Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/lv238

IT202-008-S2024 - [IT202] Milestone 1 2024

Submissions:

Submission Selection

1 Submission [active] 4/22/2024 1:55:48 PM

Instructions

COLLAPSE

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
- 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, Logout, and Profile (include "[Client]" in the output messages to differentiate between server-side validations)

Instructions:

Make sure you're in Milestone1 with the latest changes pulled
Ensure Milestone1 has been deployed to heroku dev
Gather the requested evidence and fill in the explanations per each prompt
Save the submission and generate the output PDF
Put the output PDF into your local repository folder
add/commit/push it to GitHub
Merge Milestone1 into dev
Locally checkout dev and pull the changes
Create and merge a pull request from dev to prod to deploy Milestone1 to prod
Upload this output PDF to Canvas

Branch name: Milestone1

Tasks: 26 Points: 10.00

User Registration (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of form on website page

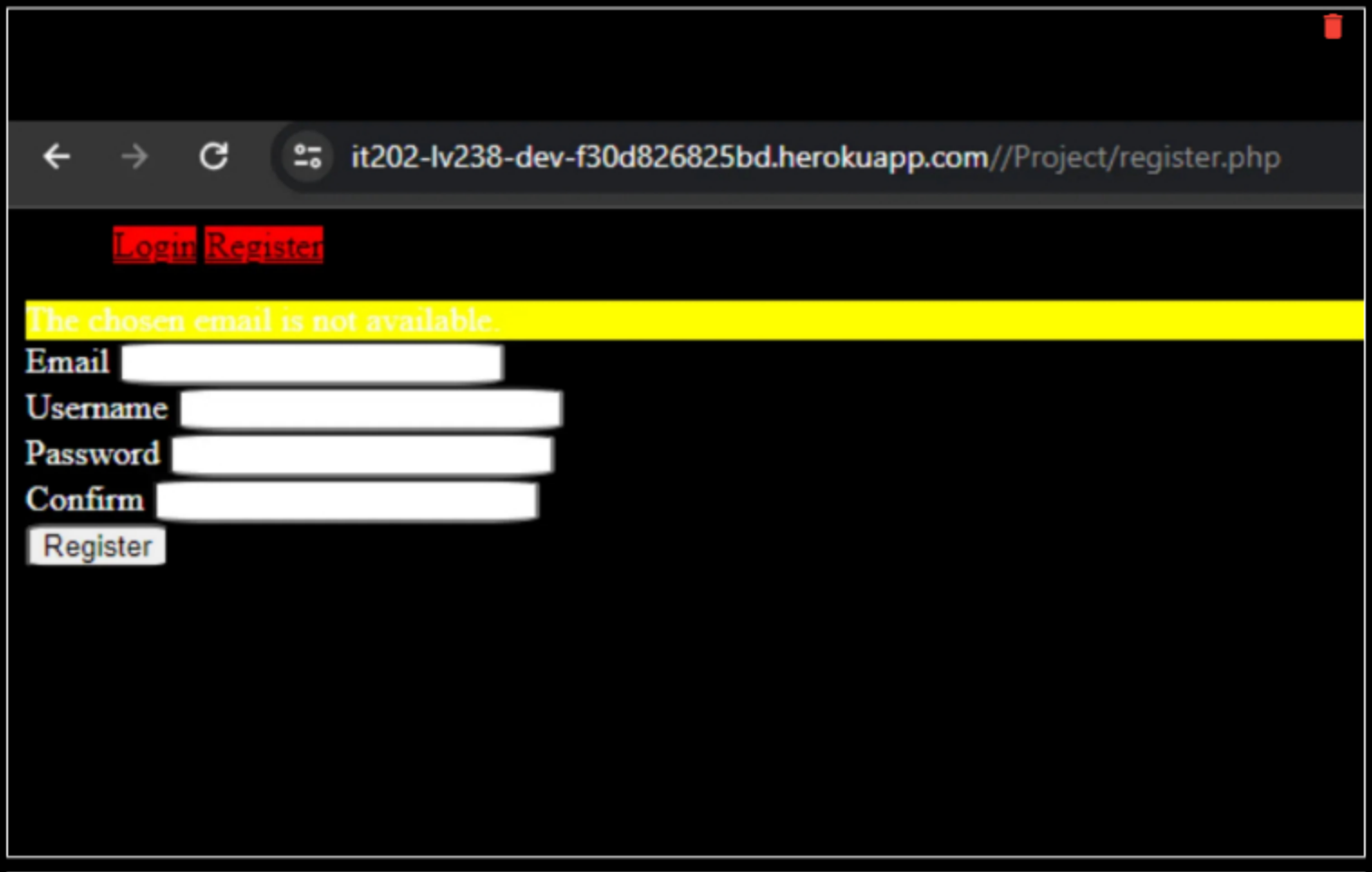
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	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
#4	1	Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)
#5	1	Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)
#6	1	Demonstrate user-friendly message of new account being created

Task Screenshots:

Gallery Style: Large View

Small Medium Large



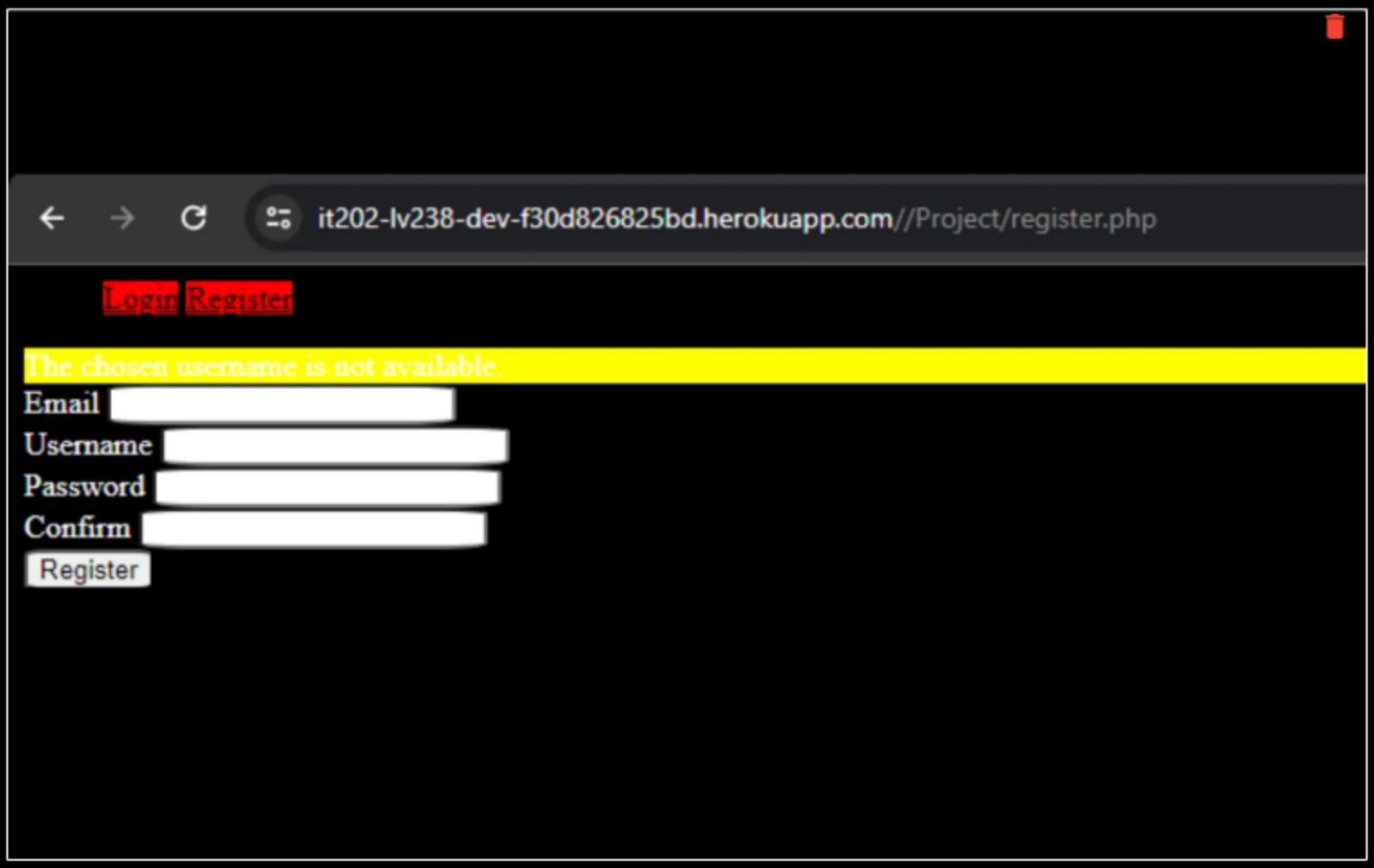
emial not available

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

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



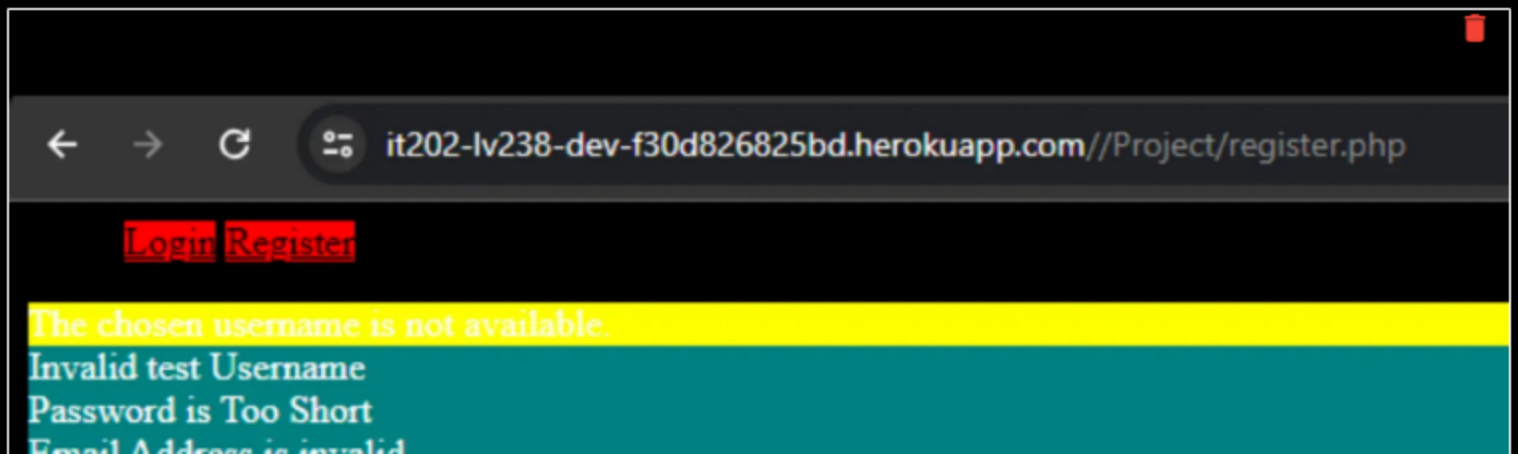
user not available

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

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



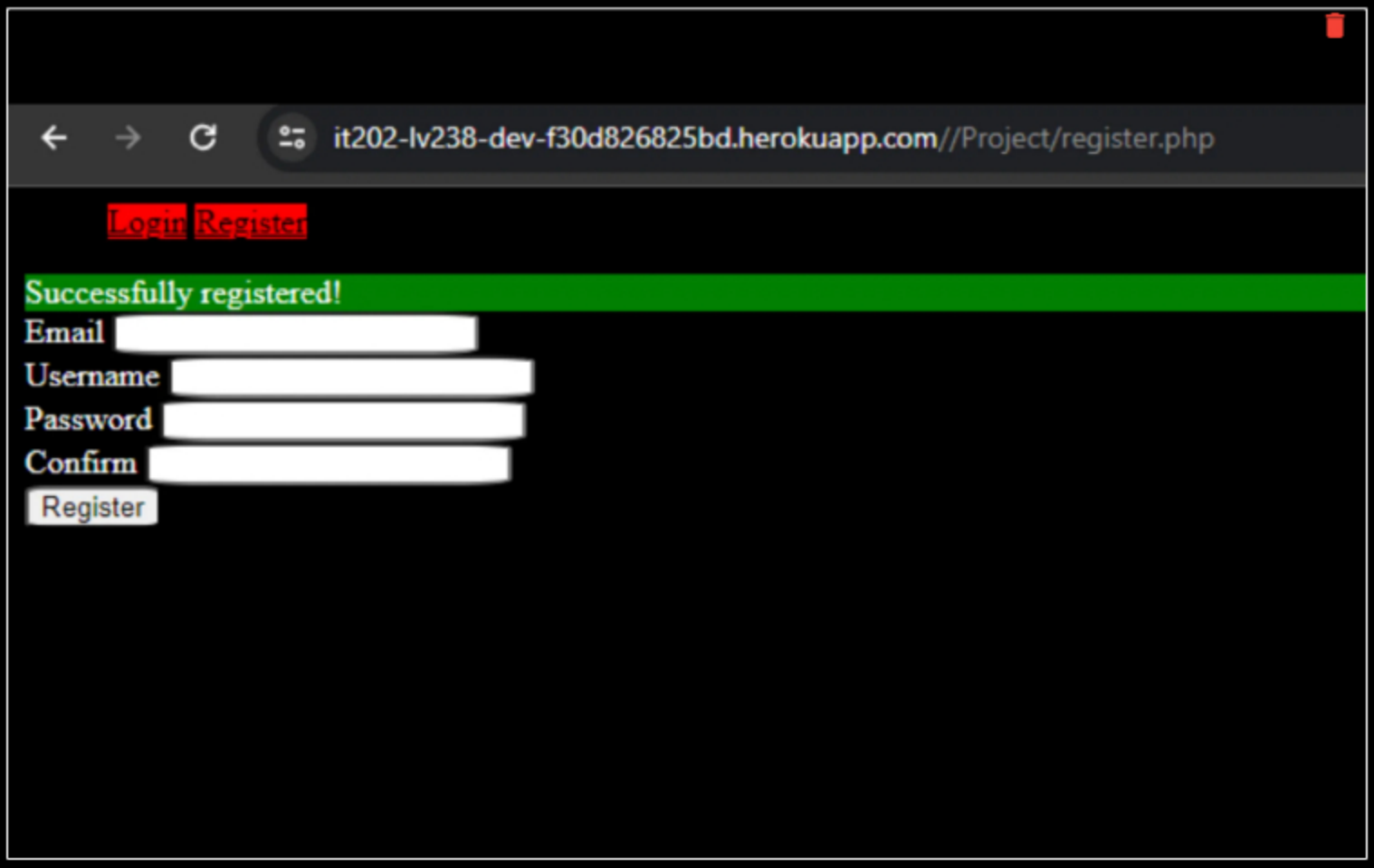
js validation is in blue

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

#3 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



successful register

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

#6 Demonstrate user-friendly message of new account being created

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the form code

Details:
Should have appropriate input types for the fiel

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```php
<?php
require(__DIR__ . "/../../partials/nav.php");
reset_session();
?>
<form onsubmit="return validate(this)" method="POST">
    <div>
        <label for="email">Email</label>
        <input type="email" id = "email"name="email" required />
    </div>
    <div>
        <label for="username">Username</label>
        <input type="text" id ="username"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" id = "confirm" name="confirm" required minlength="8" />
    </div>
    <input type="submit" value="Register" />
</form>
<script>
                                                    //lv238 4/22/24
```

form, UCID at bottom

COLLAPSE

Task #3 - Points: 1

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

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the JavaScript validations (include any extra files related)
#2	1	Show the PHP validations (include any lib content)
#3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
function validate(form) {
    let errorState = true;
    let jemail = form.email.value;
    let jpassword = form.password.value;
    let jusername = form.username.value;
    let jconfirm = form.confirm.value;
    let validRegex = new RegExp(/^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-
    let valReg2 = new RegExp(/^([a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,})$/);
    let userReg = new RegExp(/^[a-z0-9_-]{3,16}$/);

    if (!/^[a-z0-9_-]{3,16}$/.test(jusername))
    {
        flash("Invalid test Username", "info");
        errorState = false;                          //lv238 4/22/24
    }
    if (jpassword.length = 0)
    {
        flash("Password Cannot Be Empty", "info");
        errorState = false;
    }
    if(jpassword.length < 8)
    {       You, 13 hours ago • done with milestone 1 i think
        flash("Password is Too Short", "info");
        errorState = false;
    }
    if(!valReg2.test(jemail))
    {
        flash("Email Address is invalid", "info");
        errorState = false;
    }
```

part 1 of the js code used to validate. Initiates the regex value and pull all of the form info. some usage of it

Checklist Items (2)

#1 Show the JavaScript validations (include any extra files related)

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

```
if(jconfirm.length === 0)
{
    flash("Confirm Password Field Should Not Be Empty", "info");
    errorState = false;
}                                                    //LV238 4/22/24
```

```
    if(jpassword!=jconfirm)
    {
        flash("Passwords Must Match", "info");
        errorState = false;
    }



    //TODO update clientside validation to check if it should
    //valid email or username
    return errorState;
} <- #46-86 /^
```

the rest of the usage of the validation. uses regexp.test() to see if it matches, the others are simple js checks on certain properties.

Checklist Items (2)

#1 Show the JavaScript validations (include any extra files related)

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

```
if (isset($_POST["email"]) && isset($_POST["password"]) && isset($_POST["confirm"]) && isset($_POST["username"])) {
    $email = se($_POST, "email", "", false);
    $username = se($_POST, "username", "", false);
    $password = se($_POST, "password", "", false);
    $confirm = se($_POST, "confirm", "", false);

    //TODO 3
    $hasError = false;
    if (empty($email))
    {
        flash("Email must not be empty", "danger");
        $hasError = true;
    }
    //hoping change appears in git                                  lv238 4/22/24      You, 1 second ago • Uncomm
    //sanitize
    $email = sanitize_email($email);
    //validate
    if (!is_valid_email($email))
    {
        flash("Invalid email address", "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))
    {
        flash("password must not be empty", "danger");
        $hasError = true;
    }
```

php val part 1

Checklist Items (2)

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

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

part 2 of php val

Checklist Items (2)

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

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

COLLAPSE

Task #4 - Points: 1

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

Checklist

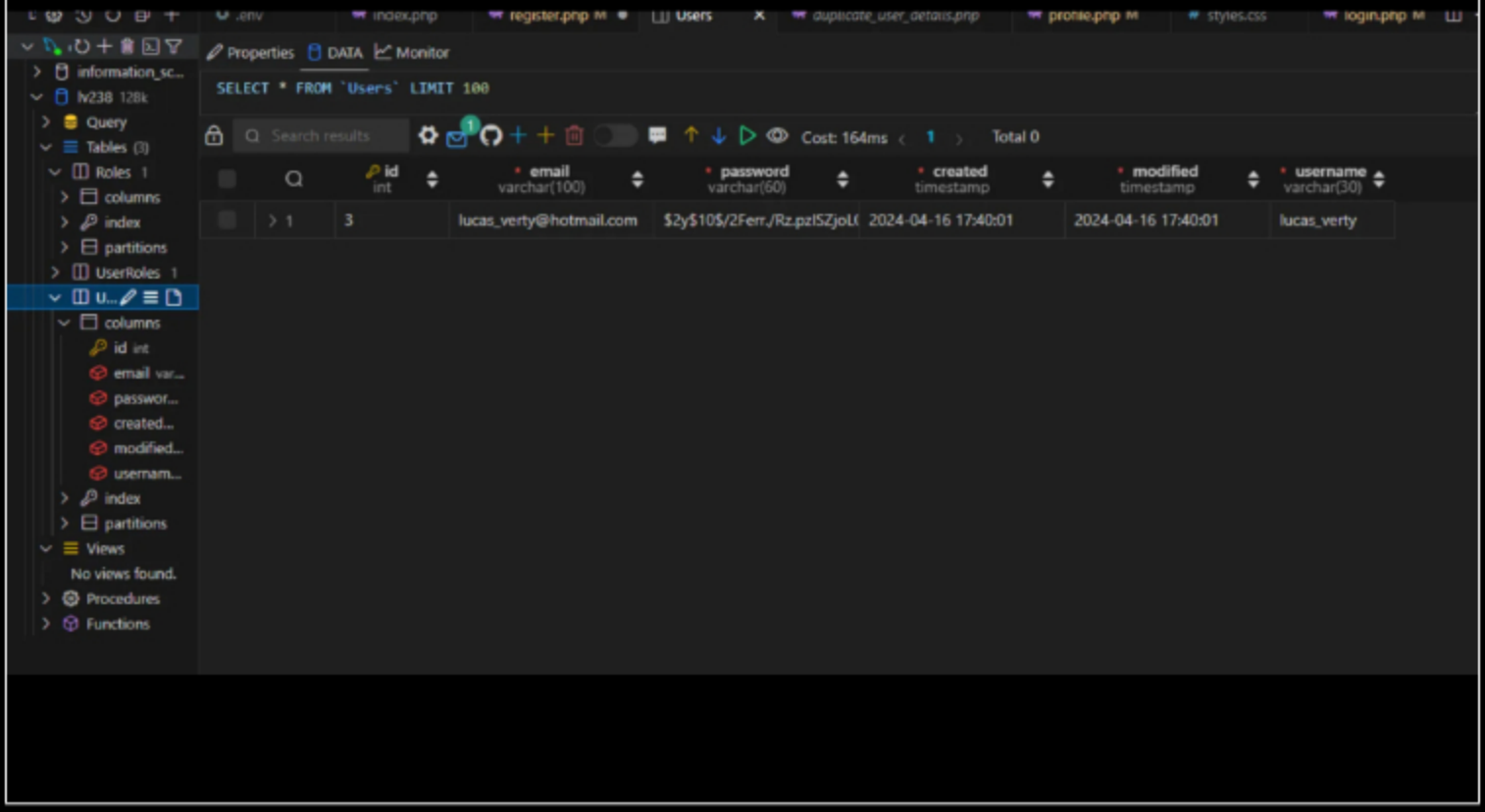
*The checkboxes are for your own tracking

#	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)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Users table (i created and deleted 2 users)

Checklist Items (3)

#1 Password should be hashed

#2 Should have email, password, username (unique), created, modified, and id fields

#3 Ensure left panel or database name is present (should contain your ucid)

^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

Response:

the page will load into a form, the form will ask for an Email, an Username, a Password, and a confirmation of said Password.
>it will first has the password, get the database assigned to us, then insert into the already made users table that information

>if table has one of these values for a UNIQUE value, it'll give an error and give an appropriate error message for it
>otherwise it will store that information in the database

COLLAPSE

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/LucasValMelo/lv238-IT202-008/pull/25

User Login (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of form on website page

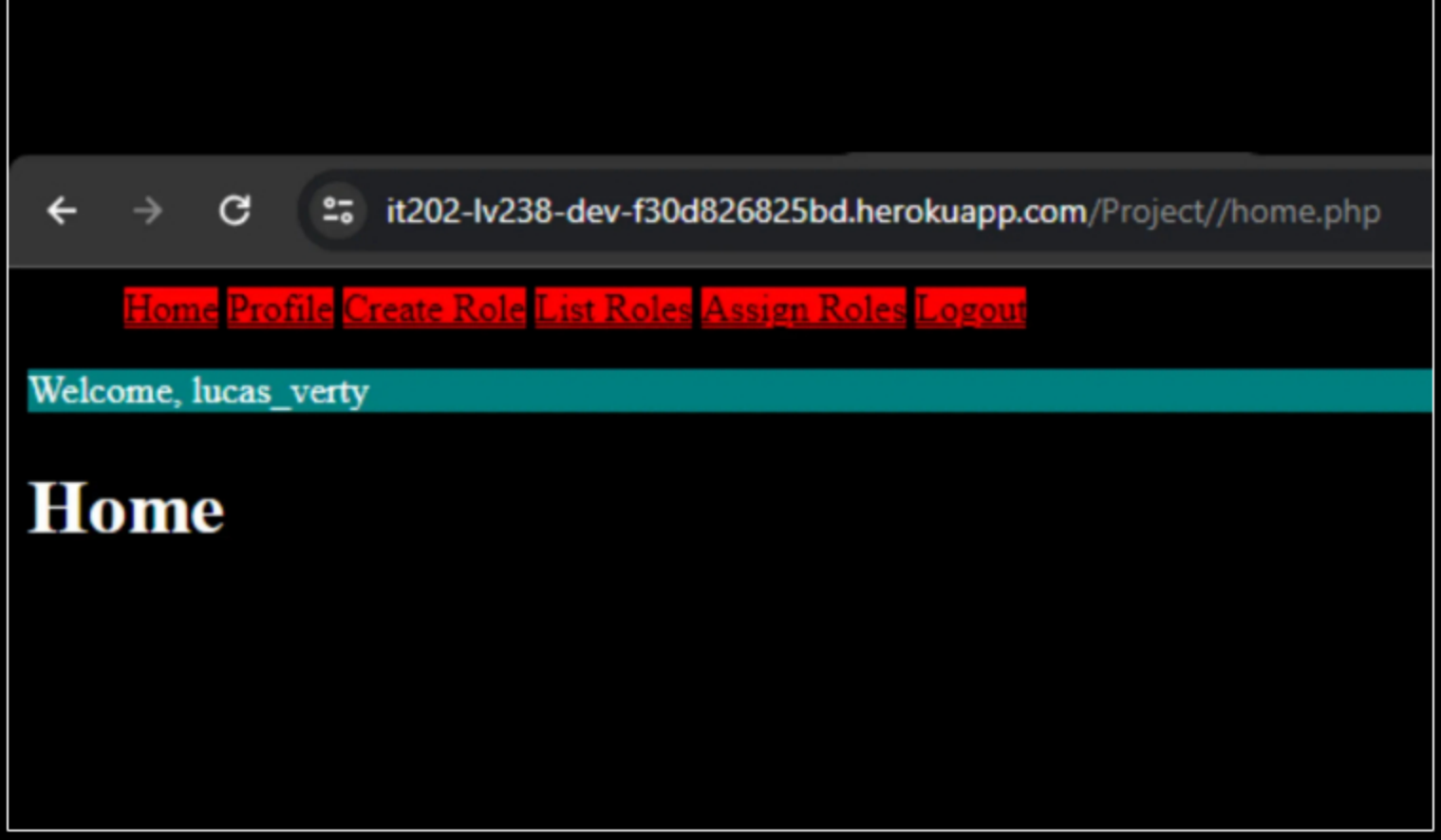
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Include correct data where username is used to login
#4	1	Include correct data where email is used to login
#5	1	Show success login message
#6	1	Demonstrate JavaScript validation for each field [can be combined] (username format, email format, password format, and each field being required)
#7	1	Demonstrate user-friendly message of when an account doesn't exist
#8	1	Demonstrate user-friendly message of when password doesn't match what's in the DB
#9	1	Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)
#10	1	Demonstrate session data being set (captured from server logs)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



login successful

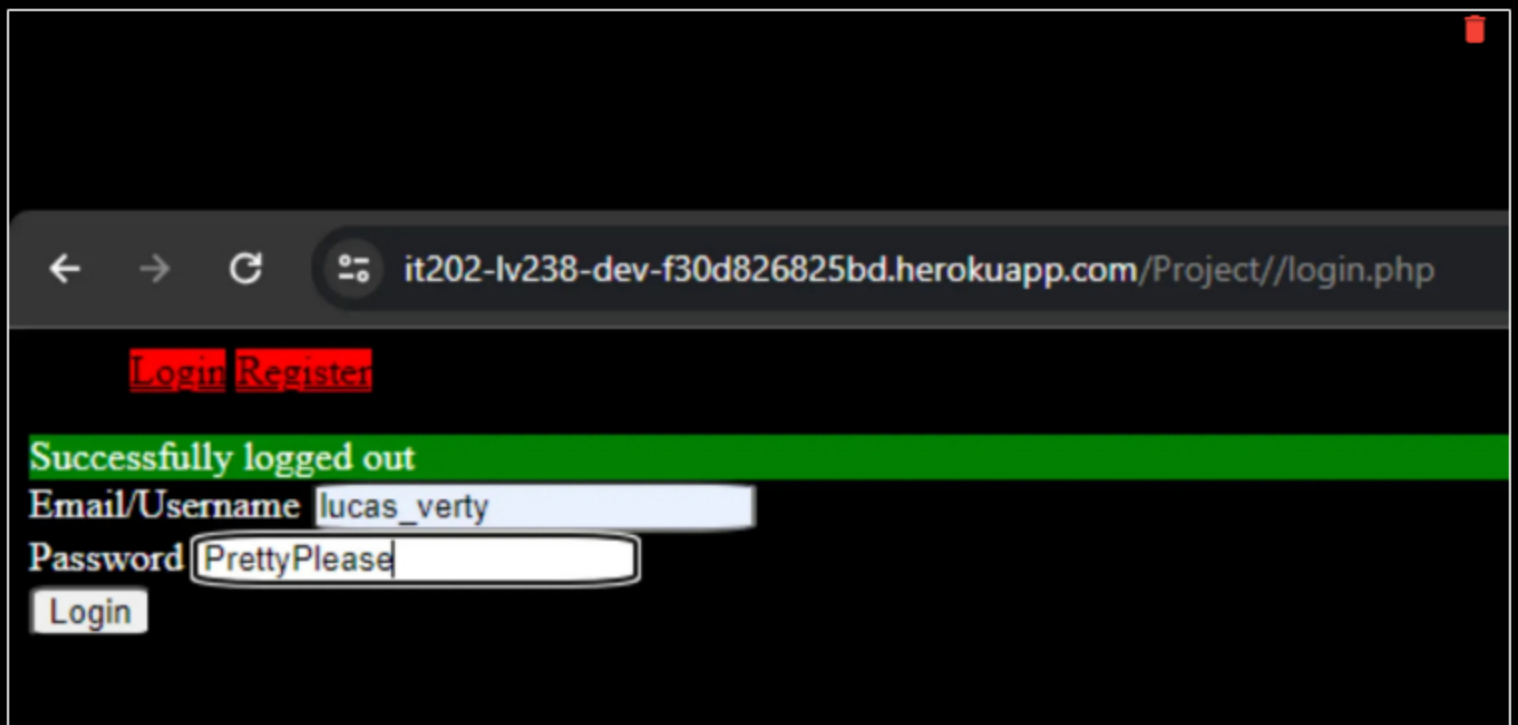
Checklist Items (4)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

#5 Show success login message

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



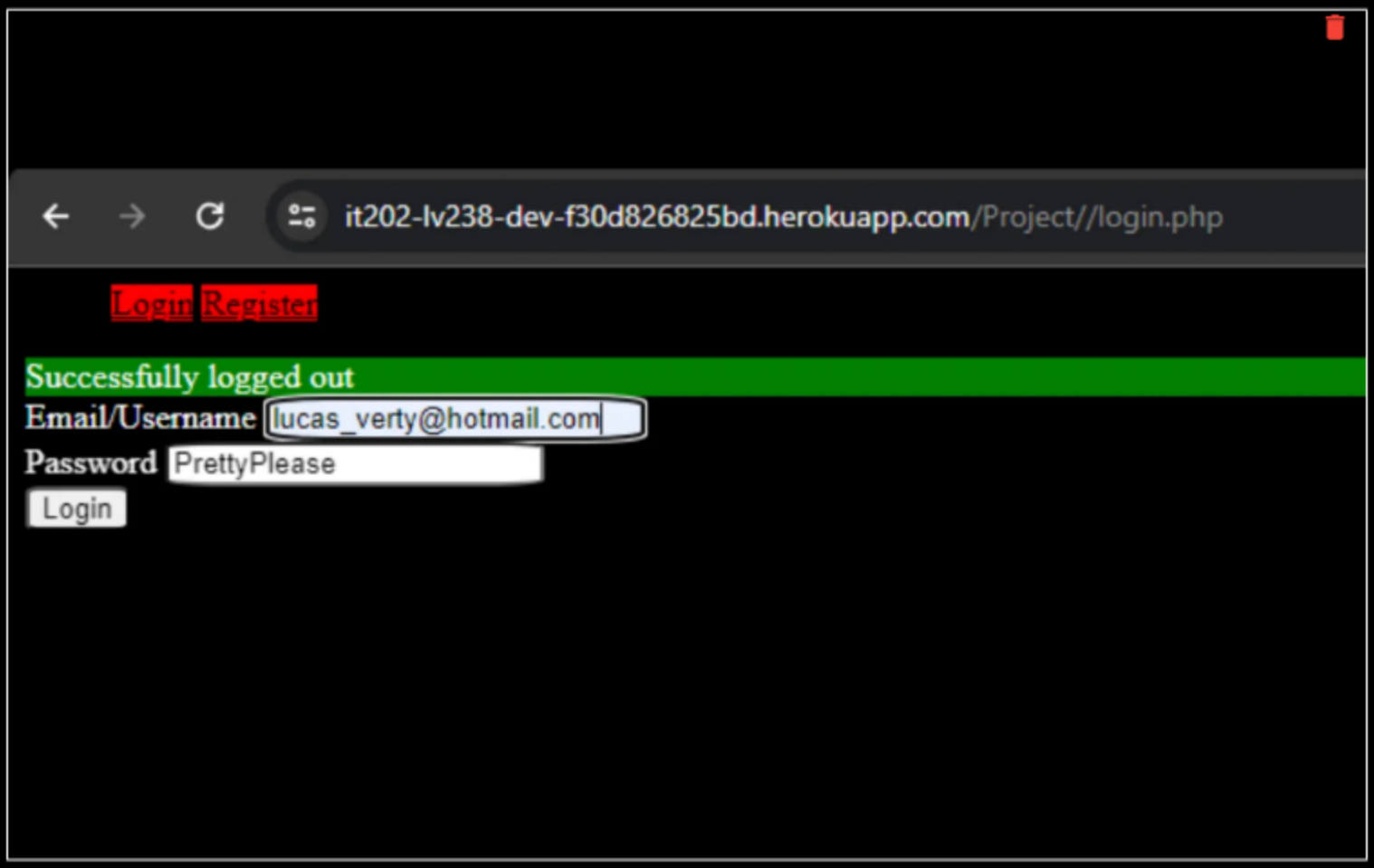
correct username data

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

#3 Include correct data where username is used to login



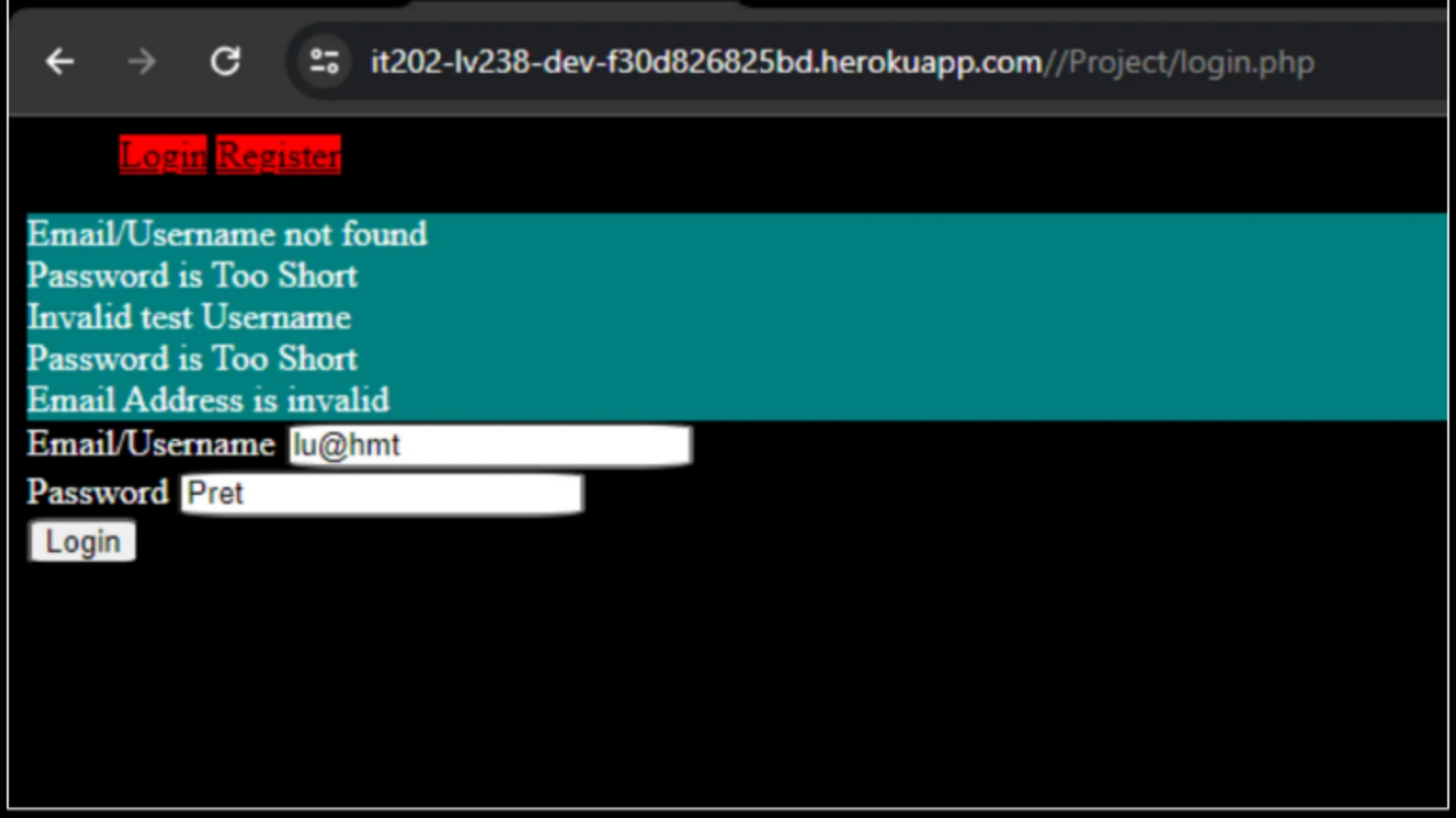
email

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

#4 Include correct data where email is used to login



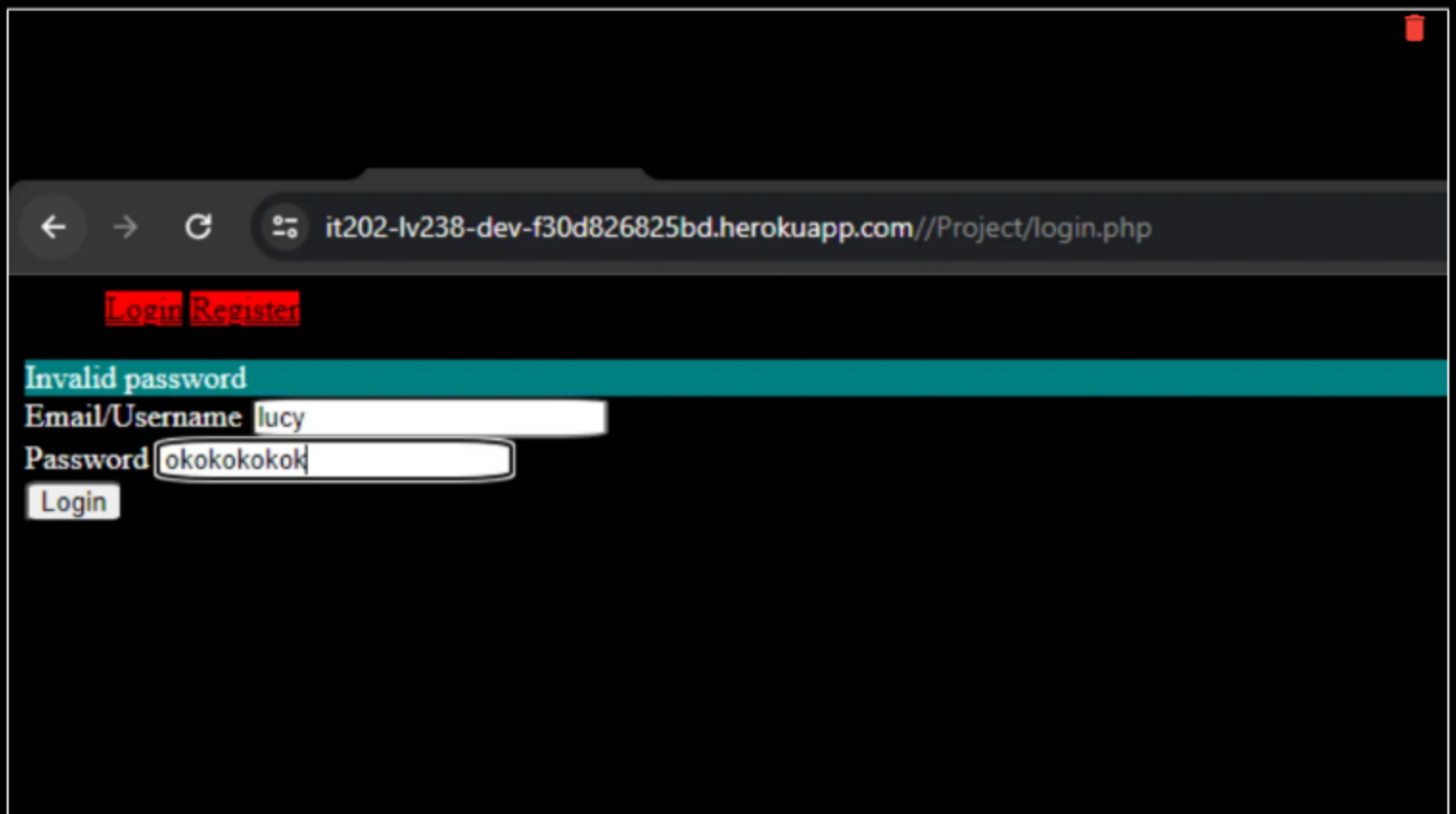
Login page

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

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



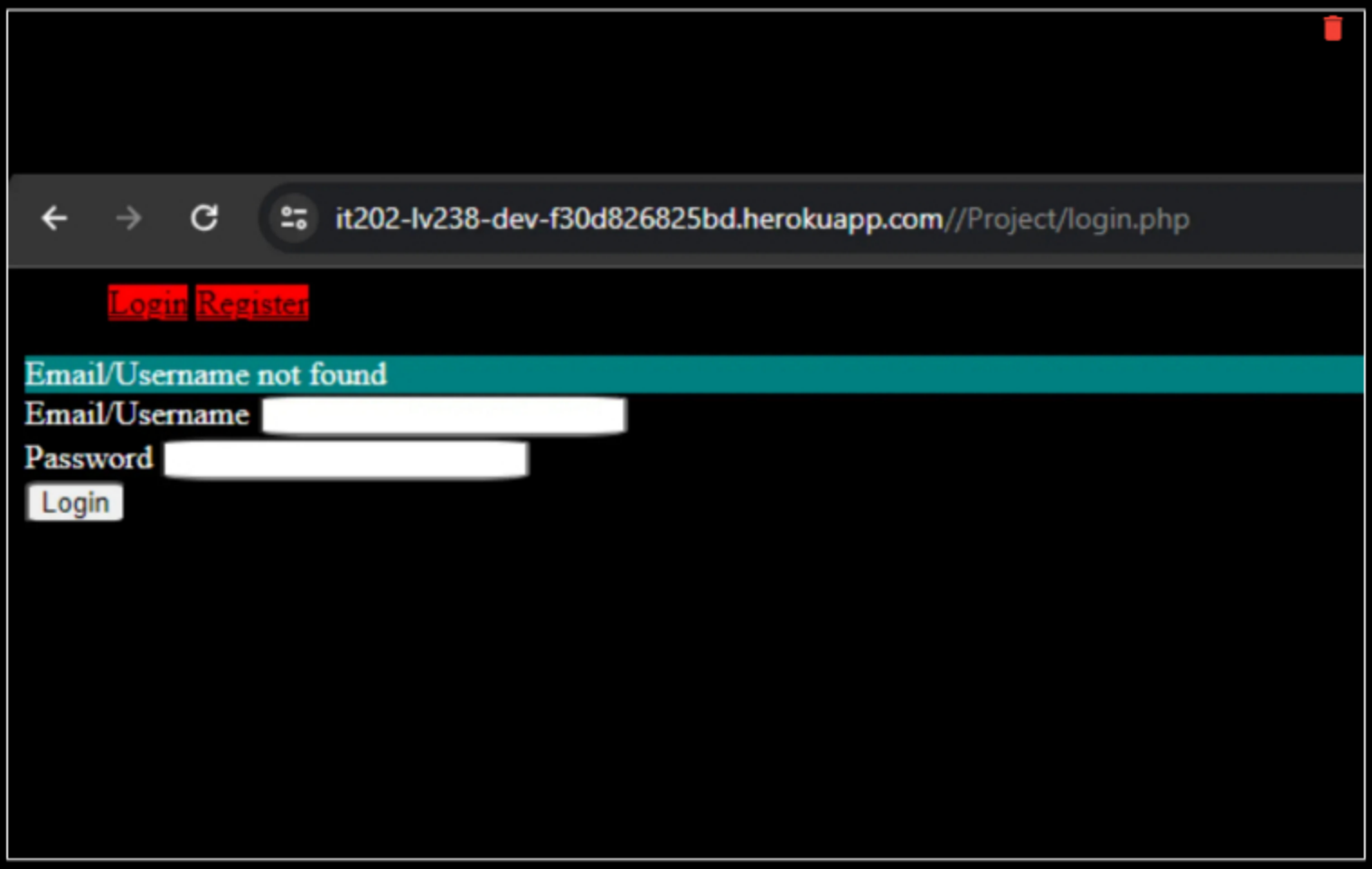
password doesnt match

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

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



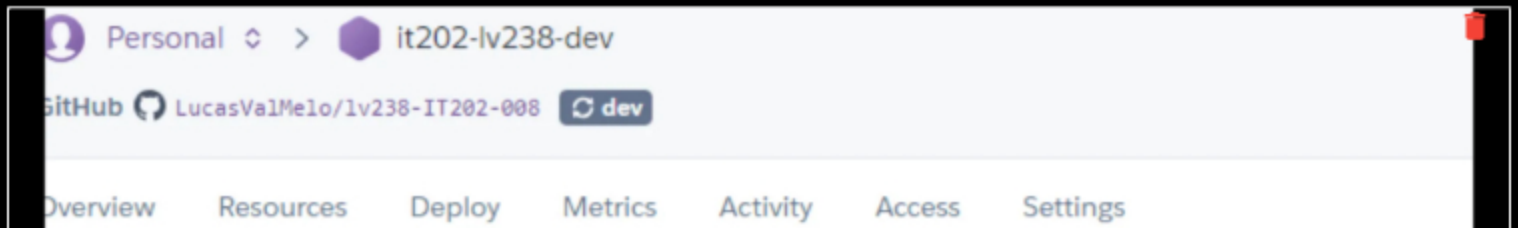
account doesn't exist'

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

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



heroku ss of data being set

Checklist Items (2)

#1 Heroku dev url should be present in the address bar

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

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the form code

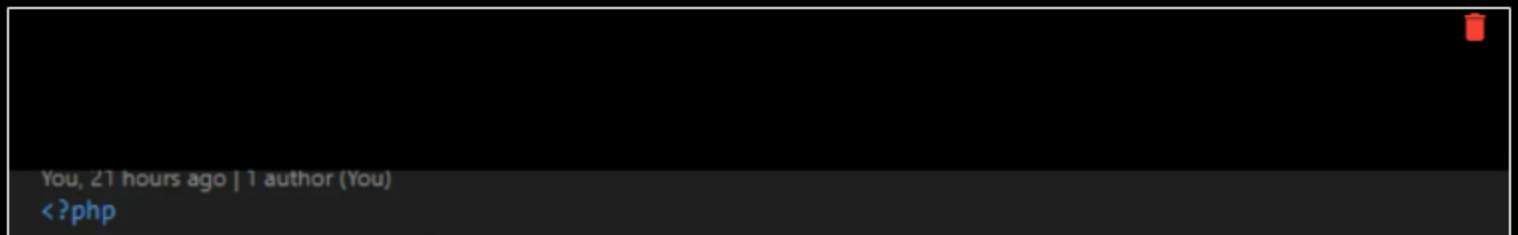
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Should have proper input types for the fields (note in the caption if you're using two fields for Username/Email or a combined field)
#2	1	Show JavaScript validations (include any extra files related)
#3	1	Show PHP validations (include any lib content)
#4	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```php
require(__DIR__ . "/../../partials/nav.php");
?>
<form onsubmit="return validate(this)" method="POST">
    <div>
        <label for="email">Email/Username</label>
        <input type="text" id = "email" 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>
<script>
    //                                                  lv238 4/23/24      You, 21 hours ago
```

form code

Checklist Items (2)

#1 Should have proper input types for the fields (note in the caption if you're using two fields for Username/Email or a combined field)

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

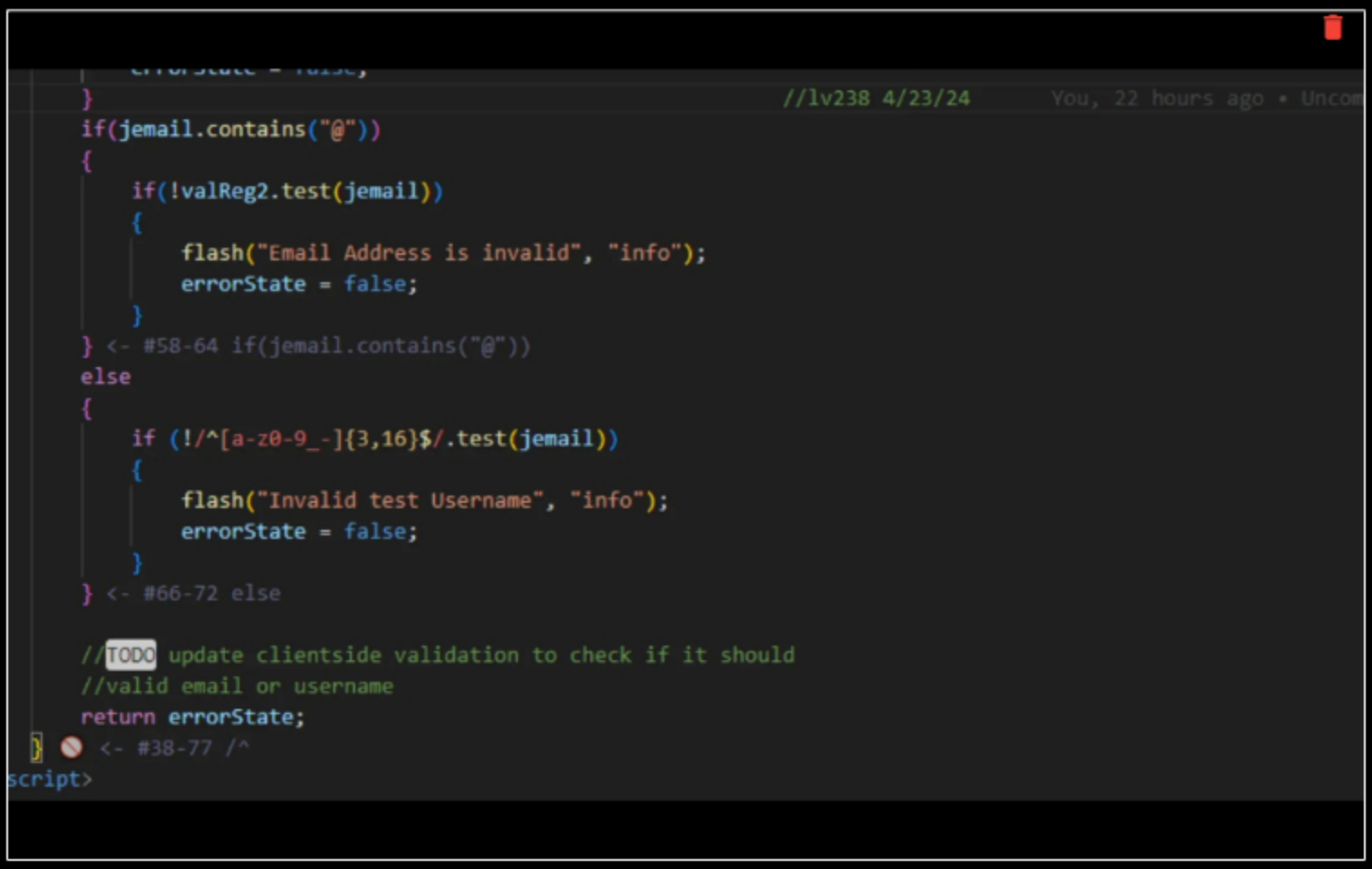
```javascript
<script>
    function validate(form) {
        let errorState = true;
        let jemail = form.email.value;
        let jpassword = form.password.value;
        let jusername = form.username;
        let validRegex = new RegExp(/^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)*$/);
        let valReg2 = new RegExp(/^([a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,})$/);
        let userReg = new RegExp(/^[a-z0-9_-]{3,16}$/);

        /*if (!/^[a-z0-9_-]{3,16}$/.test(jusername))
        {
            flash("Invalid test Username", "info");
            errorState = false;
        }*/
        if (jpassword.length = 0)
        {
            flash("Password Cannot Be Empty", "info");
            errorState = false;
        }
        if(jpassword.length < 8)
        {
            flash("Password is Too Short", "info");
            errorState = false;
        }                                                   //lv238 4/23/24
```

js validation p1

Checklist Items (2)

#2 Show JavaScript validations (include any extra files related)

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



```
    }                                                    //lv238 4/23/24      You, 22 hours ago • Uncom
    if(jemail.contains("@"))
    {
        if(!valReg2.test(jemail))
        {
            flash("Email Address is invalid", "info");
            errorState = false;
        }
    } <- #58-64 if(jemail.contains("@"))
    else
    {
        if (!/^[a-z0-9_-]{3,16}$/.test(jemail))
        {
            flash("Invalid test Username", "info");
            errorState = false;
        }
    } <- #66-72 else

    //TODO update clientside validation to check if it should
    //valid email or username
    return errorState;
} <- #38-77 /^
script>
```

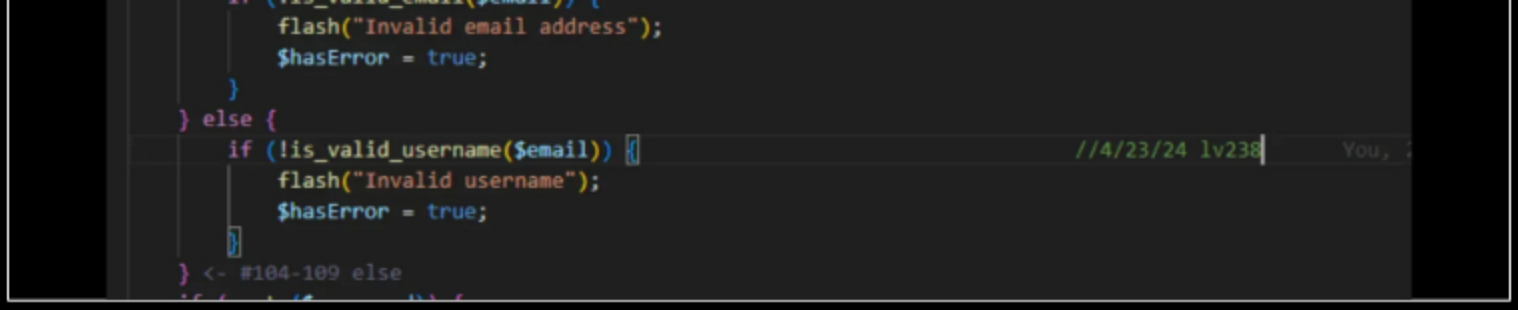
js validation p2

Checklist Items (2)

#2 Show JavaScript validations (include any extra files related)

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

```
//TODO 2: add PHP Code
if (isset($_POST["email"]) && isset($_POST["password"])) {
    $email = se($_POST, "email", "", false);
    $password = se($_POST, "password", "", false);

    //TODO 3
    $hasError = false;
    if (empty($email)) {
        flash("Email must not be empty");
        $hasError = true;
    }
    if (str_contains($email, "@")) {
        //sanitize
        //$email = filter_var($email, FILTER_SANITIZE_EMAIL);
        $email = sanitize_email($email);
        //validate
        /*if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            flash("Invalid email address");
            $hasError = true;
        }*/
        if (!is_valid_email($email)) {
```

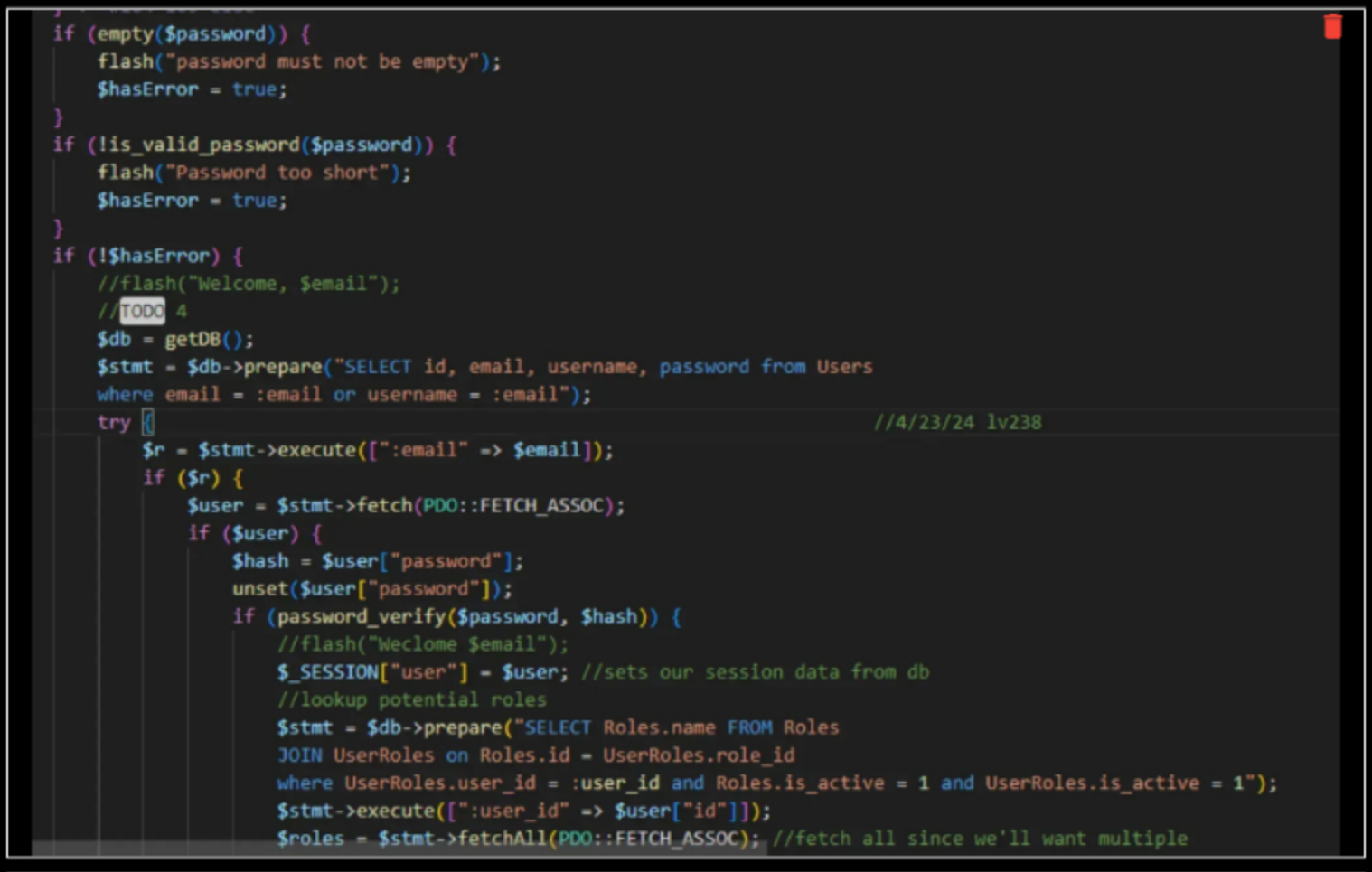


php val p1

Checklist Items (2)

#3 Show PHP validations (include any lib content)

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

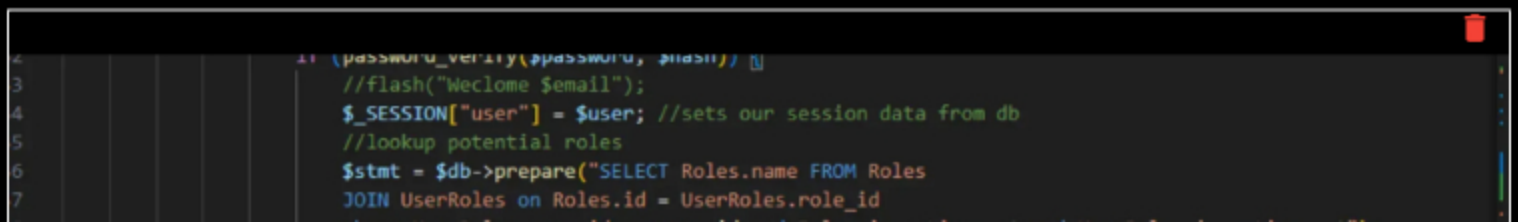


php val p2

Checklist Items (2)

#3 Show PHP validations (include any lib content)

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



```php
                    where UserRoles.user_id = :user_id and Roles.is_active = 1 and UserRoles.is_active = 1");
                    $stmt->execute([":user_id" => $user["id"]]);
                    $roles = $stmt->fetchAll(PDO::FETCH_ASSOC); //fetch all since we'll want multiple
                    //save roles or empty array
                    if ($roles) {
                        $_SESSION["user"]["roles"] = $roles; //at least 1 role
                    } else {
                        $_SESSION["user"]["roles"] = []; //no roles
                    }
                    flash("Welcome, " . get_username());                    //lv238 4.23.24
                    die(header("Location: home.php"));
                } else {
                    flash("Invalid password");
                }
            } else {
                flash("Email/Username not found");
            }
        } <- #128-154 if ($user)
    } catch (Exception $e) {
        flash("<pre>" . var_export($e, true) . "</pre>");
    }
} <- #124-158 try
} <- #122-159 $stmt = $db->prepare
?>
```

php val 3

Checklist Items (2)

#3 Show PHP validations (include any lib content)

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

```php
...
<?php

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);
}                                                   //lv238 4.23.24
function is_valid_username($username)
{
    return preg_match('/^[a-z0-9_-]{3,16}$/', $username);
}
function is_valid_password($password)
{
    return strlen($password) >= 8;
}
```

lib 1

Checklist Items (2)

Checklist Items (2)

#3 Show PHP validations (include any lib content)

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

```php
<?php

/**
 * Passing $redirect as true will auto redirect a logged out user to the $destination.
 * The destination defaults to login.php
 */
function is_logged_in($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, $destination = "logi...
function has_role($role)
{
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {                          //4/23/24 lv238
        foreach ($_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($role)
function get_username()
{
    if (is_logged_in()) { //we need to check for login first because "user" key may not exist
        return se($_SESSION["user"], "username", "", false);
```

lib2

Checklist Items (2)

#3 Show PHP validations (include any lib content)

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

```php
} <- #8-16 function is_logged_in($redirect = false, $destination = "logi...
function has_role($role)
{
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {                          //4/23/24 lv238
        foreach ($_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($role)
function get_username()
{
    if (is_logged_in()) { //we need to check for login first because "user" key may not exist
        return se($_SESSION["user"], "username", "", false);
    }
    return "";
} <- #29-34 function get_username()
function get_user_email()
{
    if (is_logged_in()) { //we need to check for login first because "user" key may not exist
```

```php
        return se($_SESSION["user"], "email", "", false);
    }
    return "";
} <- #36-41 function get_user_email()
function get_user_id()
{
    if (is_logged_in()) { //we need to check for login first because "user" key may not exist
        return se($_SESSION["user"], "id", false, false);
    }
    return false;
```

lib3

Checklist Items (2)

#3 Show PHP validations (include any lib content)

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

^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

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Don't just show code, translate things to plain English
#2	1	Explain how the session works and why/how it's used

Response:

login page has a form that shows an Email/Username and Password
>The form is filled out, when submit is pressed it loads up the database of Users
>it'll see if the data entered matches the entry in the database
>it will then set the password to a hash and unset the password, if the two equal out with the hased password, it'll set the session with the matching user, including its roles

>once the session is loaded, it loads you into the Home Page, flashing a Welcome message and role permissions (if they have a role)

^COLLAPSE^

Task #4 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/LucasValMelo/lv238-IT202-008/pull/30

User Logout (1 pt.)
COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Capture the following screenshots

Checklist *The checkboxes are for your own tracking

#	Points	Details
☐ #1	1	Screenshot of the navigation when logged in (site)
☐ #2	1	Screenshot of the redirect to login with the user-friendly logged-out message (site)
☐ #3	1	Screenshot of the logout-related code showing the session is destroyed (code). Ensure ucid/date comment is present.

Task Screenshots:

Gallery Style: Large View

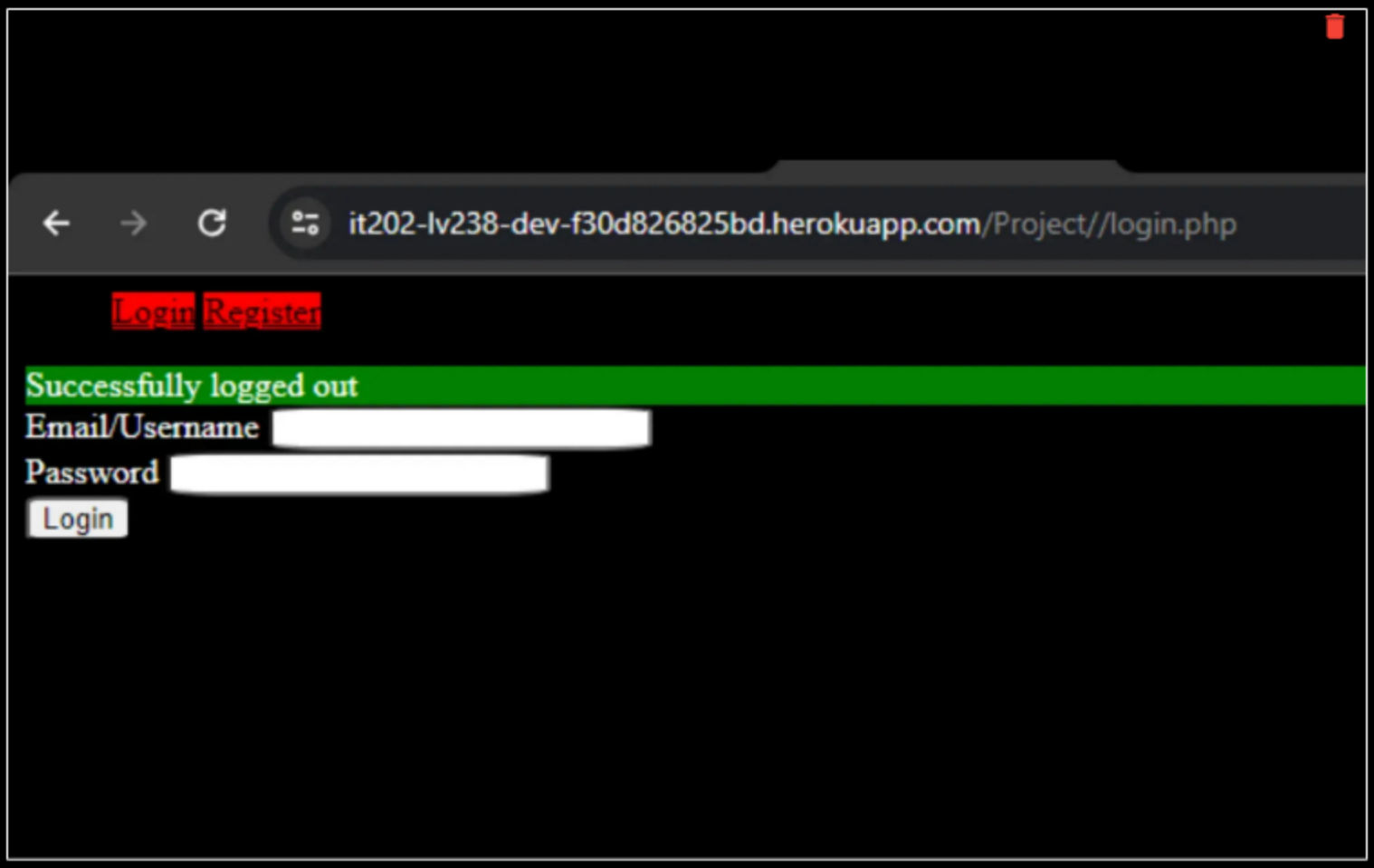
Small Medium Large

code for logout

Checklist Items (1)

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

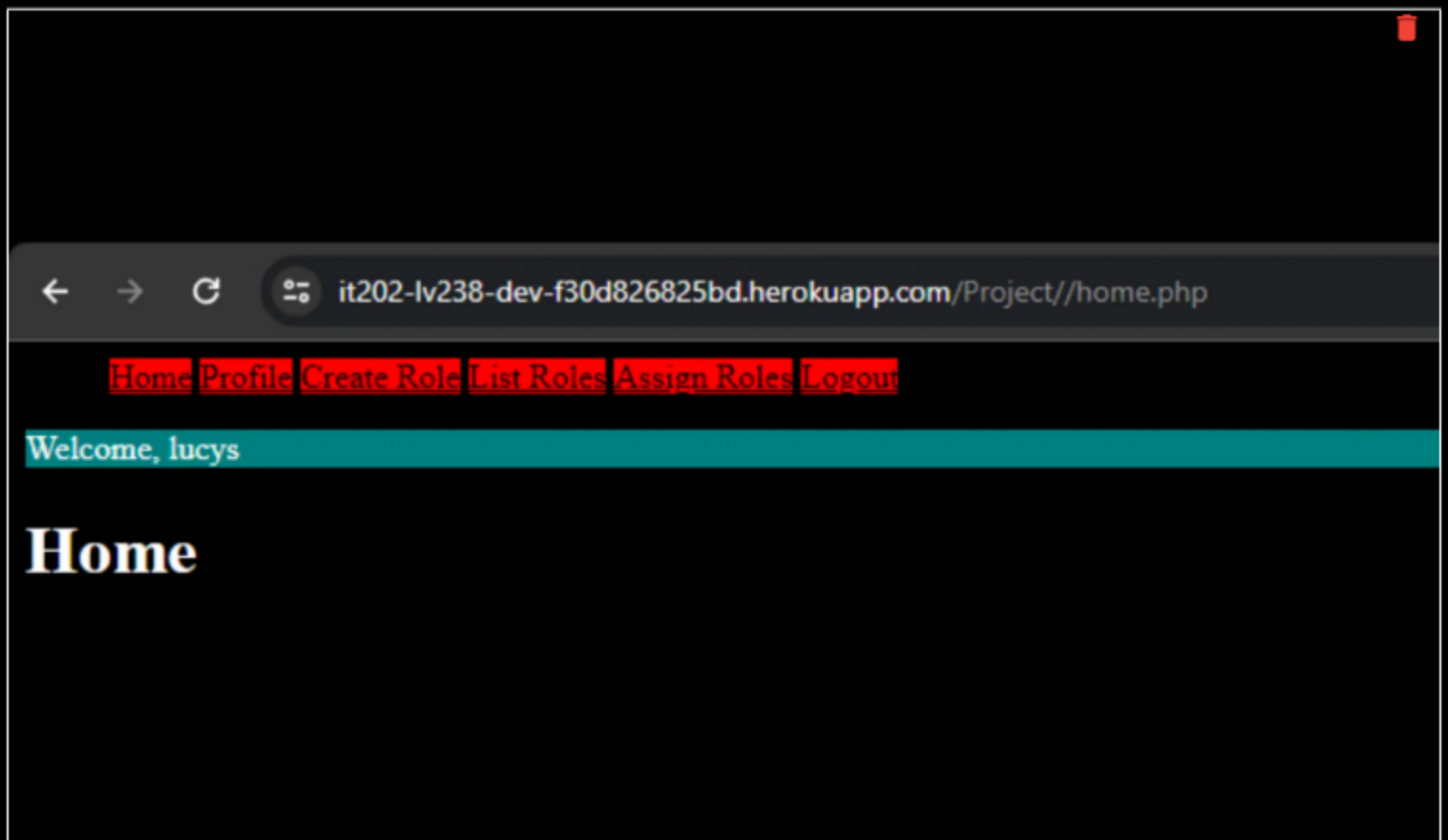
#3 Screenshot of the logout related code showing the session is destroyed (code). Ensure acid/date comment is present.



logout shown

Checklist Items (1)

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



navigation

Checklist Items (1)

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

COLLAPSE

Task #2 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/LucasValMelo/Iv238-IT202-008/pull/29

Basic Security Rules and Roles (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Authentication Screenshots

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Screenshot of the function that checks if a user is logged in
#2	1	Screenshot of the login check function being used (i.e., profile likely). Also caption what pages it's used on
#3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)
#4	1	Demonstrate the user-friendly message of trying to manually access a login-protected page while being logged out

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
*/
function is_logged_in($redirect = false, $destination = "login.php")
{
    $isLoggedIn = isset($_SESSION["user"]);
    if ($redirect && !$isLoggedIn) {
```

```php
    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, $destination = "logi...
function has_role($role)
{
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {                    //4/23/24 lv238
        foreach ($_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;
```

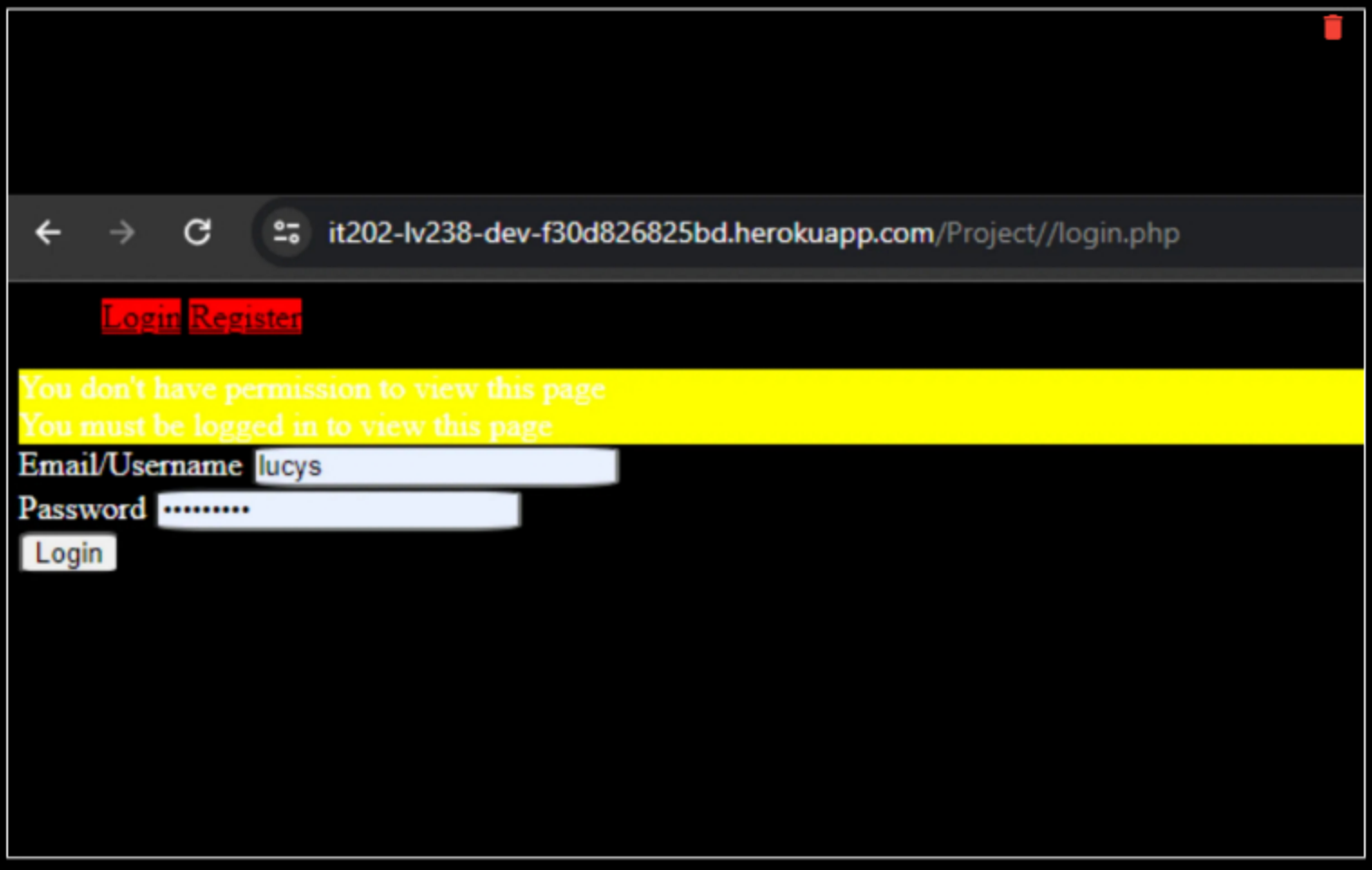
function code

Checklist Items (3)

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

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

#4 Demonstrate the user-friendly message of trying to manually access a login-protected page while being logged out



user message

Checklist Items (2)

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

#4 Demonstrate the user-friendly message of trying to manually access a login-protected page while being logged out

public_html > Project > profile.php > ...

You, 23 hours ago | 1 author (You)

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

function used in profile

Checklist Items (1)

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

COLLAPSE

Task #2 - Points: 1

Text: Authorization Screenshots

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Screenshot of the function that checks for a specific role
#2	1	Screenshot of the role check function being used. Also caption what pages it's used on
#3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)
#4	1	Demonstrate the user-friendly message of trying to manually access a role-protected page while being logged out

Task Screenshots:

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```php
if (password_verify($password, $hash)) {
    //flash("Weclome $email");
    $_SESSION["user"] = $user; //sets our session data from db      You, 2 weeks ago • finished
    //lookup potential roles
    $stmt = $db->prepare("SELECT Roles.name FROM Roles
    JOIN UserRoles on Roles.id = UserRoles.role_id
    where UserRoles.user_id = :user_id and Roles.is_active = 1 and UserRoles.is_active = 1");
    $stmt->execute([":user_id" => $user["id"]]);
    $roles = $stmt->fetchAll(PDO::FETCH_ASSOC); //fetch all since we'll want multiple
    //save roles or empty array
    if ($roles) {
        $_SESSION["user"]["roles"] = $roles; //at least 1 role
    } else {
        $_SESSION["user"]["roles"] = []; //no roles
    }
    flash("Welcome, " . get_username());                    //lv238 4.23.24
    die(header("Location: home.php"));
} else {
```

the code to check if a user has a role in login

Checklist Items (2)

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

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

```php
 */
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, $destination = "logi...
function has_role($role)
{
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {                    //4/23/24 lv238
        foreach ($_SESSION["user"]["roles"] as $r) {
            if ($r["name"] === $role) {
                return true;
            }
        } <- #20-24 foreach ($_SESSION["user"]["roles"] as $r)
```

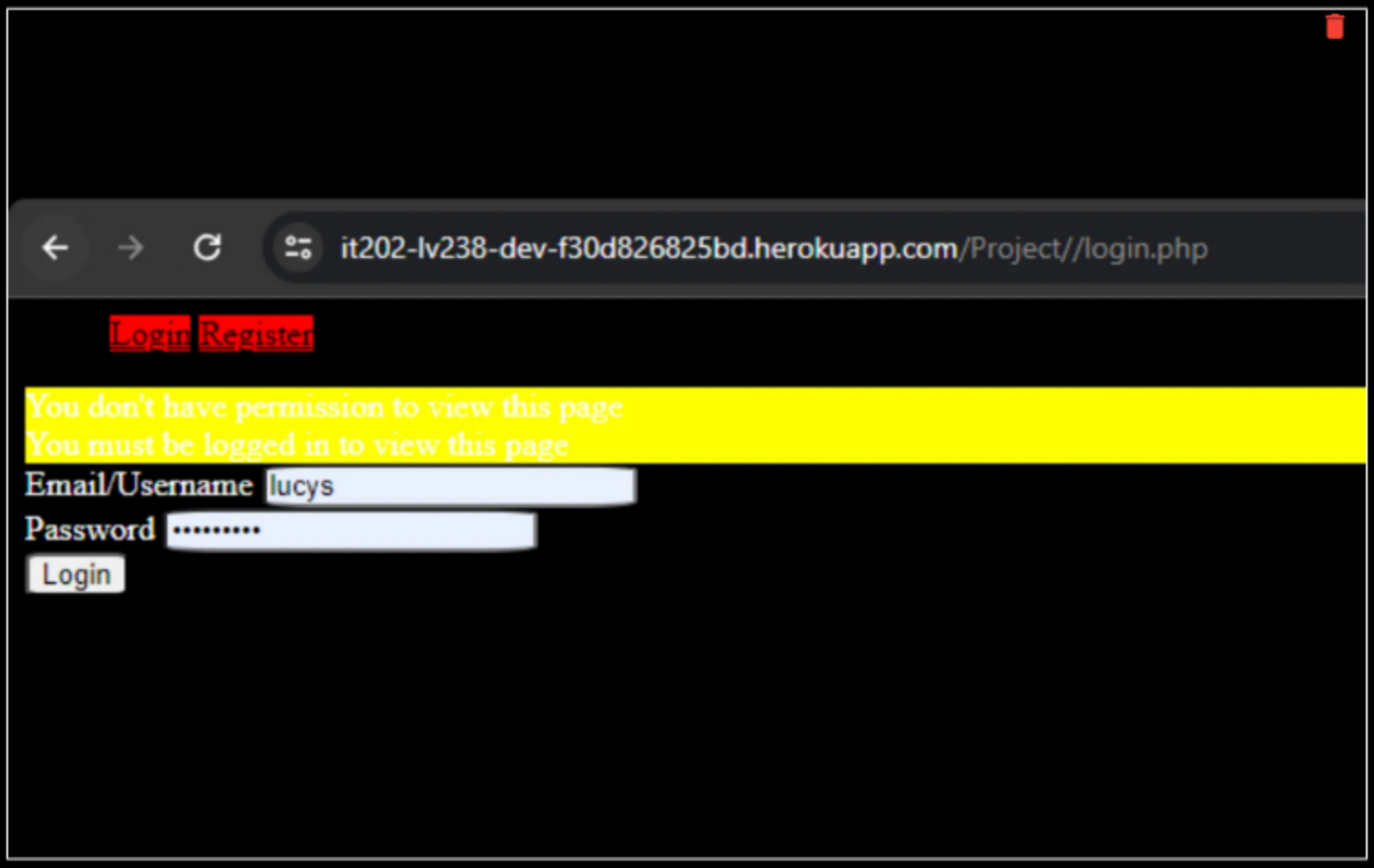
```
    } <- #19-25 if (is_logged_in() && isset($_SESSION["user"]["roles"]))
    return false;
} <- #18-27 function has_role($role)
function get_username()
{
```

function

Checklist Items (2)

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

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



user message

Checklist Items (1)

#4 Demonstrate the user-friendly message of trying to manually access a role-protected page while being logged out

^COLLAPSE^

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

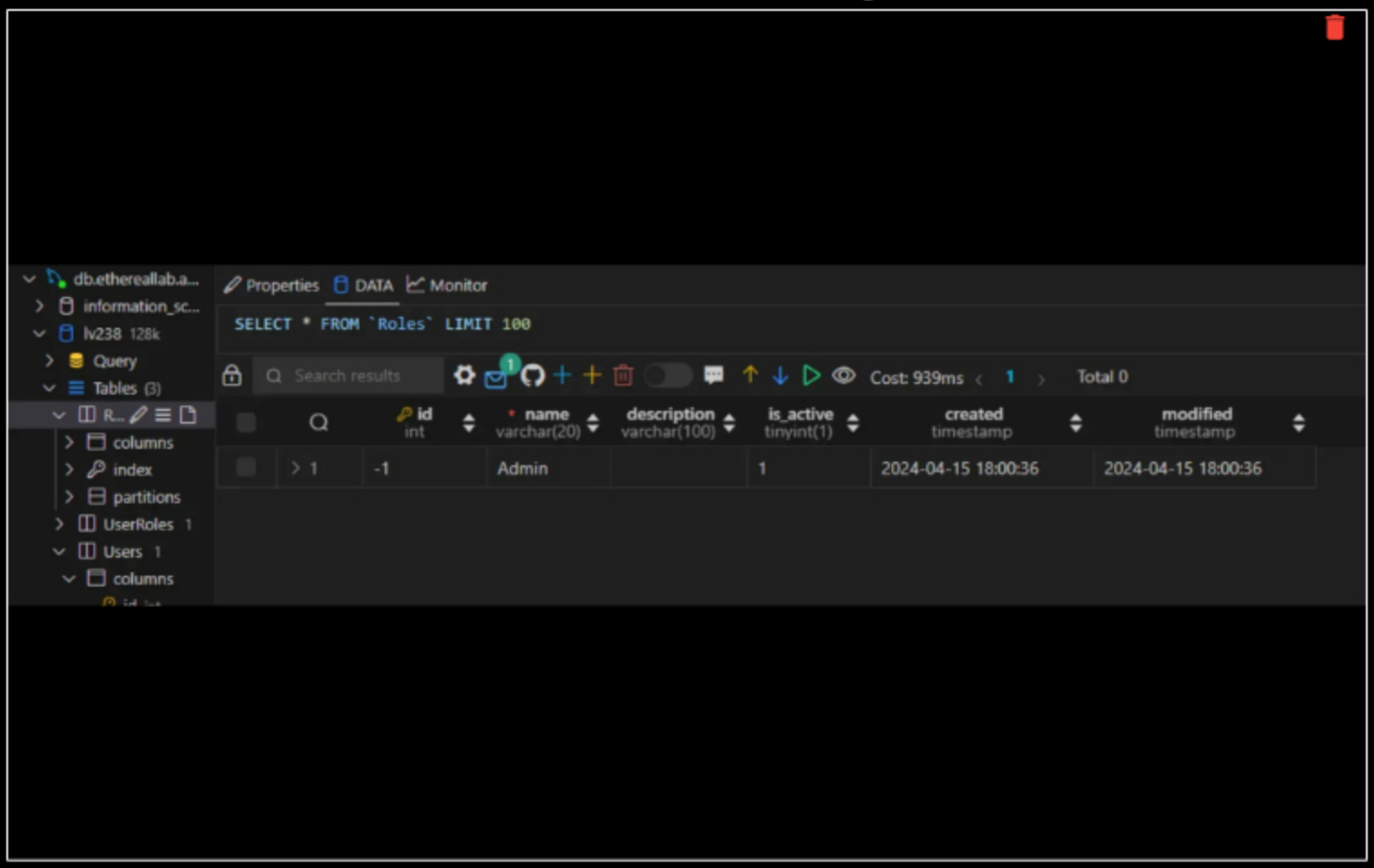
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	At least one valid and enabled User->Role reference (UserRoles table)
#2	1	UserRoles Table should have id, user_id, role_id, is_active, created, and modified columns
#3	1	Roles Table should have id, name, description, is_active, modified, and created columns
#4	1	At least one valid and enabled Role (Roles table)
#5	1	Ensure left panel or database name is present in each table screenshot (should contain your ucid)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



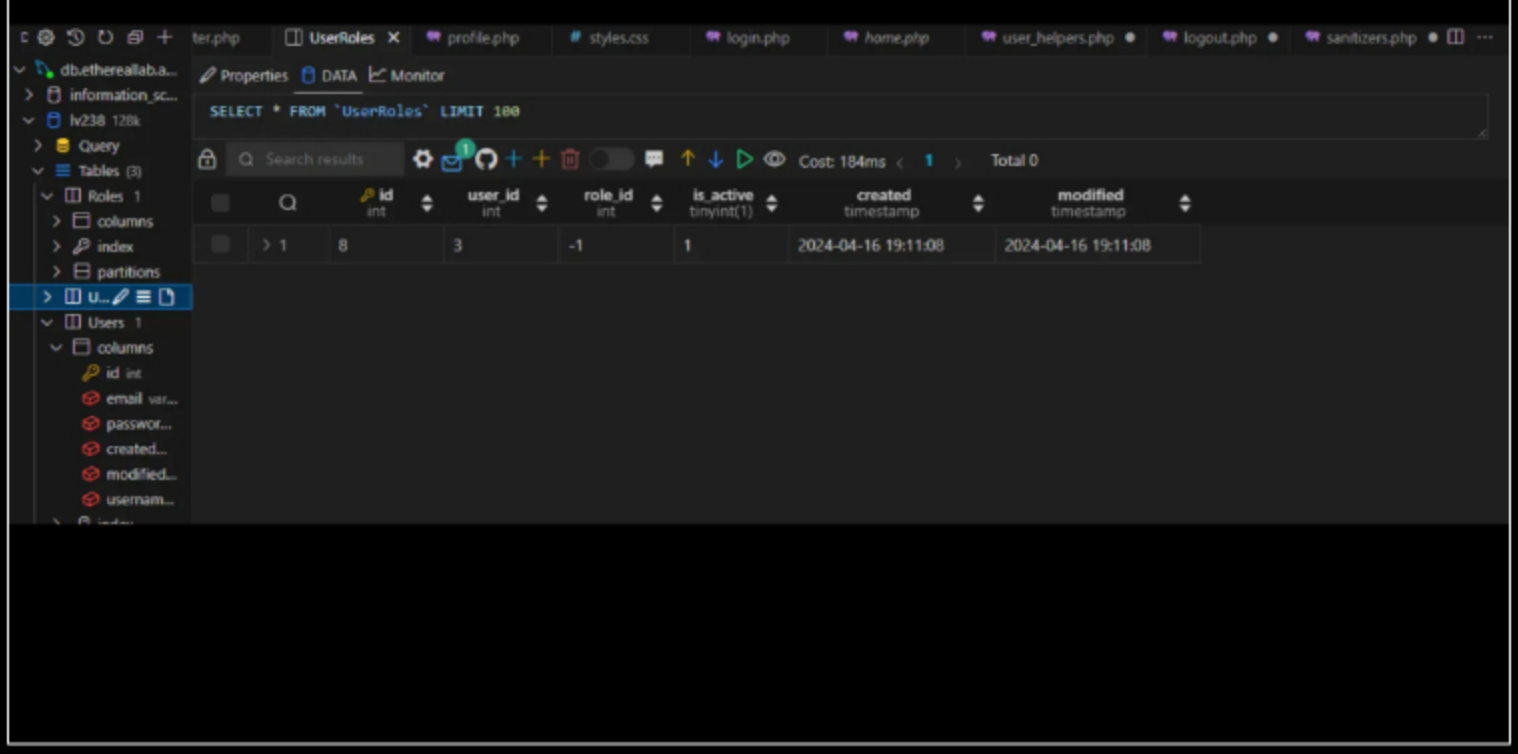
Roles

Checklist Items (3)

#3 Roles Table should have id, name, description, is_active, modified, and created columns

#4 At least one valid and enabled Role (Roles table)

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



userRoles

Checklist Items (3)

#1 At least one valid and enabled User->Role reference (UserRoles table)

#2 UserRoles Table should have id, user_id, role_id, is_active, created, and modified columns

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

COLLAPSE

Task #4 - Points: 1

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

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	What's the purpose of the UserRoles table?
#2	1	How does Roles.is_active differ from UserRoles.is_active?

Response:

The purpose of USerRoles is to get the foreign keys from users and Roles and connect the two within that database. Roles is active is to choose the state of the role in general, "no one can have this role active/this role cannot be activated" and UserRoles.is_active is an endpoint one, "you specifically do not have this role active."

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/LucasValMelo/lv238-IT202-008/pull/28

User Profile (2 pts.)
COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: View Profile Website Page

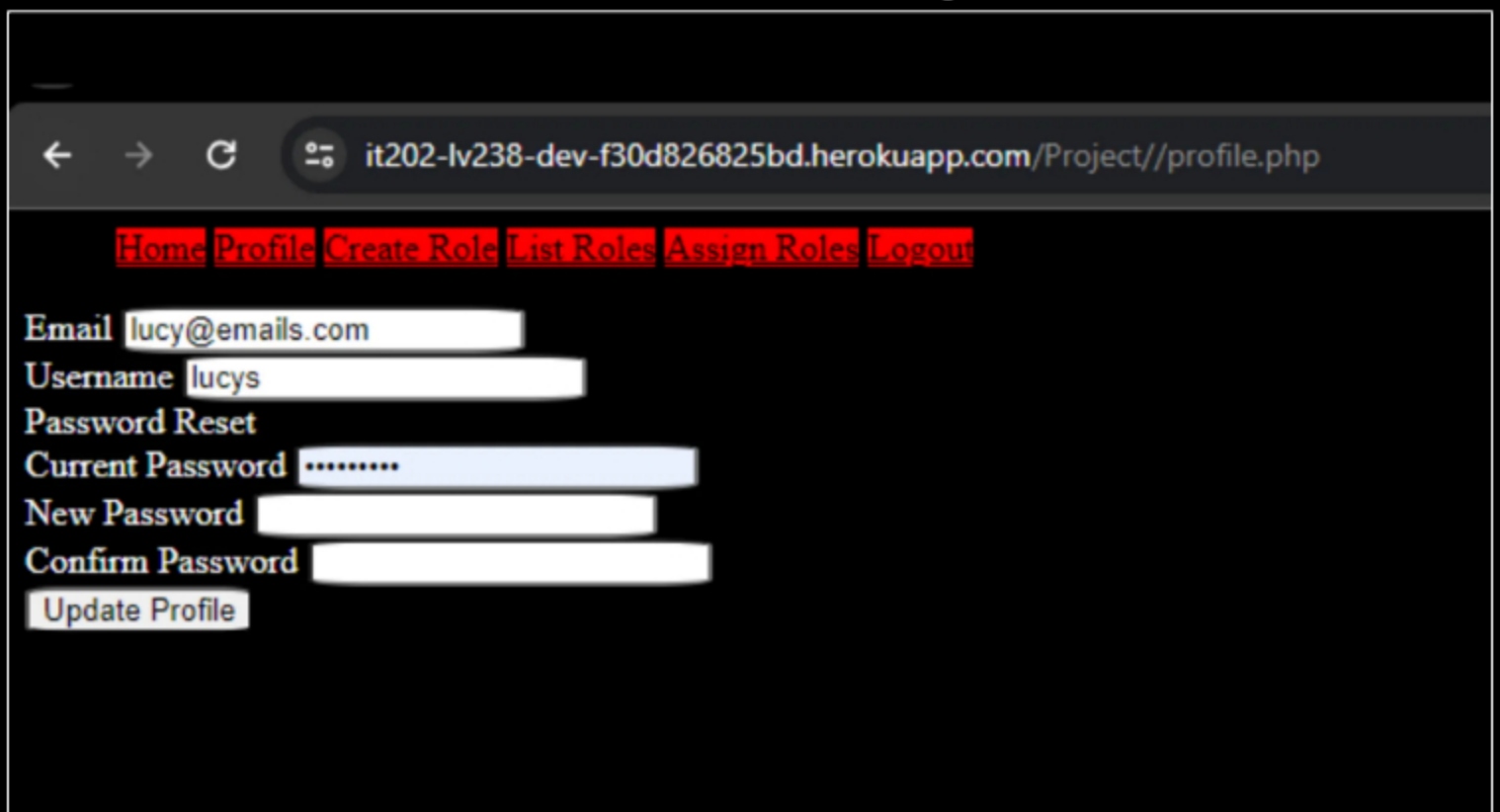
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Show the profile form correctly populated on page load (username, email)
#4	1	Should have the following fields: username, email, current password, new password, confirm password (or similar)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



the profile page

Checklist Items (4)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

#3 Show the profile form correctly populated on page load (username, email)

#4 Should have the following fields: username, email, current password, new password, confirm password (or similar)

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

Response:

When you load the page in Profile, you get a form, that form takes in fields, Email, Username, Password, and Confirm Password, these values are used to update the database and change the information to that userID

COLLAPSE

Task #3 - Points: 1

Text: Edit Profile Website Page

Checklist

*The checkboxes are for your own tracking

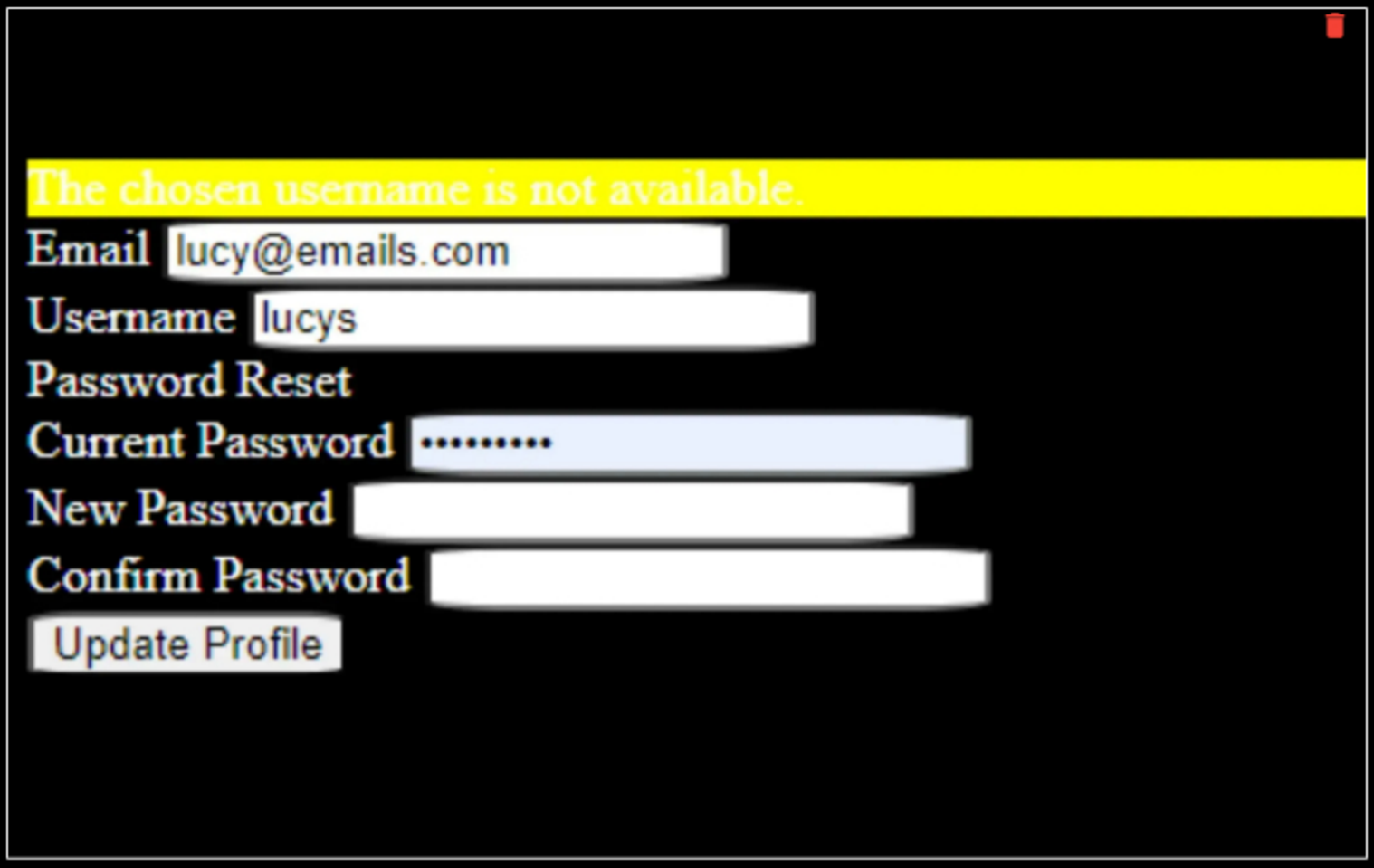
#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Demonstrate with before and after of a username change (including success message)
#4	1	Demonstrate with a before and after of an email change (including success message)
#5	1	Demonstrate the success message of updating password
#6	1	Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)

#7	1	Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



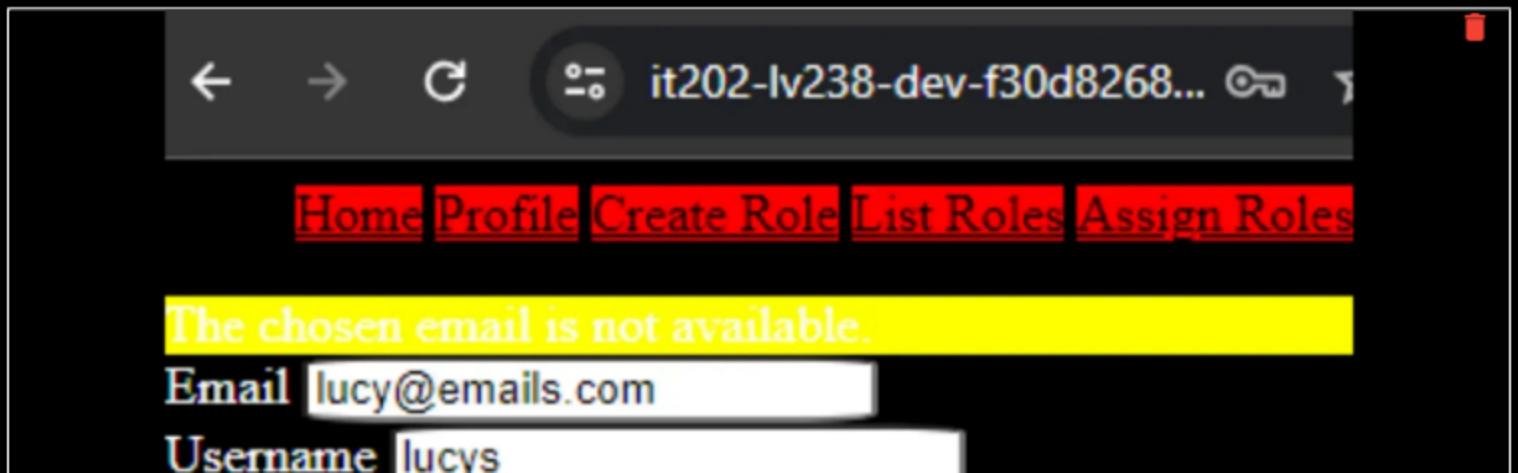
username not available

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

#7 Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)



Password Reset
Current Password ••••••••••
New Password
Confirm Password
Update Profile

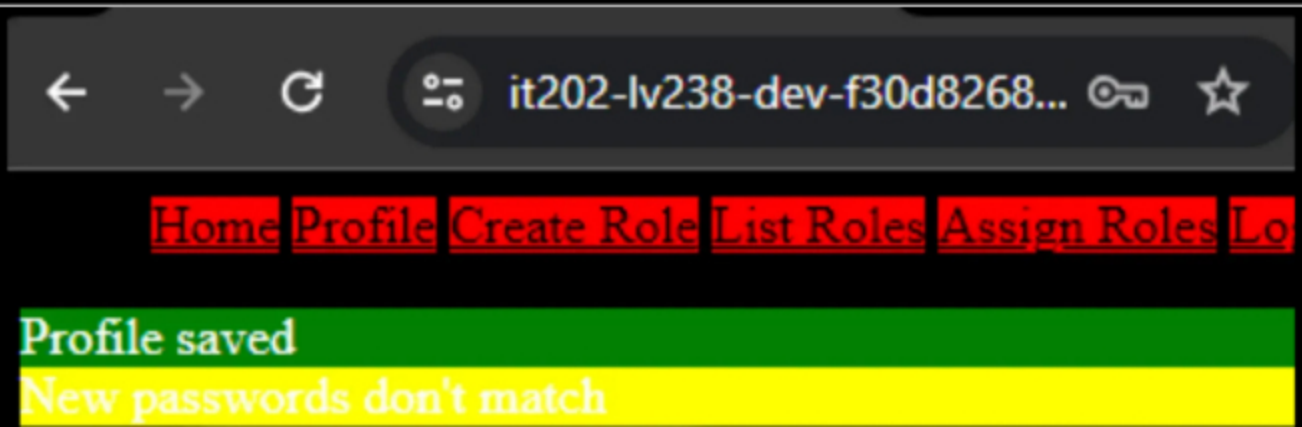
email not available

Checklist Items (0)

Current password is invalid
Email lucy@emails.com
Username lucys
Password Reset
Current Password ••••••••••
New Password
Confirm Password
Update Profile

not password in database

Checklist Items (0)



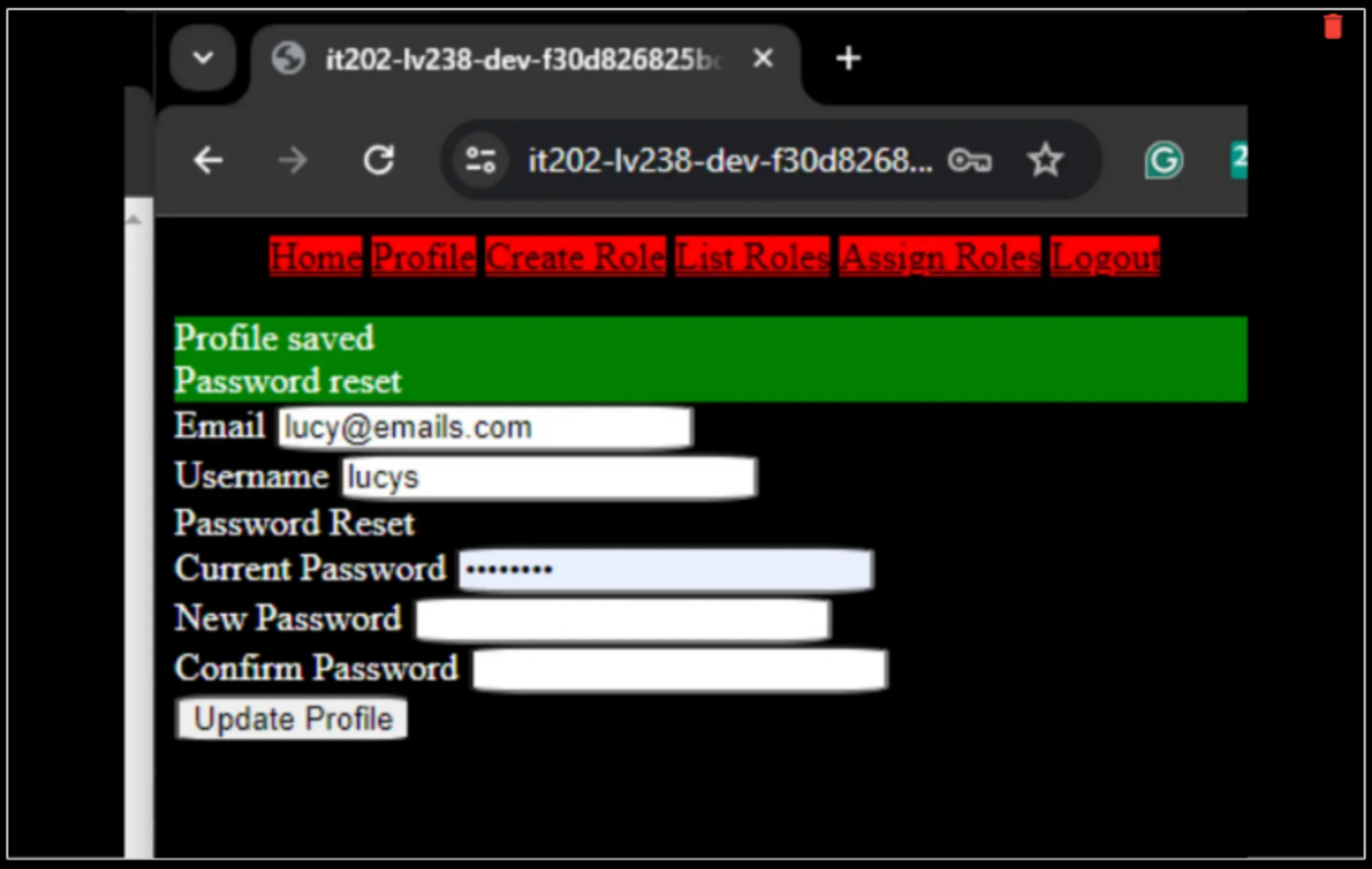
passwords dont match

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

#7 Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)



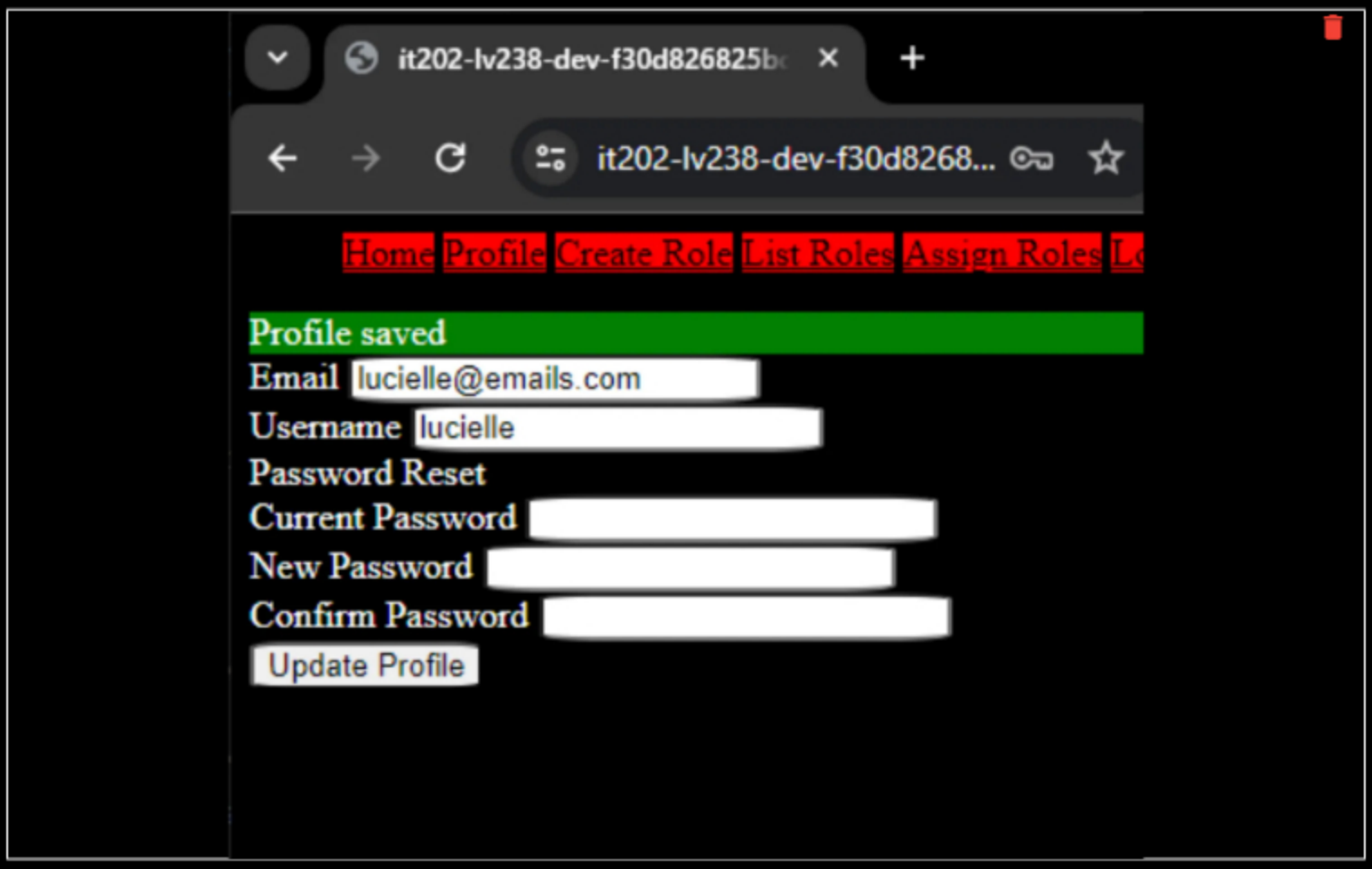
successful password

Checklist Items (3)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

#5 Demonstrate the success message of updating password



the before is every other screenshot

Checklist Items (2)

#3 Demonstrate with before and after of a username change (including success message)

#4 Demonstrate with a before and after of an email change (including success message)

COLLAPSE

Task #4 - Points: 1

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

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Updating Username/Email
#2	1	Updating password
#3	1	Don't just show code, translate things to plain English

Response:

what it does is it updates the entry in the Users database relating to the ID that is being used, so if someone wishes to

change their name and email, it wouldn't delete the account as a new one, as the id in the Users table will remain the same. changing the password is the same thing, it updates the entry in the database to the new one if your password and confirm match

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/LucasValMelo/lv238-IT202-008/pull/25

Misc (1 pt.)

COLLAPSE

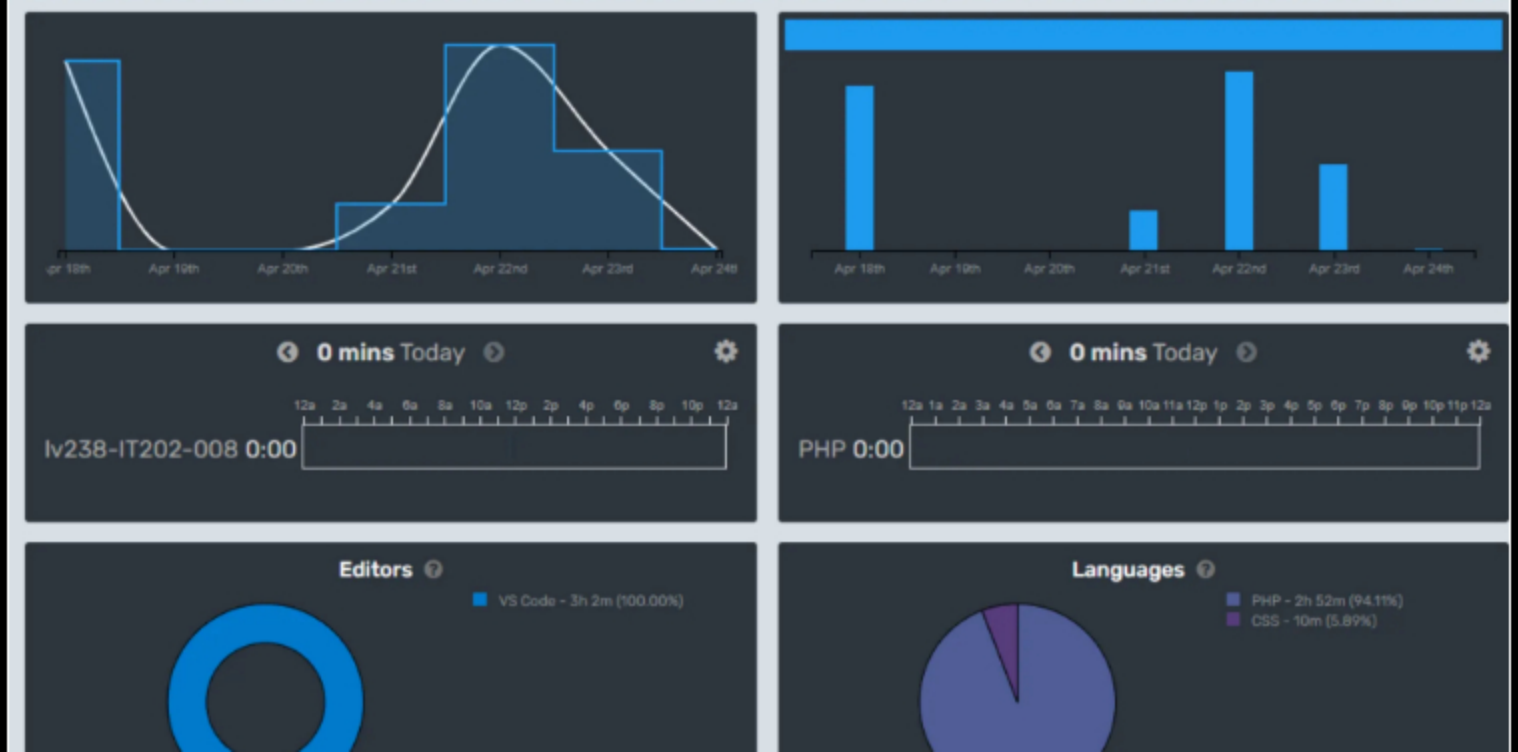
Task #1 - Points: 1

Text: Screenshot of wakatime

Task Screenshots:

Gallery Style: Large View

Small Medium Large



wakatime ss

COLLAPSE

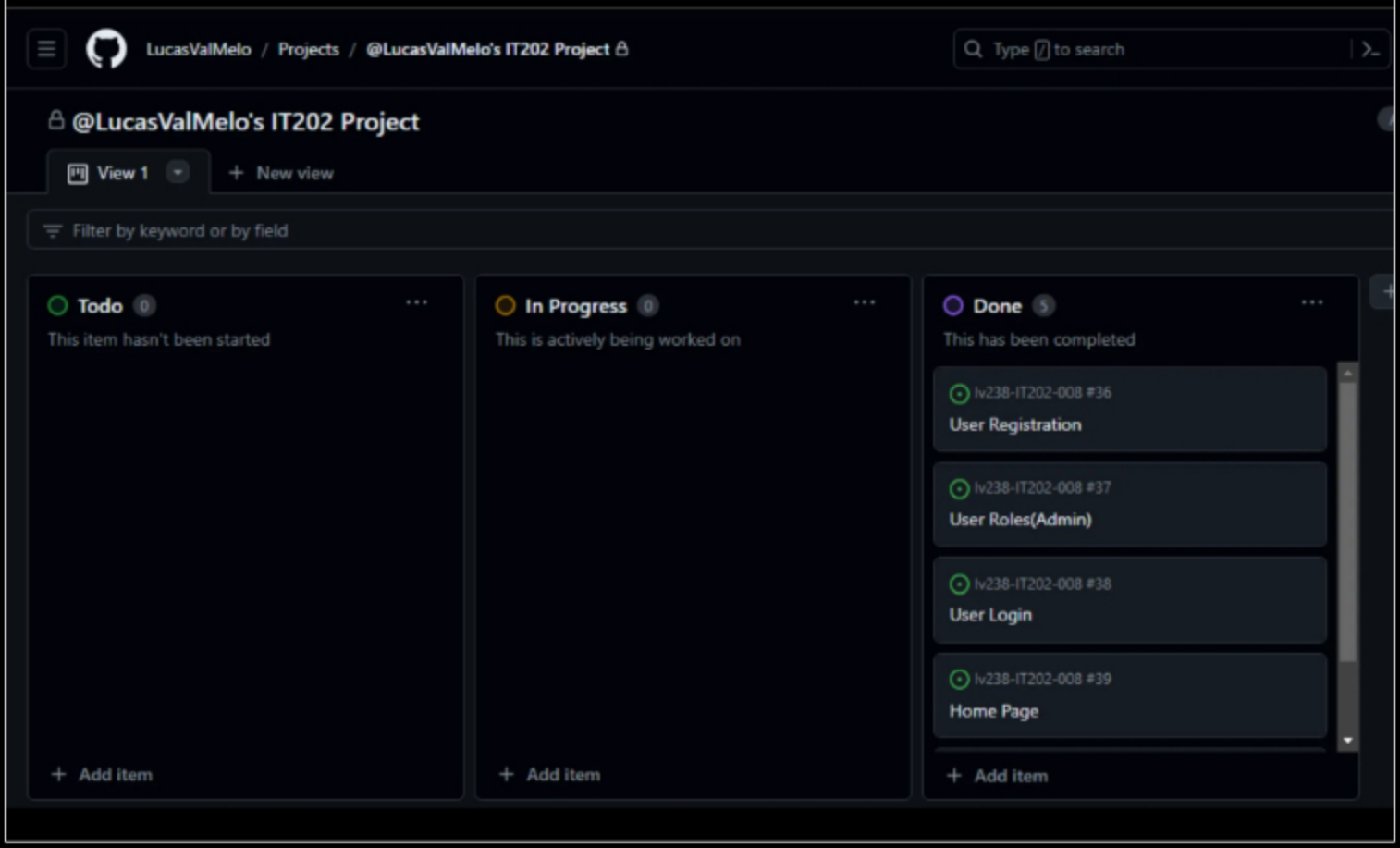
Task #2 - Points: 1

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



proj board

COLLAPSE

Task #3 - Points: 1

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

URL #1

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

^COLLAPSE ^

Task #4 - Points: 1

Text: Provide a direct link to the login page from your prod instance

URL #1

https://it202-lv238-prod-87030a0bf20e.herokuapp.com//Project/login.php

End of Assignment