

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

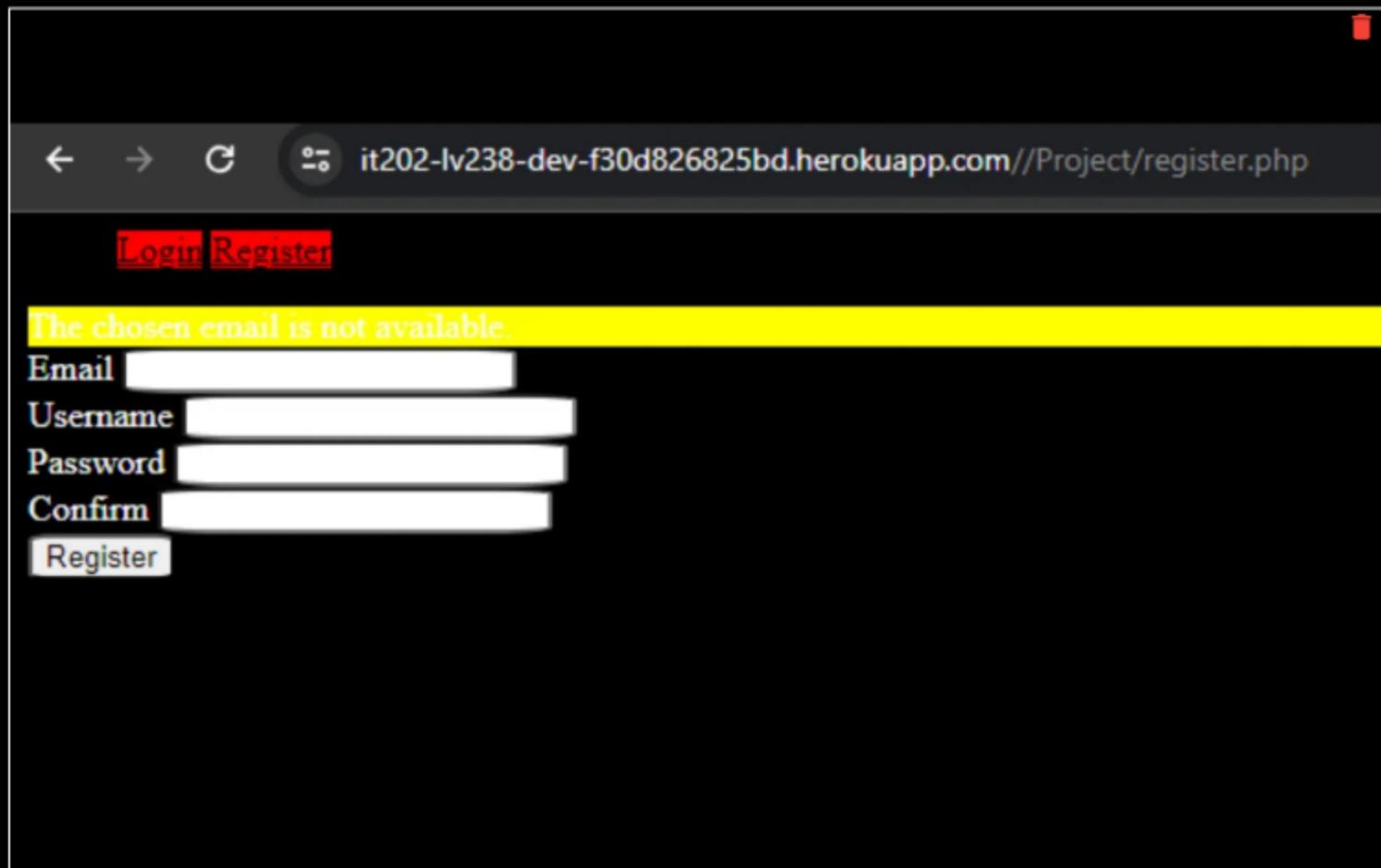
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



The chosen email is not available.

Email

Username

Password

Confirm

emial not available

#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)

The chosen username is not available

Email

Username

Password

Confirm

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)

The chosen username is not available

Invalid test Username

Password is Too Short

Email Address is invalid

Email Address is invalid

Passwords Must Match

Email

Username

Password

Confirm

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)

Successfully registered!

Email

Username

Password

Confirm

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

Task #2 - Points: 1

Text: Screenshot of the form code

 Details:

Should have appropriate input types for the fields

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```
<?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

Task #3 - Points: 1

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

#	Points	Details
<input checked="" type="checkbox"/> #1	1	Show the JavaScript validations (include any extra files related)
<input checked="" type="checkbox"/> #2	1	Show the PHP validations (include any lib content)
<input checked="" type="checkbox"/> #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._#$%&*+/=?_-]{1,}@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+){1,}+$/);
    let valReg2 = new RegExp(/^(a-zA-Z0-9._-)+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,})$/);
    let userReg = new RegExp(/^a-zA-Z0-9_-){3,16}$/);

    if (!/^a-zA-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;
} ❸ <- #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
    //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;
    }
}

```

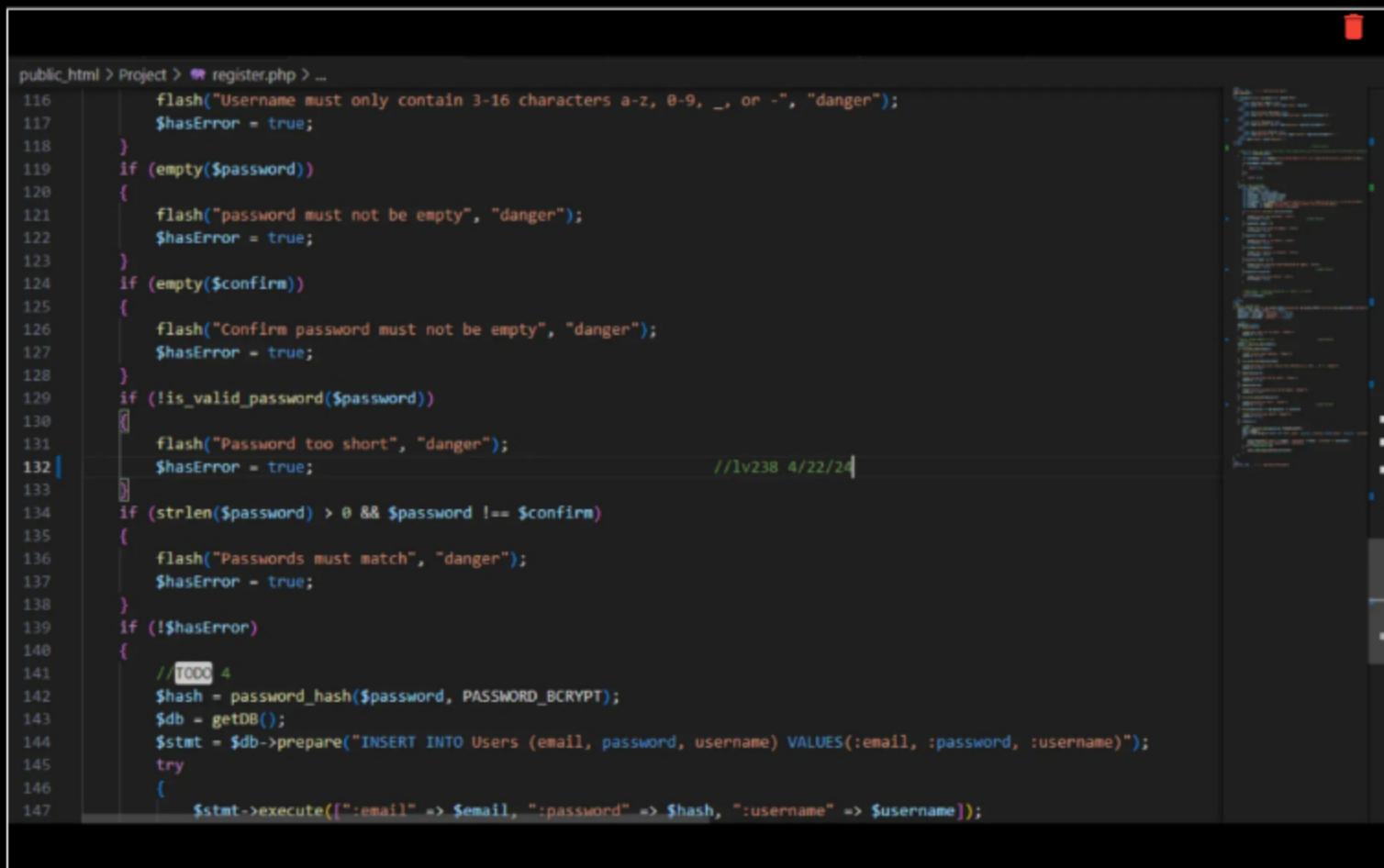
lv238 4/22/24 You, 1 second ago * Uncomm

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)



```
public_html > Project > register.php > ...
116     flash("Username must only contain 3-16 characters a-z, 0-9, _, or -, "danger");
117     $hasError = true;
118 }
119 if (empty($password))
120 {
121     flash("password must not be empty", "danger");
122     $hasError = true;
123 }
124 if (empty($confirm))
125 {
126     flash("Confirm password must not be empty", "danger");
127     $hasError = true;
128 }
129 if (!is_valid_password($password))
130 {
131     flash("Password too short", "danger");
132     $hasError = true; //lv238 4/22/24
133 }
134 if (strlen($password) > 0 && $password !== $confirm)
135 {
136     flash("Passwords must match", "danger");
137     $hasError = true;
138 }
139 if (!$hasError)
140 {
141     // TODO: 4
142     $hash = password_hash($password, PASSWORD_BCRYPT);
143     $db = getDB();
144     $stmt = $db->prepare("INSERT INTO Users (email, password, username) VALUES(:email, :password, :username)");
145     try
146     {
147         $stmt->execute([":email" => $email, ":password" => $hash, ":username" => $username]);
148     }
149 }
```

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)

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
<input checked="" type="checkbox"/> #1	1	Password should be hashed
<input checked="" type="checkbox"/> #2	1	Should have email, password, username (unique), created, modified, and id fields
<input checked="" type="checkbox"/> #3	1	Ensure left panel or database name is present (should contain your ucid)

Task Screenshots:

Gallery Style: Large View

Properties DATA Monitor

SELECT * FROM 'Users' LIMIT 100

Cost: 164ms Total 0

	id	email	password	created	modified	username
	3	lucas_verty@hotmail.com	\$2y\$10\$/2Ferr/Rz.pzISZjol	2024-04-16 17:40:01	2024-04-16 17:40:01	lucas_verty

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)

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 hash 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

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



The screenshot shows a web browser window with the following details:

- Address Bar:** it202-lv238-dev-f30d826825bd.herokuapp.com/Project//home.php
- Header:** Home Profile Create Role List Roles Assign Roles Logout
- User Information:** Welcome, lucas_verty
- Main Content:** A large, bold "Home" heading.
- Message Bar:** 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)

The screenshot shows a web browser window with the following details:

- Address Bar:** it202-lv238-dev-f30d826825bd.herokuapp.com/Project//login.php
- Header:** Login Register
- User Information:** Successfully logged out
- Form Fields:** Email/Username: lucas_verty, Password: PrettyPlease
- Buttons:** Login

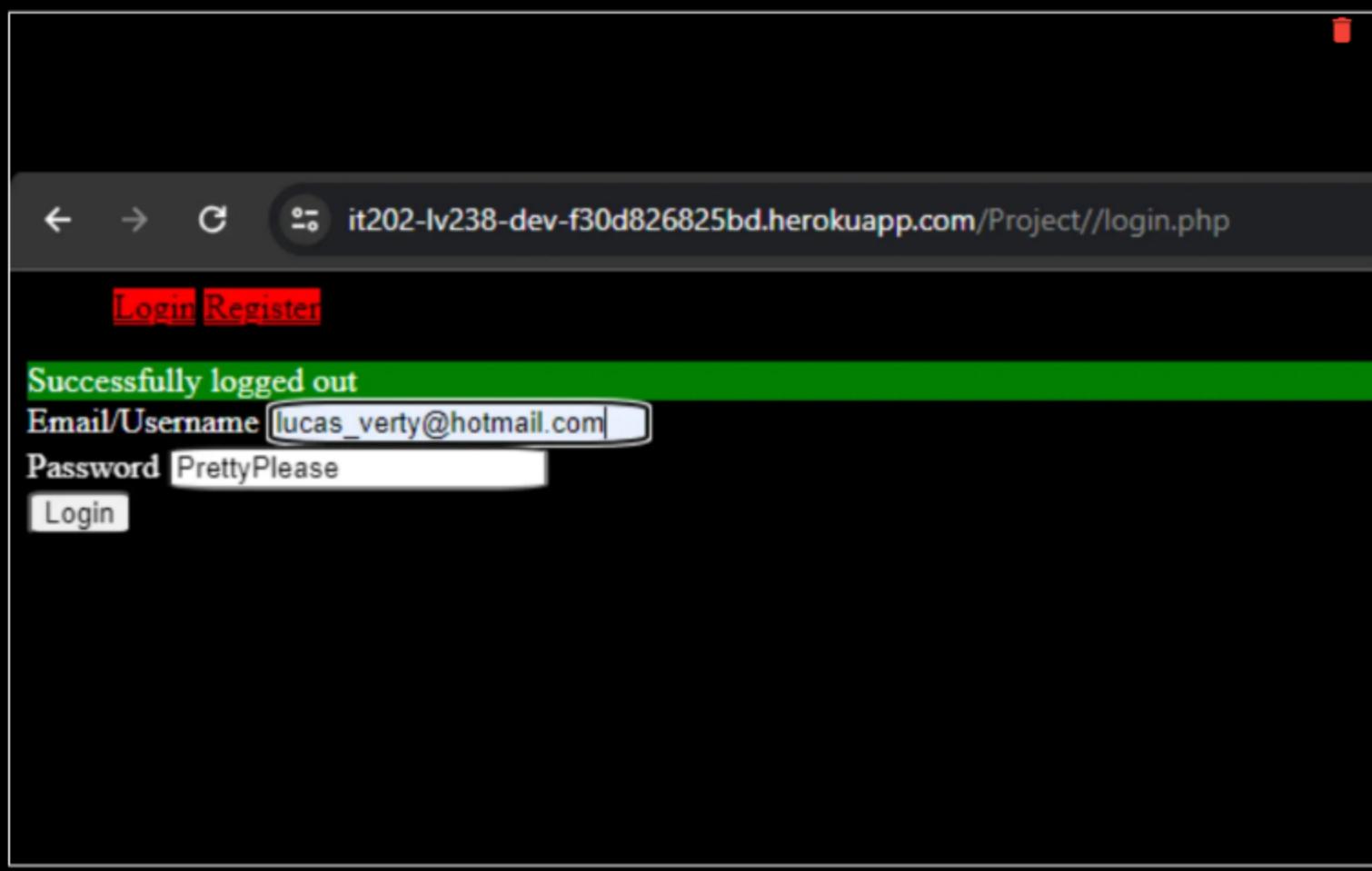
correct username data

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



it202-lv238-dev-f30d826825bd.herokuapp.com/Project//login.php

Login Register

Successfully logged out

Email/Username

Password

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](#) [Register](#)

Email/Username not found

Password is Too Short

Invalid test Username

Password is Too Short

Email Address is invalid

Email/Username

Password

[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)

← → C it202-lv238-dev-f30d826825bd.herokuapp.com//Project/login.php

[Login](#) [Register](#)

Invalid password

Email/Username

Password

[Login](#)

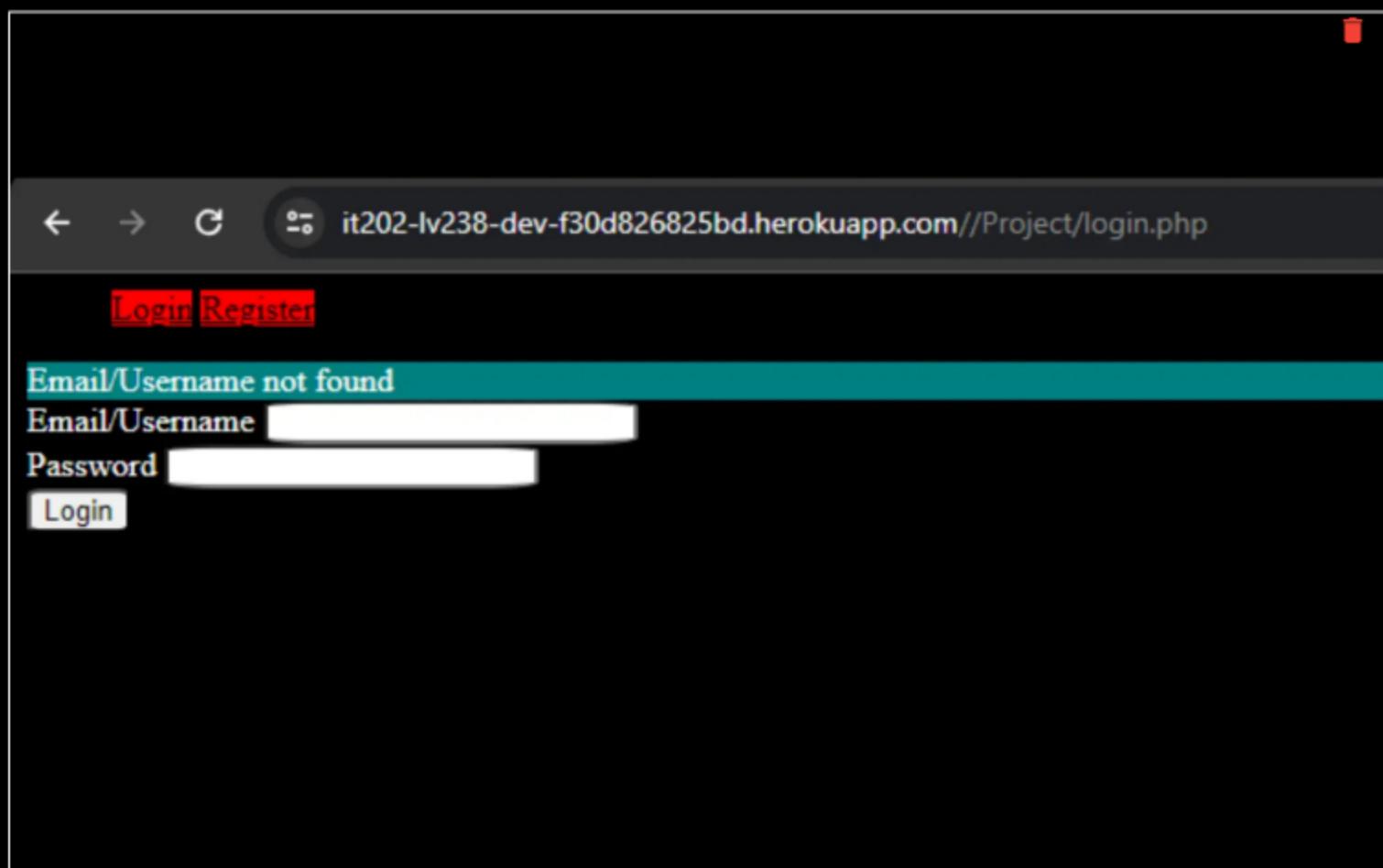
password doesn't match

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



A screenshot of a web browser showing a login page. The address bar displays the URL: `it202-lv238-dev-f30d826825bd.herokuapp.com//Project/login.php`. The page has a dark background with light-colored text. At the top, there are two buttons: `Login` and `Register`. Below them, a red error message box contains the text `Email/Username not found`. Below the message box are two input fields: one for `Email/Username` and one for `Password`. Both fields are currently empty. At the bottom of the page is a `Login` button.

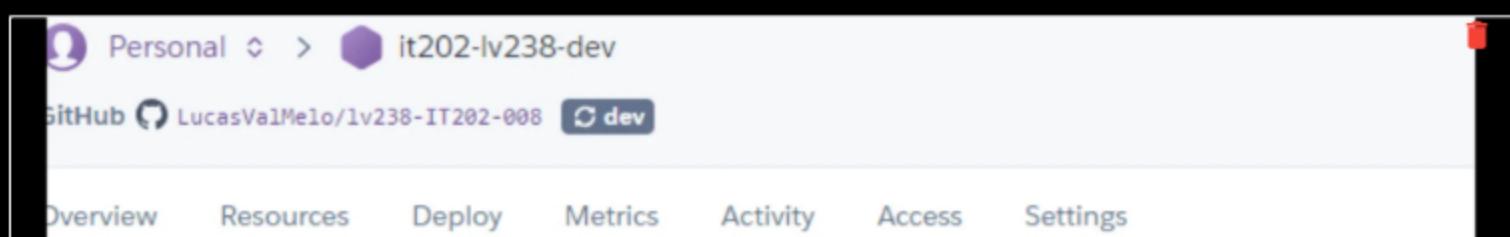
account doesn't exist'

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



Application Logs

```
2024-04-24T17:16:58.651985+00:00 app[web.1]:      'color' => 'info',
2024-04-24T17:16:58.651986+00:00 app[web.1]:      ),
2024-04-24T17:16:58.651997+00:00 app[web.1]:      ),
2024-04-24T17:16:58.652008+00:00 app[web.1]:      'user' =>
2024-04-24T17:16:58.652008+00:00 app[web.1]:      array (
2024-04-24T17:16:58.652020+00:00 app[web.1]:      'id' => 10,
2024-04-24T17:16:58.652035+00:00 app[web.1]:      'email' => 'lucy@email.com',
2024-04-24T17:16:58.652053+00:00 app[web.1]:      'username' => 'lucys',
2024-04-24T17:16:58.652066+00:00 app[web.1]:      'roles' =>
2024-04-24T17:16:58.652076+00:00 app[web.1]:      array (
2024-04-24T17:16:58.652076+00:00 app[web.1]:      0 =>
2024-04-24T17:16:58.652087+00:00 app[web.1]:      array (
2024-04-24T17:16:58.652099+00:00 app[web.1]:      'name' => 'Admin',
2024-04-24T17:16:58.652099+00:00 app[web.1]:      ),
```

Autoscroll with output

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)

Task #2 - Points: 1

Text: Screenshot of the form code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #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)
<input type="checkbox"/> #2	1	Show JavaScript validations (include any extra files related)
<input type="checkbox"/> #3	1	Show PHP validations (include any lib content)
<input type="checkbox"/> #4	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```
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>
    //
```

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)

```
<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,})$/);
    let userReg = new RegExp(/^(a-zA-Z0-9_-){3,16}$/);

    /*if (!/^a-zA-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)

```
        errorState = false;
    }
    if(jemail.contains("@"))
    {
        if(!valReg2.test(jemail))
        {
            flash("Email Address is invalid", "info");
            errorState = false;
        }
    } <- #58-64 if(jemail.contains("@"))
else
{
    if (!/^[a-zA-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)) {
```

```
        if (!is_email($email)) {
            flash("Invalid email address");
            $hasError = true;
        }
    } else {
        if (!is_valid_username($email)) {
            flash("Invalid username");
            $hasError = true;
        }
    } <- #104-109 else
    if ($hasError) {
        flash("Please correct the errors above.");
        return;
    }
}
```

//4/23/24 lv238 You, 2

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)

```
if (empty($password)) {
    flash("password must not be empty");
    $hasError = true;
}
if (!is_valid_password($password)) {
    flash("Password too short");
    $hasError = true;
}
if (!$hasError) {
    //flash("Welcome, $email");
    //TODO 4
    $db = getDB();
    $stmt = $db->prepare("SELECT id, email, username, password from Users
    where email = :email or username = :email");
    try {
        $r = $stmt->execute([":email" -> $email]);
        if ($r) {
            $user = $stmt->fetch(PDO::FETCH_ASSOC);
            if ($user) {
                $hash = $user["password"];
                unset($user["password"]);
                if (password_verify($password, $hash)) {
                    //flash("Weclome $email");
                    $_SESSION["user"] = $user; //sets our session data from db
                    //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
                }
            }
        }
    }
}
```

//4/23/24 lv238

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)

```
4     if (password_verify($password, $hash)) {
5         flash("Weclome $email");
6         $_SESSION["user"] = $user; //sets our session data from db
7         //lookup potential roles
8         $stmt = $db->prepare("SELECT Roles.name FROM Roles
9         JOIN UserRoles on Roles.id = UserRoles.role_id
10        where UserRoles.user_id = :user_id and Roles.is_active = 1 and UserRoles.is_active = 1");
11        $stmt->execute([":user_id" -> $user["id"]]);
12        $roles = $stmt->fetchAll(PDO::FETCH_ASSOC); //fetch all since we'll want multiple
13    }
14 }
```

```
8
9     where UserRoles.user_id = :user_id and Roles.is_active = 1 and UserRoles.is_active = 1");
10    $stmt->execute([":user_id" => $user["id"]]);
11    $roles = $stmt->fetchAll(PDO::FETCH_ASSOC); //fetch all since we'll want multiple
12    //save roles or empty array
13    if ($roles) {
14        $_SESSION["user"]["roles"] = $roles; //at least 1 role
15    } else {
16        $_SESSION["user"]["roles"] = []; //no roles
17    }
18    flash("Welcome, " . get_username());
19    die(header("Location: home.php"));
20 } else {
21     flash("Invalid password");
22 }
23 } else {
24     flash("Email/Username not found");
25 }
26 } <- #128-154 if ($user)
27 } catch (Exception $e) {
28     flash("<pre>" . var_export($e, true) . "</pre>");
29 }
30 } <- #124-158 try
31 } <- #122-159 $stmt = $db->prepare
32 >>
```

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

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($username)
{
    return preg_match('/^[\w\-\_]{3,16}$/', $username);
}
function is_valid_password($password)
{
    return strlen($password) >= 8;
}
```



lib 1

Checklist Items (2)

Checklist Items (2)

#3 Show PHP validations (include any lib content)

```
...
<?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($_SESSION["user"]["roles"])) {
        foreach ($_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)

```
} <- #8-16 function is_logged_in($redirect = false, $destination = "logi...
function has_role($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($role)
function get_username()
{
    if (is_logged_in()) { //we need to check for login first because "user" key may not exist
        return se($_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
        return se($_SESSION["user"], "email", "", false);
    }
}
```

```

        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;
}

```

A View Bookmarks 16:10 Sat 1/4 Coverage 1 LITE 0 CODE RUN

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)

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
<input checked="" type="checkbox"/> #1	1	Don't just show code, translate things to plain English
<input checked="" type="checkbox"/> #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)

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 ^](#)

Task #1 - Points: 1

Text: Capture the following screenshots

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Screenshot of the navigation when logged in (site)
<input checked="" type="checkbox"/> #2	1	Screenshot of the redirect to login with the user-friendly logged-out message (site)
<input checked="" type="checkbox"/> #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

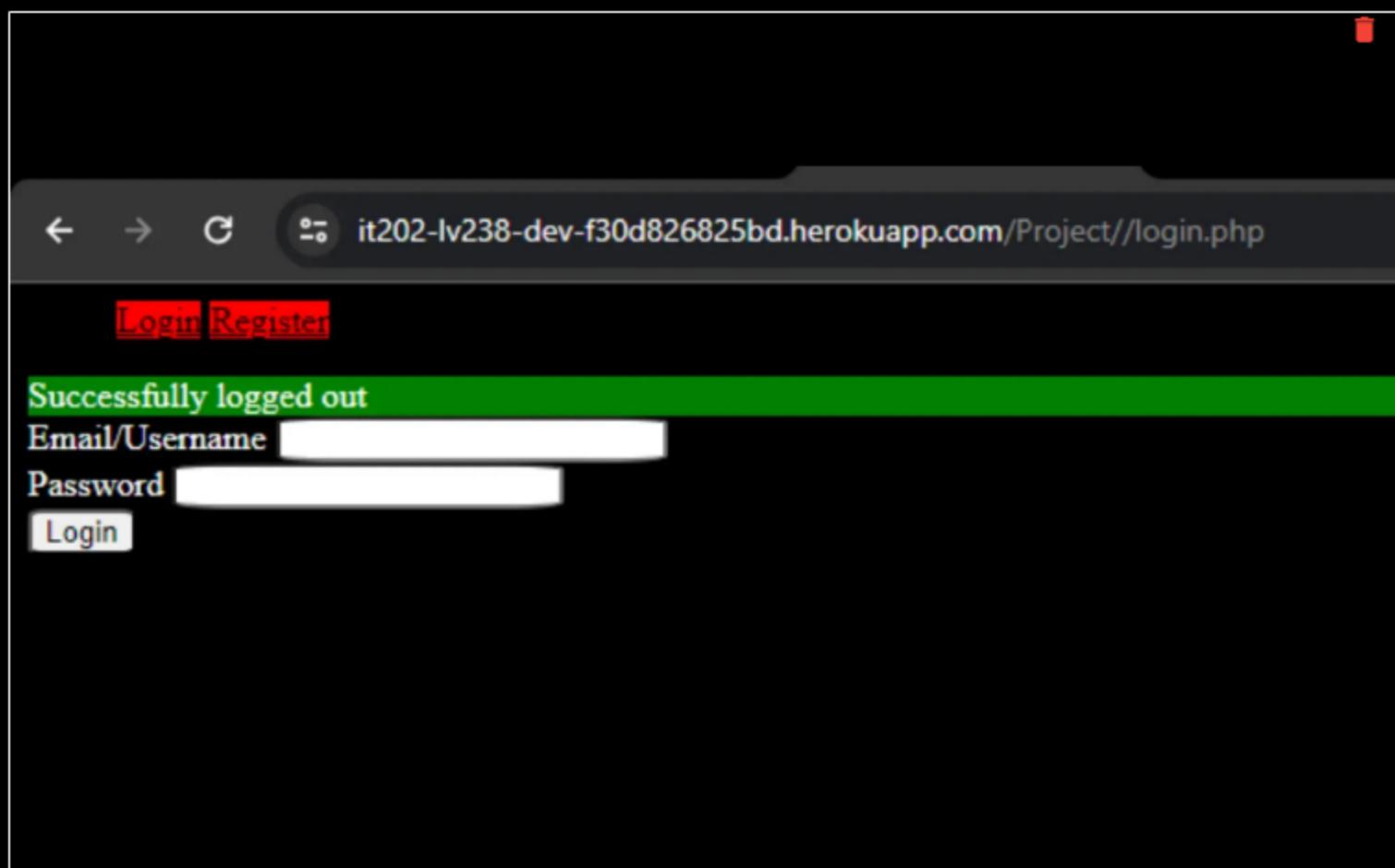
```
public_html > Project > logout.php
You, last week | 1 author (You)
1 <?php
2 session_start();
3 require(__DIR__ . "/../../lib/functions.php");
4 reset_session();
5 require(__DIR__ . "/../../partials/flash.php");
6 flash("Successfully logged out", "success");
7 header("Location: login.php");
```

code for logout

Checklist Items (1)

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

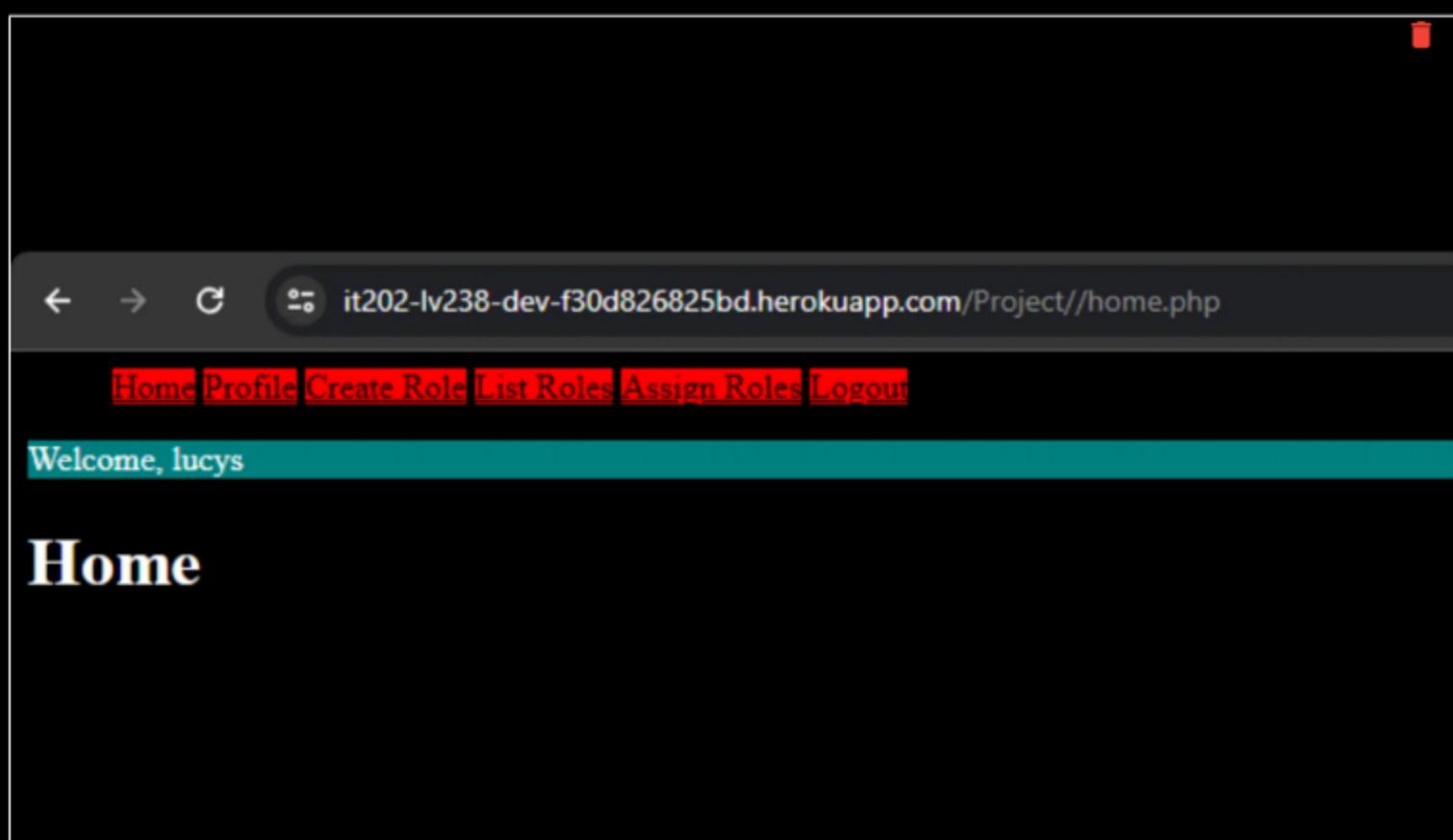
#3 Screenshot of the logout related code showing the session is destroyed (code). Ensure acid/date comment is present.



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)

Task #2 - Points: 1

Text: Include pull request links related to this feature

i Details:

Should end in /pull/#

URL #1

<https://github.com/LucasValMelo/ly238-IT202-008/pull/29>

Basic Security Rules and Roles (2 pts.)

Task #1 - Points: 1

Text: Authentication Screenshots

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Screenshot of the function that checks if a user is logged in
<input type="checkbox"/> #2	1	Screenshot of the login check function being used (i.e., profile likely). Also caption what pages it's used on
<input type="checkbox"/> #3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)
<input type="checkbox"/> #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) {
```

```

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($_SESSION["user"]["roles"])) {
        foreach ($_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;
} //4/23/24 lv238

```

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

← → C it202-lv238-dev-f30d826825bd.herokuapp.com/Project//login.php

Login Register

You don't have permission to view this page
You must be logged in to view this page

Email/Username lucys

Password

Login

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)

```
1  <?php
2  require_once(__DIR__ . "/../../partials/nav.php");
3  is_logged_in(true);
4  ?>
5  <?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

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

Small

Medium

Large

```

if (password_verify($password, $hash)) {
    //flash("Welcome $email");
    $_SESSION["user"] = $user; //sets our session data from db
    //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());
    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)

```

/*
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($_SESSION["user"]["roles"])) {
        foreach ($_SESSION["user"]["roles"] as $r) {
            if ($r["name"] === $role) {
                return true;
            }
        }
    } <- #20-24 foreach ($_SESSION["user"]["roles"] as $r)

```

//4/23/24 lv238

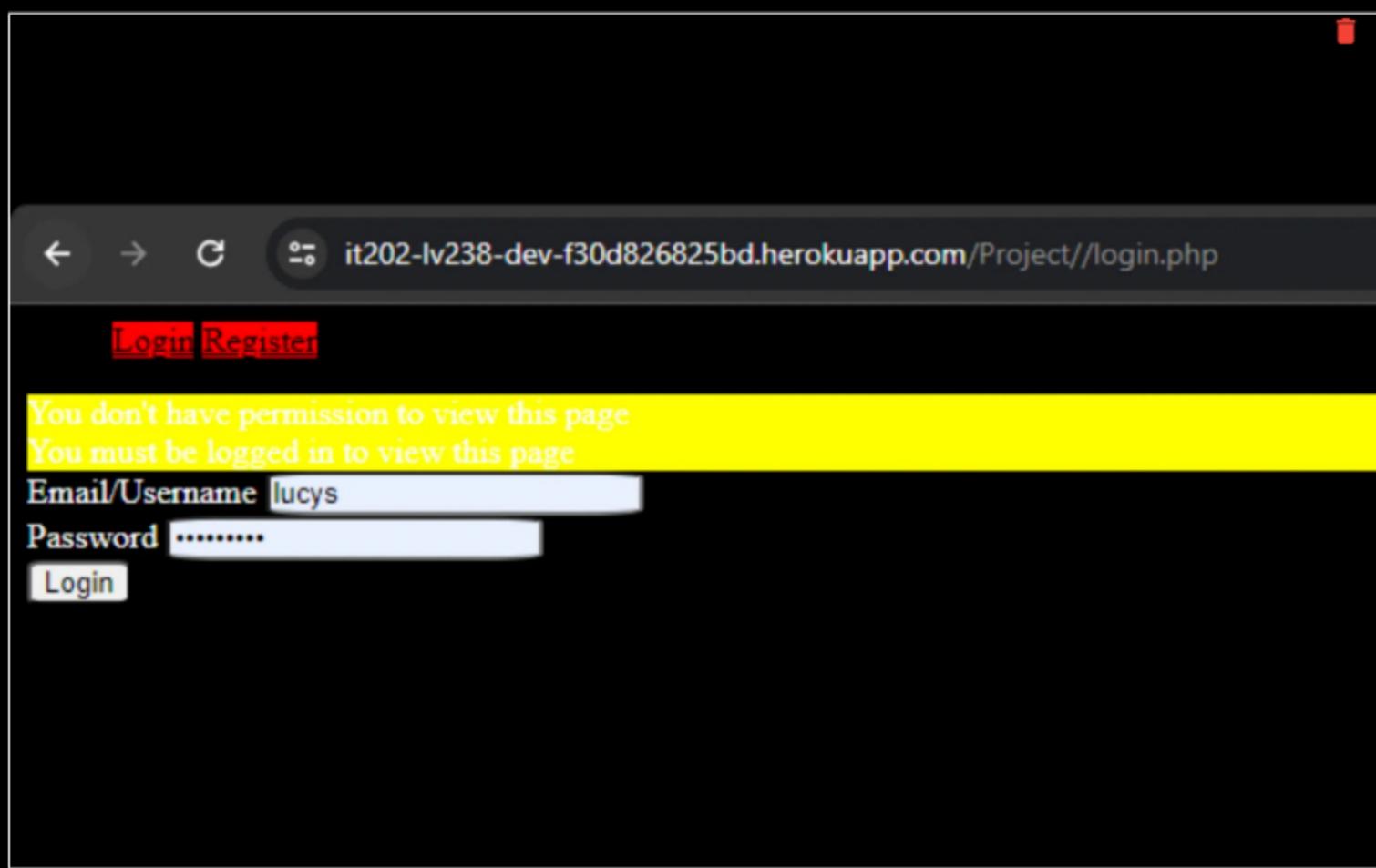
```
} <- #19-25 if (is_logged_in() && isset($_SESSION["user"]["roles"]))  
  return false;  
} <- #18-27 function has_role($role)  
function get_username()  
{
```

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

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

Checklist

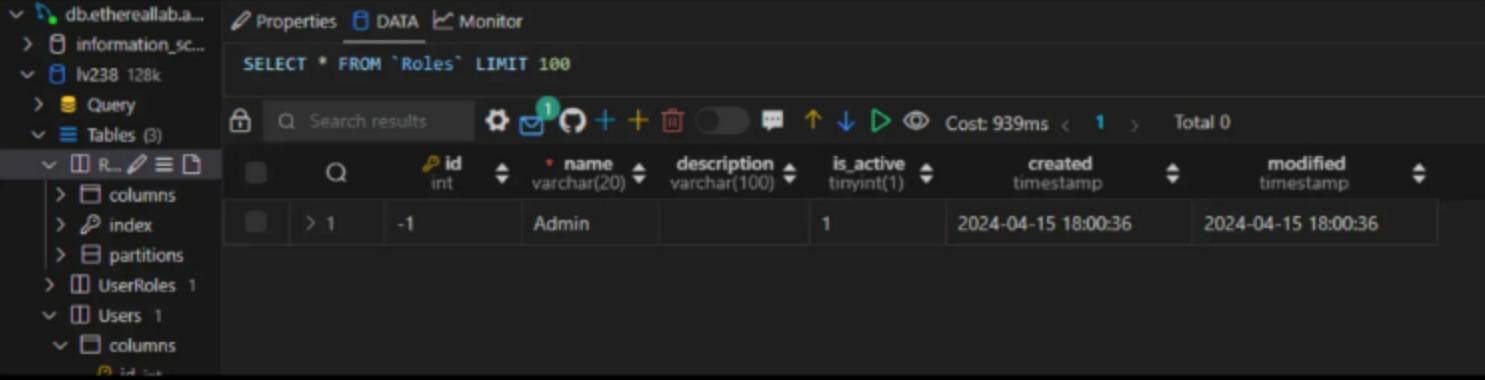
*The checkboxes are for your own tracking

#	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



Large View Screenshot of the Roles table in a database management tool. The table structure is as follows:

	id	name	description	is_active	created	modified
1	1	Admin		1	2024-04-15 18:00:36	2024-04-15 18:00:36

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)

The screenshot shows a MySQL database interface with the following details:

- Database:** db.etherallab...
- Table:** UserRoles
- Query:** SELECT * FROM 'UserRoles' LIMIT 100
- Result:** A table with 1 row and 7 columns. The columns are: id (int), user_id (int), role_id (int), is_active (tinyint(1)), created (timestamp), and modified (timestamp). The data is: id=1, user_id=8, role_id=-1, is_active=1, created=2024-04-16 19:11:08, modified=2024-04-16 19:11:08.
- Tables:** Roles (1), Users (1).
- Columns:** Roles: id, user_id, role_id, is_active, created, modified, updated. Users: id, email, password, created, modified, updated.

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)

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
<input checked="" type="checkbox"/> #1	1	What's the purpose of the UserRoles table?
<input checked="" type="checkbox"/> #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."

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 ^](#)

Task #1 - Points: 1

Text: View Profile Website Page

Checklist

*The checkboxes are for your own tracking

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

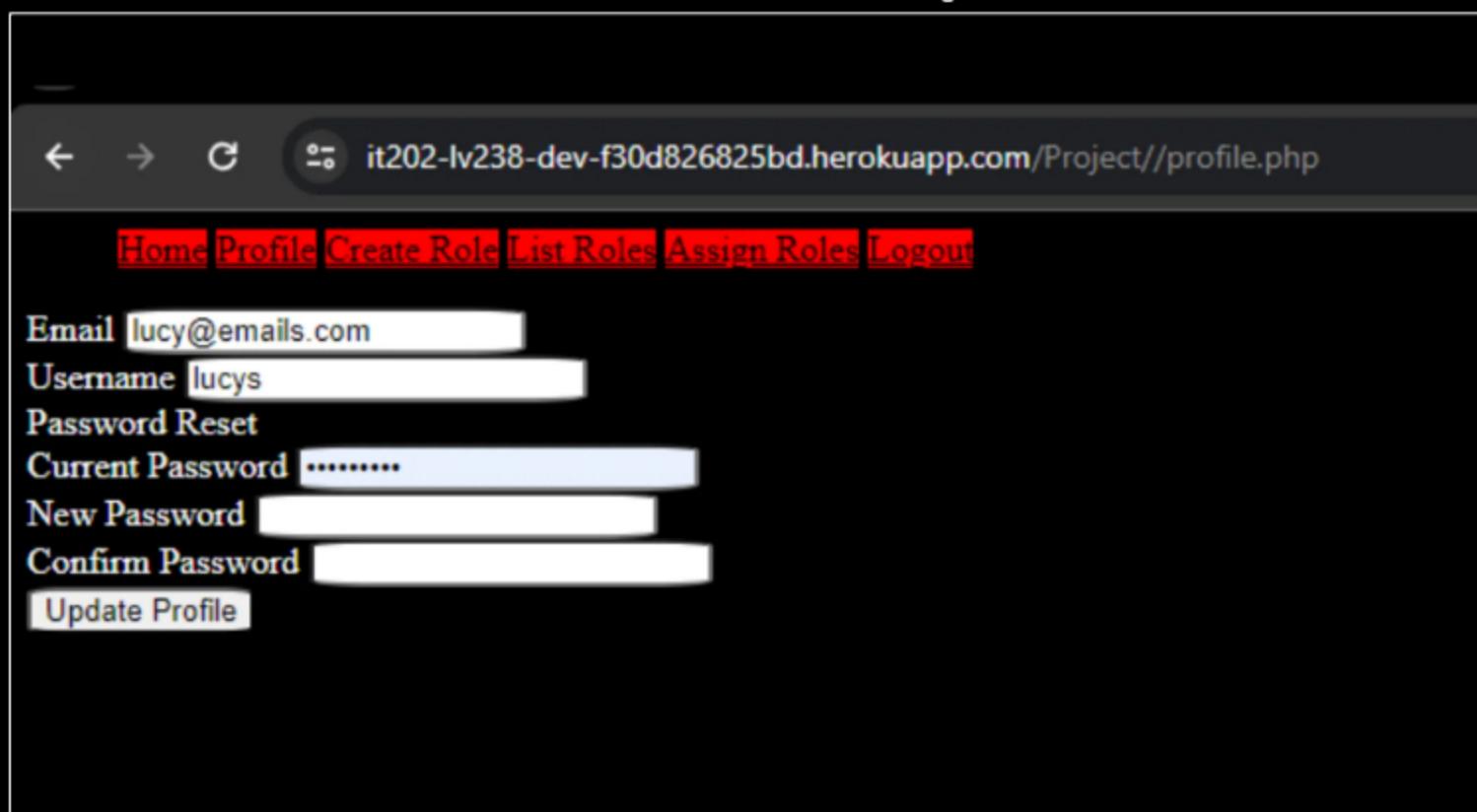
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



it202-lv238-dev-f30d826825bd.herokuapp.com/Project//profile.php

Home Profile Create Role List Roles Assign Roles Logout

Email

Username

Password Reset

Current Password

New Password

Confirm Password

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)

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

Task #3 - Points: 1

Text: Edit Profile Website Page

Checklist

*The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

The chosen username is not available.

Email

Username

Password Reset

Current Password

New Password

Confirm Password

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)

it202-lv238-dev-f30d8268...

[Home](#) [Profile](#) [Create Role](#) [List Roles](#) [Assign Roles](#)

The chosen email is not available.

Email

Username

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)



it202-lv238-dev-f30d8268...

[Home](#) [Profile](#) [Create Role](#) [List Roles](#) [Assign Roles](#) [Lo](#)

Profile saved

New passwords don't match

Email lucy@email.com

Username [lucys](#)

Password Reset

Current Password

New Password

Confirm Password

[Update Profile](#)

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)

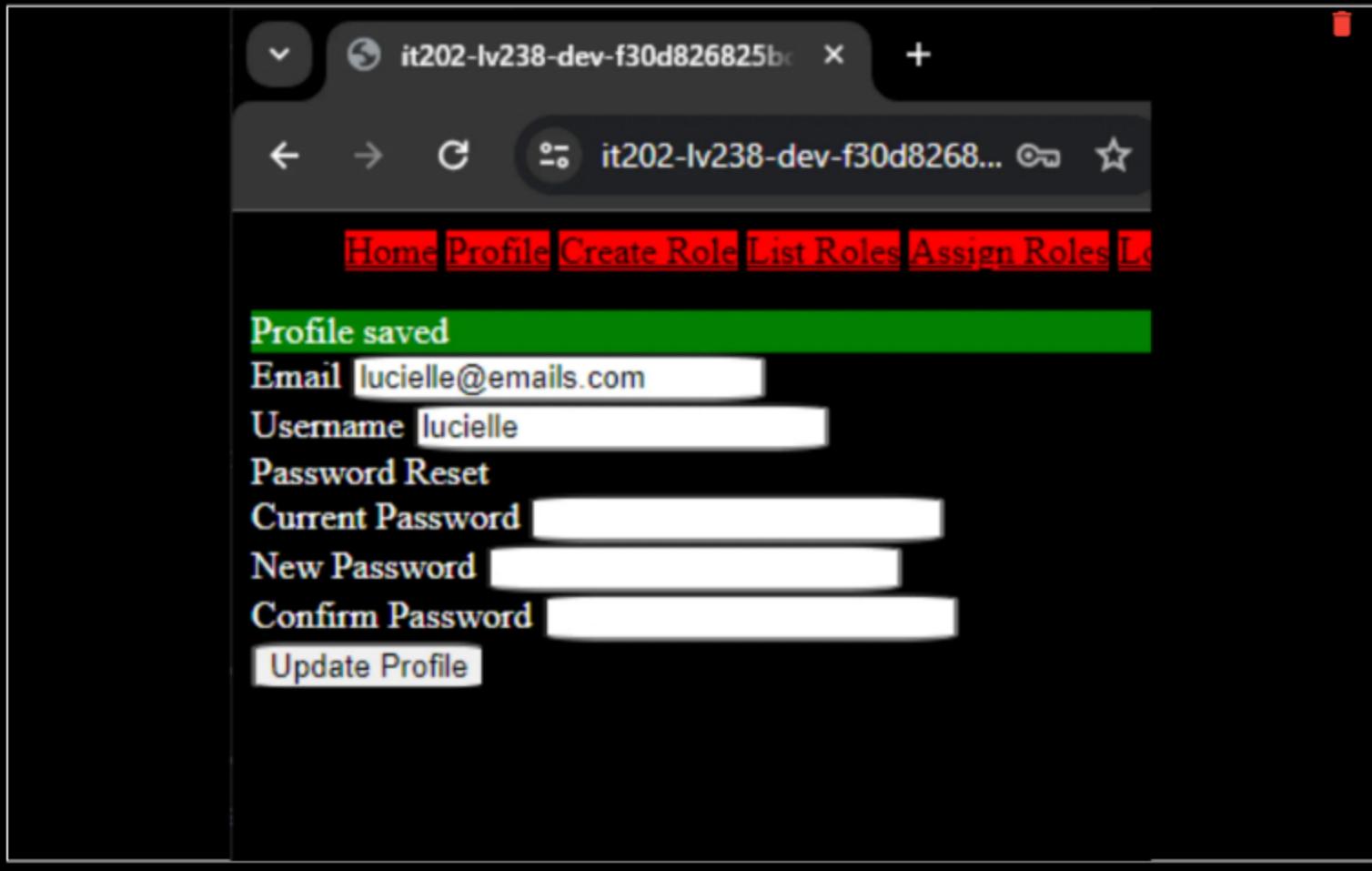
successful password

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



Profile saved

Email lucielle@emails.com

Username lucielle

Password Reset

Current Password

New Password

Confirm Password

Update Profile

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)

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	
<input checked="" type="checkbox"/> #1	1	Updating Username/Email	
<input checked="" type="checkbox"/> #2	1	Updating password	
<input checked="" type="checkbox"/> #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

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

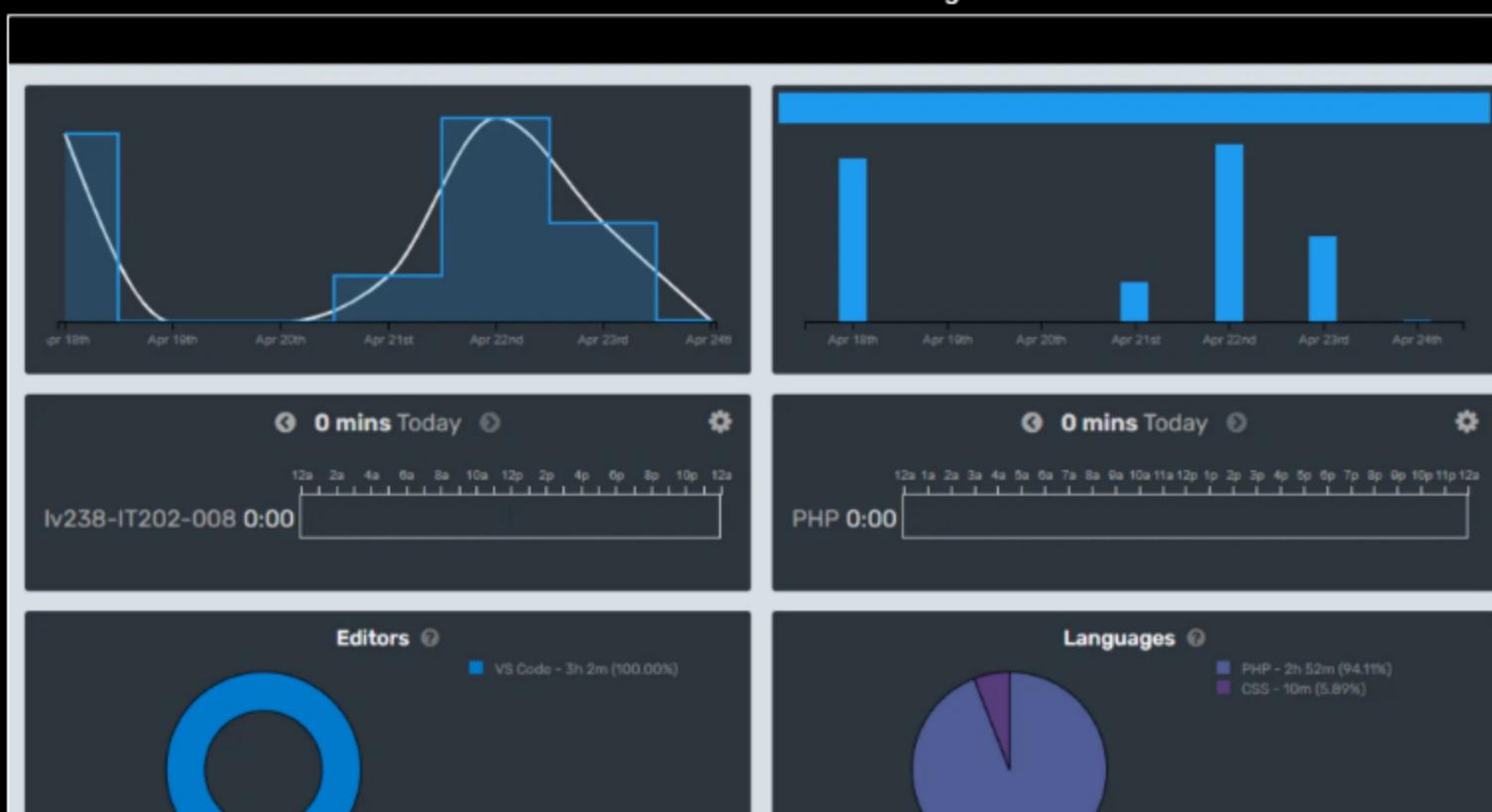
Task Screenshots:

Gallery Style: Large View

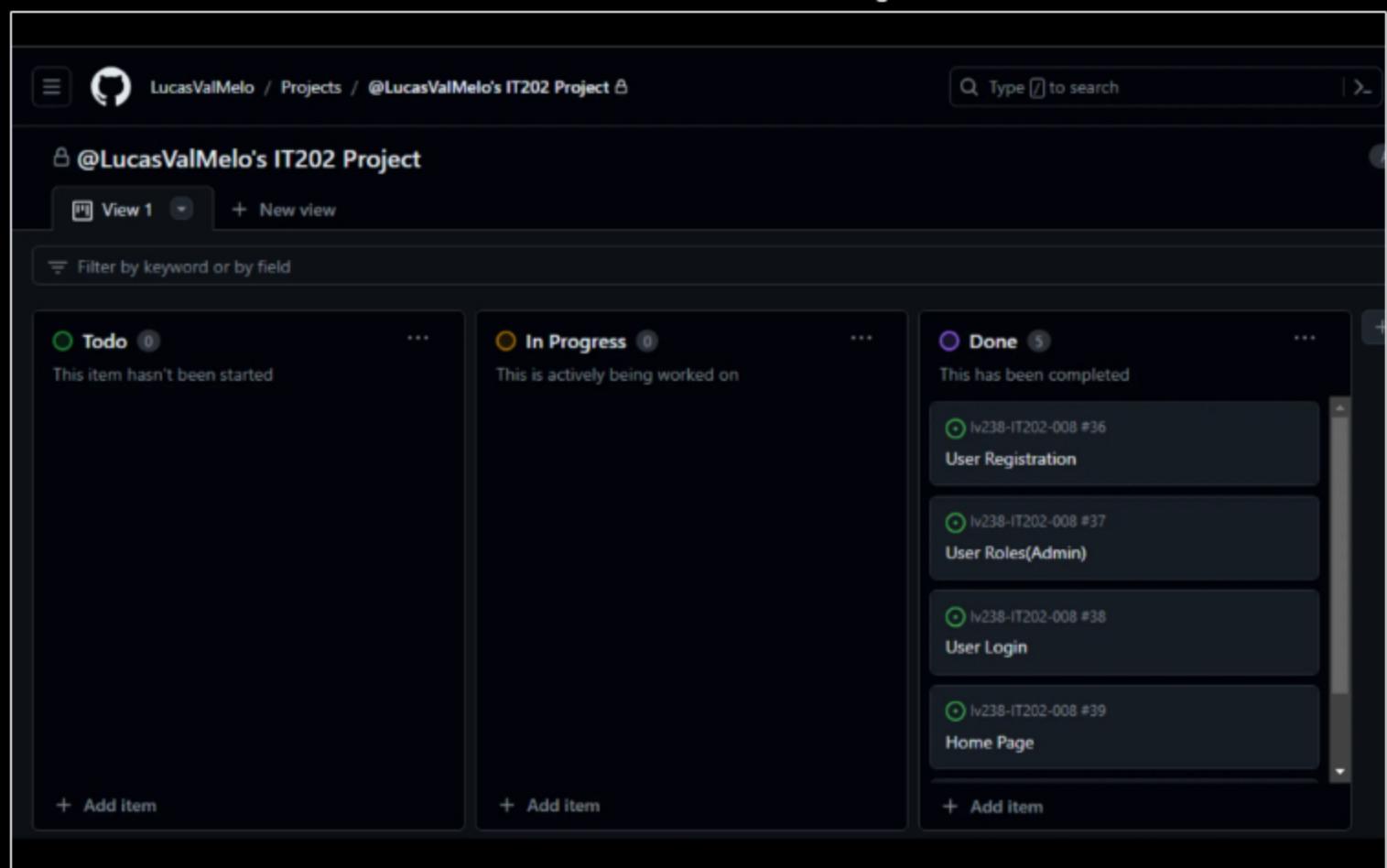
Small

Medium

Large



wakatime ss

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**

The screenshot shows a GitHub project board with three columns: Todo, In Progress, and Done. The Todo column contains one item: "This item hasn't been started". The In Progress column contains one item: "This is actively being worked on". The Done column contains four items: "lv238-IT202-008 #36 User Registration", "lv238-IT202-008 #37 User Roles(Admin)", "lv238-IT202-008 #38 User Login", and "lv238-IT202-008 #39 Home Page". Each item is represented by a card with a green circular icon and a link.

proj board**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