Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-api-project-milestone-2-2024/grade/bns24

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

Submissions:

Submission Selection

1 Submission [active] 4/14/2024 10:34:53 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

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of Table SQL

Checklist

*The checkboxes are for your own tracking

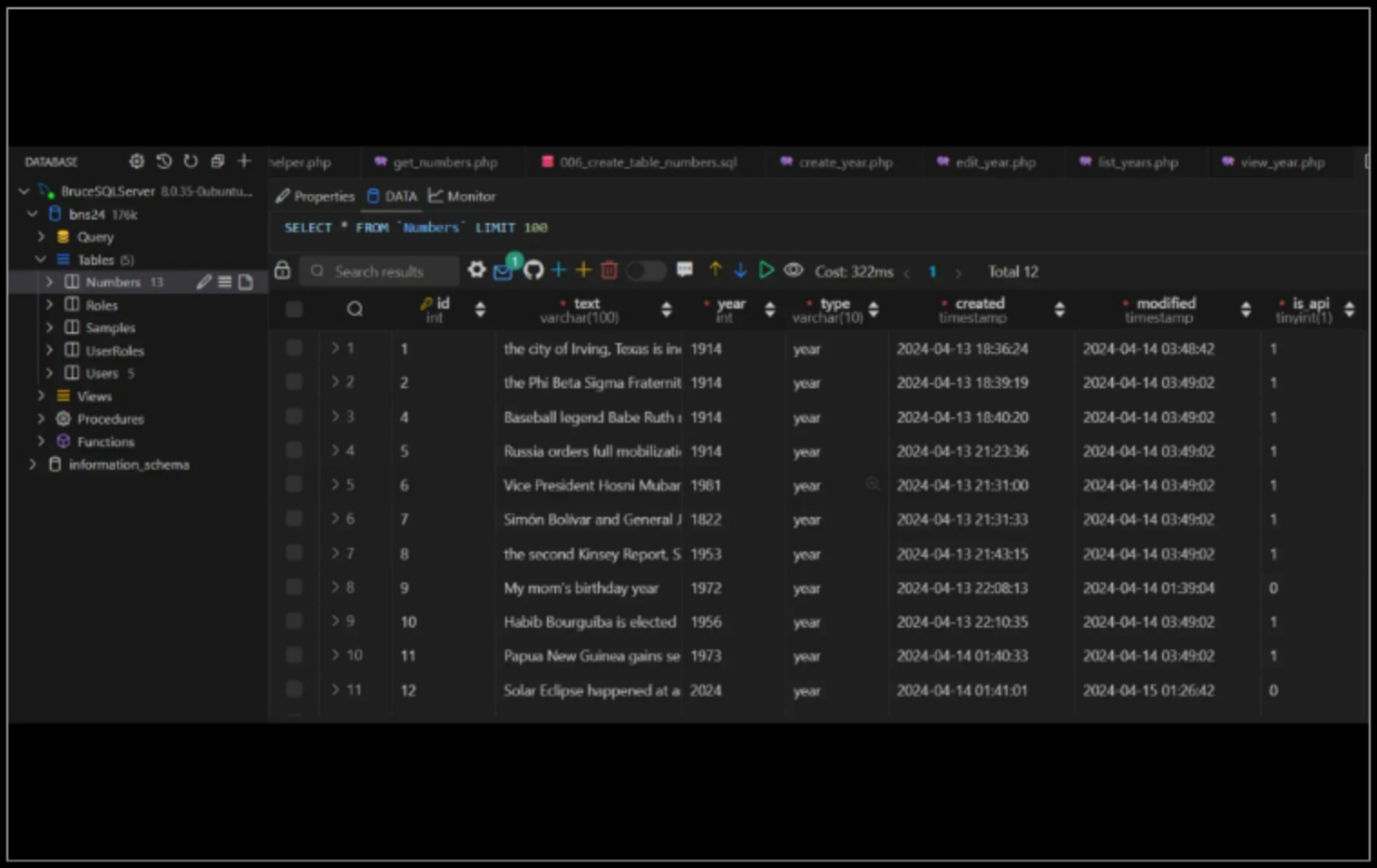
#	Points	Details

#	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



Screenshot of SQL Table `Numbers` with the 3 core columns and added columns. The datatypes for the added columns are logical.

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:

Table has the 3 core columns, as well as a "text" and "year" column, which can either be gathered from the API or from the user. Text takes in a sentence with at max 100 characters, and year is an integer. The "text" field is supposed to be used to display a fact about the "year" field. I didn't really need multiple tables to handle this because the info gathered is relatively simple.

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/BruceSiaNJIT/bns24-it202-008/pull/39

Data Creation Page (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the creation page

Checklist

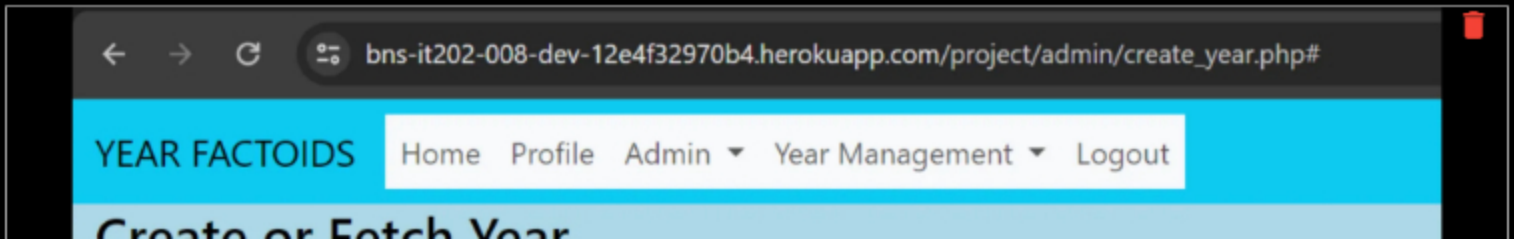
*The checkboxes are for your own tracking

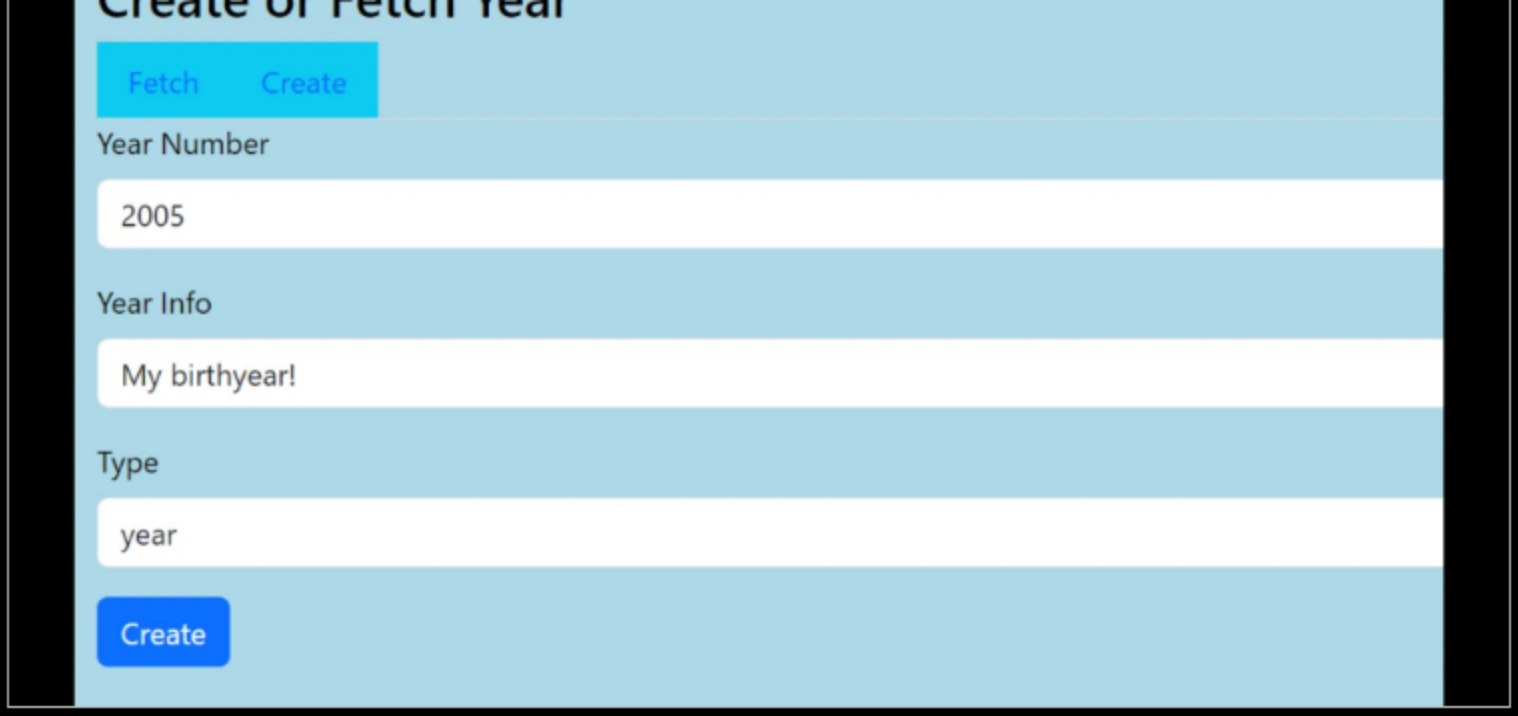
#	Points	Details
#1	1	Show potentially valid data filled in for the custom creation page
#2	1	Show how the API data is fetched for API data (must be server-side)
#3	1	Show examples of validation messages
#4	1	Show an example of successful creation message
#5	1	Design/Style should be considered (i.e., 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

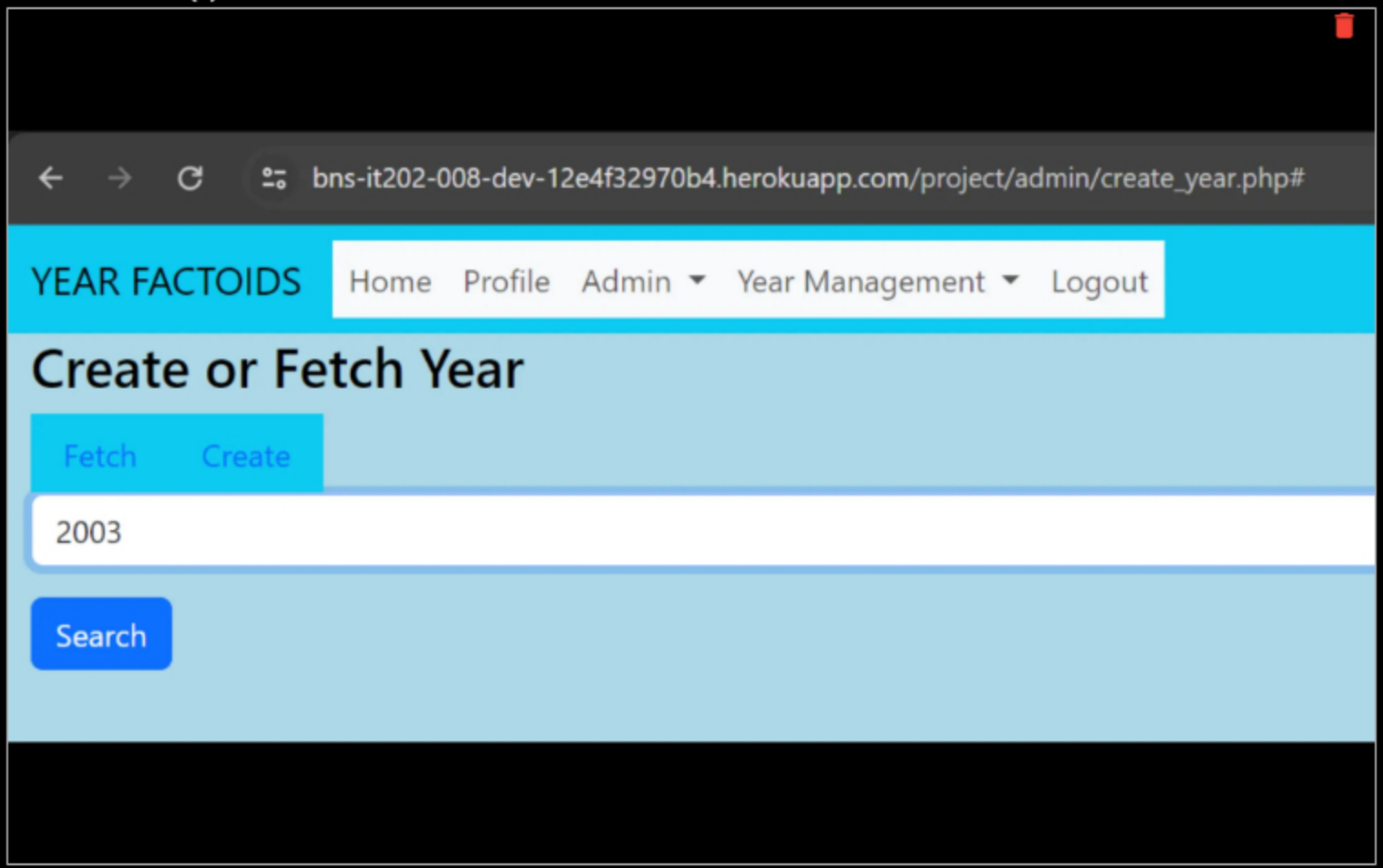
Small Medium Large





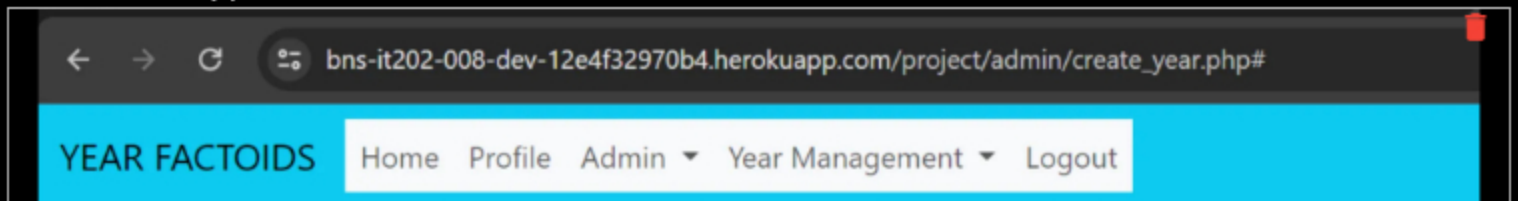
Potentially Valid data filled in for the custom creation page.

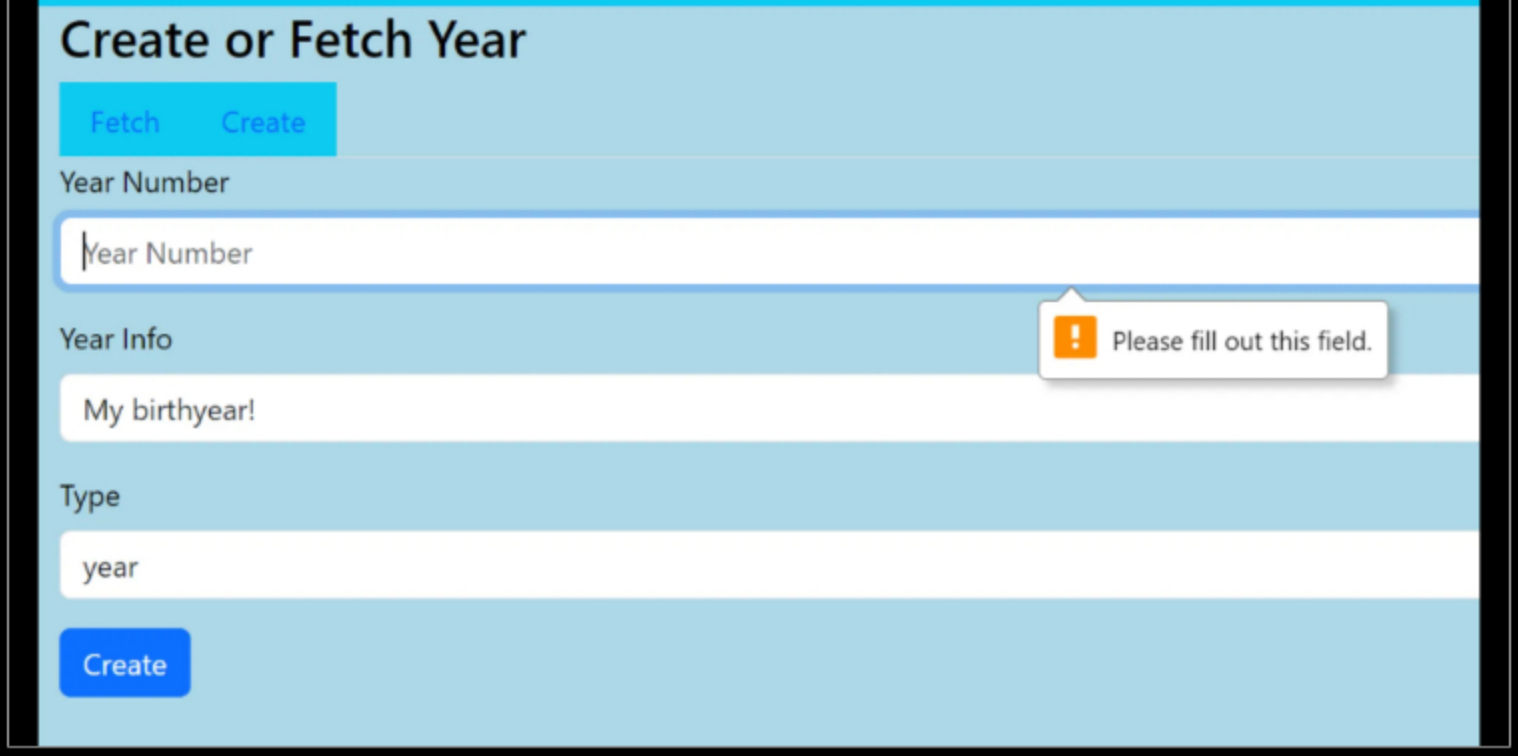
Checklist Items (0)



You put a year, then a random fact is taken from the API about the year, in this example, 2003.

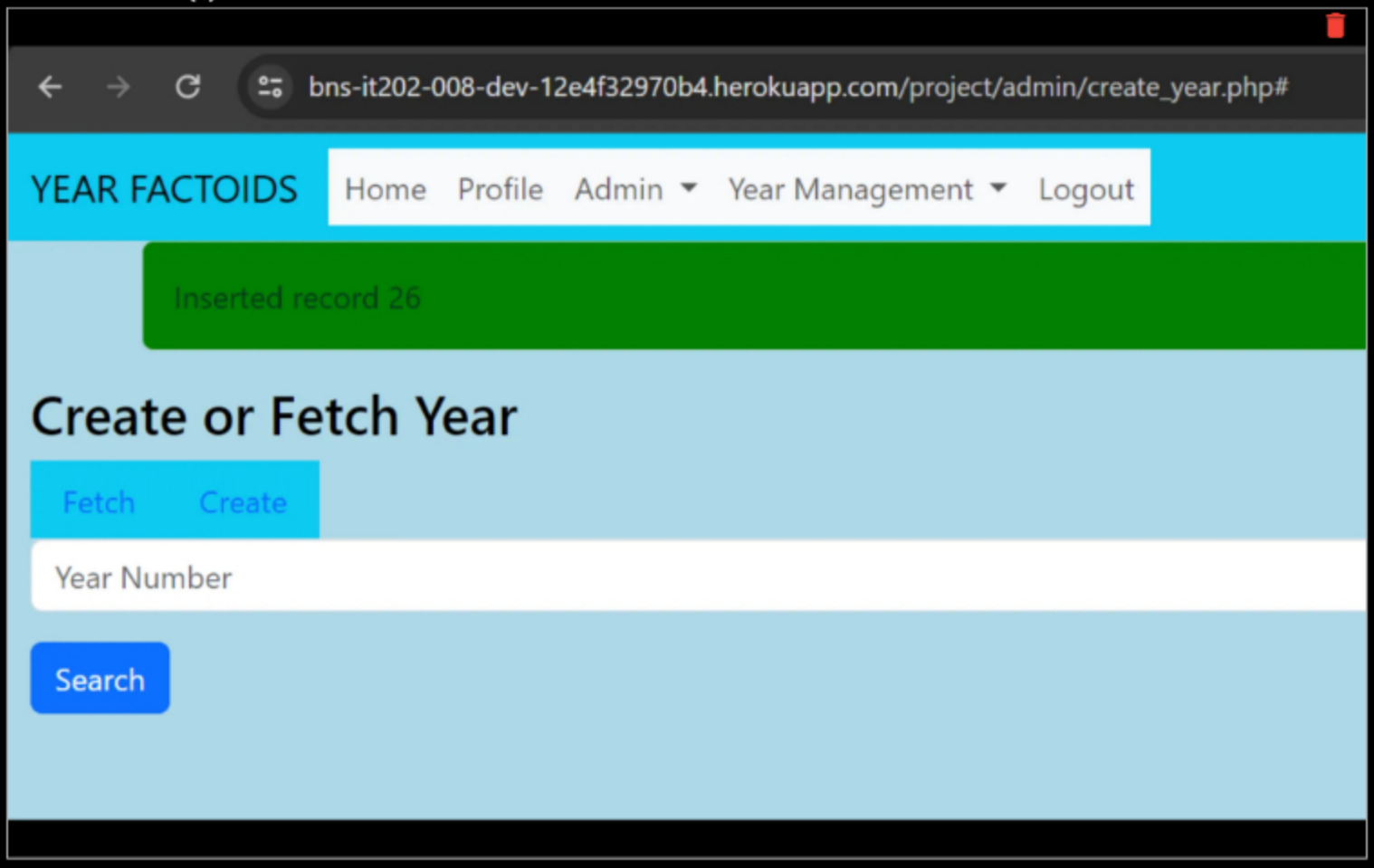
Checklist Items (0)





Validation in case the year number field is left empty.

Checklist Items (0)



Successful creation message.

Checklist Items (0)

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

public_html > project > admin > create_year.php > div.container-fluid > div#create.tab-target > form

```
<div class = "container-fluid">
    <ul class = "nav nav-tabs">
        <li class = "nav-item bg-info">
            <a class = "nav-link" href = "#" onclick = "switchTab('fetch')">Create</a>
        </li>
    </ul>

    <div id = "fetch" class = "tab-target">
        <form method="POST">
            <?php render_input(["type" => "search", "name" => "number", "placeholder" => "Year Number", "rules" => ["required" => "required"]]);/*lazy value to check if fo
            <?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 method="POST">
            <!-- bns24 04/14/24 -->
            <?php render_input(["type" => "number", "name" => "number", "placeholder" => "Year Number", "label" => "Year Number", "rules" => ["required" => "required"]]);/
            <?php render_input(["type" => "text", "name" => "text", "placeholder" => "Year Info", "label" => "Year Info", "rules" => ["required" => "required"]]);/*lazy va
            <?php render_input(["type" => "select", "name" => "type", "label" => "Type", "options" => ["year" => "year"],"rules" => ["required" => "required"]]);/*lazy val
            <?php render_input(["type" => "hidden", "name" => "action", "value" => "create"]);?>
            <?php render_button(["text" => "Search", "type" => "submit", "text" => "Create"]); ?>
        </form>
    </div>
</div>
```

Form with the correct types of datatypes that should be entered.

Checklist Items (0)

```php
<?php /* Include margin open tag */ ?>
<?php if ($_include_margin) : ?>
    <div class="mb-3">
    <?php endif; ?>
    <?php if ($_label) : ?>
        <?php /* label field  bns24 04/14/24*/ ?>
        <label class="form-label" for="<?php se($_id); ?>"><?php se($_label); ?></label>
    <?php endif; ?>

    <?php if (!in_array($_type, $_non_stanard_types)) : ?>
        <?php /* input field */ ?>
        <input type="<?php se($_type); ?>" name="<?php se($_name); ?>" class="form-control" id="<?php se($_id); ?>" value="<?php se($_value); ?>" placeholder="<?php se($_pla
        <?php echo $_rules;?> />
    <?php elseif($_type === "textarea"):?>
        <textarea class="form-control" name="<?php se($_name); ?>" id="<?php se($_id); ?>" placeholder="<?php se($_placeholder); ?>" <?php echo $_rules;?>><?php se($_value);
    <?php elseif ($_type == "select") : ?>
        <select class="form-select" name="<?php se($_name); ?>" value="<?php se($_value); ?>" id="<?php se($_id); ?>">
            <?php foreach ($_options as $k => $v) : ?>
                <option <?php echo (isset($_value) && $_value === $k ? "selected" : ""); ?> value="<?php se($k); ?>"><?php se($v); ?></option>
            <?php endforeach; ?>
        </select>
    <?php endif; ?>
```

I call render_input which leads to this code, which checks the type and makes sure the form has the right type of validation.

Checklist Items (0)

```php
    $query .= "(" . join(",", $columns) . ")";
    $query .= "VALUES (" . join(",", array_keys($params)) . ")";
    error_log("Query: " . $query);
    error_log("Params: ". var_export($params, true));
    //bns24 04/14/24
    try{
        $stmt = $db->prepare($query);
        $stmt->execute($params);
        flash("Inserted record " . $db->lastInsertId(), "success");
    }catch(PDOException $e){
        if($e->errorInfo[1] == 1062){
            flash("Another year has the same description, please use another description.", "warning");
        }else{
            error_log("Something went wrong.");
            flash("An error occurred.", "danger");
        }
    } <- #59-66 catch(PDOException $e)
```

If the query is correct and valid, then it flashes a user friendly message saying that it was inserted. Otherwise, if there is another record with the same text description, then it flashes a user friendly message saying that another year has the same description, otherwise it just says an error occurred.

Checklist Items (0)

```php
<?php
//handle year fetch bns24 04/14/24
if(isset($_POST["action"])){
    $action = $_POST["action"];
    $year = se($_POST, "number", "", false);
    if($year){
        if($action === "fetch"){
            $result = fetch_quote($year);
            if($result){
                $result["is_api"] = 1;
            }
```

The form for the fetch has an action "fetch" associated with it. If the action is "fetch" then it calls fetch_quote.

Checklist Items (0)

lib > year_api.php > fetch_quote

```php
<?php
//bns24 04/14/24

function fetch_quote($yearnum){
    $result = get("https://numbersapi.p.rapidapi.com/$yearnum/year", "NUMBER_API_KEY", ["fragment" => true, "json" => true]);

    if (se($result, "status", 400, false) == 200 && isset($result["response"])) {
        $result = json_decode($result["response"], true);
    } else {
        $result = [];
    }

    return $result;
} <- #4-14 function fetch_quote($yearnum)

?>
```

This is the fetch_quote function that was called earlier, showing the process for API data

This is the fetch_quote function that was called earlier, showing the process for API data.

Checklist Items (0)

```php
<?php
//handle year fetch bns24 04/14/24
if(isset($_POST["action"])){
    $action = $_POST["action"];
    $year = se($_POST, "number", "", false);
    if($year){
        if($action === "fetch"){
            $result = fetch_quote($year);
            if($result){
                $result["is_api"] = 1;
            }
        }else if($action === "create"){
            foreach($_POST as $k => $v){
                if(!in_array($k, ["number", "text", "type"])){
                    unset($_POST[$k]);
                }
                $result = $_POST;
            } <- #23-28 foreach($_POST as $k => $v)
        } <- #22-29 else if($action === "create")
```

Shows the difference between custom data and API data with the different action names "fetch" and "create"

Checklist Items (0)

public_html > project > admin > create_year.php > ...

```php
<?php
//note we need to go up 1 more directory bns24 04/14/24
require(__DIR__ . "/../../../partials/nav.php");

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

Checks if user has the role admin before they are able to access the page.

Checklist Items (0)

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

id int	* text varchar(100)	* year int	* type varchar(10)	* created timestamp	* modified timestamp	* is_api tinyint(1)
8	the second Kinsey Report, S	1953	year	2024-04-13 21:43:15	2024-04-14 03:49:02	1

Record fetched from the api. You can see that the is_api column is noted as "1" showing that is is from the API.

Checklist Items (0)

id int	text varchar(100)	year int	type varchar(10)	created timestamp	modified timestamp	is_api tinyint(1)
9	My mom's birthday year	1972	year	2024-04-13 22:08:13	2024-04-14 01:39:04	0

Record created via the creation form. You can see the is_api column is 0, meaning that this was not fetched from the API and was created through creation form.

Checklist Items (0)

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:

For fetching API data, the form simply has one parameter which is the year variable. The form has a POST method and the action "fetch" associated with it. A PHP script checks if the form with the action was submitted, then sets the parameter to a variable called year. If the year does exist, and if it was a "fetch" action, it called a function called fetch_quote, with the year passed in as the parameter. It also sets the is_api column to 1. The fetch quote function comes from another PHP file, which takes the year as the variable. It gets the text associated with that year number from the API using the get method and returns the result. Now personally, the API that I have was a little weird in the sense that it had the "year" column as "number" which confused things. But regardless, I create a query, and go through the key-value pairs of the result, accounting for any weird things conditions in the API. Once that is done and the query is finalized, we use a try catch and try to insert the record if the query is valid. If it is, we flash a success

message, and if it is not, then we check if the text is a duplicate. If it is, we flash a message saying to try again, and if it isn't then we flash a message saying an error occurred.

For creating custom data, we have another form with more parameters for the year, the text, and the type. The year field checks to see if it is a number, the text field checks to see if it is a text, and the type field is a dropdown. I had earlier plans to have more than one option, but I didn't have time to implement but I decided to keep it in case I want to work on it further. The same PHP script checks if the form with the action was submitted, then sets the parameter to the variable called year. If it does exist and if the action was "create", we make sure that the keys are only the ones that we need and return the result. The default value for is_api is 0, so we don't change or do anything to it. We prepare the same query statement, this time using the result variable from the custom form. We use the same try catch to see if the query is valid, if it is then we flash a success message, otherwise we check if the text field is a duplicate and flash a warning message, otherwise we let the user know that a general error occurred.

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

URL #1

https://bns24-it202-008-prod-4d42e45e31d0.herokuapp.com/project/admin/create_year.php#

URL #2

https://github.com/BruceSiaNJIT/bns24-it202-008/pull/39

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

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the list page

Checklist

*The checkboxes are for your own tracking

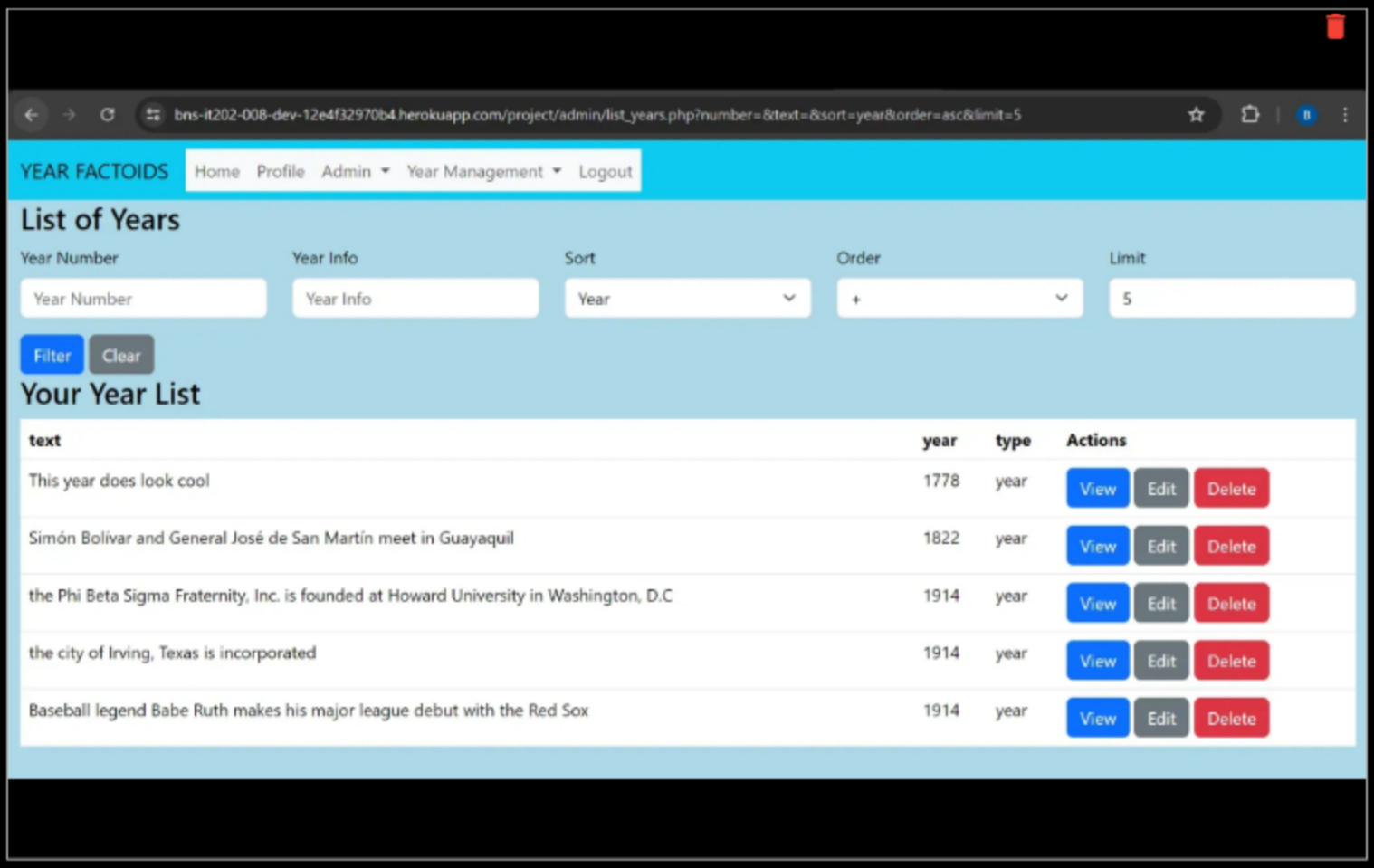
#	Points	Details
#1	1	Show the page of your entities listed (have a reasonable number shown)
#2	1	Show the filter/sort form based on your data and the required limit field
#3	1	Demonstrate a few varied filters/sorts
#4	1	Demonstrate a filter that doesn't have any records (should show an appropriate message)
#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:

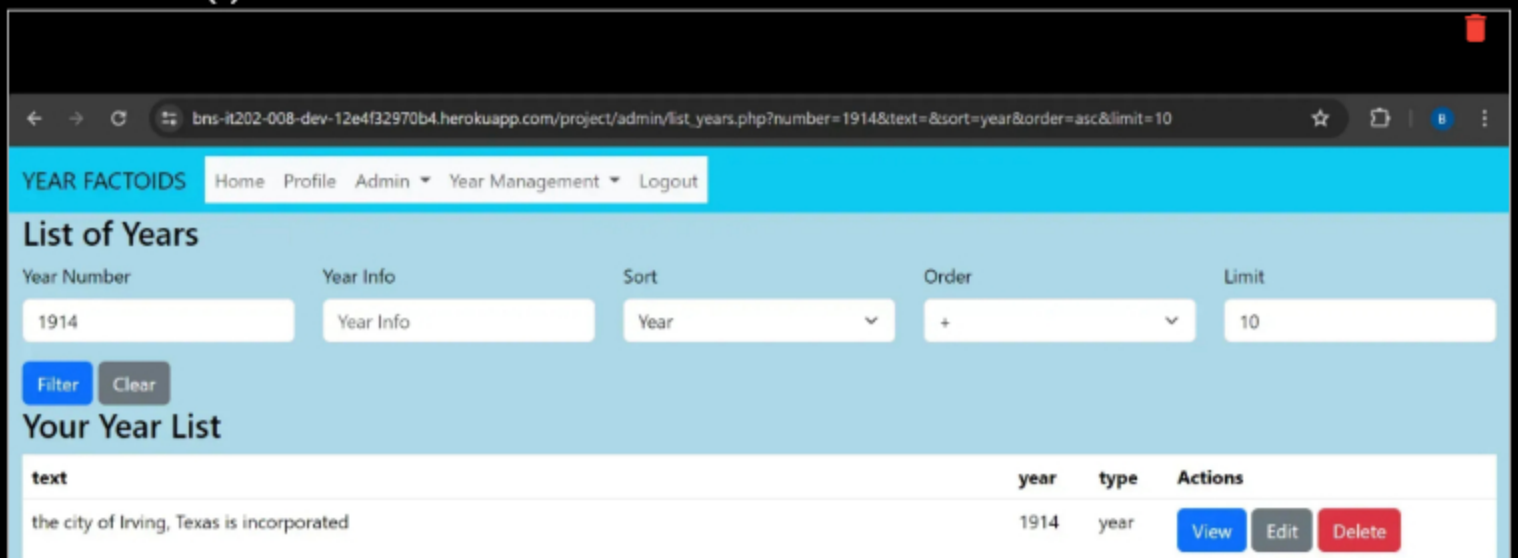
Gallery Style: Large View

Small Medium Large



The screenshot shows the page with a reasonable number of entities shown. It also shows the filter form and the required LIMIT field. Also visible view, edit, and delete link. Each list item has a summary of the data including year, type, and text.

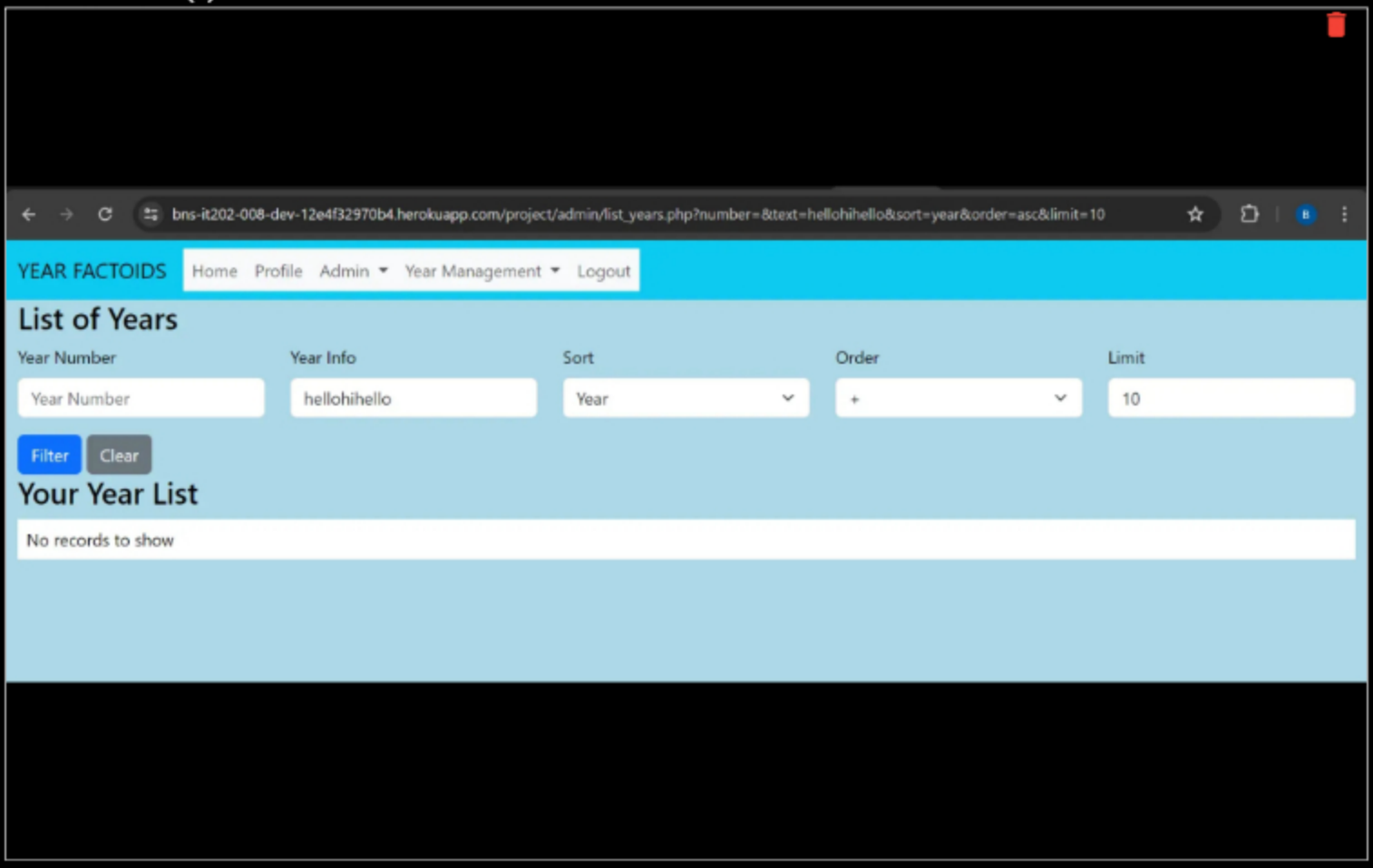
Checklist Items (0)



the Phi Beta Sigma Fraternity, Inc. is founded at Howard University in Washington, D.C	1914	year	View Edit Delete
Baseball legend Babe Ruth makes his major league debut with the Red Sox	1914	year	View Edit Delete
Russia orders full mobilization	1914	year	View Edit Delete

Filter just for the year 1914.

Checklist Items (0)



Filter for text info "hellohihello" leads to no records, and it displays that message.

Checklist Items (0)

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

```php
//build search form bns24 04/14/24
$form = [
    ["type" => "number", "name" => "number", "placeholder" => "Year Number", "label" => "Year Number", "include_margin" => false],
    ["type" => "text", "name" => "text", "placeholder" => "Year Info", "label" => "Year Info", "include_margin" => false],

    ["type" => "select", "name" => "sort", "label" => "Sort", "options" => ["year" => "Year", "text" => "Text"], "include_margin" => fals
    ["type" => "select", "name" => "order", "label" => "Order", "options" => ["asc" => "+", "desc" => "-"], "include_margin" => false],

    ["type" => "number", "name" => "limit", "label" => "Limit", "value" => "10", "include_margin" => false],
]; <- #11-19 $form =
```

A form array that will be used in the next screenshot.

Checklist Items (0)

```php
$table = ["data"=>$results, "title" => "Your Year List", "ignored_columns" => ["id"], "edit_url"=>get_url("admin/edit_year.php"), "delete
?>

<div class = "container-fluid">
    <h3>List of Years</h3>
    <form method = "GET">
        <div class = "row mb-3" style = "align-items: flex-end;">
            <?php foreach($form as $k=>$v) : ?>
                <div class = "col">
                    <?php render_input($v);?>
                </div>
            <?php endforeach;?>
        </div>
```

```
        <?php render_button([ "text" => "Search", "type" => "submit", "text" => "Filter" ]); ?>
        <a href = "?clear" class = "btn btn-secondary">Clear</a>
    <?php render_table($table) ?>
</div>
```

The form being generated, going through the form array earlier as key value pairs, and rendering them each as inputs with the values being the type.

Checklist Items (0)

```
//bns24 04/14/24
$query = "SELECT id, text, year, type FROM `Numbers` WHERE 1=1";
```

The initial query.

Checklist Items (0)

```
$yr = se($_GET, "number", "-1", false);
if(!empty($yr) && $yr > -1){
    $query .= " AND year like :year";
    $params[':year'] = "%$yr%";
}
$text = se($_GET, "text", "", false);
if(!empty($text)){
    $query .= " AND text like :text";
    $params[':text'] = "%$text%";
}

$sort = se($_GET, "sort", "year", false);
if (!in_array($sort, ["year", "text"])) {
    $sort = "year";
}
$order = se($_GET, "order", "desc", false);
if (!in_array($order, ["asc", "desc"])) {
    $order = "desc";
```

```
}

$query .= " ORDER BY $sort $order";

try{
    $limit = (int)se($_GET, "limit", "10", false);
}catch(Exception $e){
    $limit = 10;
}

if($limit < 1 || $limit > 100){
    $limit = 10;
}

$query .= " LIMIT $limit";
```

How the filter/sort is handled by adding on to the query. There is also a restriction on the limit field limiting it from between 1 and 100.

Checklist Items (0)

```
$db = getDB(); //bns24 04/14/24
$stmt = $db->prepare($query);
$results = [];
try{
    $stmt->execute($params);
    $r = $stmt->fetchAll();
    if($r){
        $results = $r;
    }
} <- #97-105 try

catch(PDOException $e){
    error_log("Error fetching year " . var_export($e, true));
    flash("Unhandled error occurred", "danger");
}

$table = ["data"=>$results, "title" => "Your Year List", "ignored_columns" => ["id"], "edit_url"=>get_url("admin/edit_year.php"), "delete_url"=>get_url("admin/delete_year.php"), "view_url"=>get_url("admin/view_year.php
?>

<div class = "container-fluid">
    <h3>List of Years</h3>
    <form method = "GET">
        <div class = "row mb-3" style = "align-items: flex-end;">
            <?php foreach($form as $k=>$v) : ?>
                <div class = "col">
                    <?php render_input($v);?>
                </div>
            <?php endforeach;?>
        </div>
        <?php render_button(["text" => "Search", "type" => "submit", "text" => "filter"]); ?>
        <a href = "?clear" class = "btn btn-secondary">Clear</a>
    <?php render_table($table) ?>
</div>
```

Once the query is finished, it executes it. We also make a new variable $results as an array and set it to everything from the statement. We then create a table using the table from results, and it is displayed in the render_table function.

Checklist Items (0)

public_html > project > admin > list_years.php > ...

```
1   <?php
2   //note we need to go up 1 more directory bns24 04/14/24
```

```php
require(__DIR__ . "/../../../partials/nav.php");

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

Checks to see if user is admin.

Checklist Items (0)

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:

We first check to see if user has admin before they are able to see the page. This is because for right now I want it so that only admin can see the database list, as well them being the only ones to view, edit, and delete the records. We build a search form using an array, which has the parameters, year number, text, sort by (year or number), and order (asc or desc), and a limit on how many records are visible. We make a form and go through the aforementioned array as key value pairs, rendering each thing using the render_input function. It's also important to mention that we use a GET method for this form so that if we were to share this link with someone, the same filters would still apply. Once the search button is clicked the form goes through. It initializes a query which we build upon. We want to save the session in case we want to edit a specific record and then go back, so that all the filters are still in place. We then go through each of the parameters. If there was a year input, we add to the query to see if there is any data where the year is the same as the inputted year. The same goes for text. For sort, we simply get whatever data we got from the form, but we make a default value just in case. We then add the limit to it, and we have conditions for the limit, where it must be greater than 1 and less than 100, and if it is not, then we default it to 10. Finally once the query is done, we filter through the database and if everything was successful then we return the results. If not, then we flash a message saying an unhandled error occurred. Using the return result, we make a table using the render_table with all the data gathered. We also add the edit, view, and delete links to the table, and add a title to it. I also chose to ignore the column id because I feel it wasn't super necessary for the user to see. Finally, it renders the table.

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

URL #1

https://bns24-it202-008-prod-4d42e45e31d0.herokuapp.com/project/admin/list_years.php

URL #2

https://github.com/BruceSiaNJIT/bns24-it202-008/pull/40

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

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the details page

Checklist

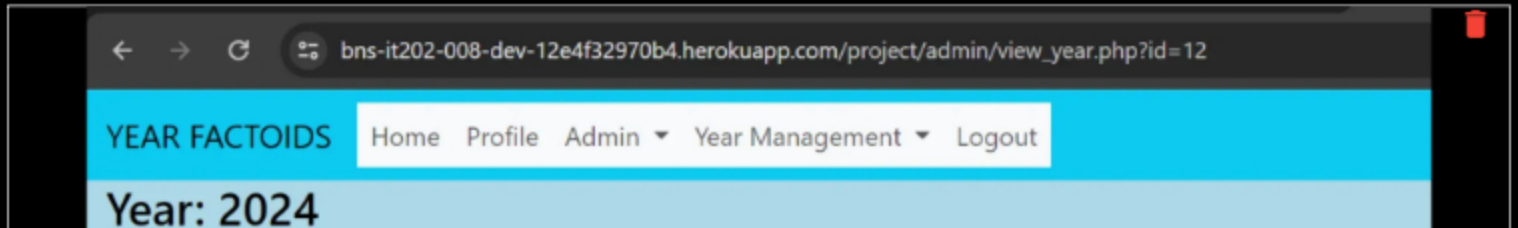
*The checkboxes are for your own tracking

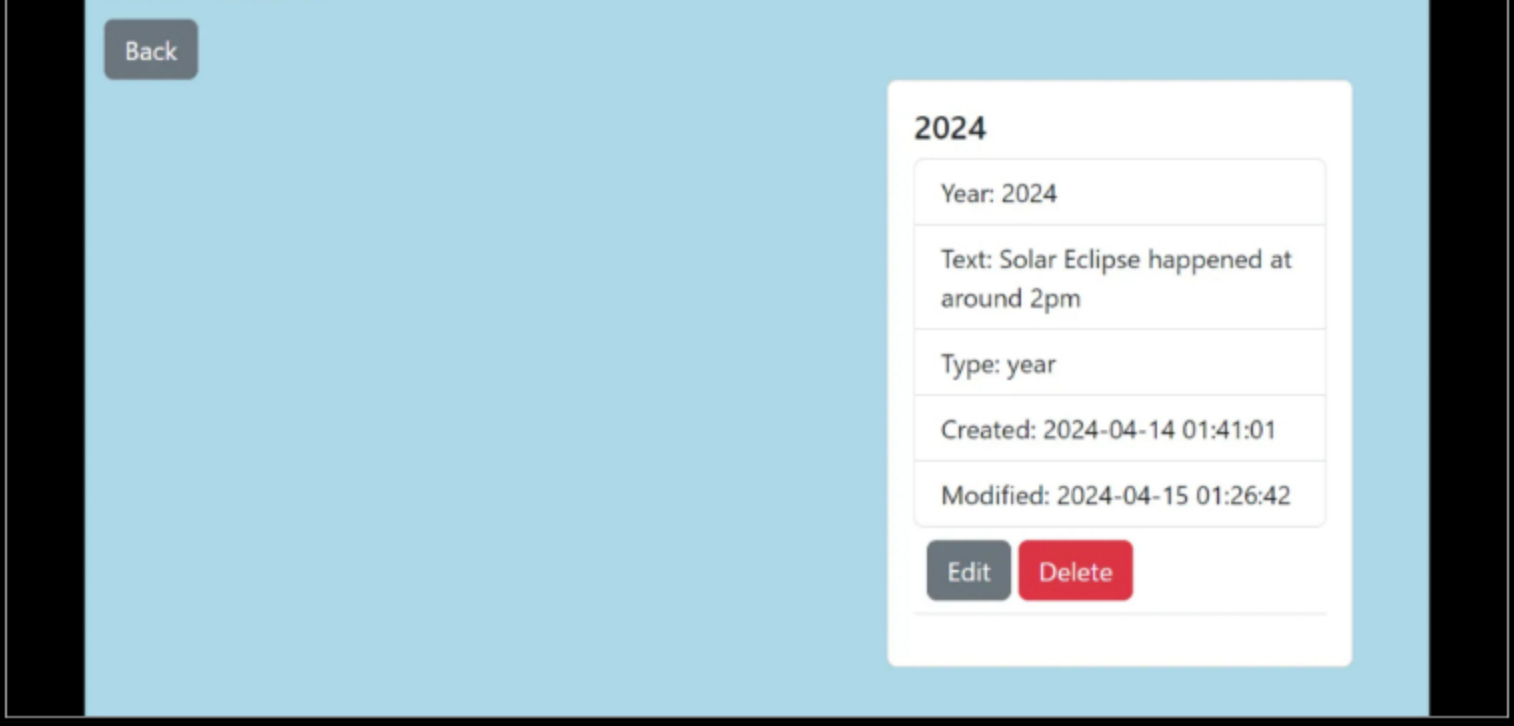
#	Points	Details
#1	1	Entity should be fetch by id (via the url)
#2	1	A missing id should redirect back to the list page with an applicable message
#3	1	Design/Style should be considered (i.e., bootstrap, custom css, etc)
#4	1	Data shown should be more detailed/inclusive than the summary view
#5	1	There 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

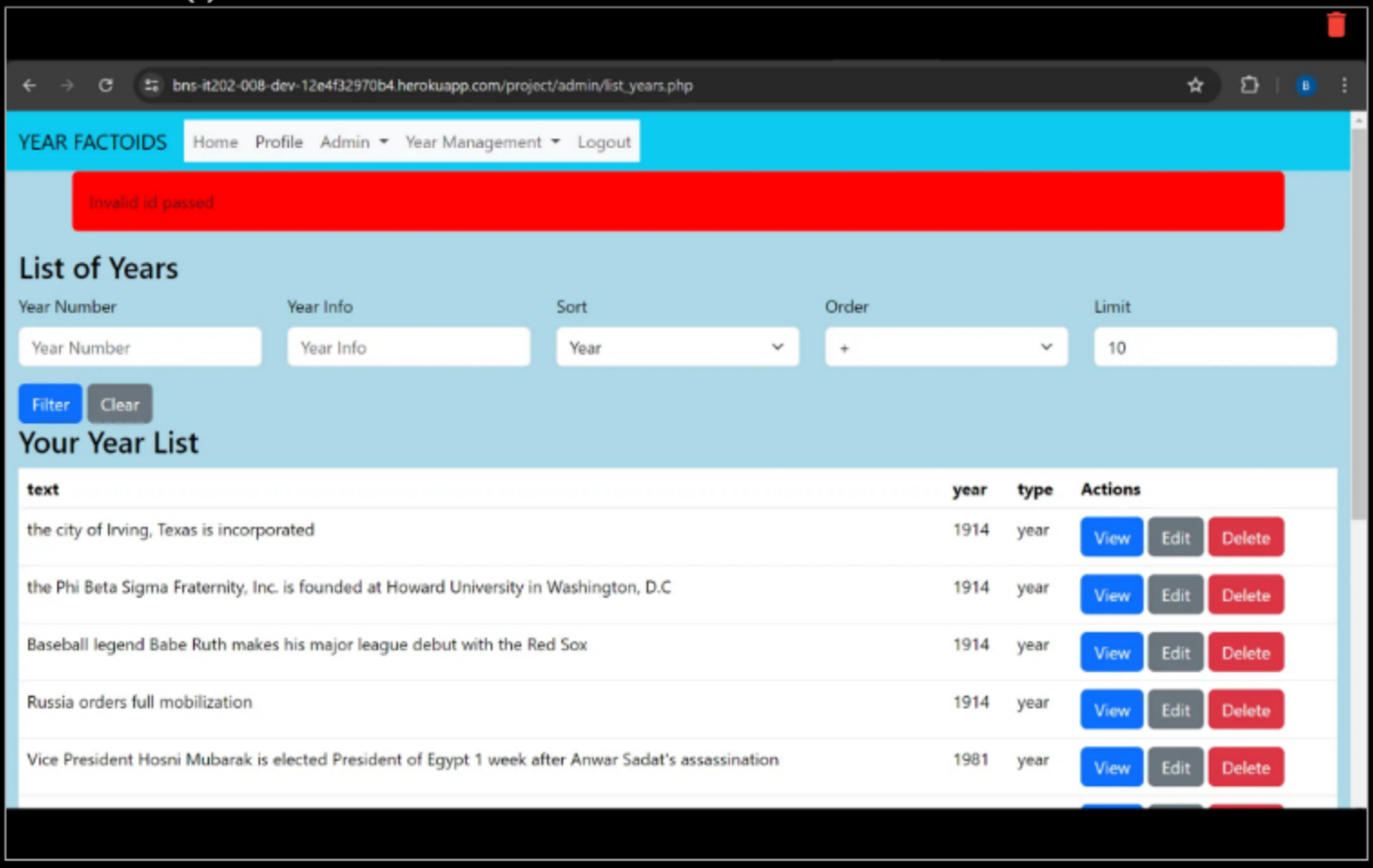
Small Medium Large





Single entity view, with the id in the url. Data is more detailed, showing the time it was created and the time is was last modified. There is a link to edit the entity and a link to delete the entity.

Checklist Items (0)



Missing id redirects to the list page with an applicable message.

Checklist Items (0)

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
$id = se($_GET, "id", -1, false);

$yr = [];
if ($id > -1) {
    //fetch bns24 04/14/24
    $db = getDB();
    $query = "SELECT id, year, text, type, created, modified FROM `Numbers` WHERE id = :id";
    try {
        $stmt = $db->prepare($query);
        $stmt->execute([":id" => $id]);
        $r = $stmt->fetch();
        if ($r) {
            $yr = $r;
        }
    } catch (PDOException $e) {
        error_log("Error fetching record: " . var_export($e, true));
        flash("Error fetching record", "danger");
    }
} else {
    flash("Invalid id passed", "danger");
    die(header("Location:" . get_url("admin/list_years.php")));
}
```

ID is fetched through GET method. DB query is also shown.

Checklist Items (0)

```php
<!-- bns24 04/24/24 -->
<div class="card mx-auto" style="width: 18rem;">
    <div class="card-body">
        <h5 class="card-title"><?php se($yr, "year", "Unknown"); ?></h5>
```

```
        <div class="card-text">
            <ul class="list-group">
                <li class="list-group-item">Year: <?php se($yr, "year", "Unknown"); ?></li>
                <li class="list-group-item">Text: <?php se($yr, "text", "Unknown"); ?></li>
                <li class="list-group-item">Type: <?php se($yr, "type", "Unknown"); ?></li>
                <li class="list-group-item">Created: <?php se($yr, "created", "Unknown"); ?></li>
                <li class="list-group-item">Modified: <?php se($yr, "modified", "Unknown"); ?></li>
            </ul>
        </div>
        <?php render_table($table) ?>
    </div>
</div>
```

Data is displayed through the card class bootstrap. Using the db query which selects the data where the id matches the passed in id, it passes in the values from that record. A render table is used to present the edit and delete.

Checklist Items (0)

^COLLAPSE ^

Task #3 - Points: 1

Text: Explain how the page works

Checklist *The checkbox are for your own tracking

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

Response:

Page first checks to see if user has role admin. It uses the id value that was passed from clicking on the view button on the list page. So long as the id is above -1, we prepare the query, selecting the data that we need such as id, year, text, type, created, and modified where the id matches the id variable from before. If any of the values are null, then we give them the string value "N/A". If the fetch doesn't work or if an invalid ID is passed, we flash the respective user-friendly messages letting the user know. In the case of invalid id, we also redirect them back to the list page. I then use a similar method to the list page for displaying the edit and delete buttons on the view page. I make a table, pass in the data, and this time ignore basically all the columns except for the action column.

For loading the page itself, we use the data to print out each value, such as the year, the type, the text, modified, and created. We also render the previous table, which brings in the earlier buttons I mentioned.

^COLLAPSE ^

Task #4 - Points: 1

Text: Add related links

Checklist *The checkbox 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/#. Same pull request shouldn't be used for each feature

URL #1

https://bns24-it202-008-prod-4d42e45e31d0.herokuapp.com/project/admin/view_year.php?id=1

URL #2

https://github.com/BruceSiaNJIT/bns24-it202-008/pull/42

Edit Data Page (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the edit page

Checklist

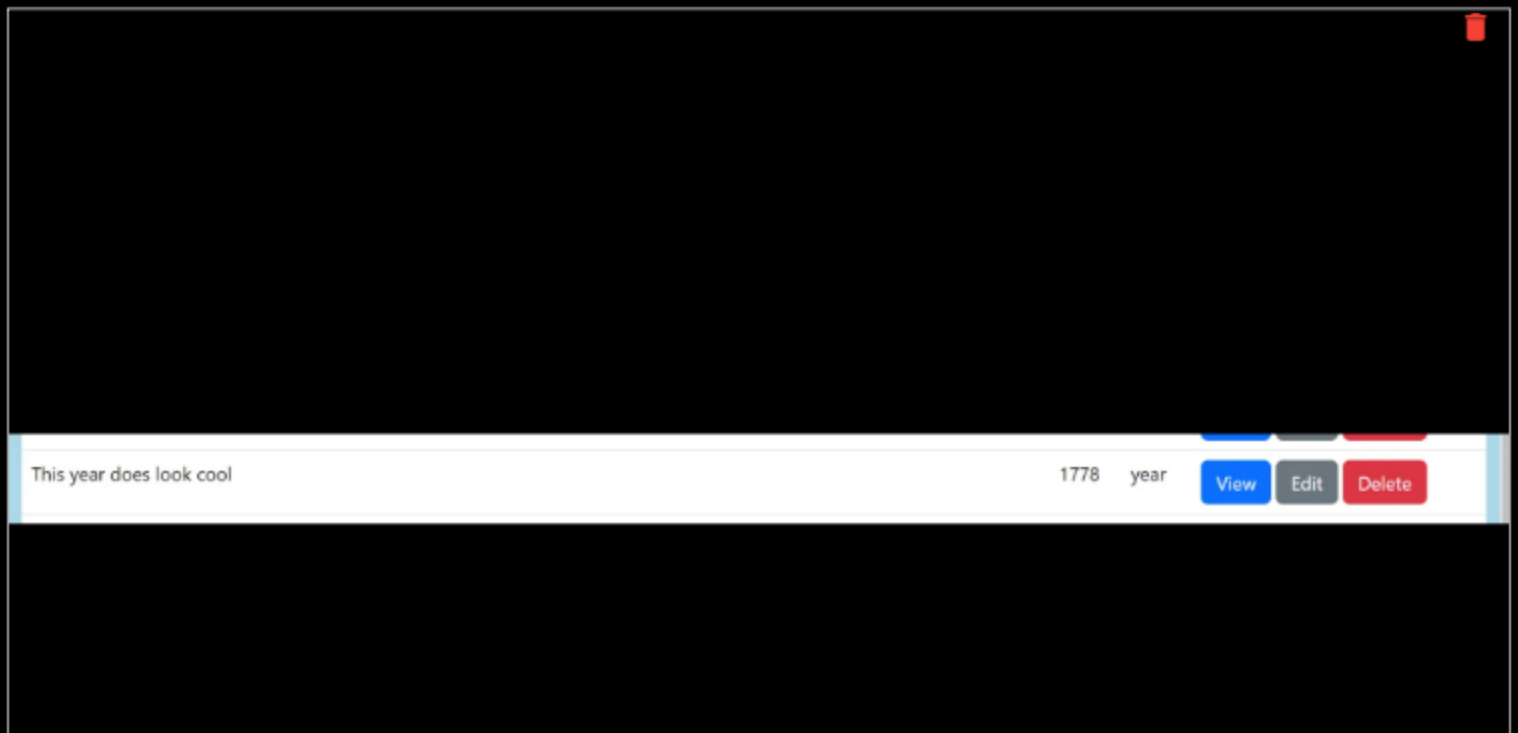
*The checkboxes are for your own tracking

#	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:

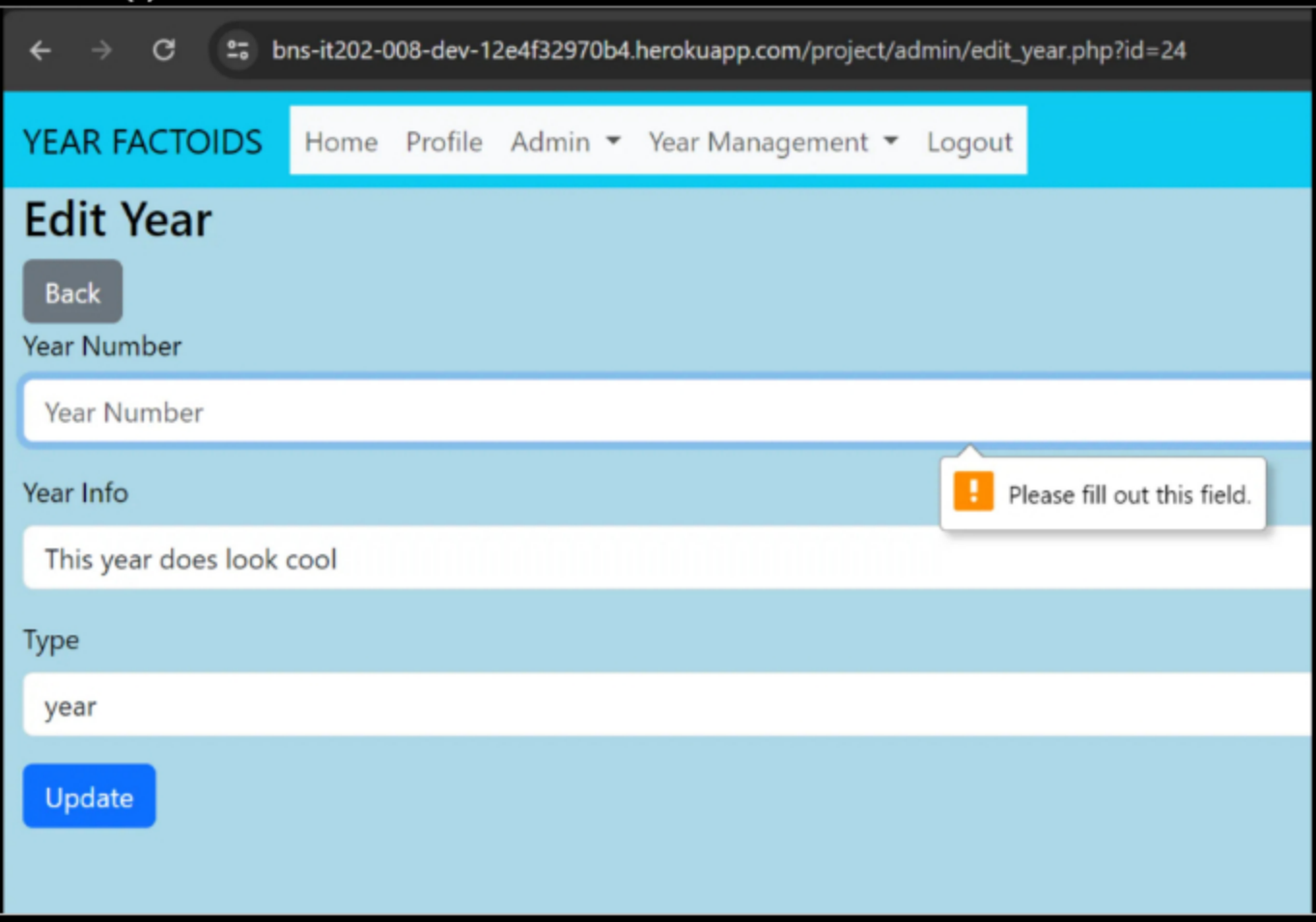
Gallery Style: Large View

Small Medium Large



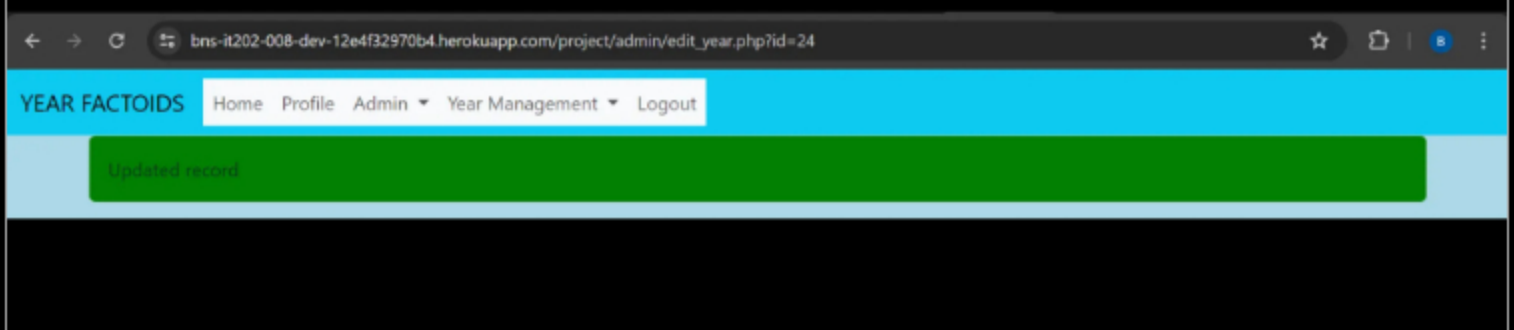
Before

Checklist Items (0)



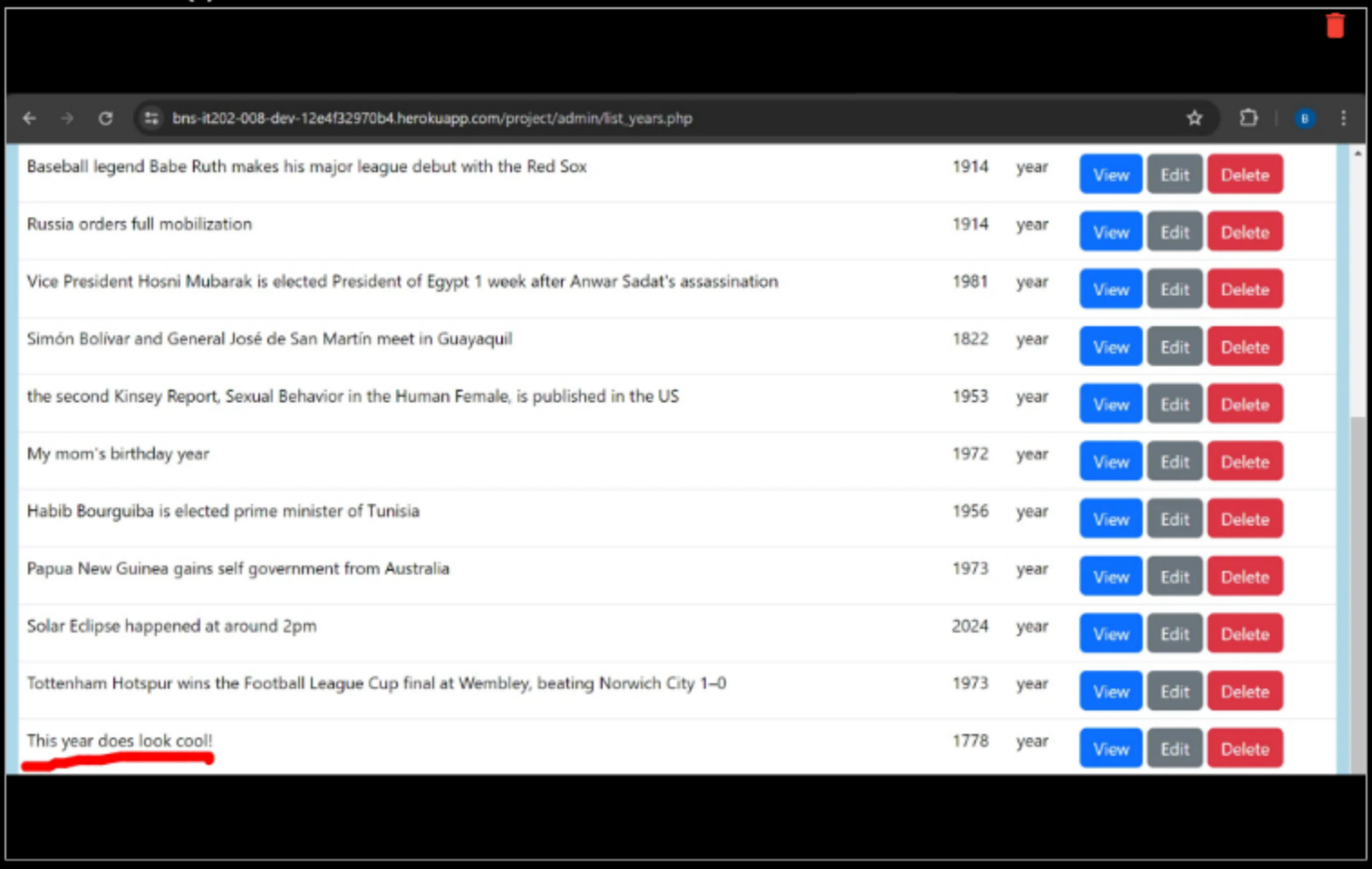
Validation

Checklist Items (0)



Successful edit flash message

Checklist Items (0)



Added an "!", after.

Checklist Items (0)

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

```php
if($yr){
    //bns24 04/14/24
    $form = [
        ["type" => "number", "name" => "number", "placeholder" => "Year Number", "label" => "Year Number", "rules" => ["required" => "required"]],
        ["type" => "text", "name" => "text", "placeholder" => "Year Info", "label" => "Year Info", "rules" => ["required" => "required"]],
        ["type" => "select", "name" => "type", "label" => "Type", "options" => ["year" => "year"],"rules" => ["required" => "required"]]
    ]; <- #77-81 $form =
    $keys = array_keys($yr);
```

Form showing the correct datatypes requested.

Checklist Items (0)

```php
<?php if ($_label) : ?>
    <?php /* label field  bns24 04/14/24*/ ?>
    <label class="form-label" for="<?php se($_id); ?>"><?php se($_label); ?></label>
<?php endif; ?>

<?php if (!in_array($_type, $_non_stanard_types)) : ?>
    <?php /* input field */ ?>
    <input type="<?php se($_type); ?>" name="<?php se($_name); ?>" class="form-control" id="<?php se($_id); ?>" value="<?php se($_value); ?>" placeholder="<?php se($_placeholder); ?>"
    <?php echo $_rules;?> />
<?php elseif($_type === "textarea"):?>
    <textarea class="form-control" name="<?php se($_name); ?>" id="<?php se($_id); ?>" placeholder="<?php se($_placeholder); ?>" <?php echo $_rules;?>><?php se($_value);?></textarea>
<?php elseif ($_type == "select") : ?>
    <select class="form-select" name="<?php se($_name); ?>" value="<?php se($_value); ?>" id="<?php se($_id); ?>">
        <?php foreach ($_options as $k => $v) : ?>
            <option <?php echo (isset($_value) && $_value === $k ? "selected" : ""); ?> value="<?php se($k); ?>"><?php se($v); ?></option>
        <?php endforeach; ?>
    </select>
<?php endif; ?>
```

Validation for each input field because it uses render_input.

Checklist Items (0)

```php
$query .= " WHERE id = :id";
$params[":id"] = $id;
var_export($query);
error_log("Query: " . $query);
error_log("Params: ". var_export($params, true));
try{
    //bns24 04/14/24
    $stmt = $db->prepare($query);
    $stmt->execute($params);
    flash("Updated record", "success");
}catch(PDOException $e){
    error_log("Something went wrong.");
}
```

Successful update has a user friendly flash message.

Checklist Items (0)

public_html > project > admin > edit_year.php > ...

```php
<?php
//note we need to go up 1 more directory
//bns24 04/14/24
require(__DIR__ . "/../../../partials/nav.php");

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

```
10  ?>
```

Role check for admin

Checklist Items (0)

```php
if($id > -1){
    //fetch
    $db = getDB();
    $query = "SELECT id, text, year, type FROM `Numbers` WHERE id = :id";
    try{
        $stmt = $db->prepare($query);
        $stmt->execute([":id"=>$id]);
        $r = $stmt->fetch();
        if($r){
            $yr = $r;
        }
    }catch(PDOException $e){
        error_log("Error fetching record: " . var_export($e, true));
        flash("Error fetching record.", "danger");
    }
}else{
    flash("Invalid id passed", "danger");
    die(header("Location: " . get_url("admin/list_years.php")));
}
```

User friendly flash message in case of errors.

Checklist Items (0)

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

	> 11	12	Solar Eclipse happened at a	2024	year	2024-04-14 01:41:01	2024-04-15 01:26:42	0

Before

Checklist Items (0)

	> 11	12	Solar Eclipse, 2:00 P.M	2024	year	2024-04-14 01:41:01	2024-04-15 04:44:11	0

After Difference is in the text that describes the year 2024.

Checklist Items (0)

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:

We first check to see if the user has the admin role. We fetch the id and select all the data that is associated with that id, and put it int an array. If there is any error during this process, then we flash a user-friendly error message. If the array with the data exists, then we prepare a separate array to build the form. We make sure that each of the form parameters has the correct types for validations (text would text, year would be number, and type would be a select). Then using the information from before, we populate the form with that data on page load, and then render each of the inputs. Next, We get the id from the get method where we prepare a query where we update the database. We take the result which comes from the POST method from the actual edit form. Because of the way the form works, I need to make sure that if the key is "number", I need to rename it to "year" so that it can actually work with my table. Using an empty array params, we are able to build the query so that it updates the data with the new information where the id matches the id that was given before. We then prepare the database for the query, and if everything works out, we flash a user-friendly success message and the database is updated, otherwise we do an error_log saying what went wrong.

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://bns24-it202-008-prod-4d42e45e31d0.herokuapp.com/project/admin/edit_year.php?id=1

URL #2

https://github.com/BruceSiaNJIT/bns24-it202-008/pull/40

Delete Handling (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots related to delete

Checklist

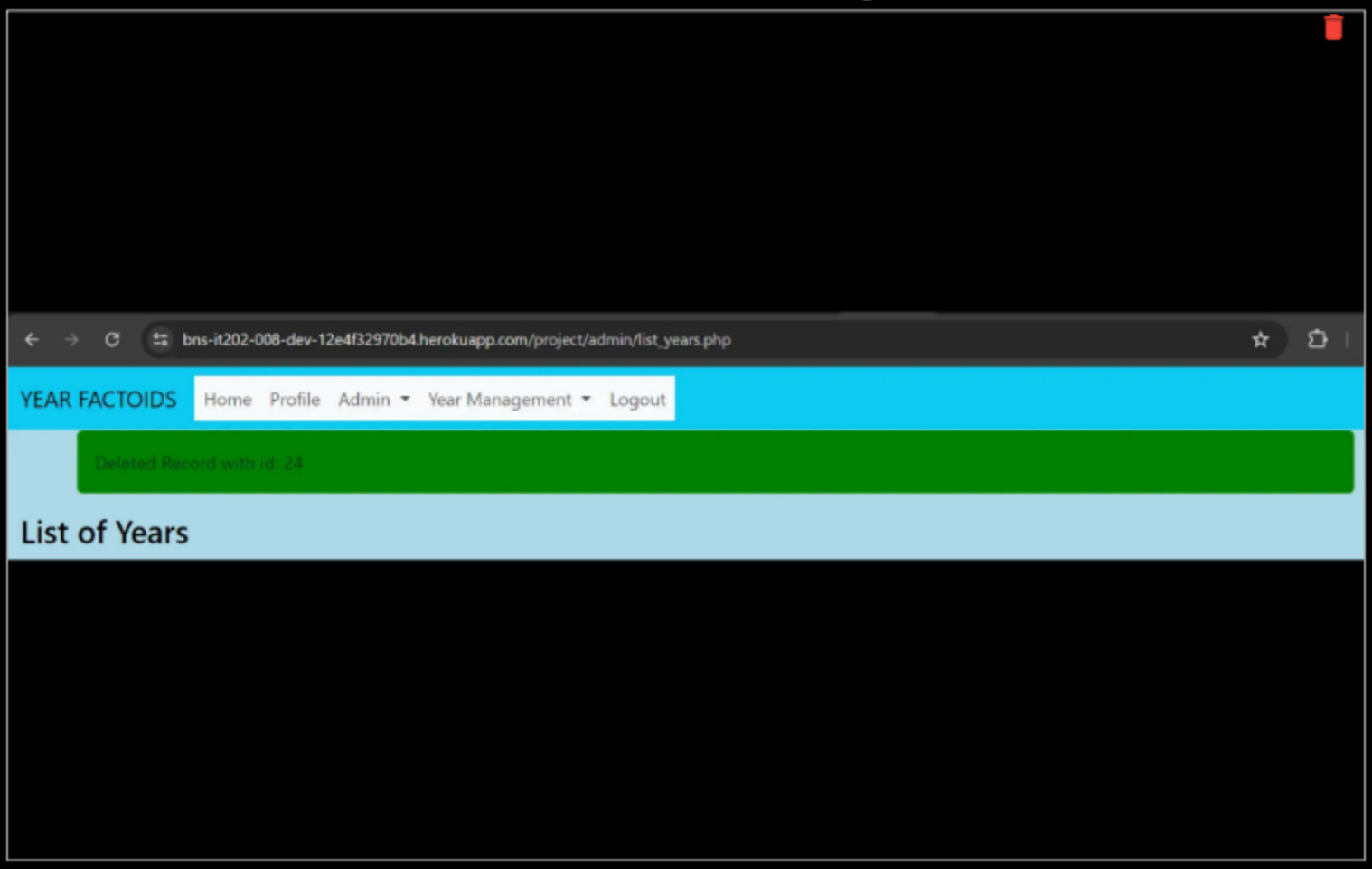
*The checkboxes are for your own tracking

#	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:

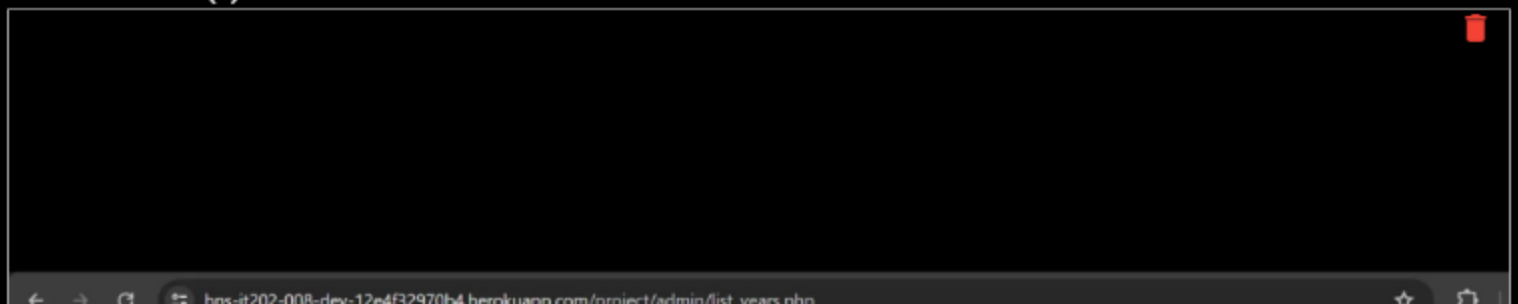
Gallery Style: Large View

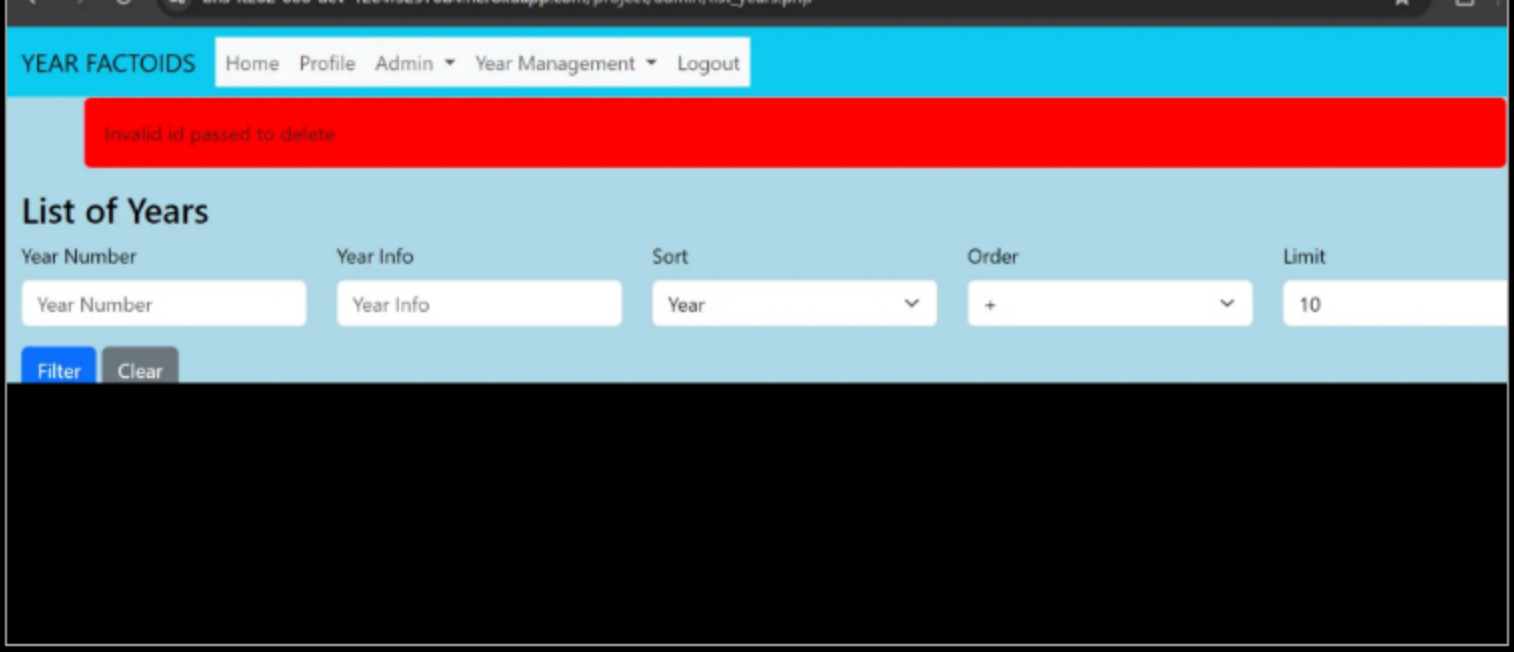
Small Medium Large



Success message for deleting a record

Checklist Items (0)





Error message when an id is not passed

Checklist Items (0)

public_html > project > admin > delete_year.php > ...

```php
<?php
//bns24 04/14/24
session_start();
require(__DIR__ . "/../../../lib/functions.php");

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

$id = se($_GET, "id", -1, false);
if($id < 1){
    flash("Invalid id passed to delete", "danger");
    die(header("Location: " . get_url("admin/list_years.php")));
}

$db = getDB();
$query = "DELETE FROM `Numbers` 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 stock $id" . var_export($e, true));
    flash("Error deleting record", "danger");
}
die(header("Location: " . get_url("admin/list_years.php")));
```

Code related to the delete processing

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the data

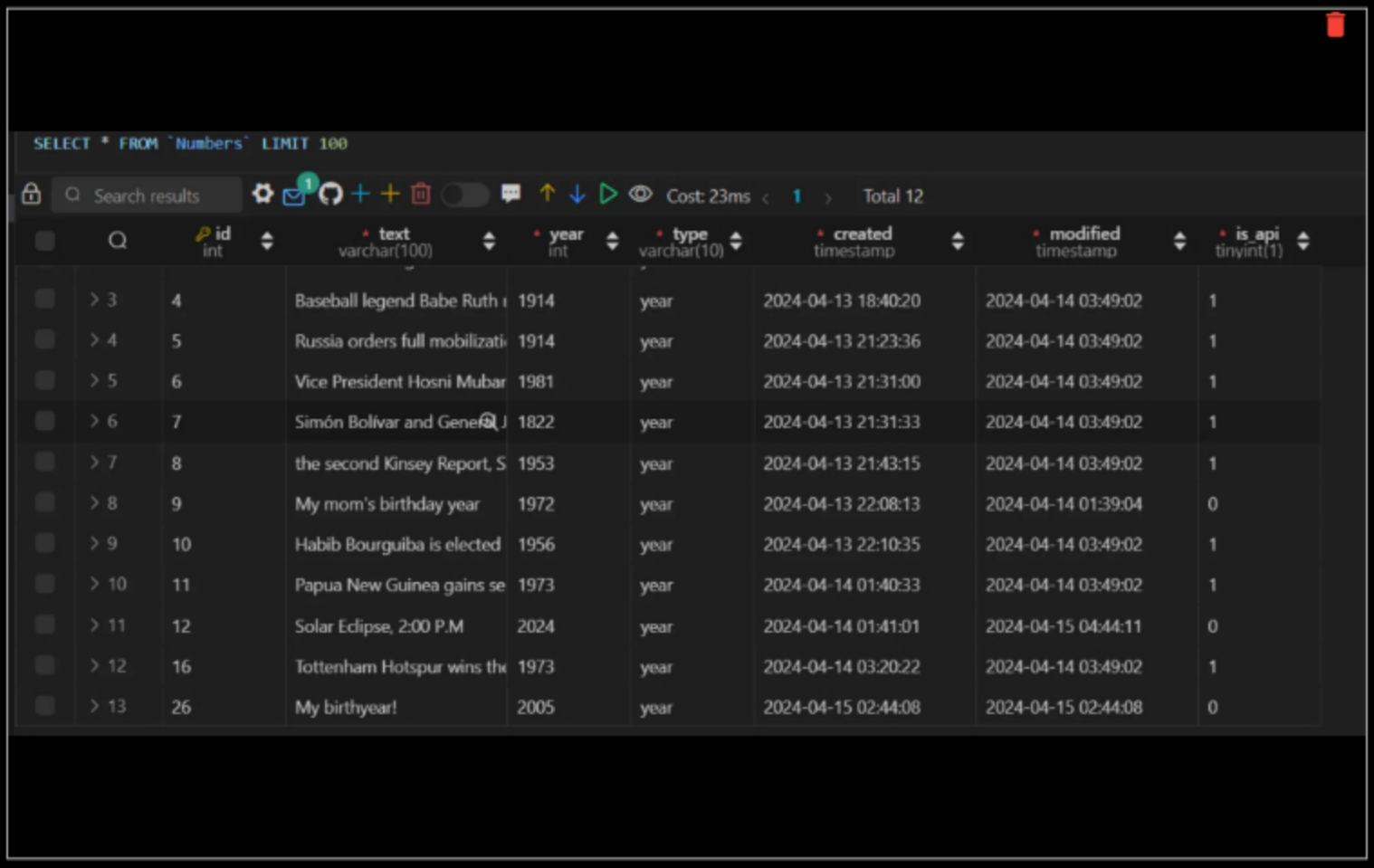
Checklist *The checkboxes are for your own tracking

#	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

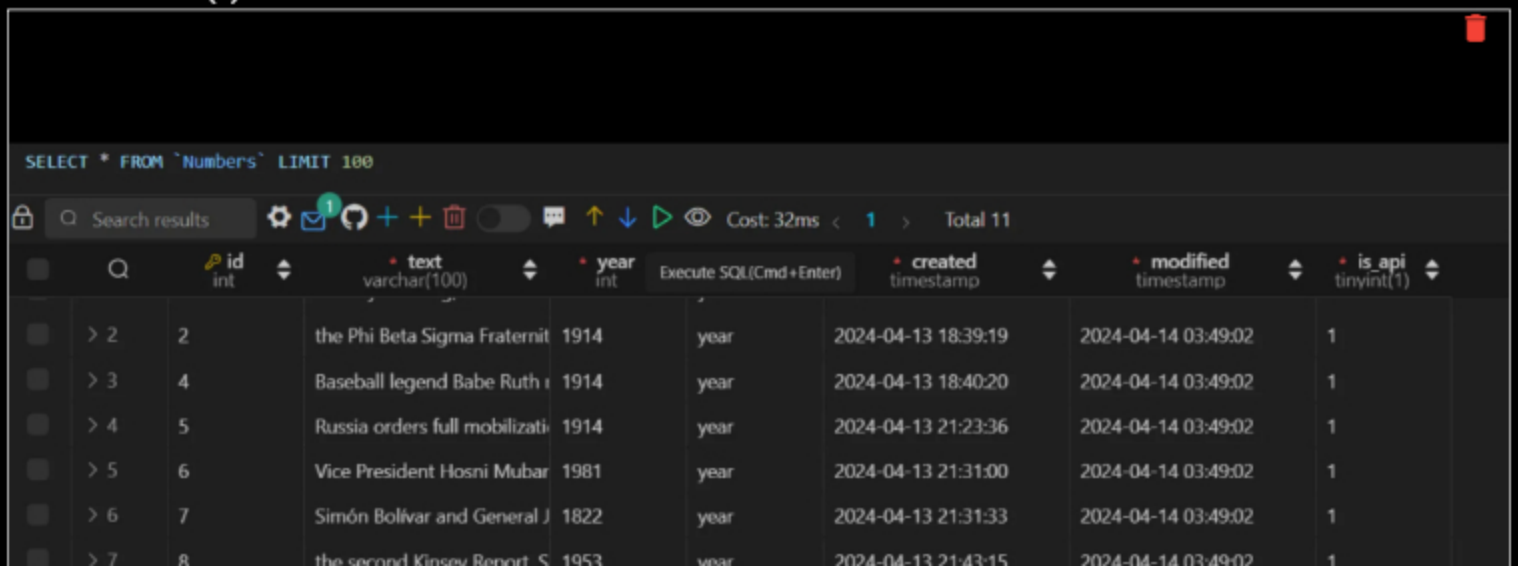
Small Medium Large



	id int	text varchar(100)	year int	type varchar(10)	created timestamp	modified timestamp	is_api tinyint(1)
3	4	Baseball legend Babe Ruth	1914	year	2024-04-13 18:40:20	2024-04-14 03:49:02	1
4	5	Russia orders full mobilizati	1914	year	2024-04-13 21:23:36	2024-04-14 03:49:02	1
5	6	Vice President Hosni Mubar	1981	year	2024-04-13 21:31:00	2024-04-14 03:49:02	1
6	7	Simón Bolívar and Gener	1822	year	2024-04-13 21:31:33	2024-04-14 03:49:02	1
7	8	the second Kinsey Report, S	1953	year	2024-04-13 21:43:15	2024-04-14 03:49:02	1
8	9	My mom's birthday year	1972	year	2024-04-13 22:08:13	2024-04-14 01:39:04	0
9	10	Habib Bourguiba is elected	1956	year	2024-04-13 22:10:35	2024-04-14 03:49:02	1
10	11	Papua New Guinea gains se	1973	year	2024-04-14 01:40:33	2024-04-14 03:49:02	1
11	12	Solar Eclipse, 2:00 P.M	2024	year	2024-04-14 01:41:01	2024-04-15 04:44:11	0
12	16	Tottenham Hotspur wins the	1973	year	2024-04-14 03:20:22	2024-04-14 03:49:02	1
13	26	My birthyear!	2005	year	2024-04-15 02:44:08	2024-04-15 02:44:08	0

id 11 will be the one deleted BEFORE

Checklist Items (0)



	id int	text varchar(100)	year int	type varchar(10)	created timestamp	modified timestamp	is_api tinyint(1)
2	2	the Phi Beta Sigma Fraternit	1914	year	2024-04-13 18:39:19	2024-04-14 03:49:02	1
3	4	Baseball legend Babe Ruth	1914	year	2024-04-13 18:40:20	2024-04-14 03:49:02	1
4	5	Russia orders full mobilizati	1914	year	2024-04-13 21:23:36	2024-04-14 03:49:02	1
5	6	Vice President Hosni Mubar	1981	year	2024-04-13 21:31:00	2024-04-14 03:49:02	1
6	7	Simón Bolívar and General J	1822	year	2024-04-13 21:31:33	2024-04-14 03:49:02	1
7	8	the second Kinsey Report, S	1953	year	2024-04-13 21:43:15	2024-04-14 03:49:02	1

> 8	9	My mom's birthday year	1972	year	2024-04-13 22:08:13	2024-04-14 01:39:04	0
> 9	10	Habib Bourguiba is elected	1956	year	2024-04-13 22:10:35	2024-04-14 03:49:02	1
> 10	12	Solar Eclipse, 2:00 P.M	2024	year	2024-04-14 01:41:01	2024-04-15 04:44:11	0
> 11	16	Tottenham Hotspur wins the	1973	year	2024-04-14 03:20:22	2024-04-14 03:49:02	1
> 12	26	My birthyear!	2005	year	2024-04-15 02:44:08	2024-04-15 02:44:08	0

After, id 11 is deleted from the database

Checklist Items (0)

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:

On both the edit page and the view single item page, there is a button for the delete. This passes in the id for delete_year.php. It first checks to see if the user is an admin, then it checks to see if the id passed is valid. If it is not, a user friendly flash message will appear stating so, and it will redirect you back to the list page. If it passes, then it prepares the query where it deletes the data where the id matches the id that was passed previously. If everything works, it flashes a user-friendly flash message saying that it removed a record with that id. Otherwise, it flashes a user-friendly message stating that there was an error while deleting the record. This is a hard delete, and cannot be recovered once deleted.

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/BruceSiaNJIT/bns24-it202-008/pull/41

Misc (1 pt.)

COLLAPSE

COLLAPSE

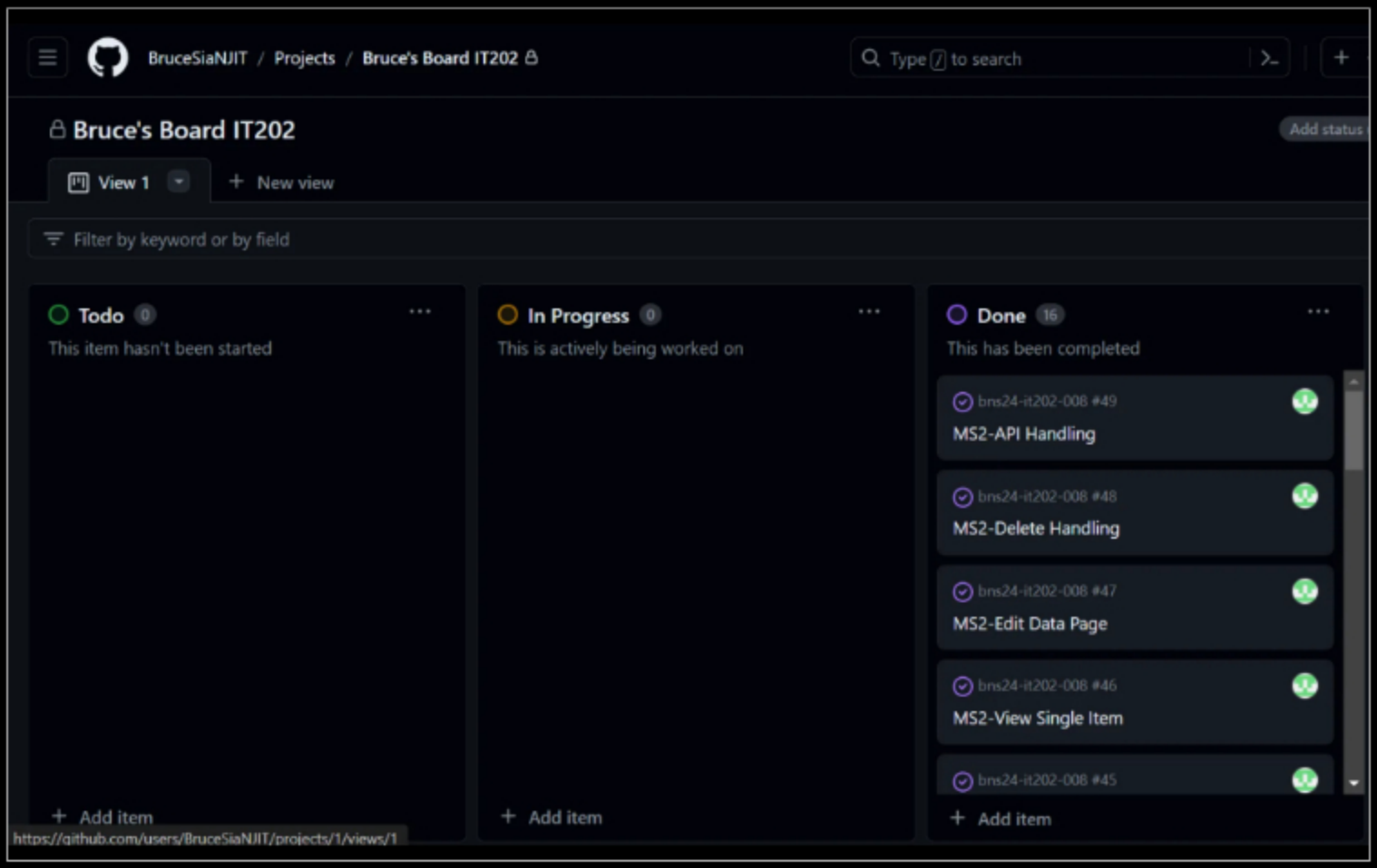
Task #1 - Points: 1

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Screenshot of the Project board, everything in the right column

COLLAPSE

Task #2 - Points: 1

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

URL #1

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

COLLAPSE

Task #3 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

This milestone I definitely learned a lot. This was kind of the first time I was really doing this on my own. Compared to Milestone 1, I mainly just copied pasted most of it. But for Milestone 2, because we all have our own unique API's, I couldn't simply just copy paste and I actually had to take what was being said in the videos and lectures and find a way to incorporate them for my specific API. It was very, very rewarding once I got certain parts to work.

COLLAPSE

Task #4 - Points: 1

Text: WakaTime Screenshot

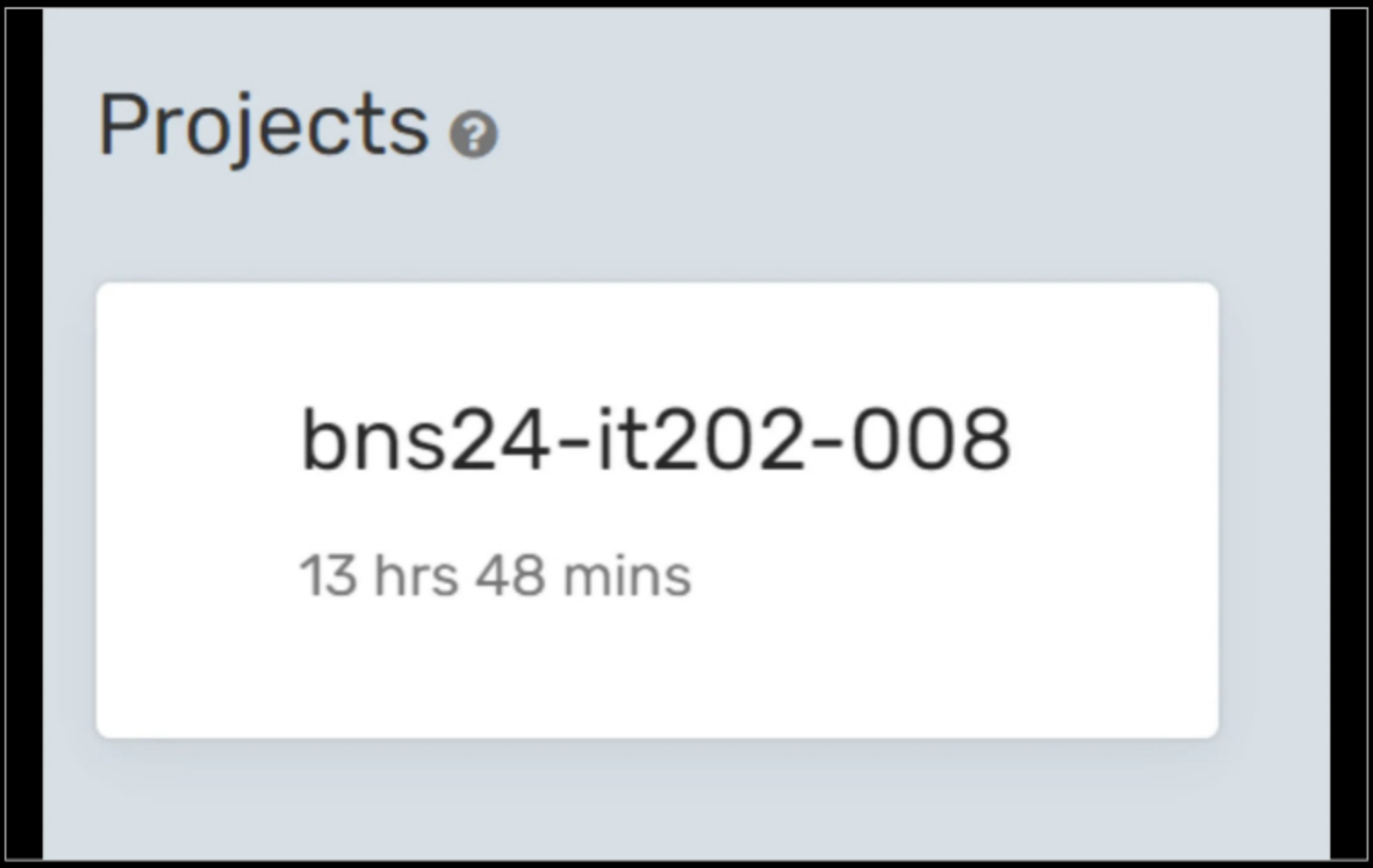
Details:

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



Screenshot of WakaTime

End of Assignment