Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-452-M2024/it202-module-6-milestone-1-2024/grade/mrs43

IT202-452-M2024 - [IT202] Module 6 Milestone 1 2024

Submissions:

Submission Selection

1 Submission [active] 7/8/2024 12:04:56 AM

Instructions

COLLAPSE

Overview Video: https://youtu.be/V7oHa8KKtss

Prereqs:

- Go through each lesson from "Project Setup and SQL" to "User Login Enhancement" and follow the branching names while gathering the code
- Merge each into Milestone1 branch
- Create a Project board on GitHub (if you haven't yet)
    - Add each major item from the proposal doc as an Issue item
    - Invite the grader(s) and myself as collaborators on the board (they're separate from your repository)
        - See Canvas announcements for the Usernames or check your collab list on your repo
- Mark the related GitHub Issues items as "done"
- Implement your own custom CSS (something much different than the default "ugly" CSS given as an example)
- Consider styling all forms/inputs, data output, navigation, etc
- Implement JavaScript validation on Register, Login, and Profile (include "[Client]" in the output messages to differentiate between server-side validations)

Instructions:

1. Make sure you're in Milestone1 with the latest changes pulled
2. Ensure Milestone1 has been deployed to heroku dev
3. Gather the requested evidence and fill in the explanations per each prompt
4. Save the submission and generate the output PDF
5. Put the output PDF into your local repository folder

6. add/commit/push it to GitHub
7. Merge Milestone1 into dev
8. Locally checkout dev and pull the changes
9. Create and merge a pull request from dev to prod to deploy Milestone1 to prod
10. Upload this output PDF to Canvas

Branch name: Milestone1

Tasks: 25 Points: 10.00

User Registration (2 pts.)

^COLLAPSE ^

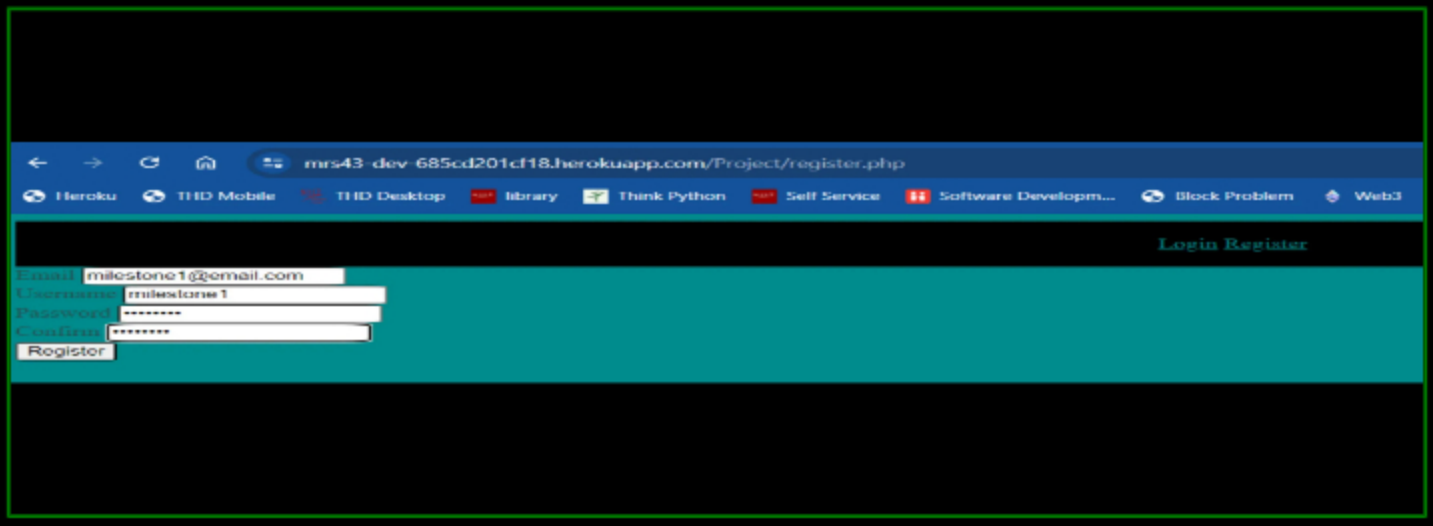
^COLLAPSE ^

Task #1 - Points: 1

Text: Screenshot of form on website page

Details:
Thoughtful CSS should be applied to all parts (must differ from the "ugly" CSS given via the lessons).
The Heroku dev URL must be present in all screenshots of the site.

#1) Screenshot of the form (ensure valid data is filled in)



Caption (required) ✓

Describe/highlight what's being shown

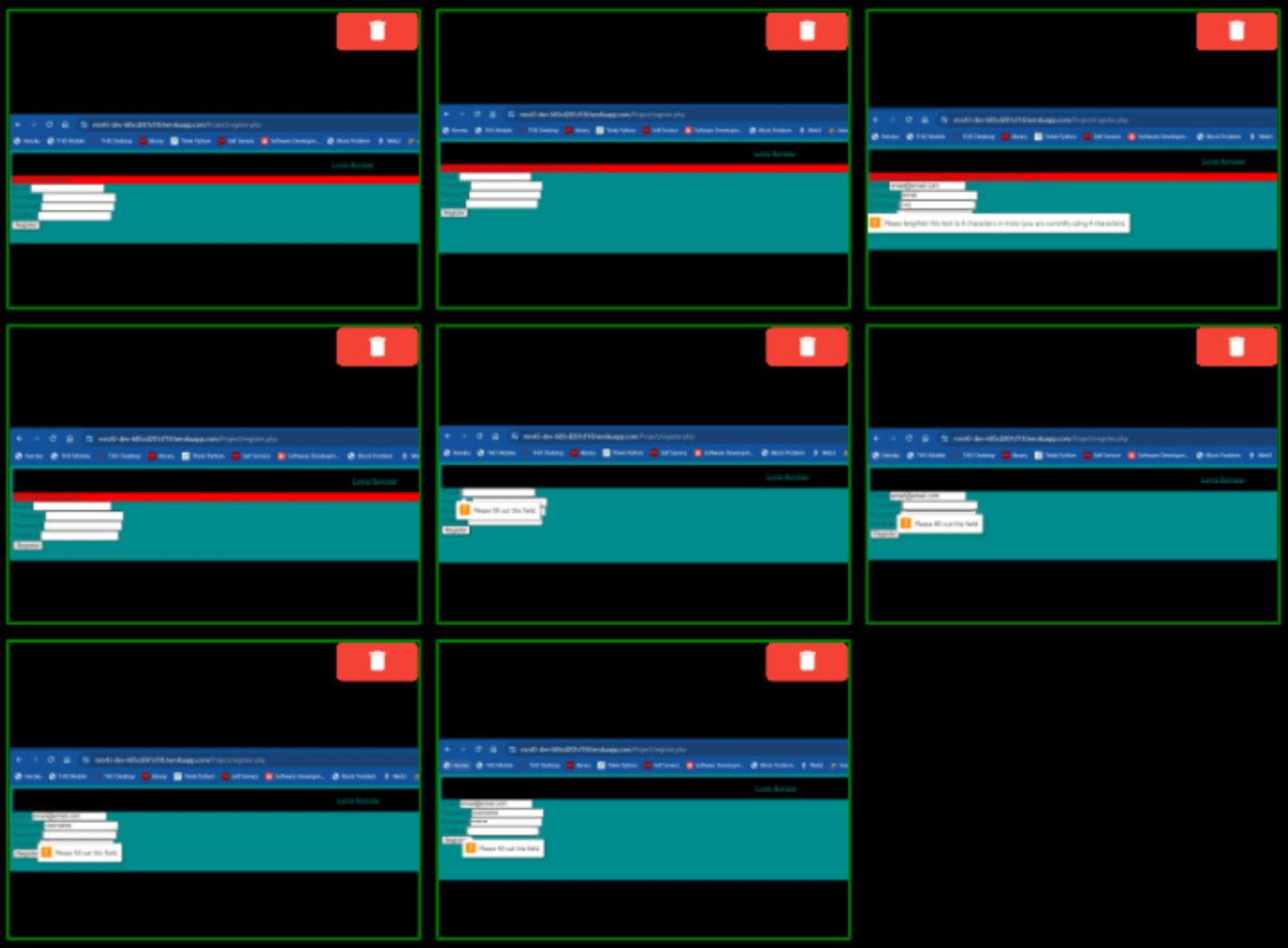
Screenshot of registration form on Heroku

URL (required) ✓

Prod link to the registration page

https://mrs43-prod-5f0e0e79560e.herokuapp.com/Project/register.php

#2) Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required

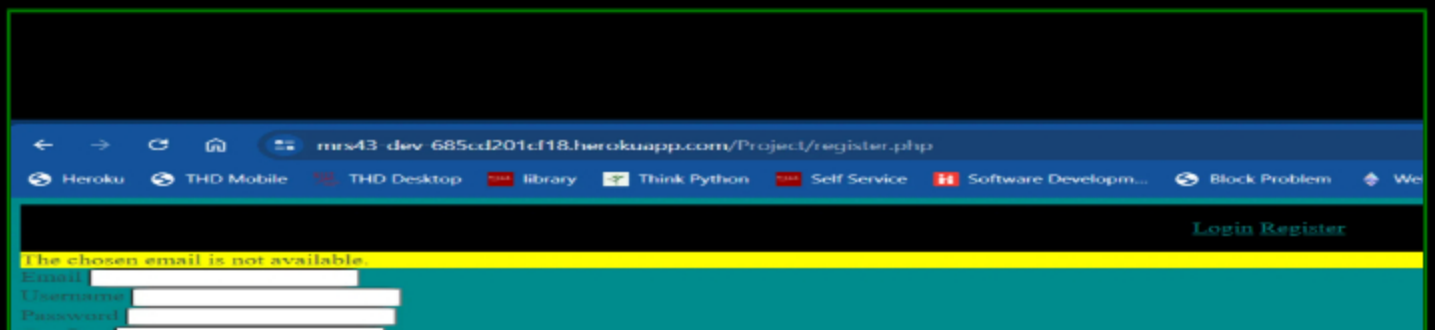


Caption (required) ✓

Describe/highlight what's being shown

1, 2, 4 show JS validation. 3 shows built in validation because it appeared before JS. 5-8 show fields are required.

#3) Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)

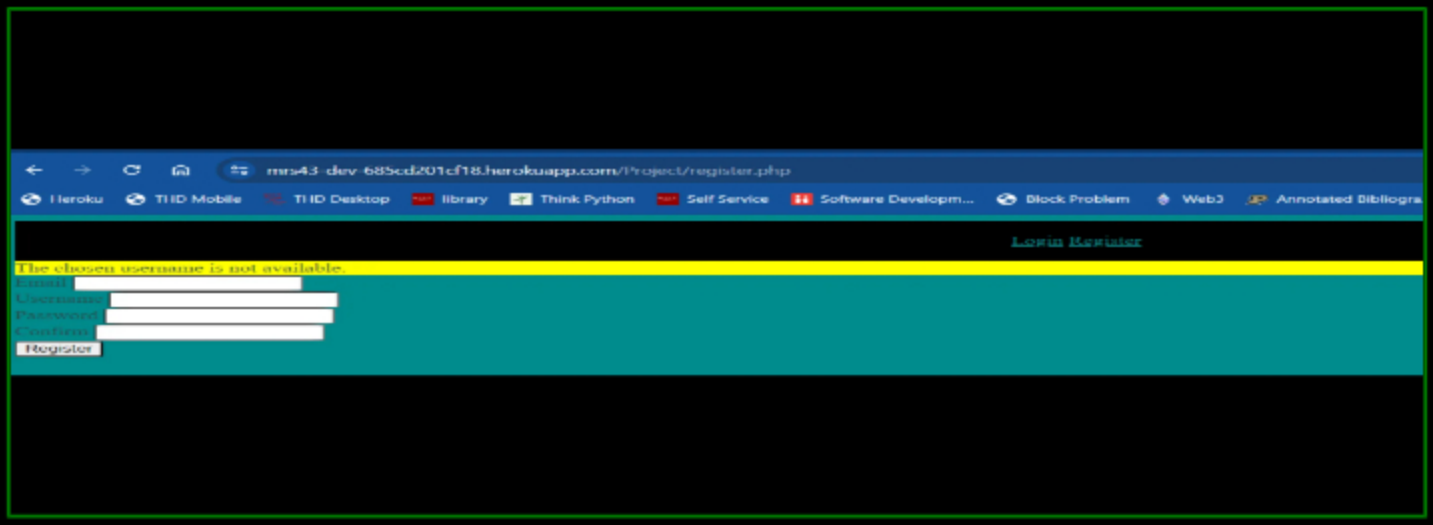


Caption (required) ✓

Describe/highlight what's being shown

Screenshot of email already in use on Heroku

#4) Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)

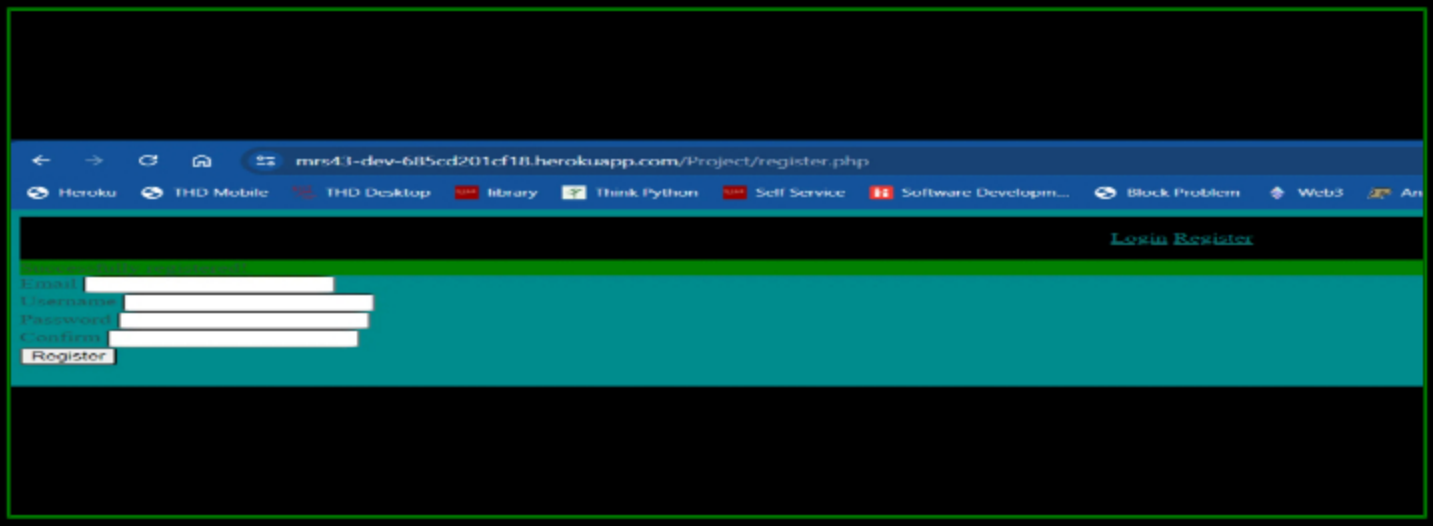


Caption (required) ✓

Describe/highlight what's being shown

Screenshot of username already in use on Heroku

#5) Demonstrate user-friendly message of new account being created



Caption (required) ✓

Describe/highlight what's being shown
Screenshot of message when new account is registered

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the form code

Details:
Should have the appropriate type attributes for the fields.
Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

#1) Show the html of the form

```
// mrs43 7-8-2024
<form onsubmit="return validate(this)" method="POST">
    <div>
        <label for="email">Email</label>
        <input type="email" name="email" required />
    </div>
    <div>
        <label for="username">Username</label>
        <input type="text" name="username" required maxlength="30" />
    </div>
    <div>
        <label for="pw">Password</label>
        <input type="password" id="pw" name="password" required minlength="8" />
    </div>
    <div>
        <label for="confirm">Confirm</label>
        <input type="password" name="confirm" required minlength="8" />
    </div>
    <input type="submit" value="Register" />
</form>
```

Caption (required) ✓
Describe/highlight what's being shown
Screenshot of the HTML of the registration form

Explanation (required) ✓
Briefly explain the html for each field including the chosen attributes
PREVIEW RESPONSE
There are four fields: email, username, password, and confirm. The email field is an email type and is required. The username field is just text (or a string), but is required and must be at most 30 characters. The password field is a password type, with an ID to make it easier to reference in other places, and is required with a minimum length of 8 characters/digits. The confirm field is also a password type, with the same requirements as the password field, but without an ID. There is also a button with the text "Register" on it.

COLLAPSE

Task #3 - Points: 1

Text: Screenshot of the client-side and server-side validation code

Details:
Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

#1) Show the JavaScript validations of the form (include any extra files related if you made separate files)

```
function validate(form) {
    //mrs43 7-8-2024
    //TODO 1: implement JavaScript validation
    //ensure it returns false for an error and true for success
    let isValid = true;
    let email = form.email.value;
    let username = form.username.value;
    let password = form.password.value;
    let confirm = form.confirm.value;

    if(!email){
        isValid = false;
        flash("An email is required", "warning");
    }
    if(!/^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$/.test(email)){
        isValid = false;
        flash("The email entered is invalid", "warning");
    }
    if(!username){
        isValid = false;
        flash("A username is required", "warning");
    }
```

```
//mrs43 7-8-2024
if(!/^[a-z0-9_-]{3,16}$/.test(username)){
    isValid = false;
    flash("The username entered is invalid", "warning");
}
if(!password){
    isValid = false;
    flash("A password is required", "warning");
}
if(!confirm){
    isValid = false;
    flash("Please re-enter your password", "warning");
}
if(!password.length() < 8){
    isValid = false;
    flash("The password entered is invalid", "warning");
}
if(password && !isEqual(password,confirm)){
    isValid = false;
    flash("The passwords must match", "warning");
}

return true;
```

Caption (required) ✓

Describe/highlight what's being shown

Screenshots of JS validation in VSCode

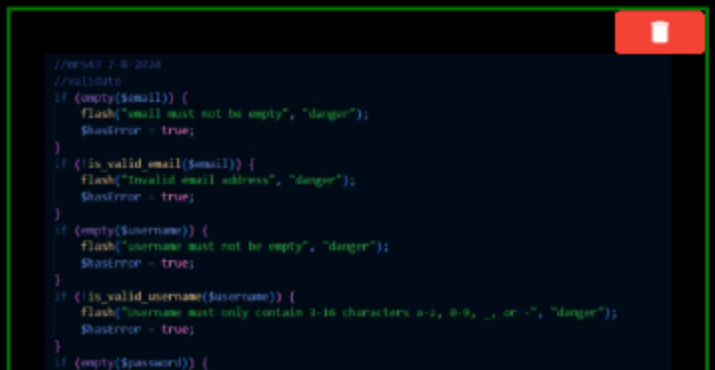
Explanation (required) ✓

Explain in concise steps how this logically works

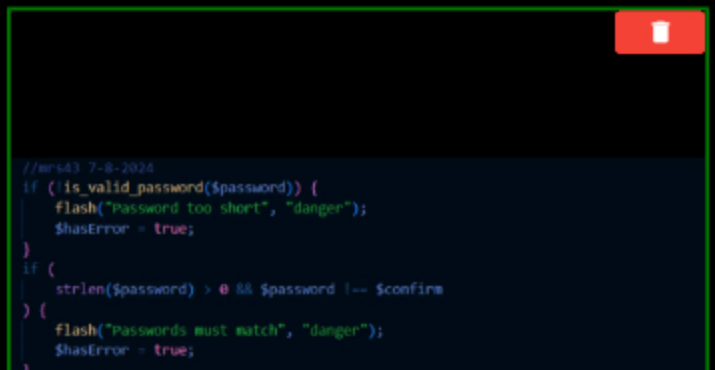
PREVIEW RESPONSE

This is a collection if if statements that test the fields email, username, password, and confirm. There is a condition for each of the fields to test if it is filled out. There are also three functions that test if the email, username, and passwords entered are valid. There is one more that tests if the password value is equal to the confirm value. If the fields are not valid, there is a flash message describing the error as well as a variable isValid that is set to false.

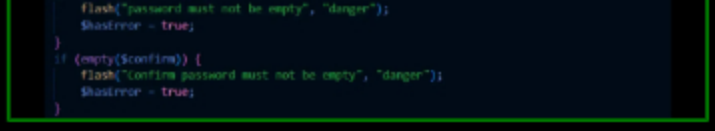
#2) Show the PHP validations (include any lib content)

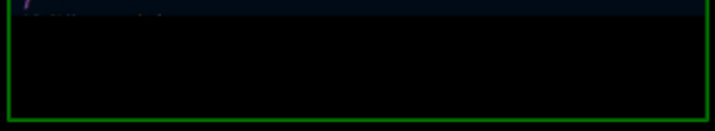


```
//mrs43 7-8-2024
//validate
if (empty($email)) {
    flash("email must not be empty", "danger");
    $hasError = true;
}
if (!is_valid_email($email)) {
    flash("Invalid email address", "danger");
    $hasError = true;
}
if (empty($username)) {
    flash("username must not be empty", "danger");
    $hasError = true;
}
if (!is_valid_username($username)) {
    flash("Username must only contain 3-16 characters a-z, 0-9, _, or -", "danger");
    $hasError = true;
}
if (empty($password)) {
```



```
//mrs43 7-8-2024
if (!is_valid_password($password)) {
    flash("Password too short", "danger");
    $hasError = true;
}
if (
    strlen($password) > 0 && $password !== $confirm
) {
    flash("Passwords must match", "danger");
    $hasError = true;
}
```





Caption (required) ✓

Describe/highlight what's being shown

Screenshots of PHP validation in VSCode

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

Similar to the JavaScript validation, there are some conditionals that check if the required fields are empty and some that make sure the values entered are valid. If the fields are not valid, a flash message is sent and the variable hasError is set to true.

COLLAPSE

Task #4 - Points: 1

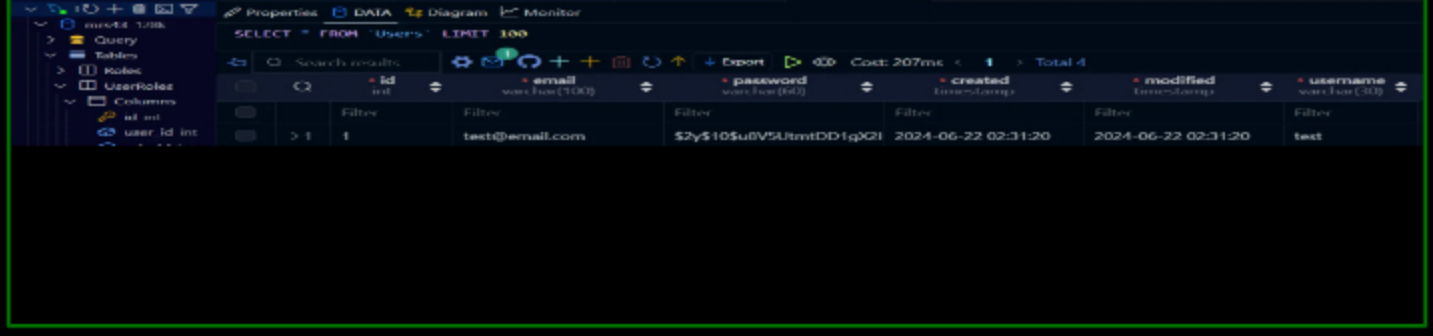
Text: Screenshot of the Users table with a valid user entry

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Password should be hashed
#2	1	Should have email, password, username (unique), created, modified, and id fields
#3	1	Ensure left panel or database name is present (should contain your ucid)

#1) Show valid data per the checklist



Caption (required) ✓

Describe/highlight what's being shown

Screenshot of Users table from VSCode

COLLAPSE

Task #5 - Points: 1

Text: Explain the registration logic in a step-by-step manner from when the page loads to when the data is saved to the DB

Details:

Don't just show code, translate things to plain English in concise steps.

May be worthwhile using a list.

Response:

The user fills out a form with the fields email, username, password, and confirm. The user's information is then validated using client-side validation (JavaScript) and server-side validation (PHP). If the user's entry is valid, the user's password is protected by hashing it, and the entry is saved to an SQL table called Users.

COLLAPSE

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

https://github.com/m-sansone/mrs43-IT202-452/pull/13

URL #2

https://github.com/m-sansone/mrs43-IT202-452/pull/23

URL #3

https://github.com/m-sansone/mrs43-IT202-452/pull/24

URL #4

https://github.com/m-sansone/mrs43-IT202-452/pull/27

URL #5
https://github.com/m-sansone/mrs43-IT202-452/pull/28

URL #6
https://github.com/m-sansone/mrs43-IT202-452/pull/29

URL #7
https://github.com/m-sansone/mrs43-IT202-452/pull/30

URL #8
https://github.com/m-sansone/mrs43-IT202-452/pull/48

URL #9
https://github.com/m-sansone/mrs43-IT202-452/pull/20

URL #10
https://github.com/m-sansone/mrs43-IT202-452/pull/52

User Login (2 pts.)

COLLAPSE

COLLAPSE

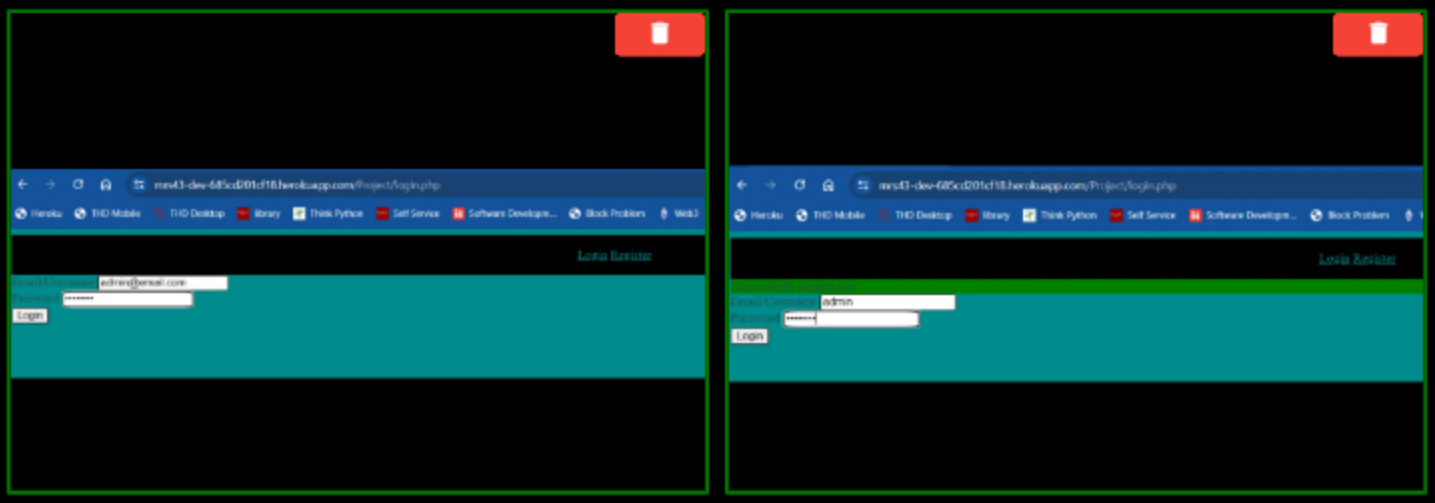
Task #1 - Points: 1

Text: Screenshot of form on website page

Details:

Thoughtful CSS should be applied to all parts (must differ from the "ugly" CSS given via the lessons).
The Heroku dev URL must be present in all screenshots of the site.

#1) Two Screenshot of the form (one with valid email data filled and one with valid username data filled)



Caption (required) ✓

Describe/highlight what's being shown

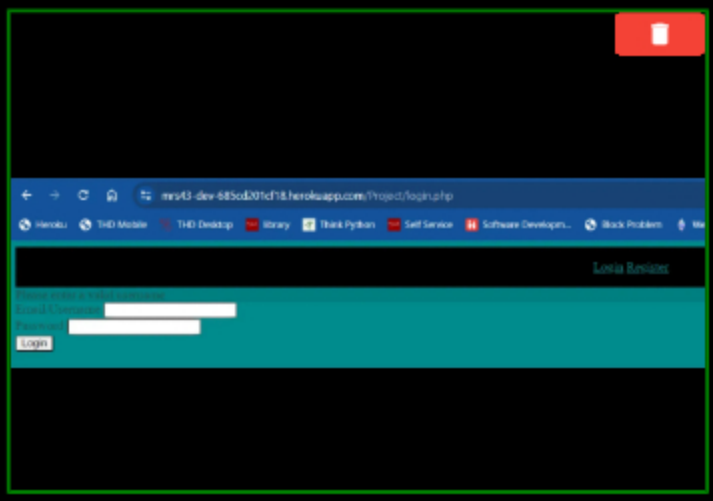
Left screenshot is valid email login for Heroku. Right one is valid username login for Heroku

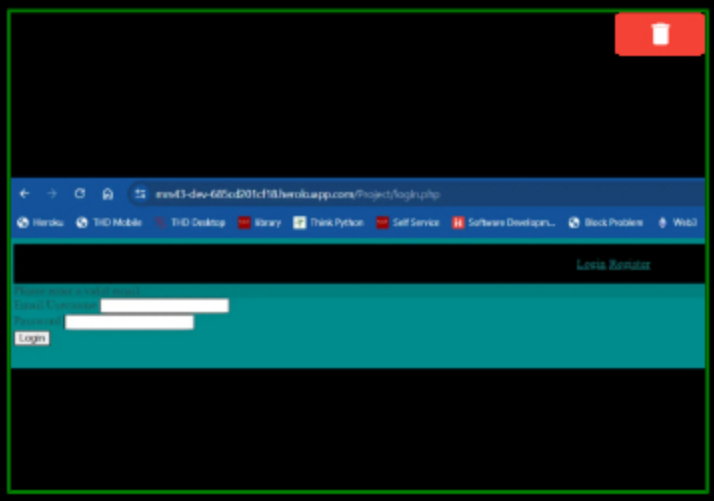
URL (required) ✓

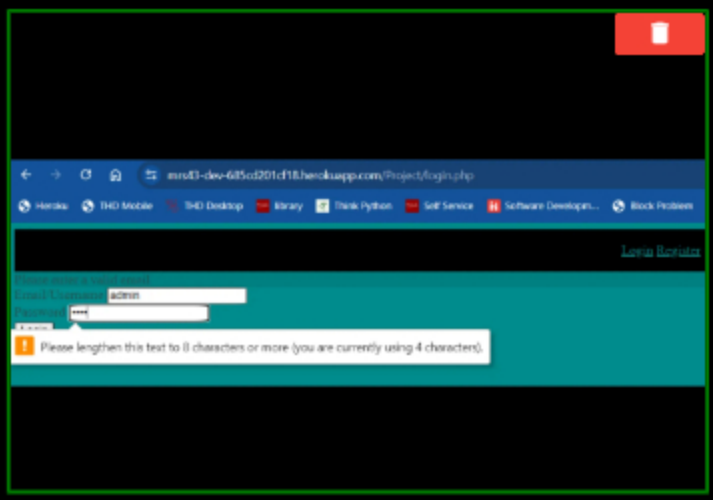
Prod link to the login page

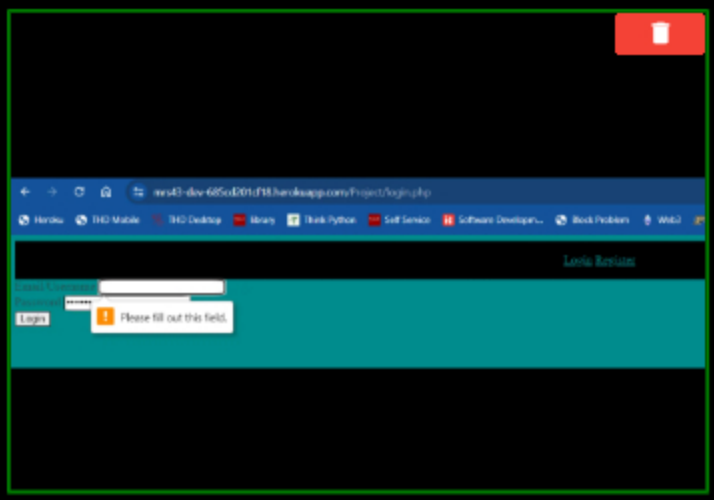
https://mrs43-prod-5f0e0e79560e.herokuapp.com/Project/login.php

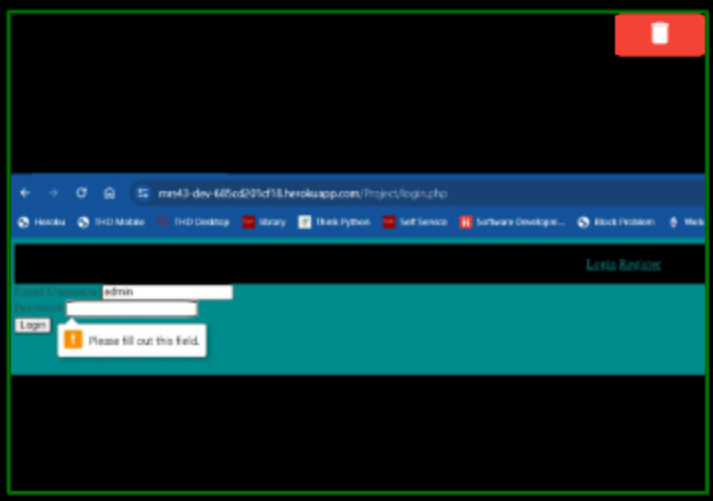
#2) Demonstrate JavaScript validation for each field [can be combined] (username format, email format, password format, and each field being required)











Caption (required) ✓

Describe/highlight what's being shown

1/2: error when wrong format username/email. 3: built in message for wrong format password (pops up before JS). 4/5: all require

#3) Demonstrate user-friendly message of when an account doesn't exist

Caption (required) ✓

Describe/highlight what's being shown

Error message when account doesn't exist

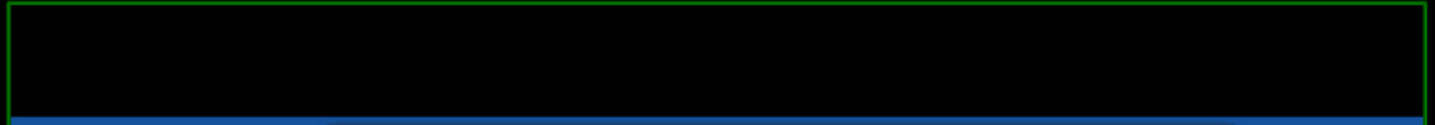
#4) Demonstrate user-friendly message of when password doesn't match what's in the DB

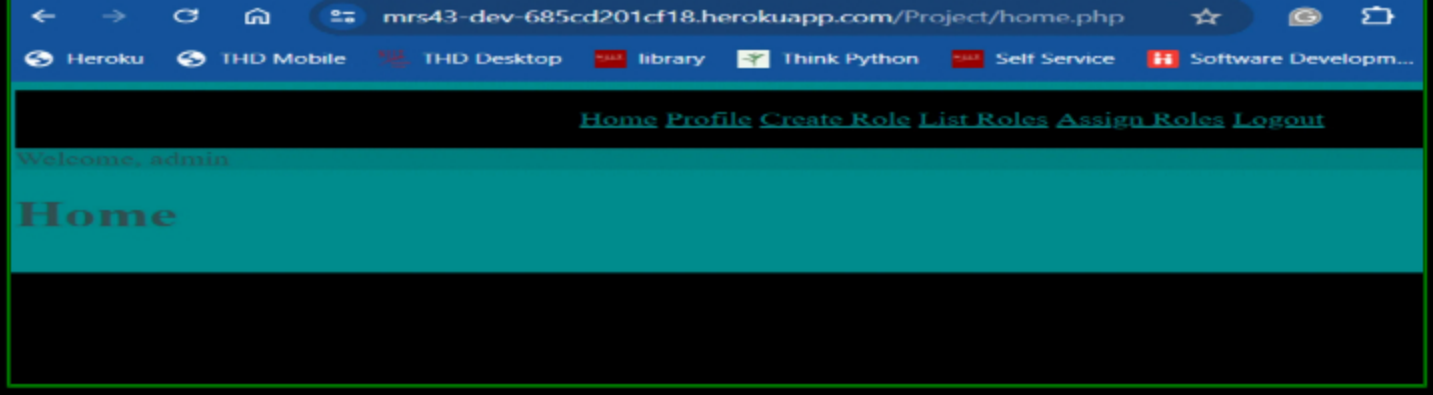
Caption (required) ✓

Describe/highlight what's being shown

Error message when wrong password is entered

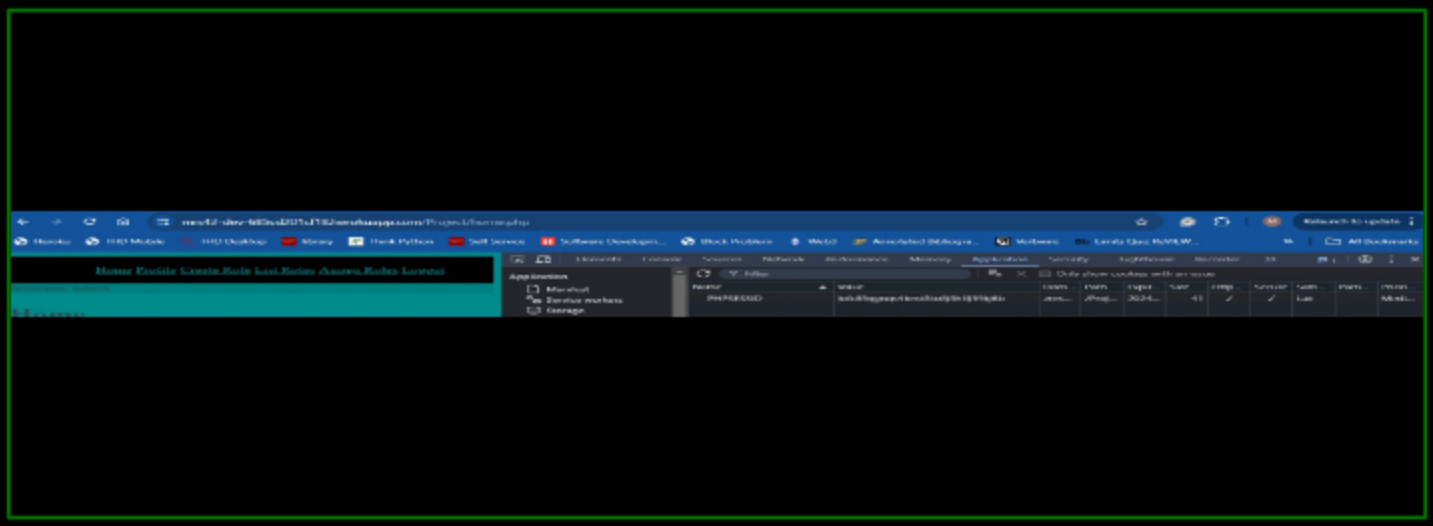
#5) Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)





Caption (required) ✓
Describe/highlight what's being shown
Success message and home page on Heroku

#6) Demonstrate session data being set (captured from server logs)



Caption (required) ✓
Describe/highlight what's being shown
cookies in developer tools on Chrome

^COLLAPSE ^

Task #2 - Points: 1

Text: Screenshot of the form code

Details:
Should have the appropriate type attributes for the fields.
Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

#1) Show the html of the form

```php
//mrs43 7-8-2024
require(__DIR__ . "/../../partials/nav.php");
?>
<form onsubmit="return validate(this)" method="POST">
    <div>
        <label for="email">Email/Username</label>
        <input type="text" name="email" required />
    </div>
    <div>
        <label for="pw">Password</label>
        <input type="password" id="pw" name="password" required minlength="8" />
    </div>
    <input type="submit" value="Login" />
</form>
```

Caption (required) ✓

Describe/highlight what's being shown

login page HTML in VSCode

Explanation (required) ✓

Briefly explain the html for each field including the chosen attributes

PREVIEW RESPONSE

There are two fields: email/username and pasword. The email/username field is a text/string type named "email" that is required. The password field is a password type, with an ID "pw" and name "password". It is required and must be at least 8 characters/digits. THere is also a submit button with the text "Login" that will validate the data when submitted.

#2) Show the JavaScript validations of the form (include any extra files related if you made separate files)

```javascript
function validate(form) {
    //mrs43 7-8-2024
    //TODO 1: implement JavaScript validation
    //ensure it returns false for an error and true for success
    let isValid = true;
    let email = form.email.value;
    let password = form.password.value;

    if(!/^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$/.test(email)){
        if(!/^[a-z0-9_-]{3,16}$/.test(email)){
            isValid = false;
            flash("Invalid username or email", "warning");
        }
    } <- #24-29 if(!/^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$/.test(e...
    if(password.length() < 1){
        isValid = false;
        flash("Please enter a password", "warning");
    }

    return isValid;
```

Caption (required) ✓

Describe/highlight what's being shown

JavaScript validation of form in VSCode

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

There are conditionals that check if each field (username/email and password) are empty or invalid. If invalid, there is a flash message explaining the error as well as a variable isValid that is set to false.

#3) Show PHP validations (include any lib content)

```php
//mrs43 7-8-2024
function sanitize_email($email = "")
{
    return filter_var(trim($email), FILTER_SANITIZE_EMAIL);
}
function is_valid_email($email = "")
{
    return filter_var(trim($email), FILTER_VALIDATE_EMAIL);
}
function is_valid_username($username)
{
    return preg_match('/^[a-z0-9_-]{3,16}$/', $username);
}
function is_valid_password($password)
{
    return strlen($password) >= 8;
}
```

Caption (required) ✓

Describe/highlight what's being shown

PHP validation of form in VSCode

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

There are conditionals that test if each field is empty or invalid. If they are, there is a flash message explaining the error and the variable hasError is set to true.

COLLAPSE

Task #3 - Points: 1

Text: Explain the login logic step-by-step from when the page loads to when the data is fetched from the DB and stored in the session

Details:

Don't just show code, translate things to plain English in concise steps.
Explain how the session works and why/how it's used

May be worthwhile using a list.

Response:

The user fills in the form with their username or email as well as their password. The user's input is then checked using client-side validation (JavaScript) and server-side validation (PHP). If the input is valid, the user's information is grabbed from the Users SQL database based on the username or email provided. The code checks if the username/email exists in the database. If it does, the password the user entered is compared to the hash. If it matches, the session is created. The code then sees if the user is assigned a role in the UserRoles database and saves the result to an array. The user is then navigated to the homescreen with a success message.

COLLAPSE

Task #4 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/m-sansone/mrs43-IT202-452/pull/25

URL #2
https://github.com/m-sansone/mrs43-IT202-452/pull/26

URL #3
https://github.com/m-sansone/mrs43-IT202-452/pull/27

URL #4
https://github.com/m-sansone/mrs43-IT202-452/pull/28

URL #5
https://github.com/m-sansone/mrs43-IT202-452/pull/29

URL #6
https://github.com/m-sansone/mrs43-IT202-452/pull/30

URL #7
https://github.com/m-sansone/mrs43-IT202-452/pull/36

URL #8
https://github.com/m-sansone/mrs43-IT202-452/pull/52

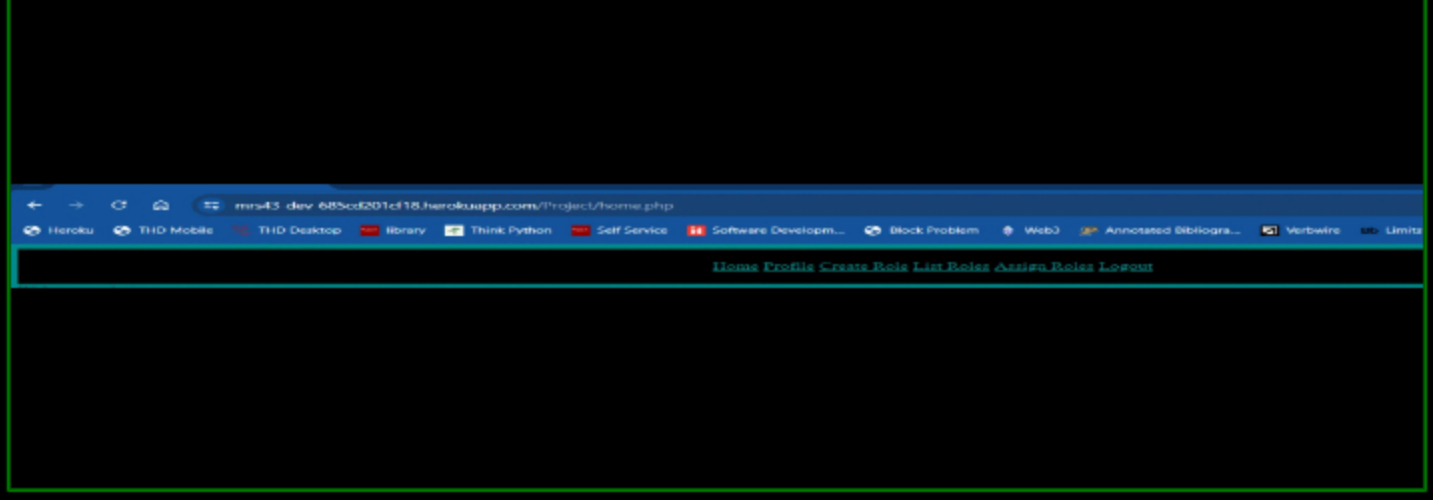
User Logout (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Capture the following screenshots

#1) Screenshot of the navigation when logged in (site)

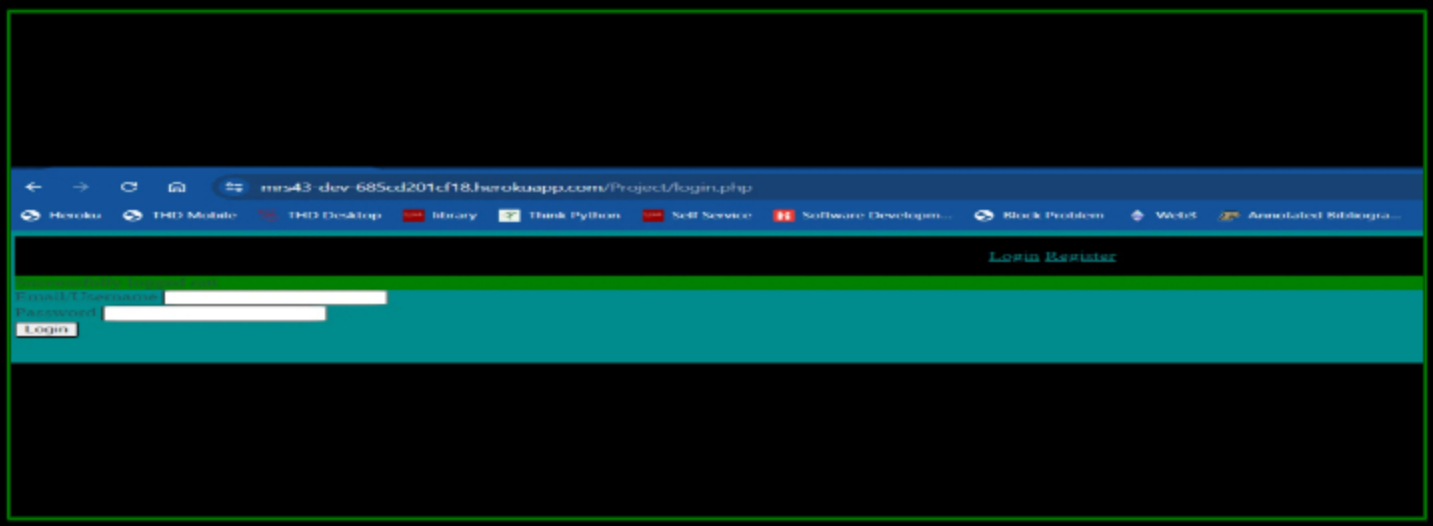


Caption (required) ✓

Describe/highlight what's being shown

Screenshot of navigation on Heroku when logged in

#2) Screenshot of the redirect to login with the user-friendly logged-out message (site)



Caption (required) ✓

Describe/highlight what's being shown

Screenshot of navigation to log in page when logged out on Heroku

#3) Screenshot of the logout-related code showing the session is destroyed (code). Ensure ucid/date comment is present.

```php
//mrs43 7-8-2024
session_start();
require(__DIR__ . "/../../lib/functions.php");
reset_session();

flash("Successfully logged out", "success");
```

```php
function reset_session()
{
    session_unset();
    session_destroy();
    session_start();
```

```
header("Location: login.php");
```

```
} <- #3-7 function reset_session()
```

Caption (required) ✓

Describe/highlight what's being shown

Screenshots on VSCode of session being destroyed when user logs out

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The session is started, then reset, before the user is logged out. When reset, the session resets the array variables from when the user logged in.

COLLAPSE

Task #2 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

https://github.com/m-sansone/mrs43-IT202-452/pull/25

URL #2

https://github.com/m-sansone/mrs43-IT202-452/pull/26

URL #3

https://github.com/m-sansone/mrs43-IT202-452/pull/28

URL #4

https://github.com/m-sansone/mrs43-IT202-452/pull/30

Basic Security Rules and Roles (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Authentication Screenshots

Details:

Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

#1) Screenshot of the function that checks if a user is logged in

```php
//mrs43 7-8-2024
function is_logged_in($redirect = false, $destination = "login.php")
{
    $isLoggedIn = isset($_SESSION["user"]);
    if ($redirect && !$isLoggedIn) {
        //if this triggers, the calling script won't receive a reply since die()/exit() terminates it
        flash("You must be logged in to view this page", "warning");
        die(header("Location: $destination"));
    } <- #10-14 if ($redirect && !$isLoggedIn)
    return $isLoggedIn;
} <- #8-16 function is_logged_in($redirect = false, $destination = "logi...
```

Caption (required) ✓

Describe/highlight what's being shown

Screenshot of is_logged_in function

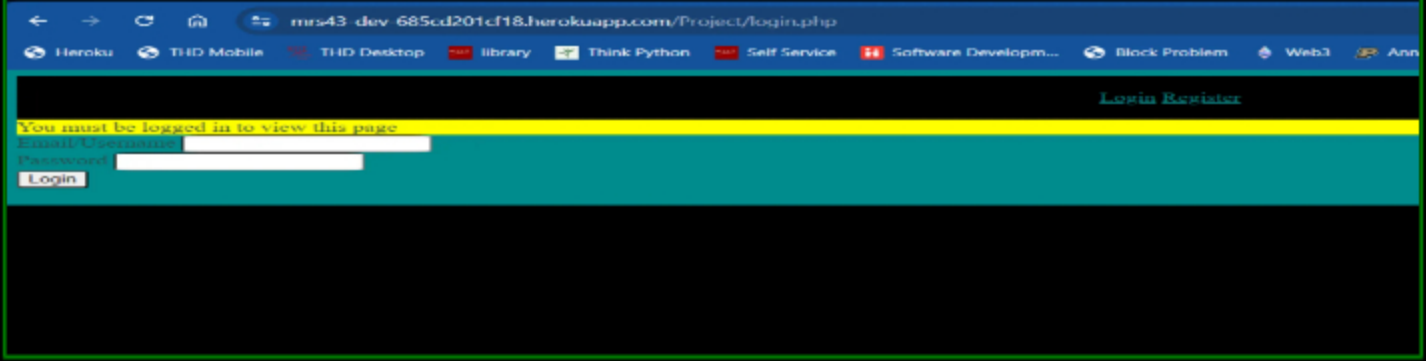
Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The code checks to see if the user is logged in via the array created when the user is logged in. If they are not, a flash message will say the user must be logged in to view the page and the user will be redirected to the login page.

#2) Screenshot of the login check function being used (i.e., profile likely). Also, caption what pages it's used on



Caption (required) ✓

Describe/highlight what's being shown

Screenshot of not logged in error message; is_logged_in used on home and profile pages

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

I logged out, then typed in the url for the profile page. The profile page first checks if the user is logged in. Because I was not logged in, I was redirected with an error message as the is_logged_in function does.

#3) Demonstrate the user-friendly message of trying to manually access a login-protected page while being logged out (must show the appropriate message)

Caption (required) ✓

Describe/highlight what's being shown

screenshot of logged out error message

COLLAPSE

Task #2 - Points: 1

Text: Authorization Screenshots

ℹ Details:

Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

#1) Screenshot of the function that checks for a specific role

```php
//mrs43 7-8-2024
function has_role($role)
{
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {
        foreach ($_SESSION["user"]["roles"] as $r) {
```

```
            if ($r["name"] === $role) {
                return true;
            }
        } <- #20-24 foreach ($_SESSION["user"]["roles"] as $r)
    } <- #19-25 if (is_logged_in() && isset($_SESSION["user"]["roles"]))
    return false;
} <- #18-27 function has_role($role)
```

Caption (required) ✓

Describe/highlight what's being shown

screenshot of function has_role

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

If the user is logged in, the function iterates through the user's roles to see if any match the one being tested for (which is used as a parameter for the function when it is called).

#2) Screenshot of the role check function being used. Also, caption what pages it's used on

```
//mrs43 7-8-2024
if (!has_role("Admin")) {
    flash("You don't have permission to view this page", "warning");
    die(header("Location: $BASE_PATH" . "/home.php"));
}
```

Caption (required) ✓

Describe/highlight what's being shown

function has_role being used; used on pages assign_roles, create_role, and list_roles

Explanation (required) ✓

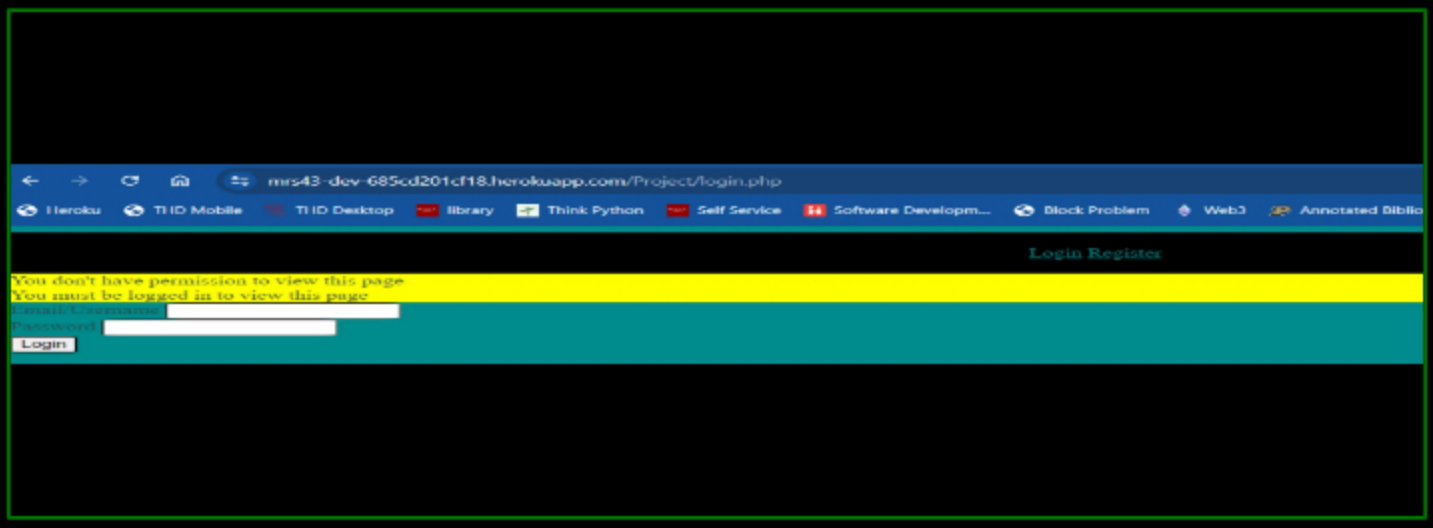
Explain in concise steps how this logically works

PREVIEW RESPONSE

The function tests if the user is logged in and has the Admin role. If not, there is an error message and the user is navigated back to the home page.

#3) Demonstrate the user-friendly message of trying to manually access a role-protected page while

being logged out (must show the appropriate message)



Caption (required) ✓

Describe/highlight what's being shown

Error message when logged out user tries to access role-protected page

COLLAPSE

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

ℹ Details:

Ensure left panel or database name is present in each table screenshot (should contain your ucid)

#1) UserRoles table with at least one valid entry (Table should have id, user_id, role_id, is_active, created, and modified columns)

Caption (required) ✓

Describe/highlight what's being shown

screenshot of UserRoles table

#2) Roles table with at least one valid entry (Table should have id, name, description, is_active, modified, and created columns)

Caption (required) ✓

Describe/highlight what's being shown

Screenshot of Roles table

^COLLAPSE^

Task #4 - Points: 1

Text: Explain how Roles and UserRoles tables work in conjunction with the Users table

#1) UserRoles

Explanation (required) ✓

What's the purpose of the UserRoles table?

PREVIEW RESPONSE

The UserRoles table has a list of each user's roles, with the user's id from the Users table and the role id from the Roles table. It also had a column called is_active that says whether the role is active. This table acts as a bridge between the Users and Roles tables to keep the data as simple as possible.

#2) Roles

Explanation (required) ✓

How does Roles.is_active differ from UserRoles.is_active?

PREVIEW RESPONSE

UserRoles.is_active is active when a specific user actively has a role. The user can turn this role on and off. Roles.is_active is for the role in general.

^COLLAPSE^

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

https://github.com/m-sansone/mrs43-IT202-452/pull/12

URL #2

https://github.com/m-sansone/mrs43-IT202-452/pull/20

URL #3

https://github.com/m-sansone/mrs43-IT202-452/pull/27

URL #4

https://github.com/m-sansone/mrs43-IT202-452/pull/29

URL #5

https://github.com/m-sansone/mrs43-IT202-452/pull/30

URL #6

https://github.com/m-sansone/mrs43-IT202-452/pull/35

^COLLAPSE^ User Profile (2 pts.)

^COLLAPSE^

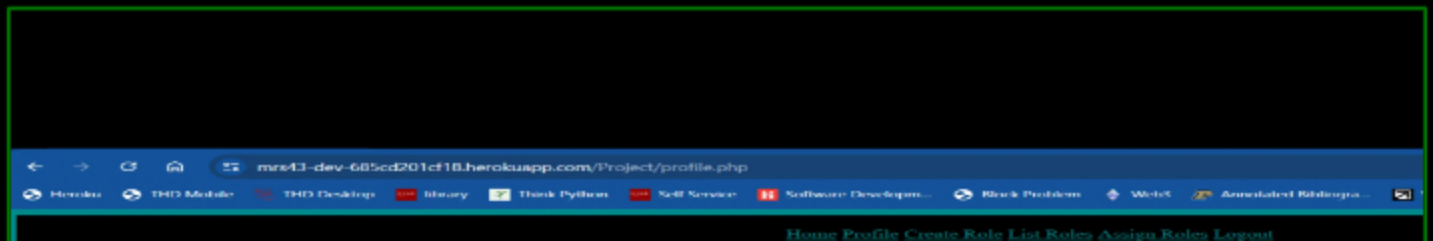
Task #1 - Points: 1

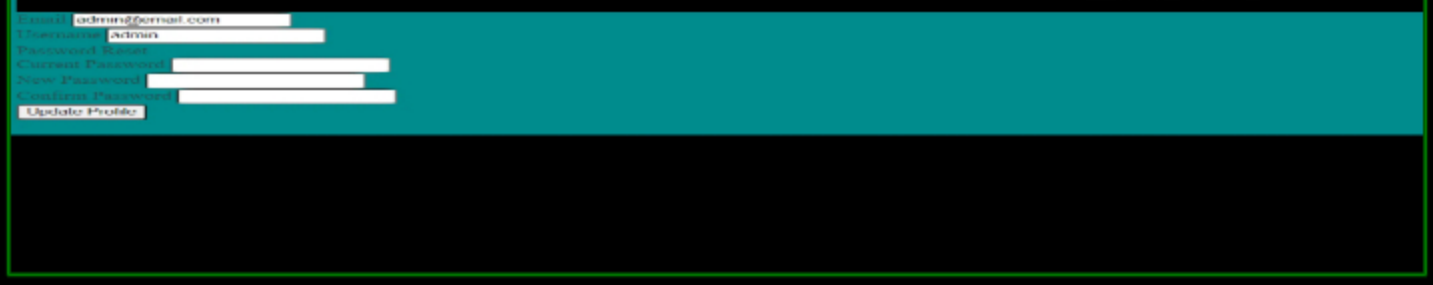
Text: View Profile Website Page

Details:

Thoughtful CSS should be applied to all parts (must differ from the "ugly" CSS given via the lessons).
The Heroku dev URL must be present in all screenshots of the site.

#1) Show the profile form correctly populated on page load (username, email) Form should have the following fields: username, email, current password, new password, confirm password (or similar)





Caption (required) ✓
Describe/highlight what's being shown
profile page from Heroku

COLLAPSE

Task #2 - Points: 1

Text: Explain the logic step-by-step of how the data is loaded and populated when the profile page is visited

Details:
Don't just show code, translate things to plain English in concise steps.

May be worthwhile using a list.

Response:

The page first checks if the user is logged in. If so, the user's email and username are retrieved and displayed in the HTML form. If fields are edited and submitted on the form, then the code tries to update the table accordingly. If it works, a success message is displayed. If not, an error message is shown when an exception is caught. The code then tries to retreieve the user's information and resume the user's session.

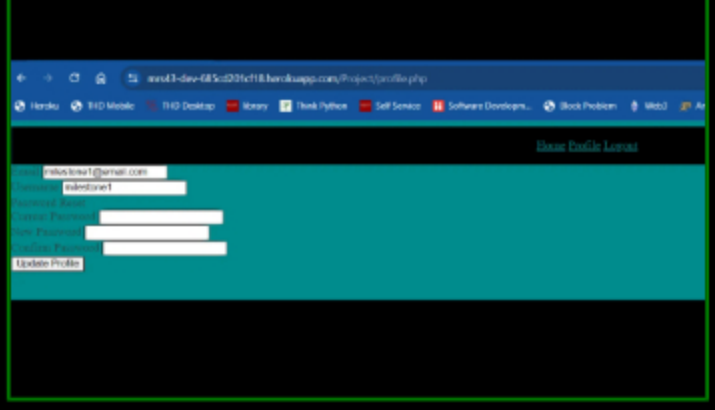
COLLAPSE

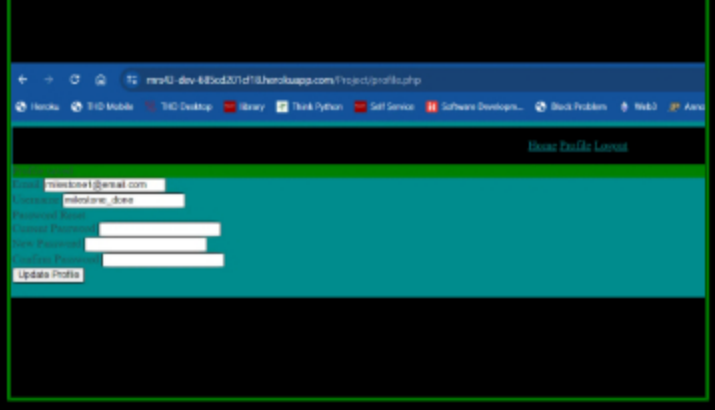
Task #3 - Points: 1

Text: Edit Profile Website Page

Details:
Thoughtful CSS should be applied to all parts (must differ from the "ugly" CSS given via the lessons).
The Heroku dev URL must be present in all screenshots of the site.

#1) (Two Screenshots) Demonstrate with before and after of a username change (including success message)





Caption (required) ✓

Describe/highlight what's being shown

before and after username change

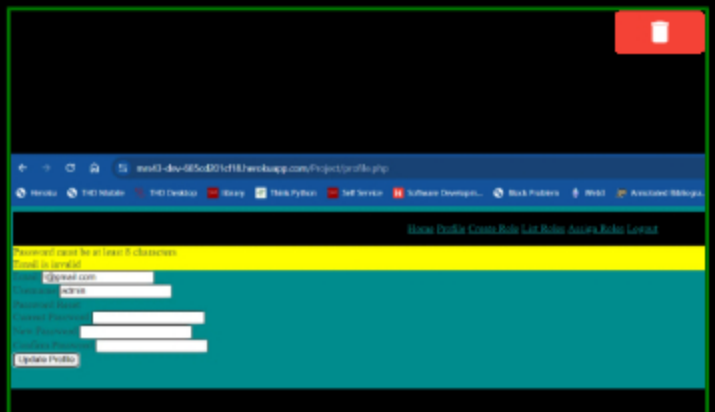
#2) Demonstrate the success message of updating password

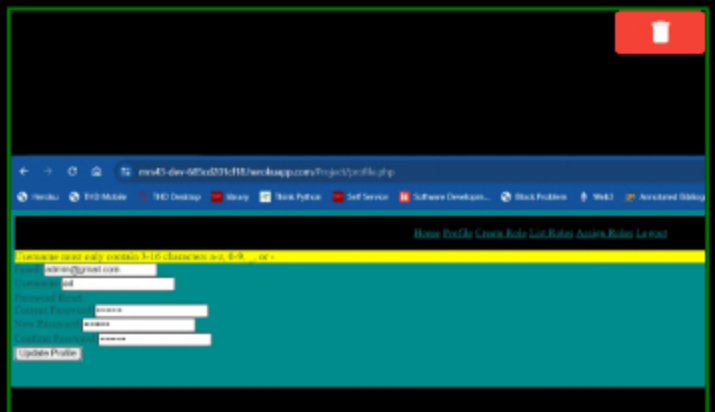
Caption (required) ✓

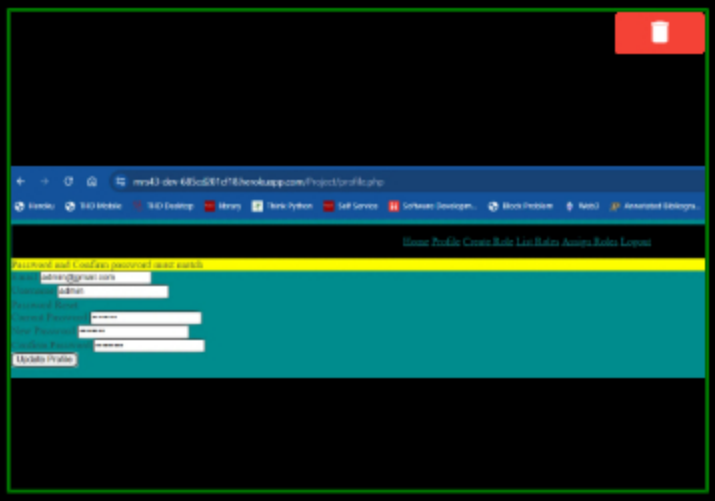
Describe/highlight what's being shown

success message when password is updated

#3) Demonstrate all JavaScript user-friendly validation messages [can be combined] (email format, username format, password format, new password matching confirm password)





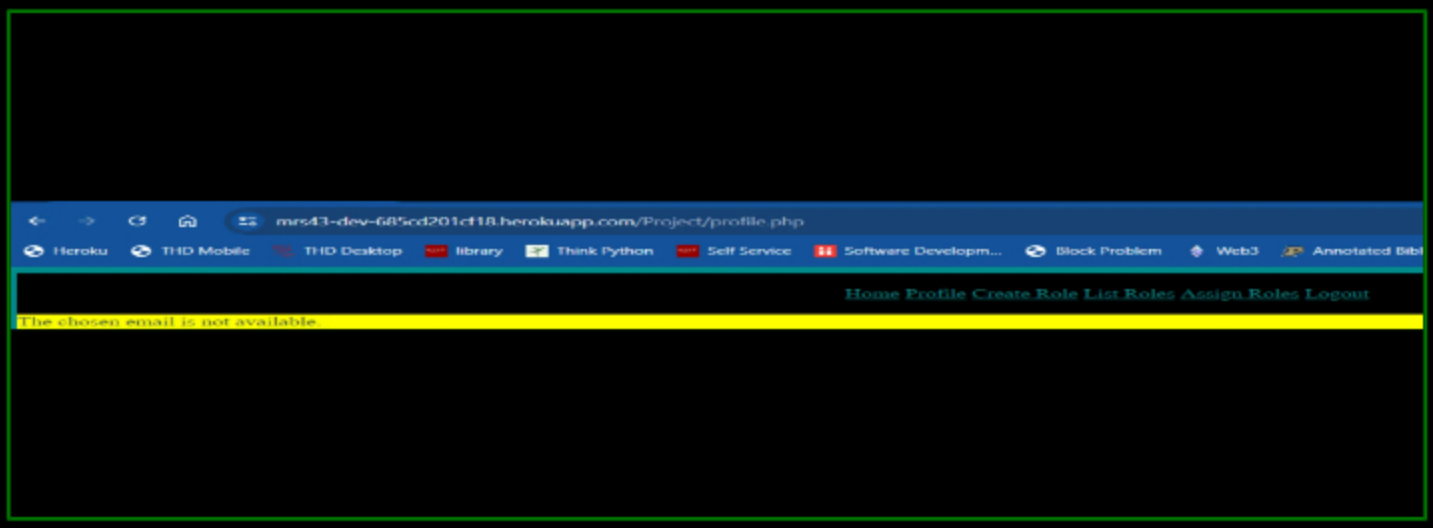


Caption (required) ✓

Describe/highlight what's being shown

screenshots from Heroku with JS validation messages

#4) Demonstrate PHP user-friendly validation message (desired email is already in use)

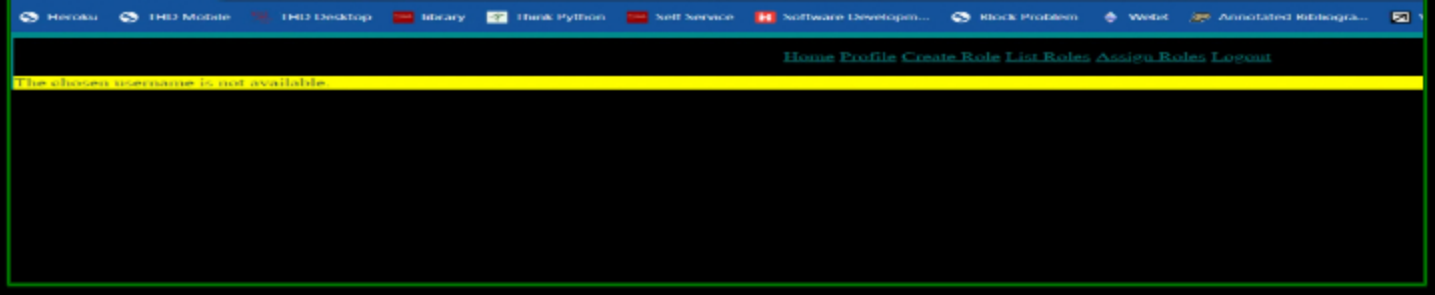


Caption (required) ✓

Describe/highlight what's being shown

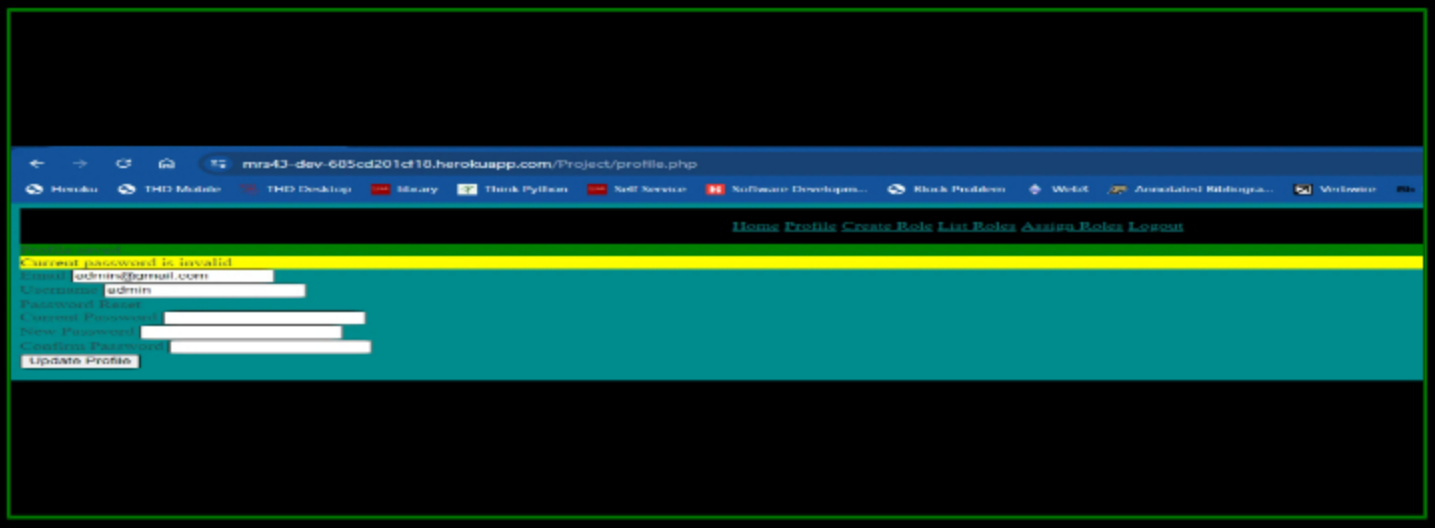
error message for taken email

#5) Demonstrate PHP user-friendly validation message (Desired username is already in use)



Caption (required) ✓
Describe/highlight what's being shown
error message of taken username

#6) Demonstrate PHP user-friendly validation message (Current password doesn't match what's in the DB)



Caption (required) ✓
Describe/highlight what's being shown
error message for invalid password

COLLAPSE

Task #4 - Points: 1

Text: Explain the logic step-by-step of how the data is checked and saved for the following scenarios

Details:
Don't just show code, translate things to plain English

#1) Updating Username/Email

Explanation (required) ✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

The user inputs a requested email/username and submits the form. On submission, the data is checked using client-side validation (JavaScript). If the data is valid, the form is submitted and the user's email, username and id are retrieved. The code will try to update the user's information using the retreived information (which is stored as parameters). If it works, a success message is shown. If not, an error will be caught and an error message will be shown. Fresh data will be retrieved from the Users table and fetches the user record. If the user exists, the session will be updated with the user's input. If they do not, an error message will be shown.

#2) Updating password

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The current, new, and confirm passwords are retrieved and saved to variables. If they all exist and the new and confirm passwords match, the code will see if the current password is correct. If the new and confirm passwords do not match, an error message will be shown. If the current password is incorrect, an error message will be shown. The Users table will then be updated with the new password based on the user id and with a hashed password. A success message will be shown.

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

https://github.com/m-sansone/mrs43-IT202-452/pull/29

URL #2

https://github.com/m-sansone/mrs43-IT202-452/pull/30

URL #3

https://github.com/m-sansone/mrs43-IT202-452/pull/52

Misc (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of wakatime

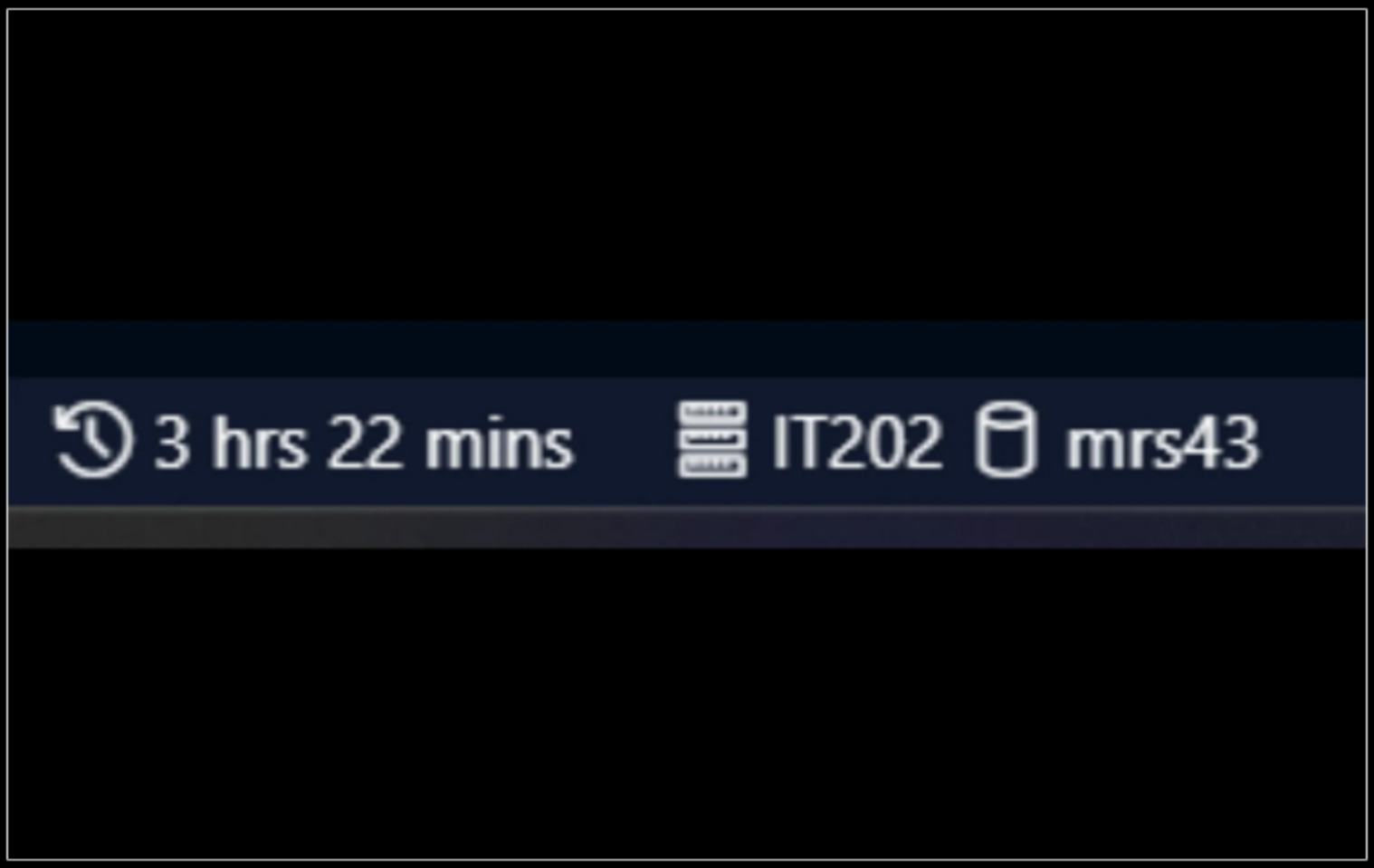
Details:

Note: The duration of time isn't directly related to the grade, the goal is to just make sure time is being tracked

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Screenshot of wakatime in VSCode

COLLAPSE

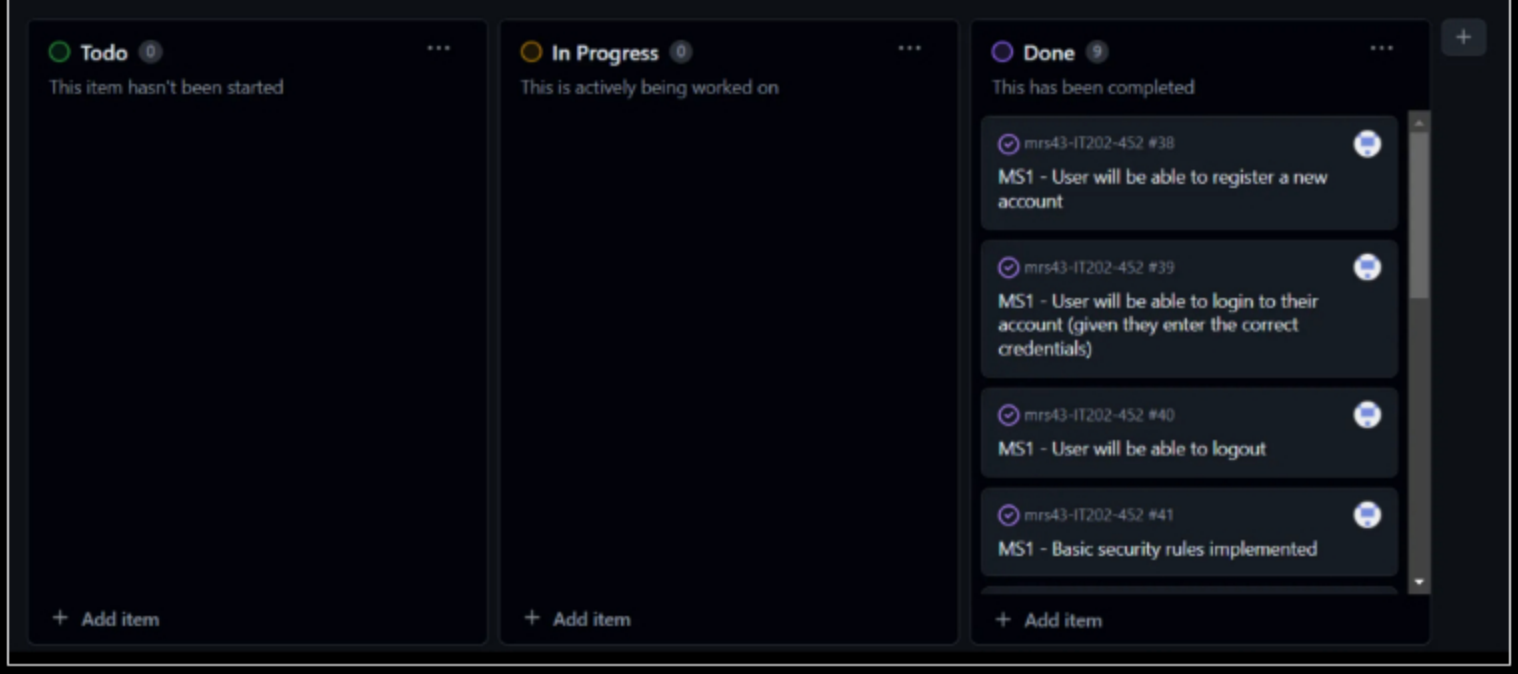
Task #2 - Points: 1

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

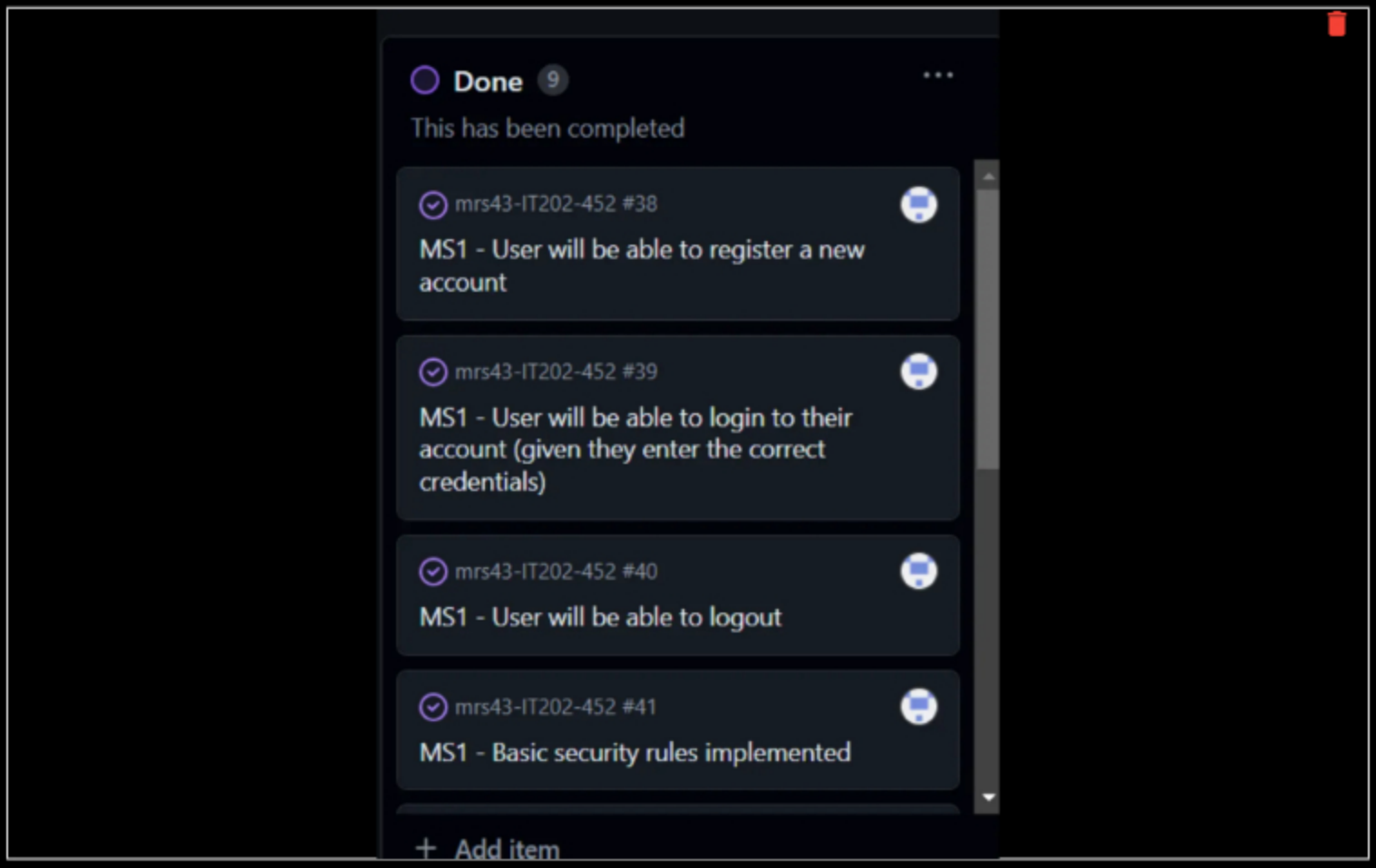
Task Screenshots:

Gallery Style: Large View

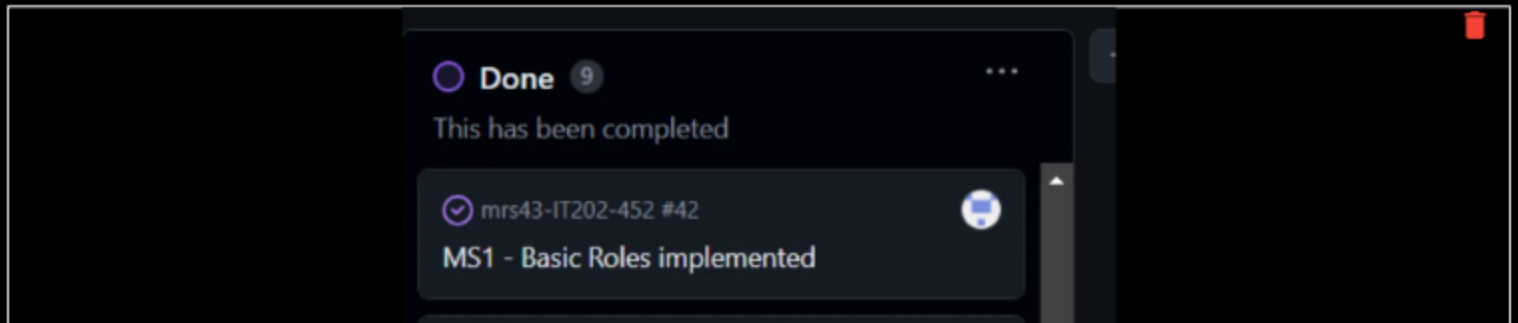
Small Medium Large

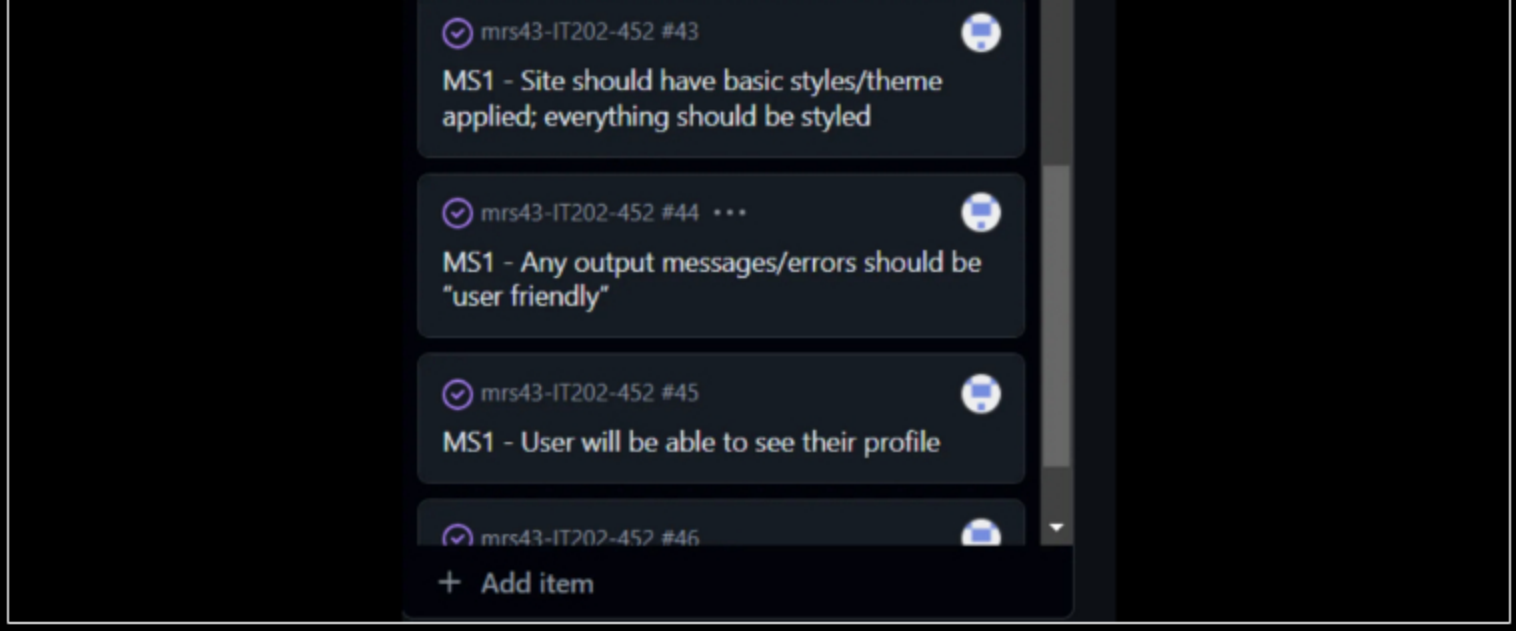


Screenshot of project board from GitHub with all tasks in Done column

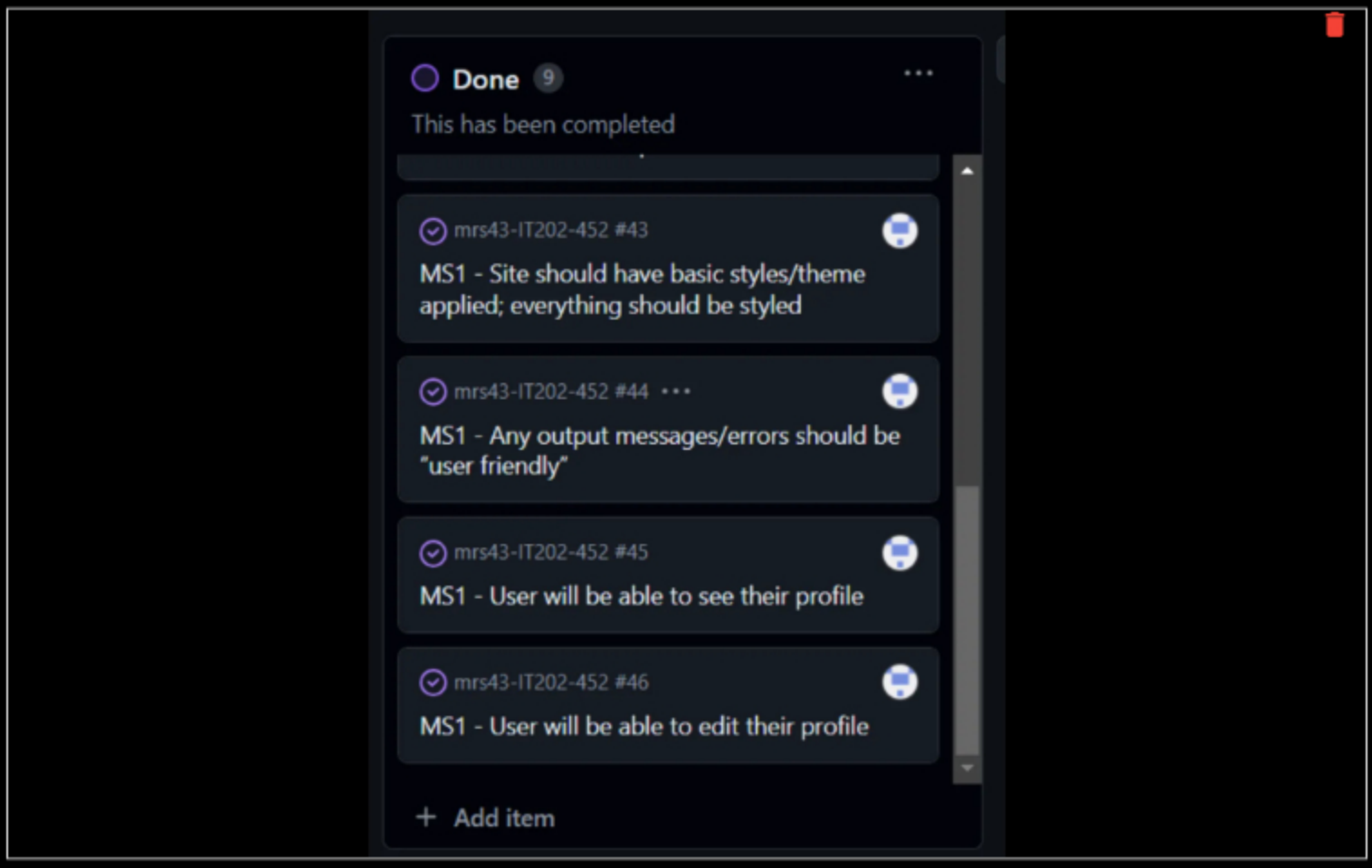


screen shot 1/3 of tasks in done column





screen shot 2/3 of tasks in done column



screen shot 3/3 of tasks in done column

COLLAPSE

Task #3 - Points: 1

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

URL #1

https://github.com/users/m-sansone/projects/2

End of Assignment