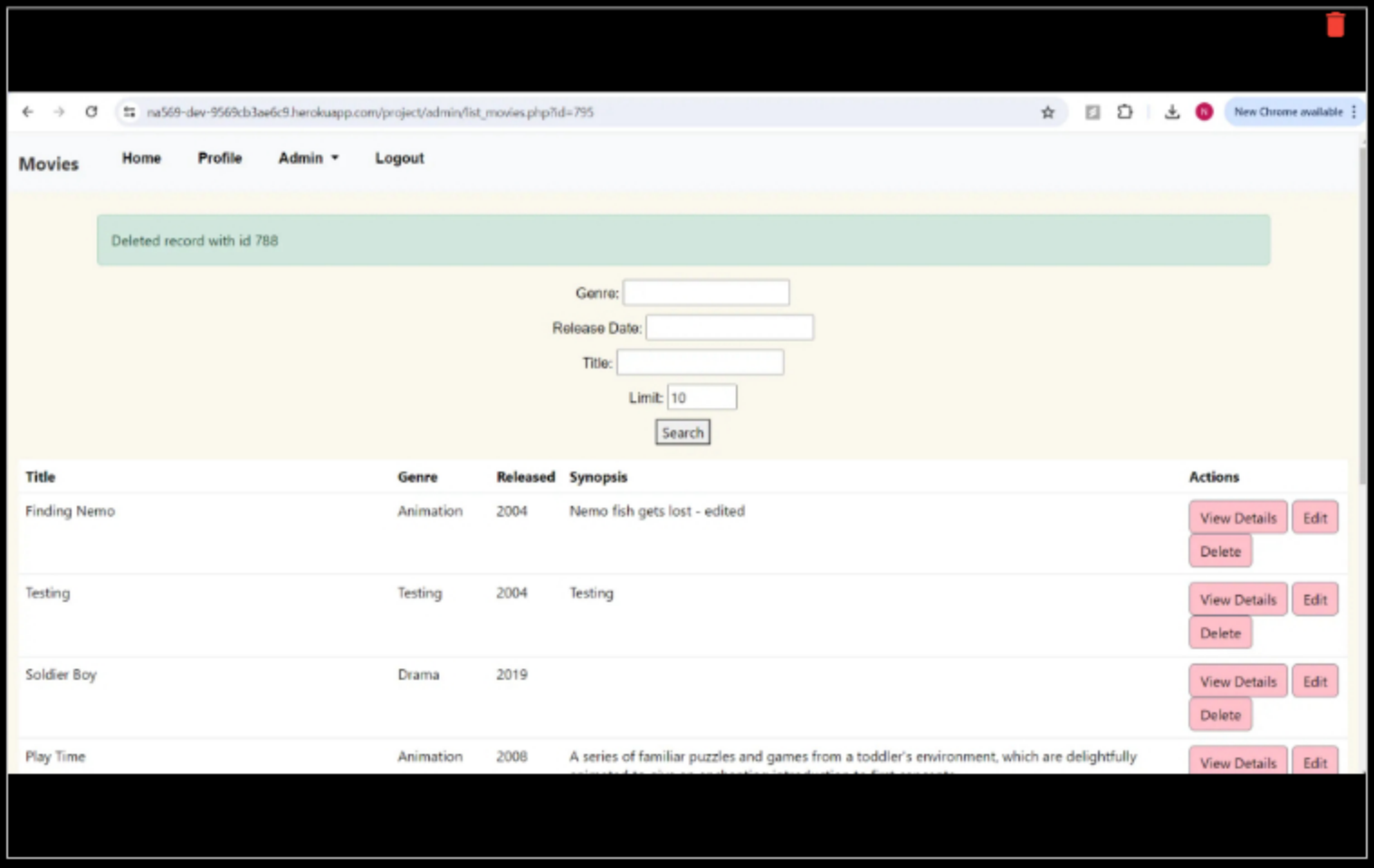
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the success message of a delete
#2	1	Show any error messages of a failed delete (like id not being passed)
#3	1	Show the code related to the delete processing

Task Screenshots:

Gallery Style: Large View

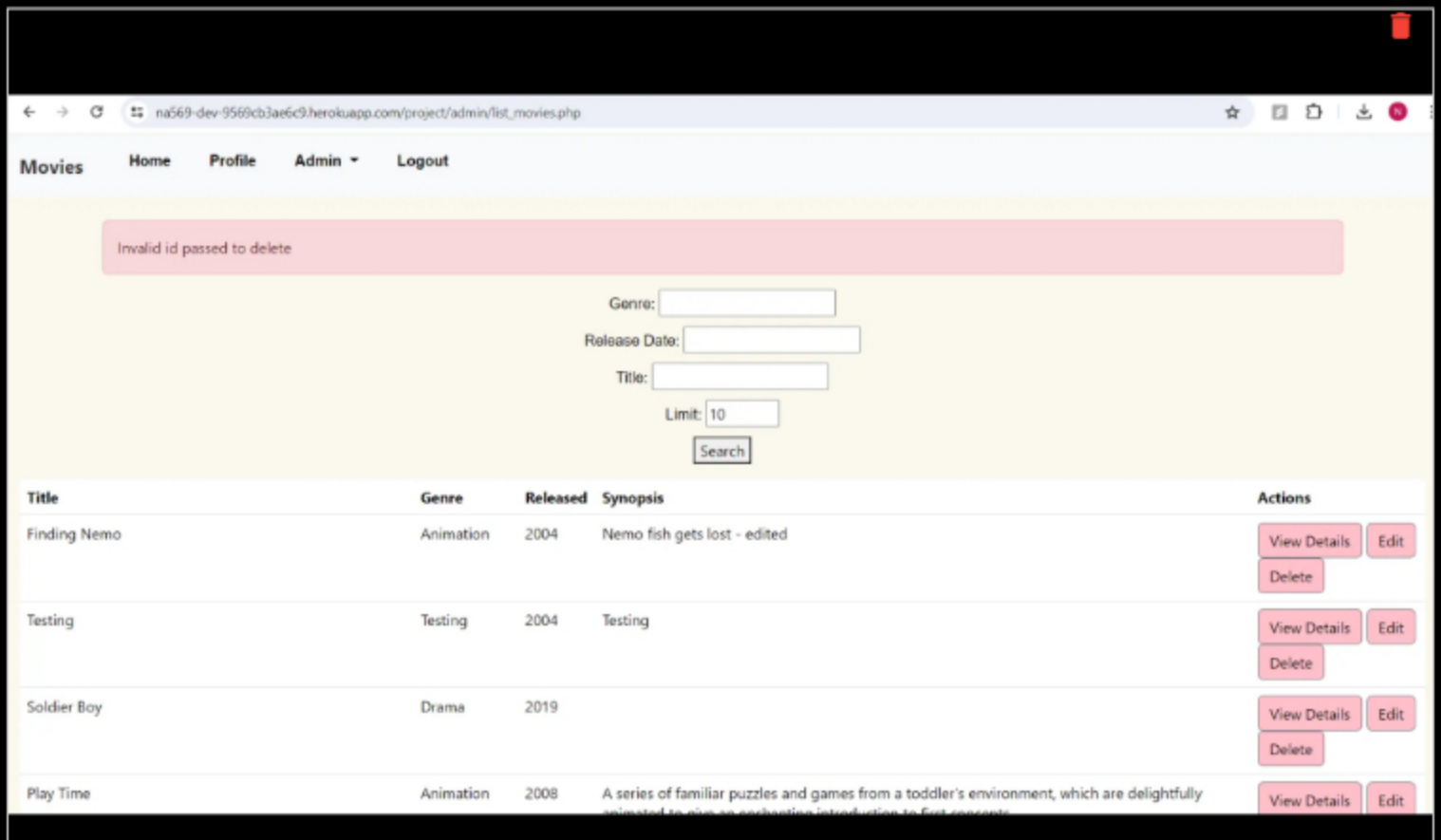
Small Medium Large



shows deletion of record from database

Checklist Items (1)

#1 Show the success message of a delete



showcases error message of failed delete

Checklist Items (1)

#2 Show any error messages of a failed delete (like id not being passed)

```php
    die(header("Location: $BASE_PATH" . "/home.php"));
}
//na569, 4/24/24       You, 2 days ago • Uncommitted changes
$id = se($_GET, "id", -1, false);
if ($id < 1) {
    flash("Invalid id passed to delete", "danger");
    die(header("Location: " . get_url("admin/list_movies.php")));
}

$db = getDB();
$query = "DELETE FROM `Movies` WHERE id = :id";
try {
    $stmt = $db->prepare($query);
    $stmt->execute([":id" => $id]);
    flash("Deleted record with id $id", "success");
} catch (Exception $e) {
    error_log("Error deleting movie $id" . var_export($e, true));
    flash("Error deleting record", "danger");
}

if (isset($_SERVER['HTTP_REFERER'])) {
    $previousPage = $_SERVER['HTTP_REFERER'];
} else {
    $previousPage = get_url("admin/list_movies.php");
}

if (!empty($_SESSION['filter'])) {
    $filterParams = http_build_query($_SESSION['filter']);
    if (strpos($previousPage, '?') !== false) {
        $separator = '&';
    } else {
        $separator = '?';
    }
    $previousPage .= $separator . $filterParams;
} <- #33-41 if (!empty($_SESSION['filter']))

die(header("Location: " . $previousPage));
```

code related to delete processing

Checklist Items (1)

#3 Show the code related to the delete processing

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the data

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show a before and after screenshot of the DB data
#2	1	Note what changed (i.e., record removed or soft delete value changed)
#3	1	Clearly caption screenshots

Task Screenshots:

Gallery Style: Large View

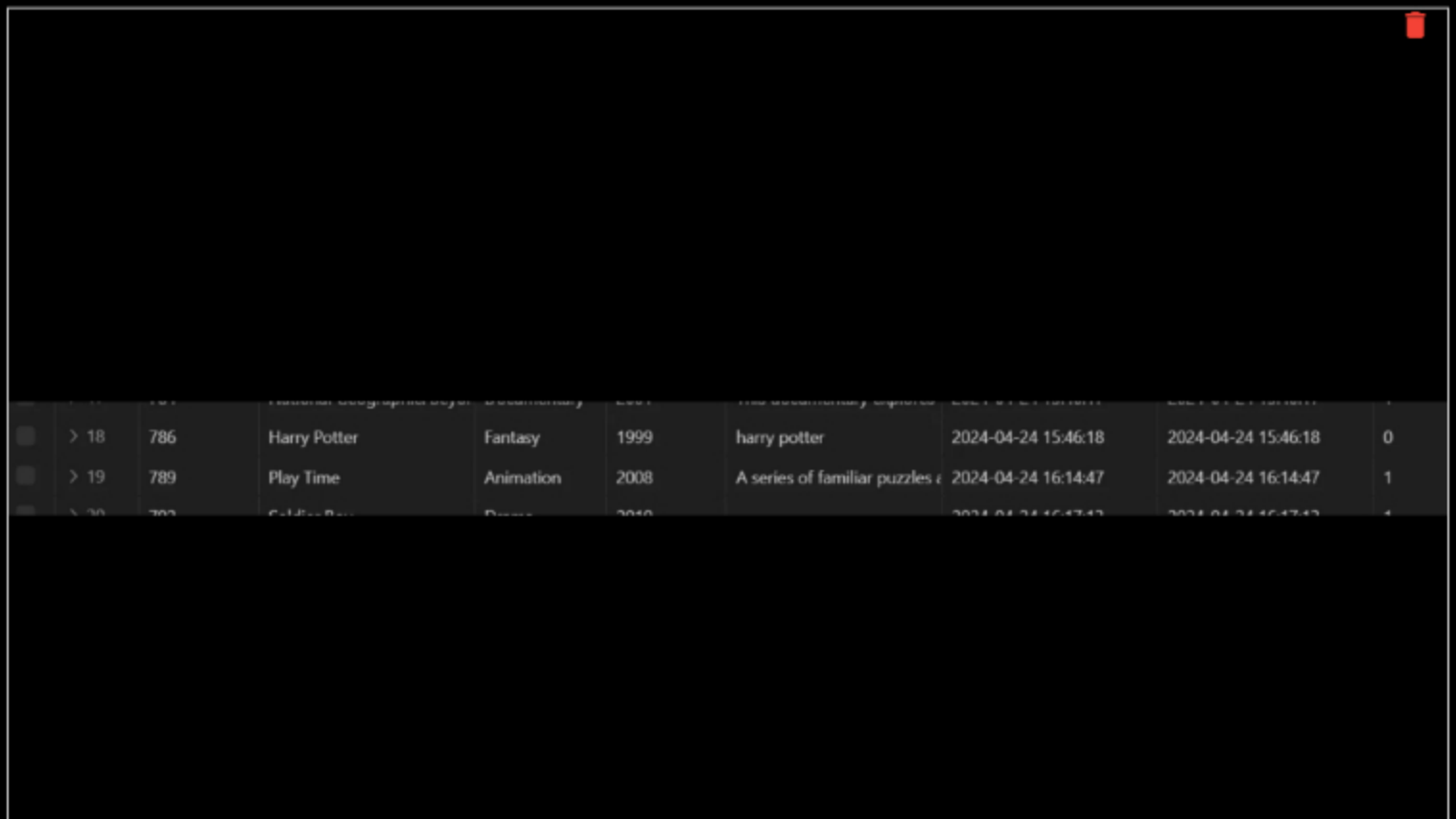
Small Medium Large

18	786	Harry Potter	Fantasy	1999	harry potter	2024-04-24 15:46:18	2024-04-24 15:46:18	0
19	788	Cesar Millan: Better Human	Reality-TV	2021		2024-04-24 15:55:10	2024-04-24 15:55:10	1
20	789	Play Time	Animation	2008	A series of familiar puzzles a	2024-04-24 16:14:47	2024-04-24 16:14:47	1

before deleting movie with ID 788

Checklist Items (1)

#1 Show a before and after screenshot of the DB data



18	786	Harry Potter	Fantasy	1999	harry potter	2024-04-24 15:46:18	2024-04-24 15:46:18	0
19	789	Play Time	Animation	2008	A series of familiar puzzles a	2024-04-24 16:14:47	2024-04-24 16:14:47	1

after deleting record 788 (record removed)

Checklist Items (1)

#1 Show a before and after screenshot of the DB data

COLLAPSE

Task #3 - Points: 1

Text: Explain the delete logic

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Is it a soft or hard delete
#2	1	Are there any necessary roles or restrictions? (can only delete their data, can only be done by admin, etc)
#3	1	Provide the high-level steps for handling the DB lookup and handling the delete

Response:

This is a hard delete. Deletes can only be done by the admins. The movie ID is retrieved using $_GET("id"). getDB() connects to the database, and an SQL query is created to delete the movie from the Movies table based on the movie ID. If the movie record is deleted, a success flash message showcases. Otherwise, a danger flash message informs user of the issue. It then redirects back to the original page with the previous filters set.

COLLAPSE

Task #4 - Points: 1

Text: Add the pull request link for the branch related to this feature

Details:

Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1

https://github.com/nafisa37/na569-it202-008/pull/53

URL #2

https://github.com/nafisa37/na569-it202-008/pull/49

Misc (1 pt.)

COLLAPSE

COLLAPSE

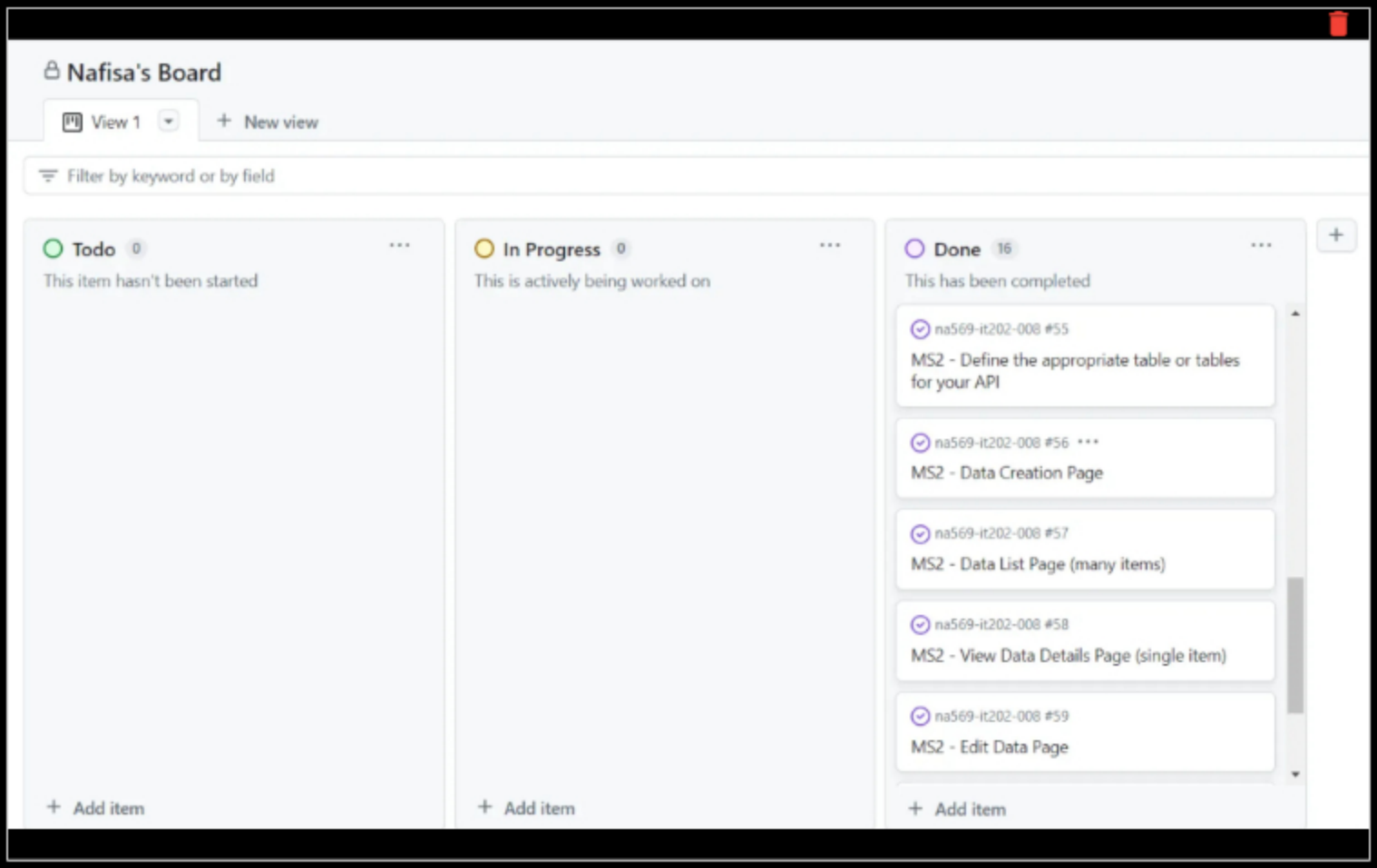
Task #1 - Points: 1

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

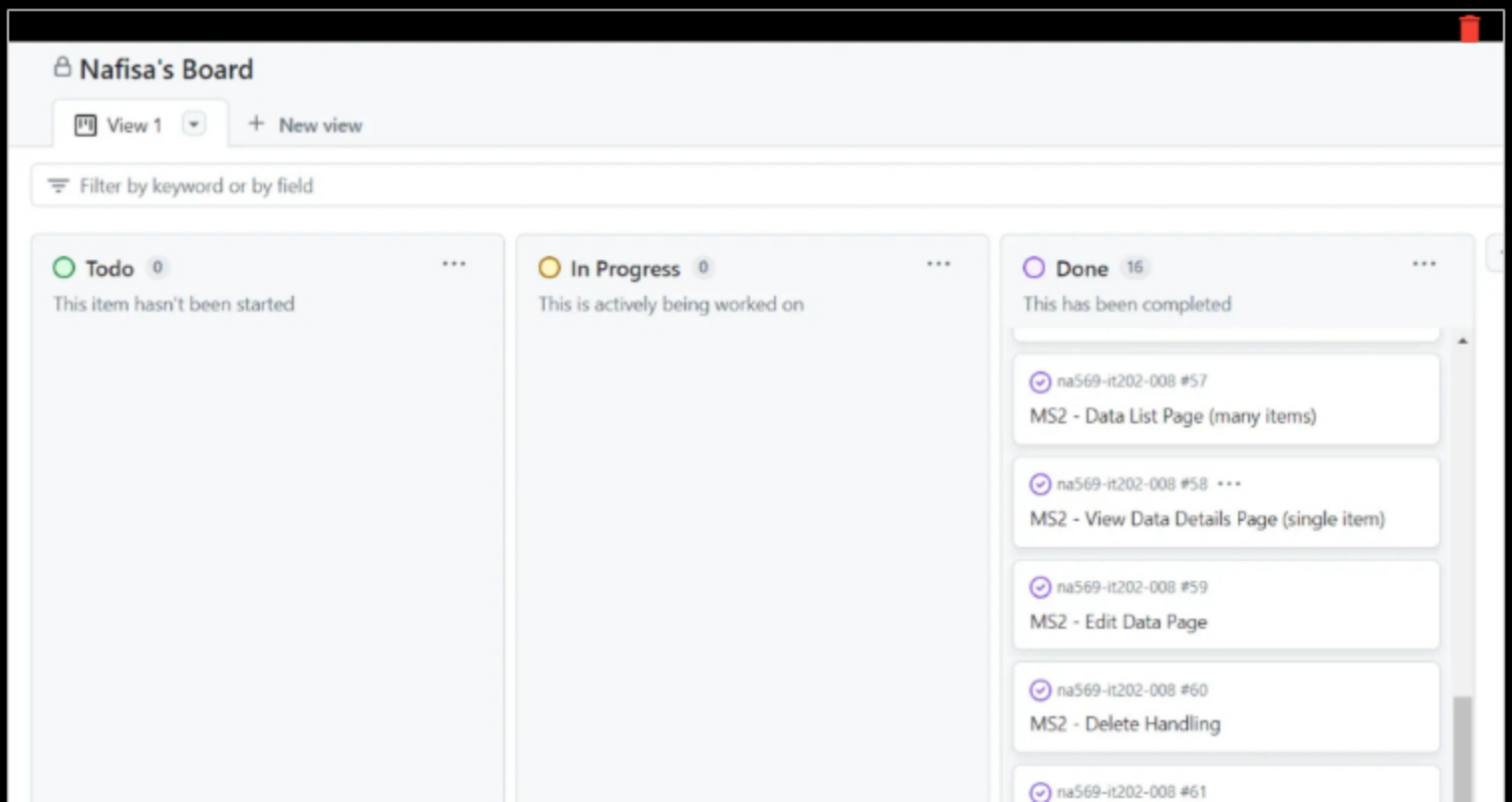
Task Screenshots:

Gallery Style: Large View

Small Medium Large



project board will all tasks completed - part 1



project board from GitHub will all tasks completed - part 2

COLLAPSE

Task #2 - Points: 1

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

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

COLLAPSE

Task #3 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

My biggest issue was getting the API correctly. I had to filter out the data properly so I only received the information I wanted, and get it to enter the database correctly. It took me a couple days to get this working, but the rest of the page did not prove to be that difficult with the help of the starter code provided to us.

COLLAPSE

Task #4 - Points: 1

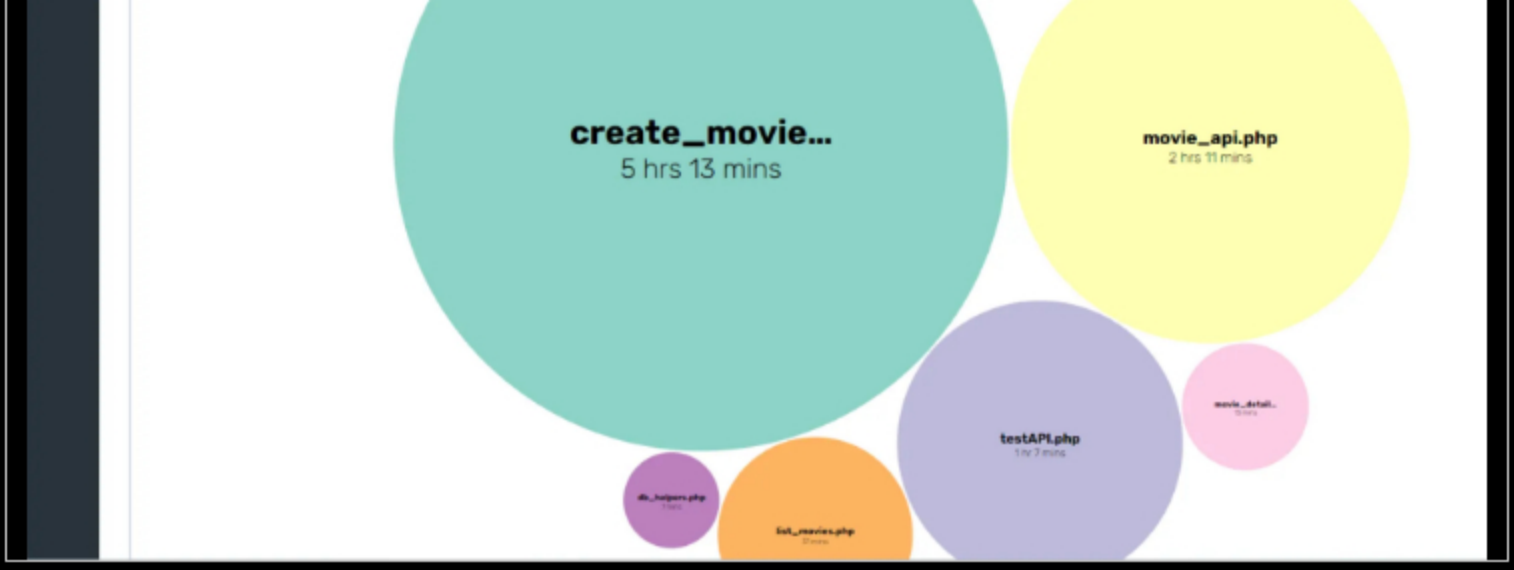
Text: WakaTime Screenshot

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: Large View

Small Medium Large



waka screenshot showcases how much time for each file

End of Assignment