Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-452-M2024/it202-api-project-milestone-2-2024-m24/grade/dr475

IT202-452-M2024 - [IT202] API Project Milestone 2 2024 M24

Submissions:

Submission Selection

1 Submission [active] 7/23/2024 3:55:51 PM

Instructions

COLLAPSE

Implement the Milestone 2 features from the project's proposal document: https://docs.google.com/document/d/1XE96a8DQ52Vp49XACBDTNCq0xYDt3kF29cO88EWVwfo/view 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: 23 Points: 10.00

Define Appropriate Tables for Data (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

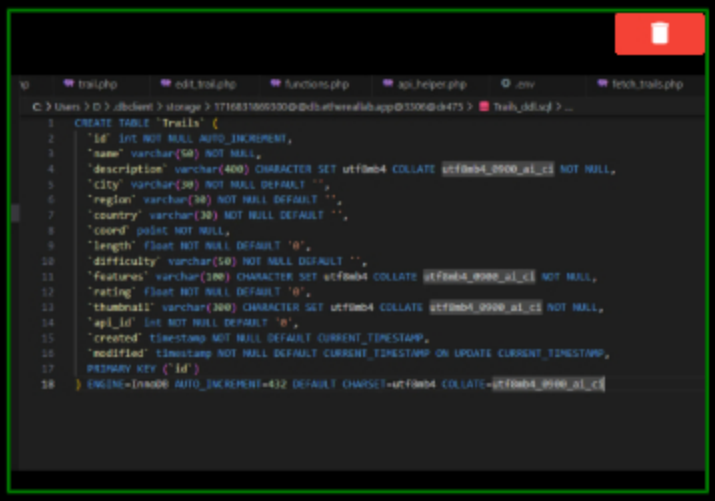
Text: Screenshots of Table SQL

Checklist *The checkboxes 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

#1) Screenshot of the table (you can duplicate this subtask as needed)



```
CREATE TABLE `Trails` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(50) NOT NULL,
  `description` varchar(400) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `city` varchar(30) NOT NULL DEFAULT '',
  `region` varchar(30) NOT NULL DEFAULT '',
  `country` varchar(30) NOT NULL DEFAULT '',
  `coord` point NOT NULL,
  `length` float NOT NULL DEFAULT '0',
  `difficulty` varchar(50) NOT NULL DEFAULT '',
  `features` varchar(100) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `rating` float NOT NULL DEFAULT '0',
  `thumbnail` varchar(300) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci NOT NULL,
  `api_id` int NOT NULL DEFAULT '0',
  `created` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `modified` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=432 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
CREATE TABLE `User_Trails` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int NOT NULL,
  `trail_id` int NOT NULL,
  PRIMARY KEY (`id`),
  KEY `trail_id` (`trail_id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `User_Trails_ibfk_1` FOREIGN KEY (`trail_id`) REFERENCES `Trails` (`id`),
  CONSTRAINT `User_Trails_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `Users` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

Caption (required) ✓

Describe/highlight what's being shown

SQL Table structures

Explanation (required) ✓

Explain the columns and what data they represent (briefly), also note any normalization that may have been necessary

PREVIEW RESPONSE

`Trails`:

Name, Description, city, region(state), country, length, difficulty, features, thumbnail - Retrieved from the API

Coord: I created a geometric point using the `lat` and `long` from the API so I can find the distance from each coord in the future (When user searches for coords given a lat, long, and radius, I am able to calculate the distance in SQL and search for trails that are in that distance)

API_ID: Just the ID from the API

Rating: For use in future user_ratings

`User_Trails`:

user_id and trail_id are both foreign keys (From `Users` and `Trails` respectively)

The purpose of this table is to keep track of trails that users submitted.

COLLAPSE

Task #2 - 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/davidredziniak/dr475-IT202-452-M2024/pull/58

URL #2
https://github.com/davidredziniak/dr475-IT202-452-M2024/pull/64

url
https://github.com/davidredziniak/dr475-IT202-4

url
https://github.com/davidredziniak/dr475-IT202-4

+ ADD ANOTHER URL

Data Creation Page (2 pts.)

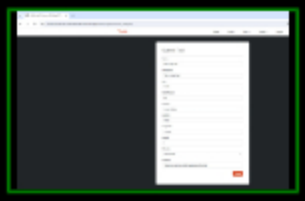
COLLAPSE

COLLAPSE

Task #1 - Points: 1
Text: Screenshots of the creation page

Details:
Heroku dev url must be visible in all relevant screenshots

#1) Show potentially valid data



Caption (required) ✓
Describe/highlight what's being shown
The user trail submission page

#2) Show how the API data is

Caption (required) ✓
Describe/highlight what's being shown
API data fetching via admin page

Explanation (required) ✓
Briefly explain the code
PREVIEW RESPONSE
This is a screenshot of the admin/fetch_trails.php

#3) Show examples of validation

Caption (required) ✓
Describe/highlight what's being shown
Error validation on form information for both admin and user trail page

#4) Show an example of successful

Caption (required)
Describe/highlight what's being shown
Missing caption

page. An admin can pass through params (lat, long, and radius) in order to fetch API data using the parameters. The resulting data is then processed and submitted to the DB, allowing users to then search for trails by country, length, lat, long, or difficulty.

#5) Design/Style should be

Caption (required) ✓

Describe/highlight what's being shown

Bootstrap Styling

Explanation (required) ✓

Briefly explain your design choices

PREVIEW RESPONSE

I went with bootstrap because it is super simple to use (grid system is great). I went with a simplistic design with a navbar for quick navigation and each page has a similar container style.

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of creation page code

Details:
Include ucid/date comments for each code screenshot

#1) Form should have correct data

Caption (required) ✓
Describe/highlight what's being shown
Form for user submit_trail.php, and admin fetch_trails.php

Explanation (required)
✓
Briefly talk about each field type and why it was chosen
PREVIEW RESPONSE
I used mainly text inputs so I can just validate the form using regex, but for the user submit_trail page I used number input fields for the length of the trail, and a select input for "difficulty" to choose between four different options.

#2) Form should have correct

Caption (required) ✓
Describe/highlight what's being shown
Admin fetch_trails and user submit_trail page JS validation

Explanation (required)
✓
Briefly explain the validations
PREVIEW RESPONSE
For each of the forms, it calls the validate() function passing in the form.

For each form, it checks if the radius/lengths are in range, if any of the fields are empty and if they are of correct length, and tests the regex for values that require it (name, lat, long).

#3) Successful creation

Caption (required) ✓
Describe/highlight what's being shown
User and Admin successful insert to DB

Explanation (required)
✓
Explain how duplicate/existing data is handled
PREVIEW RESPONSE
There is currently no duplicate trail checking for user submissions, but there is for admin API data fetching.

Using a crafted SQL query, it checks whether or not the API_ID exists in the `Trails` table before inserting the trail data. If it exists, it skips it, if it doesn't it is able to insert it.

#4) Any errors should have

Caption (required) ✓
Describe/highlight what's being shown
User and Admin error handling for inserting to DB

Explanation (required)
✓
Briefly describe the scenarios
PREVIEW RESPONSE
For the user trail submission, if there is any error while submitting a new record then it shows the user a flash message. A scenarios is: there was an error submitting into the `User_Trails` table (which keeps track of user trails submissions)

For the API trail fetching, if there is any error while submitting a new record into the DB then it shows the admin a flash message that indicates there was an error. Scenario: API doesn't work (limit reached),

#5) Include

#6) Include

#7) Include

the form/process

Caption (required) ✓
Describe/highlight what's being shown
Files that are used to insert new trails to DB using API

Explanation (required)
✓
Briefly explain the steps (include how duplicates are handled)

PREVIEW RESPONSE

1. The admin goes to fetch_trails.php in the admin section
2. They insert the lat, long, and radius to get the trails in that area from the API
3. The API is called with the parameters and a response with data is received
4. fetch_trails.php uses the process_trails() function from manage_trail_data.php in lib passing it

some indicator

Caption (required) ✓
Describe/highlight what's being shown
Condition to skip over duplicate trails

Explanation (required)
✓
Briefly mention what the indicator is (i.e., api_id if the API has ids or is_api as a boolean-like column, etc)

PREVIEW RESPONSE

The `Trails` table has a column: `api_id` that is 0 if it is user-submitted and a number other than 0 to indicate the id extracted from the API.

any other rules like

Caption (required) ✓
Describe/highlight what's being shown
Admin role

Explanation (required)
✓
Briefly explain the logic/reasoning

PREVIEW RESPONSE

The page to submit user trails is not protected (any user can submit), however the admin page to use the API to fetch trails is.

the response data
5. process_trails then decodes the JSON and iterates on the trails data
6. For each iteration, it passes the JSON object to process_single_trail() which extracts and transforms the data into a better format for the database schema.
7. After iterating and processing each trail JSON object from the response, the built array of processed objects is passed to insert_trails_into_db() where it is then processed into the DB.
8. The SQL query in insert_trails_into_db allows the function to skip over duplicates.

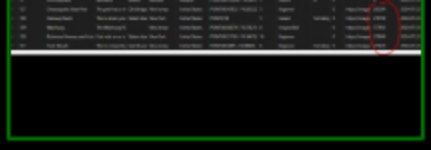
^COLLAPSE^

Task #3 - Points: 1

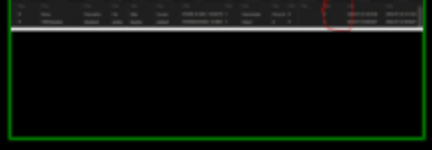
Text: Screenshot of records from DB

#1) Show at least one record fetched from the API

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



Caption (required) ✓
Describe/highlight what's being shown (note what differs from a custom record)
API_ID is not 0



Caption (required) ✓
Describe/highlight what's being shown (note what differs from the API record)
API_ID is 0

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://it202-dr475-prod-7559be2e2ad4.herokuapp.com/Project/submit_trail.php
URL https://it202-dr475-prod-7559be2e2ad4.herokua

URL #2
https://it202-dr475-prod-7559be2e2ad4.herokuapp.com/Project/admin/fetch_trails.php
URL https://it202-dr475-prod-7559be2e2ad4.herokua

URL #3
https://github.com/davidredziniak/dr475-IT202-452-M2024/pull/58
URL https://github.com/davidredziniak/dr475-IT202-4

URL #4
https://github.com/davidredziniak/dr475-IT202-452-M2024/pull/71
URL https://github.com/davidredziniak/dr475-IT202-4

URL #5
https://github.com/davidredziniak/dr475-IT202-452-M2024/pull/72
URL https://github.com/davidredziniak/dr475-IT202-4

+ ADD ANOTHER URL

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

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the list page

Details:
Heroku dev url must be visible in all relevant screenshots

#1) Show the page of your entities

Caption (required) ✓
Describe/highlight what's being shown
List of trails with country: United States

Explanation (required) ✓
Briefly describe the page
PREVIEW RESPONSE
This is a page with a listing of trails searched by the country: United States and default limit of records (10).

#2) Show the filter/sort

Caption (required) ✓
Describe/highlight what's being shown
Two forms required for searching

Explanation (required) ✓
Briefly mention what's available to the user
PREVIEW RESPONSE
The view trails page has two different forms for searching. If the user selects the left form (Location) the latitude, longitude, radius is required. If the user selects the right form (by spec), at least one of (country, max length of trail, difficulty) is required. The limit for both forms is in the range 1 to 100 records.

#3) Demonstrate a few varied

Caption (required) ✓
Describe/highlight what's being shown
3 searches - 1: Advanced trails less than 3 mi in length, 2: Trails in the US that are difficulty easy, 3: Location search

#4) Demonstrate a filter that

Caption (required) ✓
Describe/highlight what's being shown
Country: Russia, no trails found

#5) Each list item should have a link

#6) Each list item should have a

#7) Design/Style should be

Caption (required) ✓
Describe/highlight what's being shown
Details of trails

Explanation (required)
✓
Mention which users can interact with the view, edit, and delete links
PREVIEW RESPONSE
View: All users can use
Edit: Only admin/users that submitted
Delete: Only admin/users that submitted

Caption (required) ✓
Describe/highlight what's being shown
Summary details of each trail

Caption (required) ✓
Describe/highlight what's being shown
Summary details of each found trail

Explanation (required)
✓
Briefly explain your design choices
PREVIEW RESPONSE
I used bootstrap for styling the site.
The view trails container is divided into two columns, and a row for each trail. The left column is the summary details, and the right column is the available links for each trail. I wanted to keep the details page simple, though I will work on the UI/UX in milestone 3.

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the list page code

Details:
Include ucid/date comments for each code screenshot

#1) Show the filter/sort

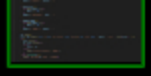
#2) Show the DB query and how the

#3) Show how the output is

#4) Show any restrictions

Caption (required) ✓
Describe/highlight what's being shown
Forms for searching

Explanation (required)
✓
Briefly explain how the code works (i.e, some stuff may be dynamic)
PREVIEW RESPONSE
There are two forms, one for searching by location, one by searching by specification. The searching by specification allows the user to input params (country, max length, difficulty) and allows the user to sort by each. If the user specified US as the country, they can also sort the result by setting the difficulty option and vice versa.



Caption (required) ✓
Describe/highlight what's being shown
Sort by specification
PHP code

Explanation (required)
✓
Briefly explain the related logic
PREVIEW RESPONSE
The filtering/sort is handled by the server given the params passed by the user.
Three params: Country, max length, difficulty
Optional param: Limit

The server first sanitizes and validates the form data, then builds the query based on the filter the user has provided. (Third image)
Then the SQL query selects id, name, city, country, length, difficulty (all included in the summary) and appends the filter after the WHERE clause, along with LIMIT if specified (10 default).

Caption (required) ✓
Describe/highlight what's being shown
HTML code generation

Explanation (required)
✓
Briefly explain the related logic
PREVIEW RESPONSE
Upon a successful search in the DB, the result array gets populated with fetched trails. The page then generates

for each trail and populates it with the trail summary data (name, length, difficulty, etc) for user output.

Caption (required) ✓
Describe/highlight what's being shown
User must be logged in

Explanation (required)
✓
Briefly explain the logic/reasoning
PREVIEW RESPONSE
The user must be logged into the site in order to view trails.

COLLAPSE

Task #3 - 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://it202-dr475-prod-7559be2e2ad4.herokuapp.com/Project/view_trails.php

URL
https://it202-dr475-prod-7559be2e2ad4.herokua

URL #2
https://github.com/davidredziniak/dr475-IT202-452-M2024/pull/65

URL
https://github.com/davidredziniak/dr475-IT202-4

URL #3
https://github.com/davidredziniak/dr475-IT202-452-M2024/pull/70

URL
https://github.com/davidredziniak/dr475-IT202-4

+ ADD ANOTHER URL

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

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the details page

Details:
Heroku dev url must be visible in all relevant screenshots

#1) Entity should be fetch by id

Caption (required) ✓
Describe/highlight what's being shown
Trail detail page by ID

#2) A missing id should

Caption (required) ✓
Describe/highlight what's being shown
Trail with an invalid ID

#3) Design/Style should be

Caption (required) ✓
Describe/highlight what's being shown
Detail page

Explanation (required)
✓
Briefly explain your design choices

#4) Data shown should be

Caption (required) ✓
Describe/highlight what's being shown
Detail page

Explanation (required)
✓
Briefly explain the details shown and how it

design choices

PREVIEW RESPONSE

The old styling CSS was ugly so I chose to use bootstrap to allow for quick styling. I stuck to the same theme as the rest of my website with white containers and a dark background to distinguish what content is important and is being shown.

details shown and how it differs from the list view

PREVIEW RESPONSE

I opted for a simplistic view with an image on the right (retrieved by the API, in the future I plan to allow thumbnail uploads on user submissions), along with a full list of details (description, location under image, features, length, city, state, country).

#5) There should be a link to edit

Caption (required) ✓

Describe/highlight what's being shown

Edit link under image

#6) There should be a link to delete

Caption (required) ✓

Describe/highlight what's being shown

Delete link under delete

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the details page code

Details:

Include ucid/date comments for each code screenshot

#1) Show how id is fetched

#2) Show the DB query to get the

#3) Show the code related to

Caption (required) ✓
Describe/highlight what's being shown
PHP code that retrieves the trail by ID

Explanation (required)
✓
Briefly explain the logic and how it's handled when the property is missing or not "valid"
PREVIEW RESPONSE
Upon page load, the id= parameter is set and is passed through one of the trail_helpers.php function get_trail_by_id(). The id is then used to select all the required trail info from the database WHERE id = trail ID. When any property is missing, it is echoed as an empty string since the trail.php page uses saferecho when calling the trail object's variables. If the thumbnail is missing it displays the alt="Hiking Trail Image" instead of an image.

Caption (required) ✓
Describe/highlight what's being shown
DB Select Query

Explanation (required)
✓
Briefly explain the logic
PREVIEW RESPONSE
Given a trail ID, it uses a SELECT query to locate a record from the DB where id = the requested trail id.

Caption (required) ✓
Describe/highlight what's being shown
Trail detail HTML generation

Explanation (required)
✓
Briefly explain the logic
PREVIEW RESPONSE
Upon a successful retrieval of trail data into an array from the DB given by the id parameter in the URL. The page is dynamically generated using the trail's data.

COLLAPSE

Task #3 - 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://it202-dr475-prod-7559be2e2ad4.herokuapp.com/project/trail.php?id=438

URL
https://it202-dr475-prod-7559be2e2ad4.herokua

URL #2
https://github.com/davidredziniak/dr475-IT202-452-M2024/pull/70

URL
https://github.com/davidredziniak/dr475-IT202-4

URL #3
https://github.com/davidredziniak/dr475-IT202-452-M2024/pull/66

URL
https://github.com/davidredziniak/dr475-IT202-4

+ ADD ANOTHER URL

Edit Data Page (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the edit page

Details:
Heroku dev url must be visible in all relevant screenshots

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

Caption (required) ✓
Describe/highlight what's being shown
Before and after editing a trail

Explanation (required) ✓
Briefly explain what you changed

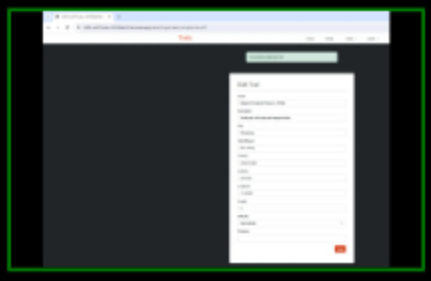
PREVIEW RESPONSE

I changed the name from:
Rutgers Ecological Preserve
to

#2) Show examples of validation messages

Caption (required) ✓
Describe/highlight what's being shown
Error Validation Messages

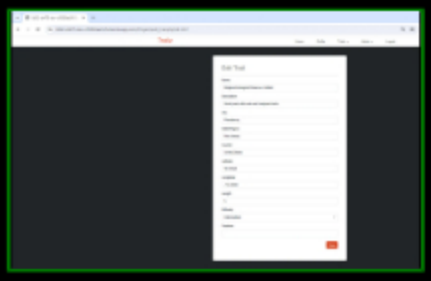
#3) Show an example of successful edit messages



Caption (required) ✓
Describe/highlight what's being shown
Successfully edited the name. The success message is uniform among all inputs

Rutgers Ecological Preserve - Edited

#4) Design/Style should be considered (i.e. bootstrap



Caption (required) ✓

Describe/highlight what's being shown

Edit trail page

Explanation (required) ✓

Briefly explain your design choices

PREVIEW RESPONSE

Implemented bootstrap, and again went with a simplistic view to edit the details. I changed the primary button color to orange to match the color palette of my site.

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of edit page code

Details:

Include ucid/date comments for each code screenshot

#1) Form should have correct data

#2) Form should have correct

#3) Successful edit should

#4) Any errors should have

Caption (required) ✓
Describe/highlight what's being shown
HTML Form

Explanation (required)
✓
Briefly explain the data type choices
PREVIEW RESPONSE
I went with the same data types as the submit_trail form.



Caption (required) ✓
Describe/highlight what's being shown
JS and PHP validation

Explanation (required)
✓
Briefly explain the validations
PREVIEW RESPONSE
First, the JS validation is called by the form and returns false if any validation fails. JS validation checks if any input is empty, if any value doesnt pass regex (lat, long, length, name of trail), if the range of the trail length is invalid (negative or 0), if the dropdown selection is valid, and if the length of the input strings exceed the length that the DB allows. The PHP validation does the same exact thing.

Caption (required) ✓
Describe/highlight what's being shown
Successful update into the DB

Explanation (required)
✓
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
PREVIEW RESPONSE

1. The user loads the edit_trail.php page with a valid id parameter set in the URL. edit_trail.php?id=388
2. On the server side, the id is parsed and then is checked if it is a valid trail using a helper function is_valid_trail() that is passed the id. If it is not valid, it redirects back to the list page. If it is valid, the logic continues.
3. Then the edit_trail.php page checks if the session user is an Admin or the user that submitted the trail. If they don't have permission, they are also redirected to the list page.

Caption (required) ✓
Describe/highlight what's being shown
DB Query for updating

Explanation (required)
✓
Briefly explain the possible errors
PREVIEW RESPONSE
A possible error: Bypassing PHP validation because of a bug, thus preparing the DB query with invalid values. It will cause an error while executing the SQL query.

4. After, the helper function get_trail_by_id() is passed the trail id and the return value is an array of keys and values that is retrieved by executing a SELECT query WHERE id = trail id from the database.
5. Since the page now has a array populated with the trail's data, it is able to continue rendering the DOM of the page and the form data is subsequently populated with the trail data from the array.
6. The user is now able to edit the form how they like, with javascript validation preventing the form from being submitted to the server.
7. Upon a valid form which passes validation, the user sends a POST request to the server with the "save" parameter set, along with all the form data in the body of the POST request

8. The PHP server receives this request and the form data in the body. It knows it is updating table data because the "save" variable is set on the edit_trail.php page.
9. The body data is sanitized and validated by checking for emptiness, testing values with regex, and making sure the length of the variables are within the constraints of the database schema.
10. If there are errors found, the error is shown to the user and the process is stopped before it can reach the DB.
11. If there are no errors, an SQL query is prepared and executed using the trail's updated data requested by the user.
12. If there is an error updating the trail, it is shown. If there is no error, the user is greeted with a success message:

"Successfully edited the trail!".
13. The updated trail data is then fetched and updated to the $trail variable (initially required to render the form) so the user sees a new form with the updated data.

#5) Include any other rules like

Caption (required) ✓
Describe/highlight what's being shown
User permissions checking

Explanation (required) ✓
Briefly explain the logic/reasoning
PREVIEW RESPONSE
Again, similar to other pages, the user has to be logged in to even view the page.
The page also requires that the user either has an admin role or is the user that submitted the trail for editing.

COLLAPSE

Task #3 - Points: 1

Text: Screenshot of records from DB

#1) Show a before and after screenshot of the record (two

Caption (required) ✓

Describe/highlight what's being shown

Db record before + after

Explanation (required) ✓

Explain what differs

PREVIEW RESPONSE

Description field was edited

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://it202-dr475-prod-7559be2e2ad4.herokuapp.com/project/edit_trail.php?id=437

URL

https://it202-dr475-prod-7559be2e2ad4.herokua

URL #2

https://github.com/davidredziniak/dr475-IT202-452-M2024/pull/67

URL

https://github.com/davidredziniak/dr475-IT202-4

+ ADD ANOTHER URL

Delete Handling (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots related to delete

Details:
Heroku dev url must be visible in all relevant screenshots

Include ucid/date comments for each code screenshot

#1) Show the success message of

Caption (required) ✓
Describe/highlight what's being shown
Successful delete

#2) Show any error messages of

Caption (required) ✓
Describe/highlight what's being shown
Id was not passed, delete failed

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The delete_trail.php page was called with the id blank. The PHP server understands the id parameter is set, but when checking if "" (blank) is a valid trail, it fails, this showing the user this error message. The same thing occurs when the ID does not exist in the database. If a user doesn't have delete permissions (not user that submitted or admin), the same message as other pages

#3) Show the code related to

Caption (required) ✓
Describe/highlight what's being shown
PHP Code for deletion

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
When a user wants to delete a trail, they call the delete_trail.php page with a GET request and the id parameter set to the trail ID they want to delete.
The Steps:

1. The PHP server checks if the user is logged in, if they aren't it redirects.
2. Then, the ID is checked using a helper function is_valid_trail()

#4) Explain the delete logic

Explanation (required)
✓
Is it a soft or hard delete? Are there any necessary roles or restrictions? (can only delete their data, can only be done by admin, etc)?
PREVIEW RESPONSE
It is a hard delete, all data is deleted from the DB.

If it is a user submitted trail, the respective record in `User_Trails` is also deleted. The `User_Trails` record needs to be deleted first before the record in `Trails` because `User_Trails` uses foreign keys.

Necessary Roles: Admin OR User that submitted the trail

(You don't have permission...") is presented, and redirected to view_trails.php page

that takes in the trail ID. If it is invalid, again, it redirects and lets the user know via an error message.

3. The PHP server checks for delete permissions (user submitted the trail OR admin). If they are neither, it redirects back to the view_trails.php page.
4. If everything has worked successfully thus far, it is time to delete the record from the database.
5. First, a SELECT query is used to select a record from `User_Trails` where trail_id = the trail ID requested to delete. If a record is found, it is deleted first, before the `Trails` record.
6. If there is no record found, it is an API submission.
7. Finally, the record from `Trails` is deleted WHERE id = trail id requested.
8. If the db query executions were

successful, the user sees the message "Successfully deleted the trail." and then is redirected to the previous page after 3 seconds (using javascript setTimeout).

9. If there were any errors, it is shown as an error message to the user.

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the data

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

Caption (required) ✓

Describe/highlight what's being shown (note precisely what changed)

Deleted the trail at ID=437

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/davidredziniak/dr475-IT202-452-M2024/pull/69

https://github.com/davidredziniak/dr475-IT202-4

+ ADD ANOTHER URL

Misc (1 pt.)

COLLAPSE

COLLAPSE

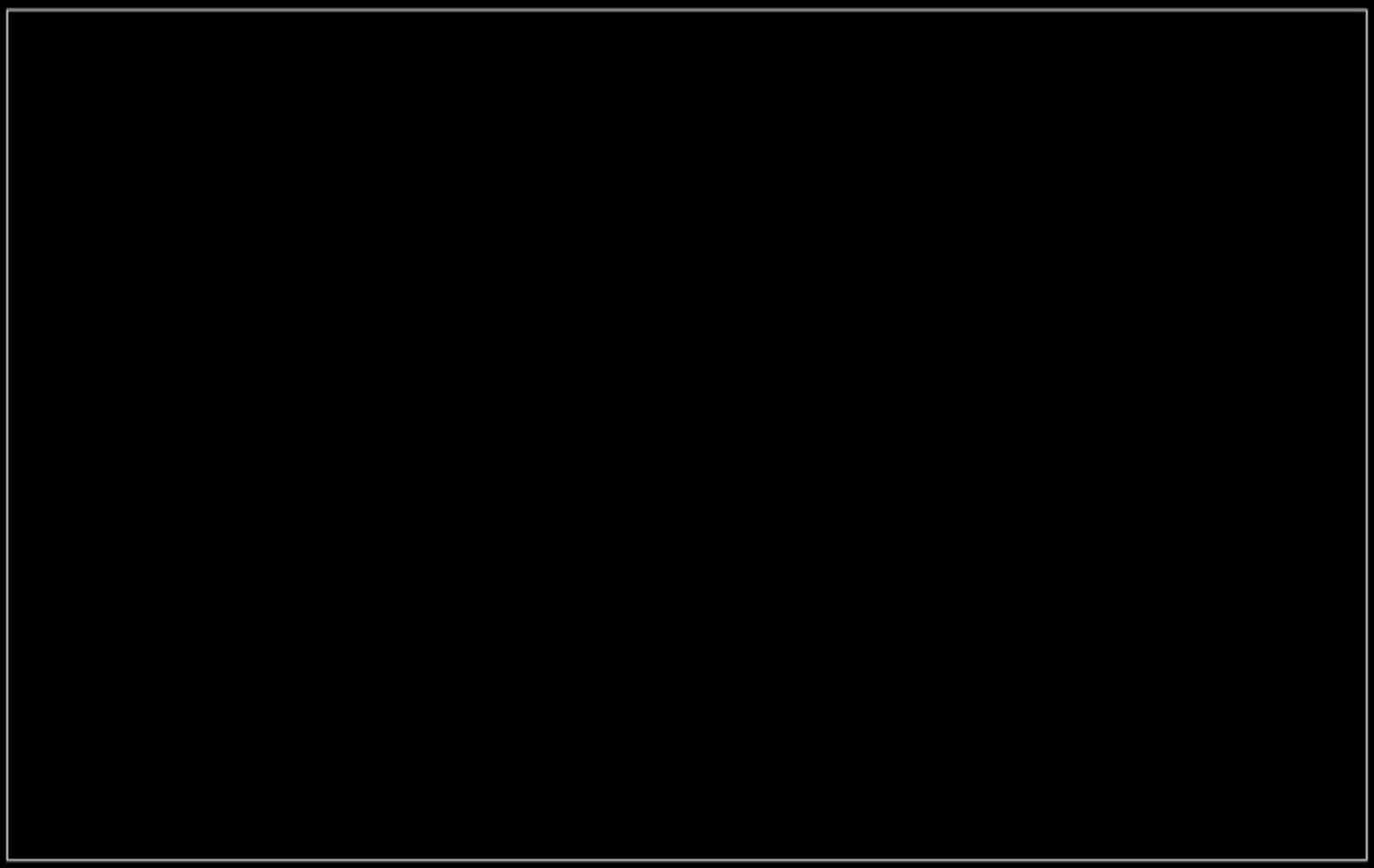
Task #1 - Points: 1

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Github Project Board

COLLAPSE

Task #2 - Points: 1

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

URL #1

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

https://github.com/users/davidredziniak/project

+ ADD ANOTHER URL

COLLAPSE

Task #3 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

Learned: I learned a lot about bootstrap styling (nav/container/row/grid system), fetching data from an API, proper handling of environmental variables, CURL, data mapping and dynamic HTML rendering.

Issues: Honestly there was a lot of trial and error regarding styling more than other aspects. A bunch of minor bugs regarding inconsistent data lengths among variables across different files (it gets hard to track as you continuously update different fields to work).

Overrall, it was a great refresh and I learned a lot. I haven't used PHP in years and now I remember why I haven't.

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

About 24 hours in the past 3 days

End of Assignment