Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-452-M2024/it202-api-project-milestone-2-2024-m24/grade/vs53

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

Submissions:

Submission Selection

1 Submission [active] 7/22/2024 1:58:57 AM

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

Task #1 - Points: 1

Text: Screenshots of Table SQL

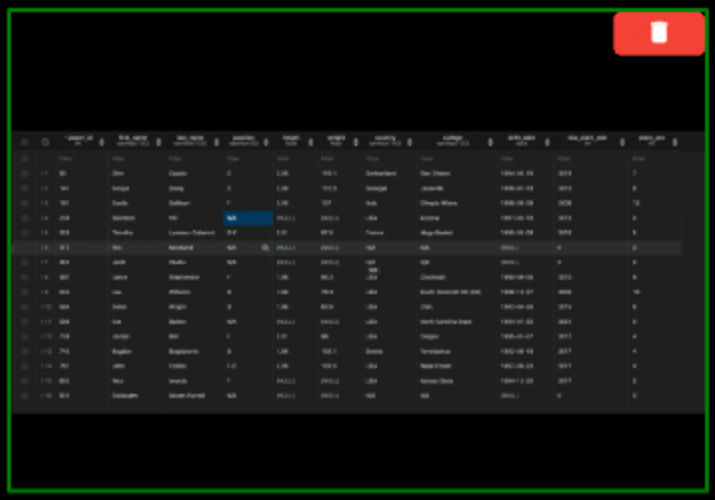
COLLAPSE

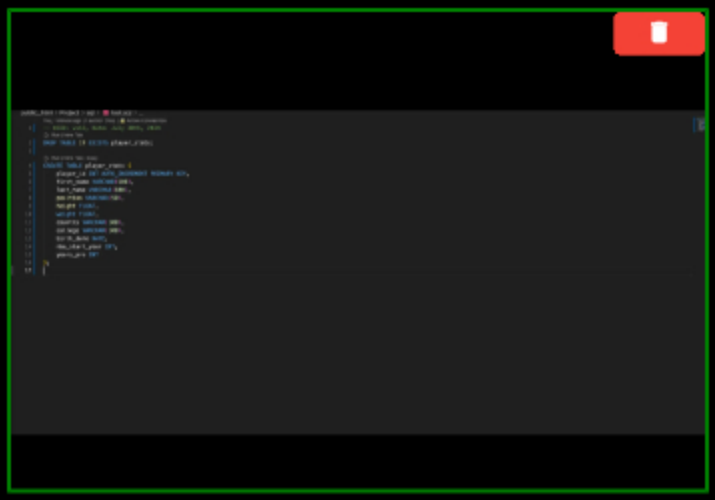
Checklist *The checkboxes are for your own tracking

#	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)





Caption (required) ✓

Describe/highlight what's being shown

Two screenshots showing the table and the sql query to make the table

Explanation (required) ✓

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

PREVIEW RESPONSE

In the player_stats table, each column represents a specific piece of information about basketball players. The player_id column is an auto-incremented primary key that uniquely identifies each player. The first_name and last_name columns store the player's first and last names. The position column indicates the player's position on the team, like guard or forward. The height and weight columns capture the player's physical stats, with height in meters and weight in kilograms. The country column shows the player's country of origin, while college notes where they played college basketball if applicable. The birth_date column records their date of birth, and the nba_start_year column notes the year they started in the NBA. Finally, the years_pro column indicates how many years they've been playing professionally. The table doesn't seem to need normalization right now, as all the data is directly related to each player and fits well in a single table.

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/VikShah/vs53-it202-452/pull/44

URL
https://github.com/VikShah/vs53-it202-452/pull/

+ ADD ANOTHER URL

Data Creation Page (2 pts.)

^COLLAPSE ^

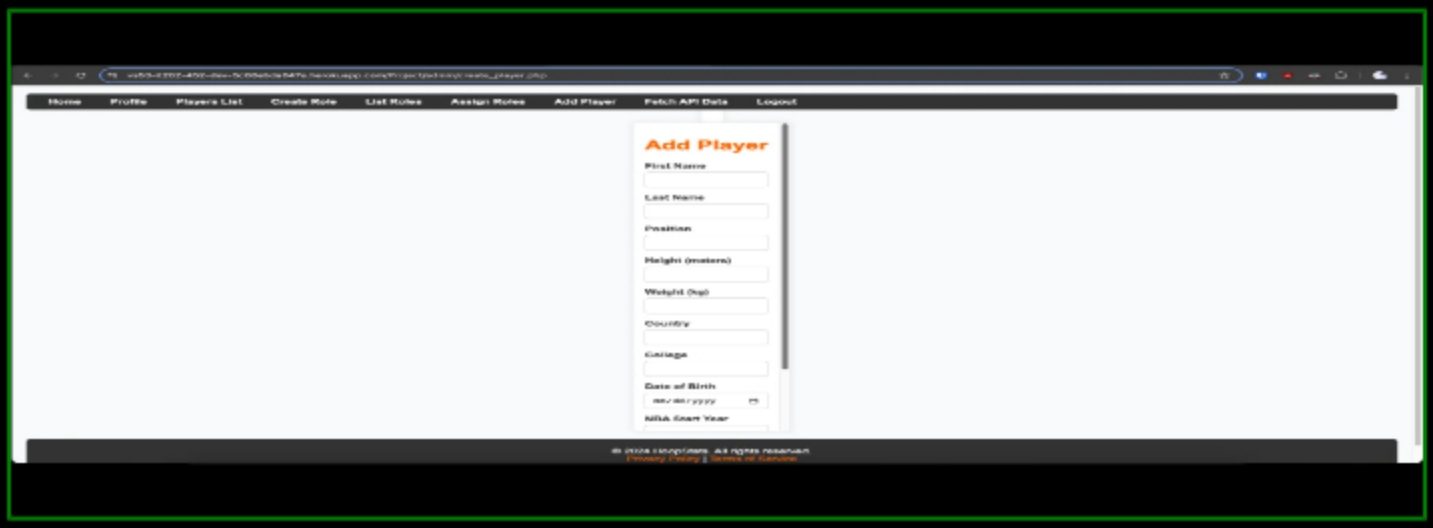
^COLLAPSE ^

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 filled in for the custom creation page



Caption (required) ✓
Describe/highlight what's being shown
PIcture of the data creating page, where you add a player is the data creation

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

Caption (required) ✓

Describe/highlight what's being shown

Picture of the code with my UCID and date showing how I fetcehd teh API and key

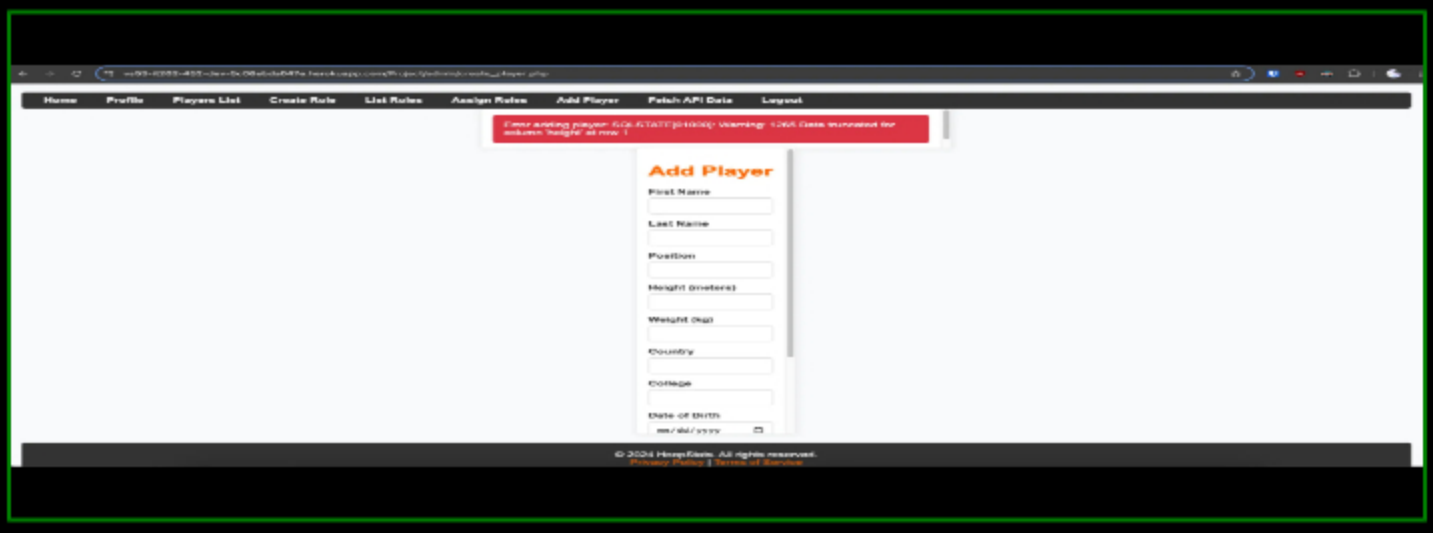
Explanation (required) ✓

Briefly explain the code

PREVIEW RESPONSE

In the fetch_api_data.php file, I wrote code to fetch player data from an NBA API and store it in a local database. I started by setting up error reporting to help catch any issues. Then, I included a functions file, which I use for database connections and other utility functions. In the main part of the script, I defined a function called fetchAndStoreApiData() that handles everything. I use my API key to access player data from the NBA API and make a request using cURL. After getting the data, I clear out the old data from the player_stats table in my database and insert the new data. The script ensures that if a player already exists in the database, their information gets updated rather than duplicated. Finally, I check the response to make sure the data is structured as expected.

#3) Show examples of validation messages

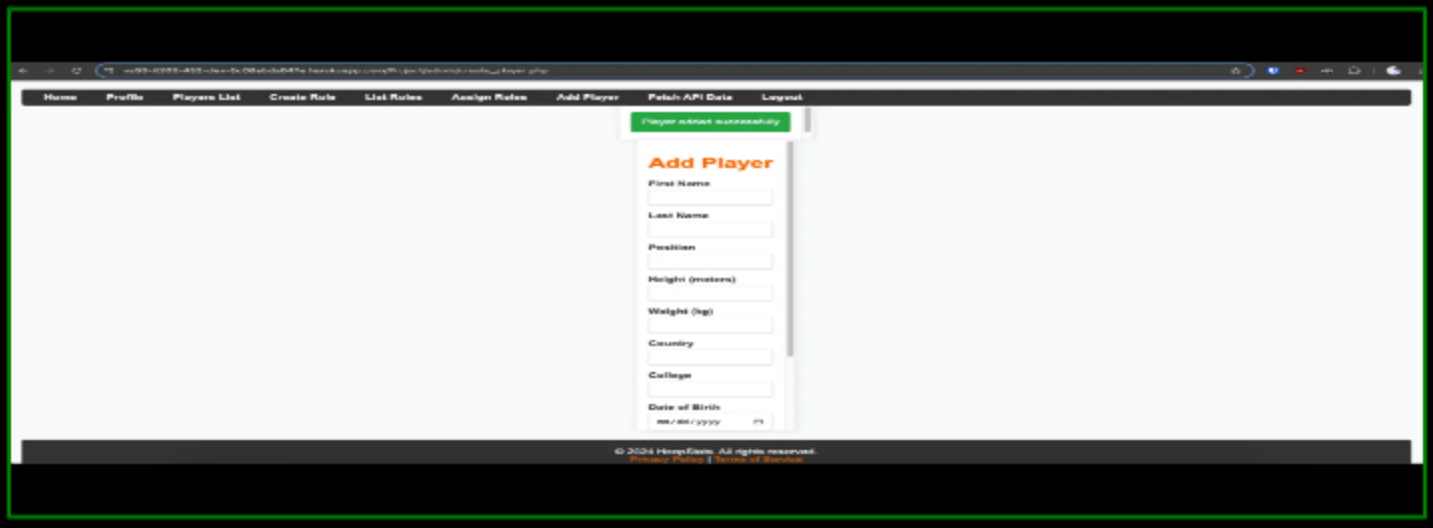


Caption (required) ✓

Describe/highlight what's being shown

An example if you don't put in the correct numbers for height and leave it with wrong parameters

#4) Show an example of successful creation message

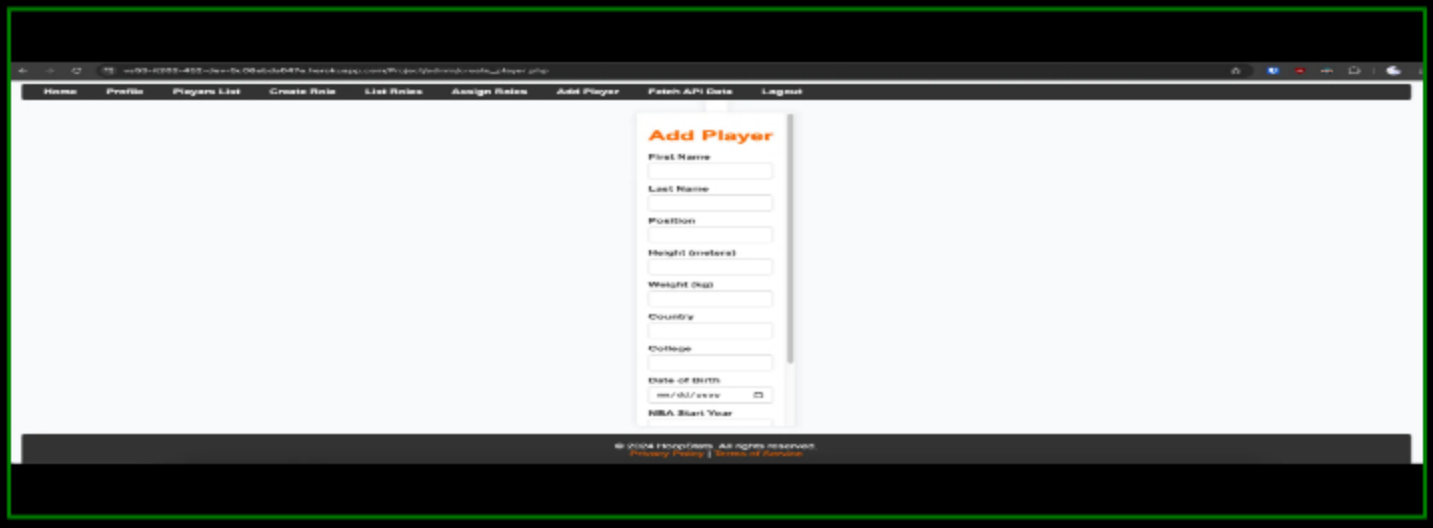


Caption (required) ✓

Describe/highlight what's being shown

An example above if you create the player and the message you get

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



Caption (required) ✓

Describe/highlight what's being shown

A picture of the design choice for the add player/data creation page

Explanation (required) ✓

Briefly explain your design choices

PREVIEW RESPONSE

For the design choices, I aimed for a clean and straightforward layout that complements the basketball theme of the website. I chose a black and orange color scheme, reflecting the colors often associated with basketball, like a basketball and the court. The design includes easy-to-navigate menus and clearly labeled sections, ensuring users can find information about players and their stats quickly. The layout uses responsive design principles, making it accessible on both desktops and mobile devices. The use of tables and structured forms helps in organizing the

data neatly, providing a user-friendly experience. I also added a footer section to provide a consistent end to the pages and offer additional navigation links or information. The overall goal was to create an interface that is both visually appealing and functional, enhancing the user experience while maintaining a sporty, energetic vibe.

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 types for each property being requested

Caption (required) ✓

Describe/highlight what's being shown

Above is the screenshot of the creation page

Explanation (required) ✓

Briefly talk about each field type and why it was chosen

PREVIEW RESPONSE

In the create_player.php form, I used different field types to ensure the data we collect is accurate. For the player's first name, last name, and position, I used type="text" because these are simple text fields and can include letters, spaces, and sometimes other characters. For height and weight, even though they're numbers, I went with text fields to handle cases like decimal points or different units of measurement. The country and college fields are also text since they are names and can contain a variety of characters. I chose type="date" for the birth date to ensure we get the correct format and valid dates. Lastly, for the NBA start year and years pro, I stuck with text fields for flexibility, even though these are typically numbers. This setup helps keep our data consistent and avoids input errors.

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

Caption (required) ✓

Describe/highlight what's being shown

3 screenshots showing all the valdations, PHP HTML And JS

Explanation (required) ✓

Briefly explain the validations

PREVIEW RESPONSE

In this code, I implemented validations to ensure data integrity when adding a player. PHP validation checks server-side that all required fields are filled and values are appropriate, like making sure height and weight are positive numbers. This helps prevent bad data from entering the database. On the client side, I used JavaScript to give users immediate feedback if they try to submit the form with missing or incorrect information, like empty fields or non-numeric input for numbers. HTML attributes like required and type enforce basic validation rules directly in the form, making sure users fill in all necessary fields and use the correct data types. These validations work together to make the data entry process smoother and more reliable.

#3) Successful creation should have a user-friendly message

#3) Successful creation should have a user-friendly message

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Explain how duplicate/existing data is handled

PREVIEW RESPONSE

Missing text

#4) Any errors should have user-friendly messages

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Briefly describe the scenarios

PREVIEW RESPONSE

Missing text

#5) Include the form/process for fetching API data

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Briefly explain the steps (include how duplicates are handled)

PREVIEW RESPONSE

Missing text

#6) Include some indicator between custom data and API data

Caption (required)

Describe/highlight what's being shown

Missing caption

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

Missing text

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

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Briefly explain the logic/reasoning

PREVIEW RESPONSE

Missing text

COLLAPSE

Task #3 - Points: 1

Text: Screenshot of records from DB

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

Caption (required)
Describe/highlight what's being shown (note what differs from a custom record)
Missing caption

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

Caption (required)
Describe/highlight what's being shown (note what differs from the API record)
Missing caption

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://github.com/VikShah/vs53-it202-452/pull/45

URL
https://github.com/VikShah/vs53-it202-452/pull/

+ ADD ANOTHER URL

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

^COLLAPSE^

^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 listed (have a reasonable number shown)

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

Explanation (required)
Briefly describe the page
PREVIEW RESPONSE
Missing text

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

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Briefly mention what's available to the user

PREVIEW RESPONSE

Missing text

#3) Demonstrate a few varied filters/sorts

Caption (required)

Describe/highlight what's being shown

Missing caption

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

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

#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, include examples from different user roles)

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

Explanation (required)
Mention which users can interact with the view, edit, and delete links
PREVIEW RESPONSE
Missing text

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

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

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

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

Explanation (required)
Briefly explain your design choices
PREVIEW RESPONSE
Missing text

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 form generation

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

Explanation (required)
Briefly explain how the code works (i.e, some stuff may be dynamic)
PREVIEW RESPONSE
Missing text

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

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

Explanation (required)
Briefly explain the related logic
PREVIEW RESPONSE
Missing text

#3) Show how the output is generated and displayed

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

Explanation (required)
Briefly explain the related logic
PREVIEW RESPONSE
Missing text

#4) Show any restrictions like role guard or login checks

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

Explanation (required)
Briefly explain the logic/reasoning
PREVIEW RESPONSE
Missing text

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
Missing URL
URL

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 (via the url)

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

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

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

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

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

Explanation (required)
Briefly explain your design choices
PREVIEW RESPONSE
Missing text

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

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

Explanation (required)
Briefly explain the details shown and how it differs from the list view
PREVIEW RESPONSE
Missing text

#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)

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

#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)

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

^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

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Briefly explain the logic and how it's handled when the property is missing or not "valid"

PREVIEW RESPONSE

Missing text

#2) Show the DB query to get the record

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)
Briefly explain the logic
PREVIEW RESPONSE
Missing text

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

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

Explanation (required)
Briefly explain the logic
PREVIEW RESPONSE
Missing text

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
Missing URL

URL

+ 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 screenshots required)

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

Explanation (required)
Briefly explain what you changed

PREVIEW RESPONSE

Missing text

#2) Show examples of validation messages

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

#3) Show an example of successful edit messages

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

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

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

Explanation (required)
Briefly explain your design choices
PREVIEW RESPONSE
Missing text

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 types for each property being requested

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

Explanation (required)
Briefly explain the data type choices

PREVIEW RESPONSE

Missing text

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

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Briefly explain the validations

PREVIEW RESPONSE

Missing text

#3) Successful edit should have a user-friendly message

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

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
Missing text

#4) Any errors should have user-friendly messages

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

Explanation (required)
Briefly explain the possible errors
PREVIEW RESPONSE
Missing text

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

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

Explanation (required)
Briefly explain the logic/reasoning
PREVIEW RESPONSE
Missing text

COLLAPSE

Task #3 - Points: 1
Text: Screenshot of records from DB

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

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

Explanation (required)
Explain what differs
PREVIEW RESPONSE
Missing text

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

Missing URL

URL

+ 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 a delete

Caption (required)

Describe/highlight what's being shown

Missing caption

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

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Explain in concise steps how this logically works

PREVIEW RESPONSE

Missing text

#3) Show the code related to the delete processing

Caption (required)

Describe/highlight what's being shown

Missing caption

Explanation (required)

Explain in concise steps how this logically works

PREVIEW RESPONSE

Missing text

#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

Missing text

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the data

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

Caption (required)

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

Missing caption

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

Missing URL

URL

+ 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

Missing Caption

COLLAPSE

Task #2 - Points: 1

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

URL #1

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

URL

https://github.com/users/VikShah/projects/1/vi

+ ADD ANOTHER URL

COLLAPSE

Task #3 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

Missing Response

COLLAPSE

Task #4 - Points: 1

Text: WakaTime Screenshot

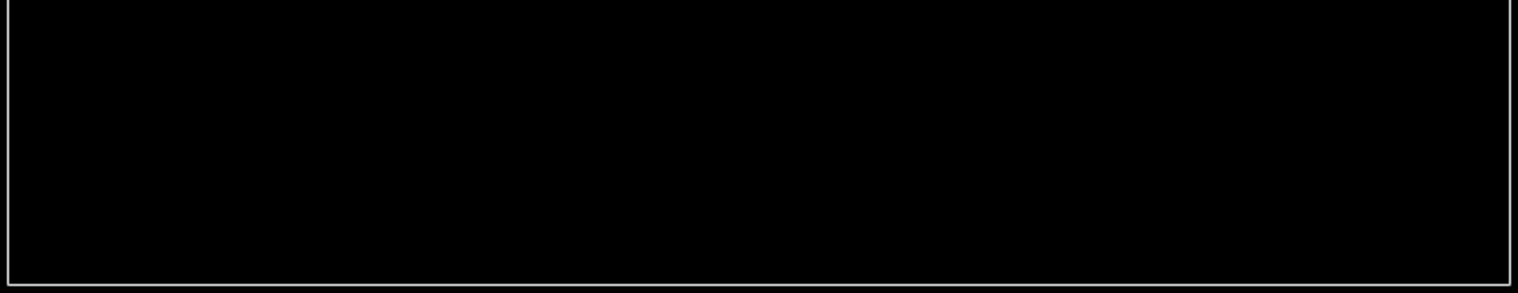
Details:

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



Missing Caption

End of Assignment