

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/ekh3>

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

Submissions:

Submission Selection

1 Submission [active] 4/1/2024 1:52:32 AM

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 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 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 checked="" type="checkbox"/> #4	1	Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)
<input checked="" type="checkbox"/> #5	1	Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)
<input checked="" type="checkbox"/> #6	1	Demonstrate user-friendly message of new account being created

Task Screenshots:

Gallery Style: Large View

Small Medium Large

The screenshot shows a registration form on a web page. The browser's title bar indicates the URL is `herokuapp.com`. The page has a blue header with 'Login' and 'Register' buttons. Below the header, there are four validation error messages displayed in yellow boxes:

- [Client] Username must only contain 3-30 characters a-z, 0-9, _ or -
- [Client] Email is not valid
- [Client] Password must be 8 characters, have 1 lowercase, 1 number and 1 special character
- [Client] Password and Confirm password must match

The form fields are as follows:

- Email: `dh@`
- Username: `hi`
- Password: `Hope You're Doing Well`
- Confirm: `HelloGrader :)`

A blue 'Register' button is at the bottom of the form.

Demonstrates the JS validation for email, username, and password formats with the matching password requirements.

Checklist Items (1)

#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)

The screenshot shows a registration form on a Heroku application. The URL in the address bar is `ekh3-r202-008-dev-b2eb0af09ac5.herokuapp.com/Project/register.php`. The form has four input fields: Email, Username, Password, and Confirm. Above the Email field, there is an error message: "The chosen email is not available." A blue "Register" button is at the bottom. The background of the page is light orange.

Email in use

Checklist Items (1)

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

The screenshot shows a registration form on a Heroku application. The URL in the address bar is `ekh3-r202-008-dev-b2eb0af09ac5.herokuapp.com/Project/register.php`. The form has four input fields: Email, Username, Password, and Confirm. Above the Email field, there is a success message: "Successfully registered!" A blue "Register" button is at the bottom. The background of the page is light orange.

Register success

Checklist Items (1)

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

The chosen username is not available.

Email

Username

Password

Confirm

Register

Username in use

Checklist Items (1)

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

[Client] Username must only contain a-z, 0-9, _ or -
[Client] Email is not valid
[Client] Password must be 8 characters, have 1 lowercase, 1 number, and 1 special character

Email

Username

Password

Confirm

Register required fields (if blank)

Checklist Items (1)

#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)

Task #2 - Points: 1

Text: Screenshot of the form code

i Details:

Should have appropriate input types for the field

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```
6  <form onsubmit="return validate(this)" method="POST">
7      <div>
8          <label for="email">Email</label>
9          <input type="email" name="email" required />
10     </div>
11     <div>
12         <label for="username">Username</label>
13         <input type="text" name="username" required maxlength="30" />
14     </div>
15     <div>
16         <label for="pw">Password</label>
17         <input type="password" id="pw" name="password" required minlength="8" />
18     </div>
19 
```

```

19         <div>
20             <label for="confirm">Confirm</label>
21             <input type="password" name="confirm" required minlength="8" />
22         </div>
23         <input type="submit" value="Register" />
24     </form>

```

Form code for register

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
<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



```

<script>
    Function validate(form) {
        // [REDACTED] 4/1/24
        // This implements Javascript validation
        // ensure it returns false for an error and true for success

        // exm3 - 4/1/24
        let pw = form.password.value;
        let con = form.confirm.value;
        let isValid = true;

        if (!verifyUsername(form.username.value))
            isValid = false;
        if (!verifyEmail(form.email.value))
            isValid = false;
        if (!verifyPassword(pw))
            isValid = false;

        // validation end

        if (pw != con)
            flash("Client] Password and Confirm password must match", "warning");
        isValid = false;
    }

    return isValid;
}

</script>
</body>
// [REDACTED] 2: add PHP Code
if (isset($_POST['email']) && isset($_POST['password']) && isset($_POST['confirm'])) {
    $email = $_POST['email'];
    $password = $_POST['password'];
    $confirm = $_POST['confirm'];

    $username = $_POST['username'];
    $error = false;
    if (empty($email)) {
        flash("Email must not be empty", "danger");
        $error = true;
    }
}

```

```

25 // exm3 - 4/1/24
26
27 function verifyUsername(user) {
28     let userPattern = /^[a-zA-Z0-9_-]{3,16}$/;
29
30     // Username verification
31     if (!userPattern.test(user) || user == "") {
32         flash("Client] Username must only contain 3-16 characters a-zA-Z0-9_-, or -, "warning");
33         return false;
34     }
35     return true;
36 }

37 function verifyEmail(email) {
38     let emailPattern = /^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z]{2,4}$/;
39
40     // Email verification
41     if (!emailPattern.test(email) || email == "") {
42         flash("Client] Email is not valid", "warning");
43         return false;
44     }
45     return true;
46 }

47 function comparePass(pw, con) {
48
49     if (pw != con) {
50         flash("Client] Password and Confirm password must match", "warning");
51         return false;
52     }
53     return true;
54 }

55 function verifyPassword(pw) {
56     let passwordPattern = /^(?=.*?[a-z])(?=.*?[0-9])(?=.*?[$!@#%^&*])[A-Za-z0-9$!@#%^&*]{8,16}$/;
57
58     if (!passwordPattern.test(pw)) {
59         flash("Client] Password must be 8 characters and have 1 lowercase, 1 number, and 1 special character", "warning");
60         return false;
61     }
62     return true;
63 }

64
65
66
67

```

JS code

Checklist Items (1)

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

Register php code

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)

```
TERMINAL OUTPUT CRASH CONSOLE PROGRAMS PORTS COMMENTS

Delta compression using up to 16 threads
remote: Resolving deltas: 100% (18/18), completed with 7 local objects.
To git@github.com:kylelemons/10-202-M6.git
 4ebe7ed..3866ca6 M51-feat-FinalizeMilestone => M51-feat-FinalizeMile
```

C:\Users\Chair-05\Downloads\y/2k_NCE_3DNC\Computer-21\new\01\DualDepth\Classes\ET200\HVIS-1030-000 (P2)-heat-Final2019Testline

Extra php functions

Checklist Items (1)

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

Task #4 - Points: 1

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

Checklist

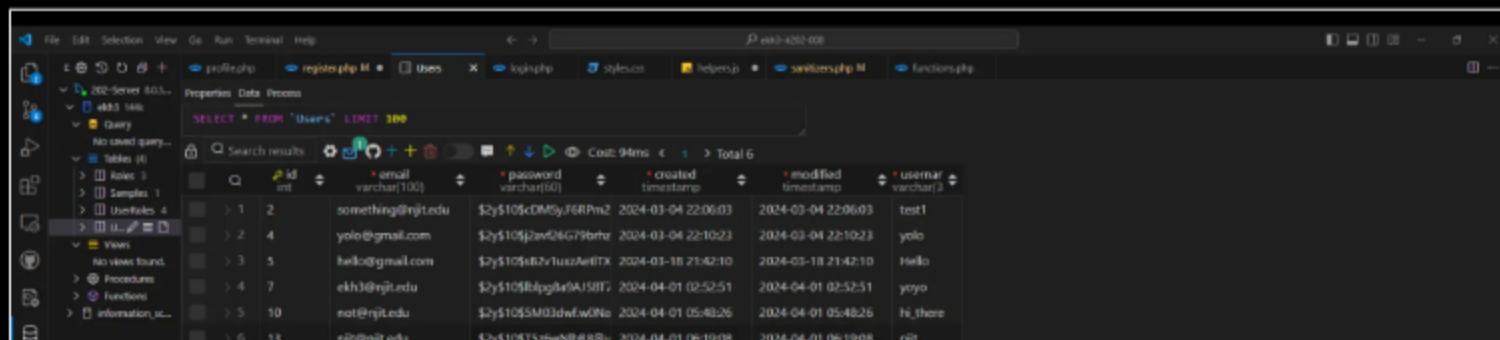
*The checkboxes are for your own tracking

#	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



User table with hashed password, unique values

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:

For registration, the user would be able to set their registration in the register page. Within the register page there would be 4 input box fields in a form which allows the user to put in their information. After entering their information and clicking register, the information from the form would initiate the validate() function in JS. It would try to validate all the patterns for each item, making sure the proper regex exists, using `regex.test(str)`. If a field is not valid, it would trigger the a flash message and not submit as the return would be false. Otherwise it would go through and the JS returns true, the PHP validations would check again stopping the process and alerting them if anything is incorrect. Lastly, it would attempt to submit it to the database and if there is a duplicate value for a unique field, it would alert the user that the username or email is unavailable. Otherwise, the results would be sent to the database and the password is hashed using bcrypt method beforehand.

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

<https://github.com/WokFriedE/ekh3-it202-008/pull/16>

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](#)

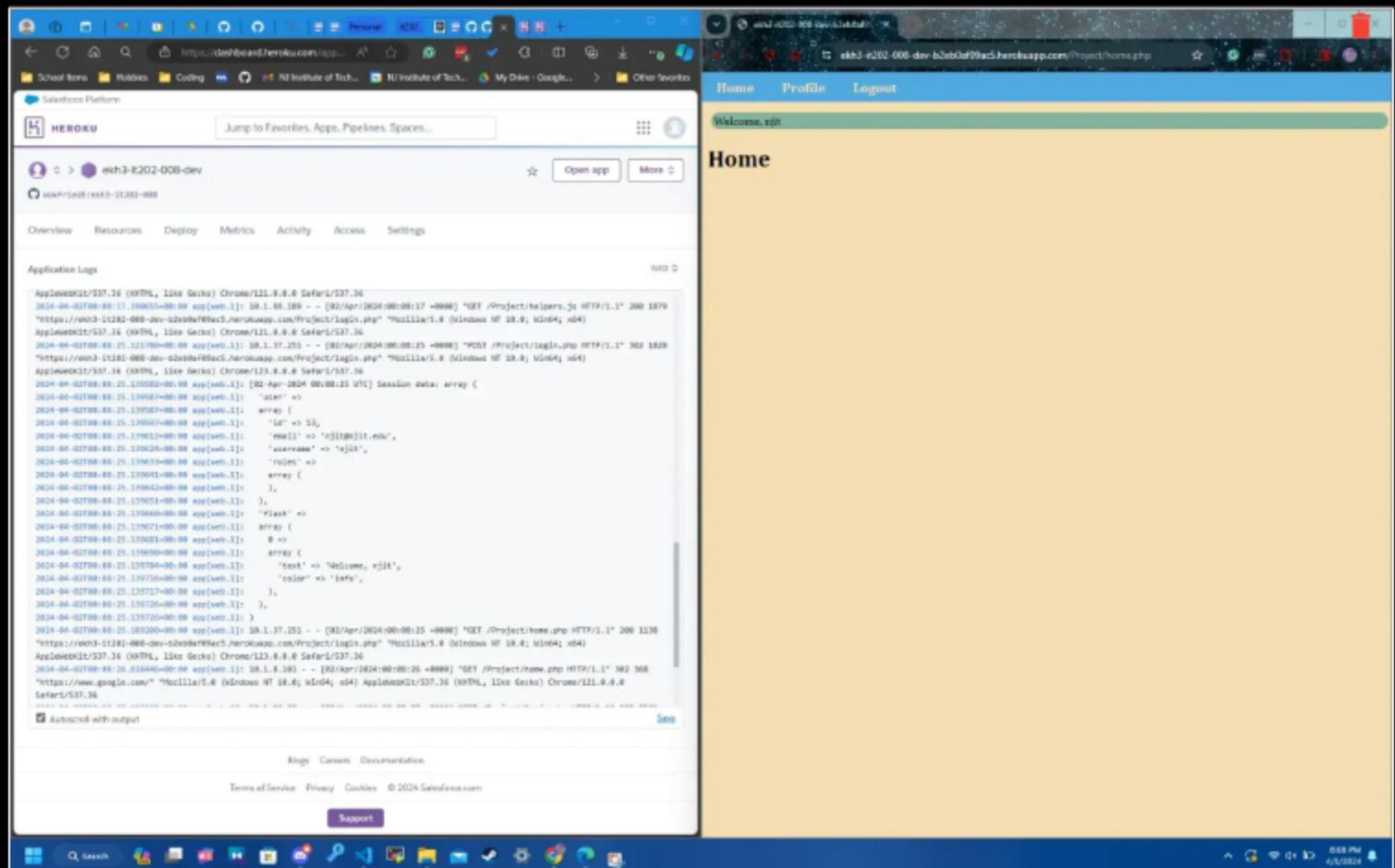
A screenshot of a user login form displayed in a web browser. The browser window title is "ekh3-it02-000-14-15v600". The URL in the address bar is "ekh3-it02-000-dev-b2eb0a91a55herokuapp.com/Project/login.php". The form has a blue header with "Login" and "Register" buttons. Below the header are two input fields: "Email/Username" containing "yoyo@njit.edu" and "Password" containing "mononoli!". At the bottom of the form is a blue "Submit" button.

valid email (admin login)

Checklist Items (2)

#2 Should have thoughtful CSS applied

#4 Include correct data where email is used to login

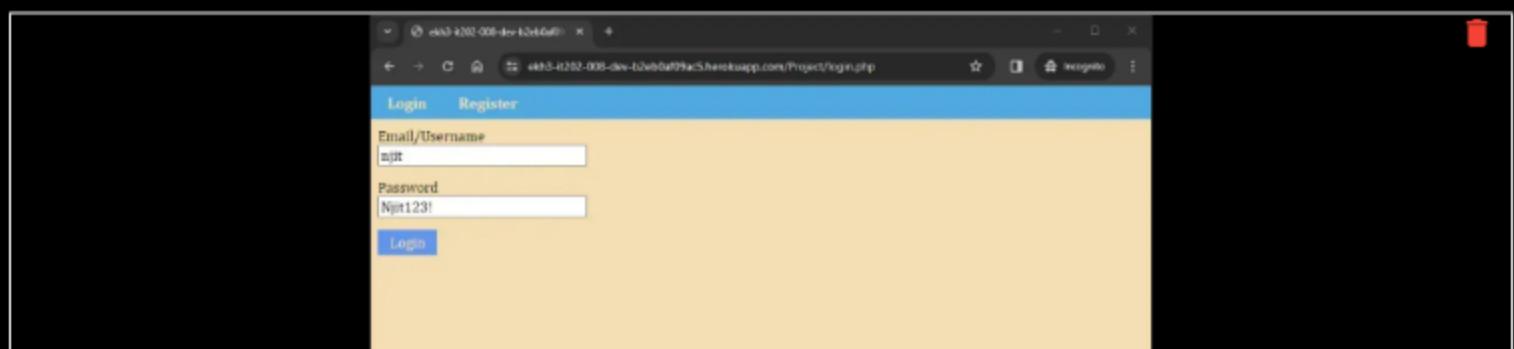


Successful login with the error log output of the session

Checklist Items (2)

#5 Show success login message

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



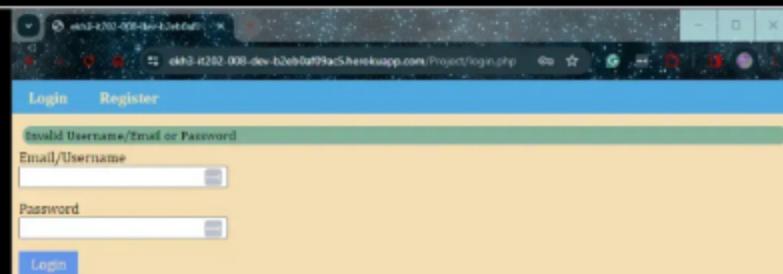
Proper username login (default user login)

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



Message for invalid password and username/email does not exist.

Checklist Items (2)

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

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

The screenshot shows a web browser window with two tabs open. The active tab is titled 'ekh3-it202-008-dev-b2eb0af09w5.herokuapp.com/Project/login.php'. The page has a blue header with 'Login' and 'Register' buttons. Below the header, there are two error messages in yellow boxes: '[Client] Username must only contain 3-30 characters a-z, 0-9, _ or -' and '[Client] Password must be 8 characters, have 1 lowercase, 1 number, and 1 special character'. There are two input fields: 'Email/Username' containing 'random!' and 'Password' containing 'yolo'. A blue 'Login' button is at the bottom.

Validation for format for password and username

Checklist Items (1)

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

The screenshot shows a web browser window with two tabs open. The active tab is titled 'ekh3-it202-008-dev-b2eb0af09w5.herokuapp.com/Project/login.php'. The page has a blue header with 'Login' and 'Register' buttons. Below the header, there are two error messages in yellow boxes: '[Client] Username must only contain 3-30 characters a-z, 0-9, _ or -' and '[Client] Password must be 8 characters, have 1 lowercase, 1 number, and 1 special character'. There are two input fields: 'Email/Username' and 'Password', both of which are currently empty. A blue 'Login' button is at the bottom.

Validation with blank fields (enforces requirement)

Checklist Items (1)

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

Task #2 - Points: 1

Text: Screenshot of the form code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input checked="" 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 checked="" type="checkbox"/> #2	1	Show JavaScript validations (include any extra files related)
<input checked="" type="checkbox"/> #3	1	Show PHP validations (include any lib content)
<input checked="" 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

```
```
<!-- ekh3 - 4/1/24 -->
<form onsubmit="return validate(this)" method="POST">
 <div>
 <label for="email">Email/Username</label>
 <input type="text" name="email" required />
 </div>
 <div>
 <label for="pw">Password</label>
 <input type="password" id="pw" name="password" required minlength="8" />
 </div>
 <input type="submit" value="Login" />
</form>
```

Login form code (used one text input for email and username)

## 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 validateForm() {
 // TODO: Implement JavaScript validation
 // ensure it returns false for an error and true for success

 //vkh3 = 4/1/24
 let cred = form.email.value;
 let isValid = true;

 if (/^(?!.*\.\w{2,3})$/.test(cred)) {
 if (!verifyEmail(cred))
 isValid = false;
 } else {
 if (!verifyUsername(cred))
 isValid = false;
 }

 // TODO: update clientSide validation to check if it should
 // valid email or username
 if (!verifyPassword(form.password.value))
 isValid = false;
 }

 return isValid;
}

// end JS validation
</script>
</pre>
// PHP: add PHP Code
if (isset($_POST["email"]) && isset($_POST["password"])) {
 $email = $_POST["email"];
 $email = filter_var($email, FILTER_SANITIZE_EMAIL);
 $password = $_POST["password"];
 $password = filter_var($password, FILTER_SANITIZE_STRING);

 //vkh3 -
 // TODO: Sanitize
 $hasError = false;
 if (empty($email)) {
 $flash["Email must not be empty"];
 $hasError = true;
 }

 if ($password == "") {
 $flash["Password must be at least 8 characters long"];
 $hasError = true;
 }

 if ($hasError) {
 // Sanitize
 $email = filter_var($email, FILTER_SANITIZE_EMAIL);
 $email = sanitize_email($email);
 }
}

//vkh3 - 4/1/24
function verifyUsername(user) {
 let userPattern = /^[a-zA-Z][a-zA-Z0-9]{3,16}$/;

 // username verification
 if (!userPattern.test(user)) || user == "" {
 flash["[Client] Username must only contain 3-16 characters a-z, 0-9, _, or -, "warning"];
 return false;
 }
 return true;
}

function verifyEmail(email) {
 let emailPattern = /^[^\s@]+@[^\s@]+\.[^\s@]{2,4}$/;

 // email verification
 if (!emailPattern.test(email)) || email == "" {
 flash["[Client] Email is not valid", "warning"];
 return false;
 }
 return true;
}

function comparePass($pw, $cn) {
 if ($pw != $cn) {
 flash["[Client] Password and Confirm password must match", "warning"];
 return false;
 }
 return true;
}

function verifyPassword($pw) {
 let passwordPattern = /^(?=.*[a-zA-Z])(?=.*[0-9])(?=.*[!@#$%^&_])[A-Za-z0-9!@#$%^&_]{8,}$/;

 if (!passwordPattern.test($pw)) {
 flash["[Client] Password must be 8 characters and have 1 lowercase, 1 number, and 1 special character", "warning"];
 return false;
 }
 return true;
}

```

## Log in code - JS validation

## 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)

## PHP verification code - safe\_echo.php is not used

### Checklist Items (1)

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

### 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:**

When the user first enters the web app, they will be directed to the login.php screen where they will be allowed to enter their username/email and password. After entering the information and clicking the login button, it would validate the values. For the username, it would check if the credential has an @ symbol, if so, check the email verification in JS. If it does not have an @, check for the username validation in JS. Afterward, it checks for the password with JS, just like the credentials. If all is right, then the form will be submitted, otherwise, it will prompt flash messages and deny the form submission. If the username or email does not exist it will prompt the user saying "Invalid Username/Email or Password." The same would be done if the password does not match the credential. If the submission is successful, then PHP will do validations using regex, similar to JS. If anything messes up, a flash message will be prompted. If everything validates, a session will be created for the nav.php to call in other pages.

The cookie and session will stay in the local cache, working as an associative array where the user information is stored. It is being used so that other pages, like the profile, can use the information from the login page. In addition, using the roles tables set, it would assign the role to the user to change their view accordingly. The way it works is that when the page opens, the navbar would pull the session cookie. The PHP on each page will use that session key to get information and change accordingly.

^COLLAPSE ^

## Task #4 - Points: 1

Text: Include pull request links related to this feature

### Details:

Should end in /pull/#

#### URL #1

<https://github.com/WokFriedE/ekh3-it202-008/pull/24>



### User Logout (1 pt.)

^COLLAPSE ^



## Task #1 - Points: 1

Text: Capture the following screenshots

### Checklist

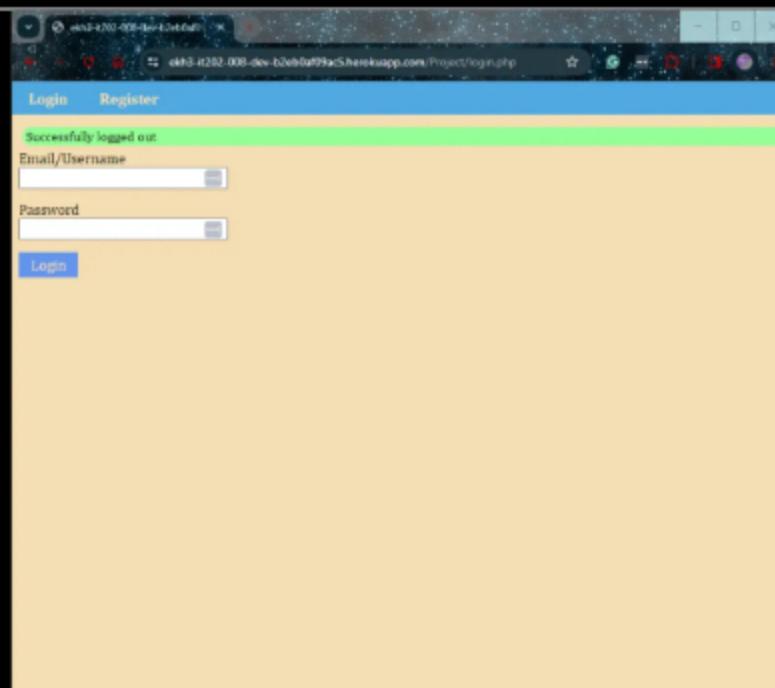
\*The checkboxes are for your own tracking

#	Points	Details
<input checked="" 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

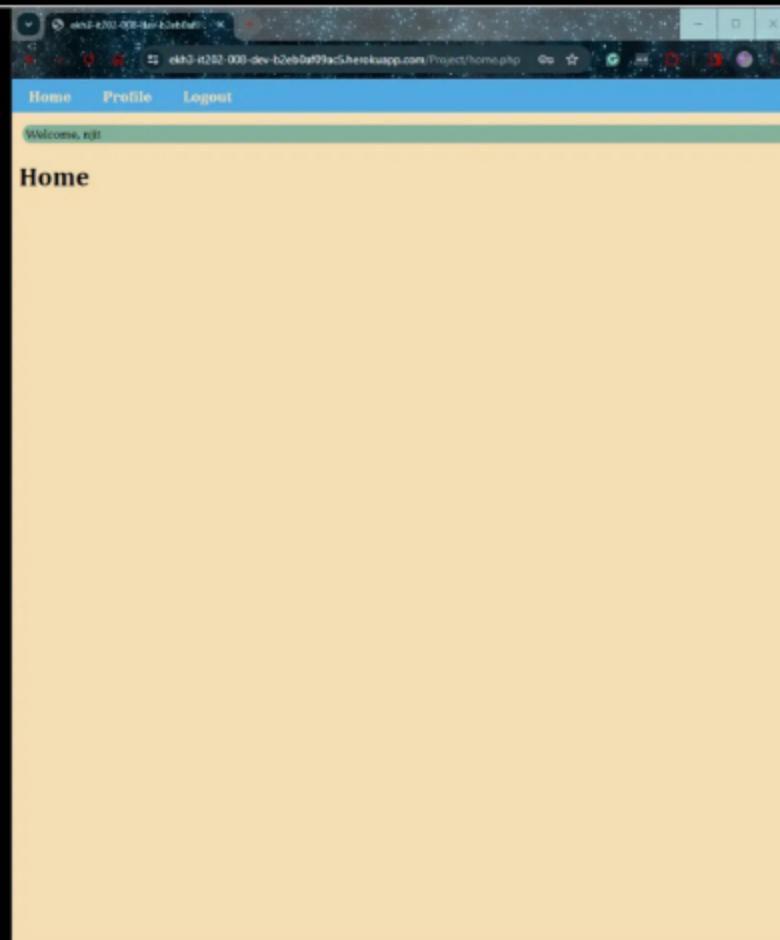
Small      Medium      Large



## Successful log out and redirect

### Checklist Items (1)

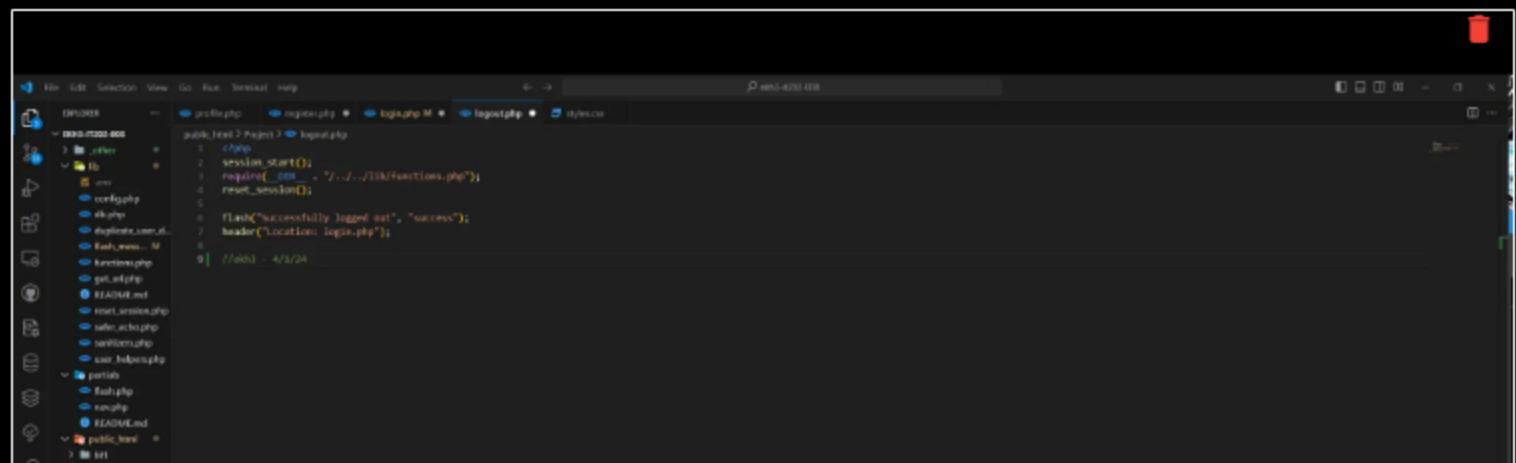
#2 Screenshot of the redirect to login with the user-friendly logged-out message (site)



## Logged in user nav - default user

### Checklist Items (1)

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



File Explorer Screenshot:

- Project
- admin
  - assign\_respc...
  - create\_role.php
  - edit\_role.php
  - index.php
  - login.php M
  - logout.php**
  - profile.php
  - README.md
  - register.php
  - styles.css
  - challenge.php
  - index.php
  - README.md
  - test01.php
  - upgrade
  - access
  - composer.json
- outline
- timeline

Log out code

### Checklist Items (1)

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

#### Task #2 - Points: 1

Text: Include pull request links related to this feature

#### i Details:

Should end in /pull/#

### URL #1

<https://github.com/WokFriedE/ekh3-it202-008/pull/14>

#### 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 checked="" type="checkbox"/> #1	1	Screenshot of the function that checks if a user is logged in
<input checked="" 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 checked="" type="checkbox"/> #3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)
<input checked="" 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

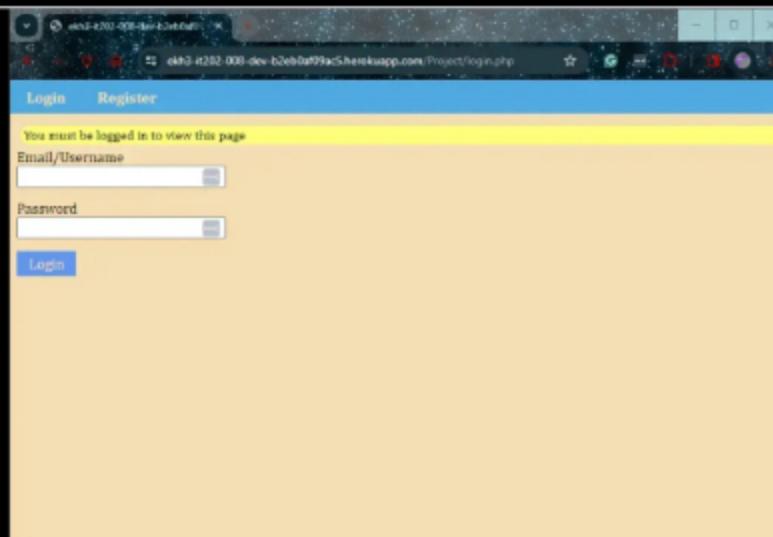
**Code for is\_logged\_in() function (right) & use in profile (left) Used in profile.php and home.php**

### Checklist Items (3)

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

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

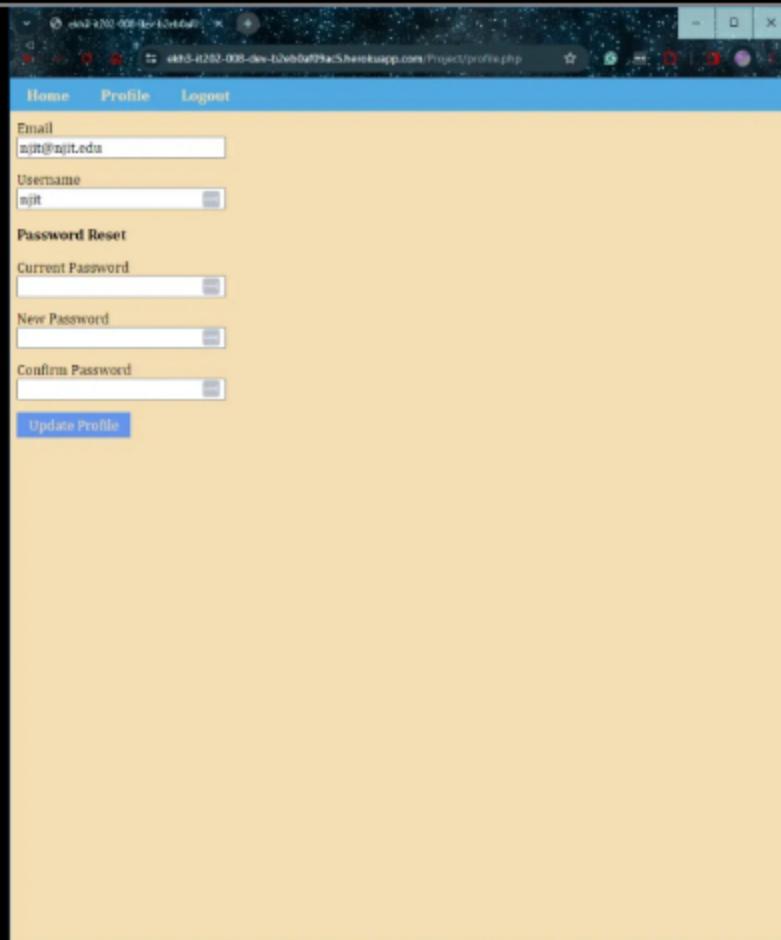
#3 Include ucid/date code comments in all screenshots (one per screenshot is sufficient)



trying ./profile.php without being logged in

#### Checklist Items (1)

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



Screenshot of profile - Login default user

#### Checklist Items (0)

Task #2 - Points: 1

Text: Authorization Screenshots

#### Checklist

\*The checkboxes are for your own tracking

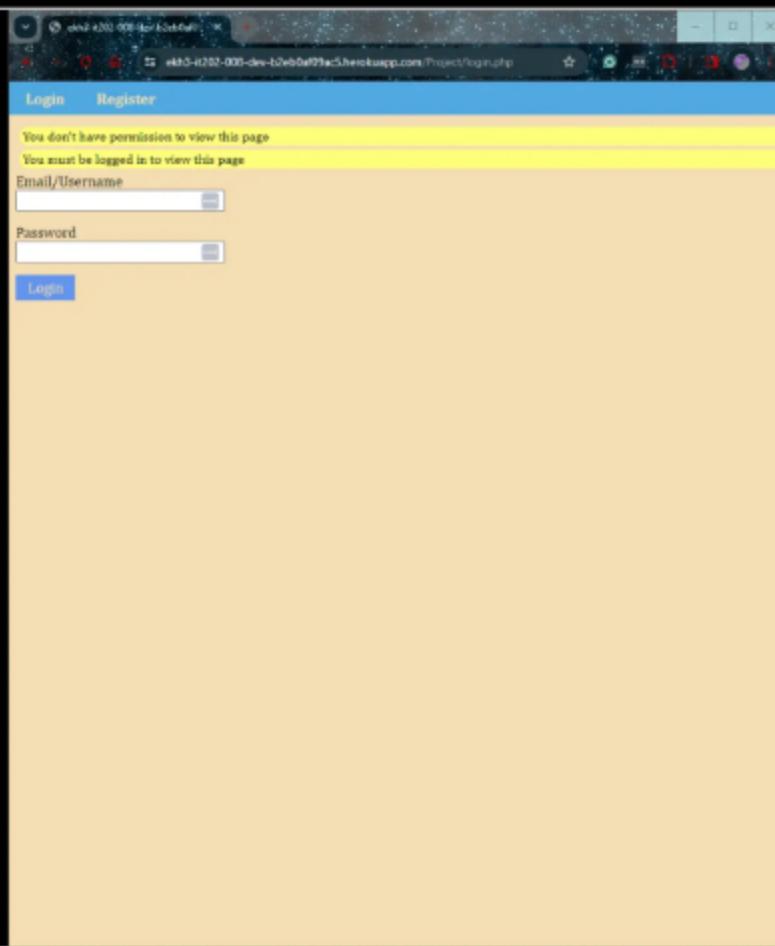
#	Points	Details
<input type="checkbox"/> #1	1	Screenshot of the function that checks for a specific role

<input type="checkbox"/> #1	1	Screenshot of the function that checks for a specific role
<input type="checkbox"/> #2	1	Screenshot of the role check function being used. 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 role-protected page while being logged out

## Task Screenshots:

### Gallery Style: Large View

**Small              Medium              Large**



trying to go to ./admin/list\_roles.php without logged in

### Checklist Items (1)

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

```

22 flash("Successfully created role '$name', 'success'");
23 } catch (PDOException $e) {
24 if ($e->errorInfo[1] == 1062) {
25 flash("A role with this name already exists, please try another", "warning");
26 } else {
27 flash(var_export($e->errorInfo, true), "danger");
28 }
29 }
30 }
31 </div>
32 </div>
33 </div>
34 </div>
35 </div>
36 </div>
37 </div>
38 </div>
39 </div>
40 </div>
41 </div>
42 </div>
43 </div>
44 </div>
45 </div>
46 </div>
47 //we need to go up 1 more directory
48 require_once(__DIR__ . "/../../../../partials/flash.php");
49 </div>

```

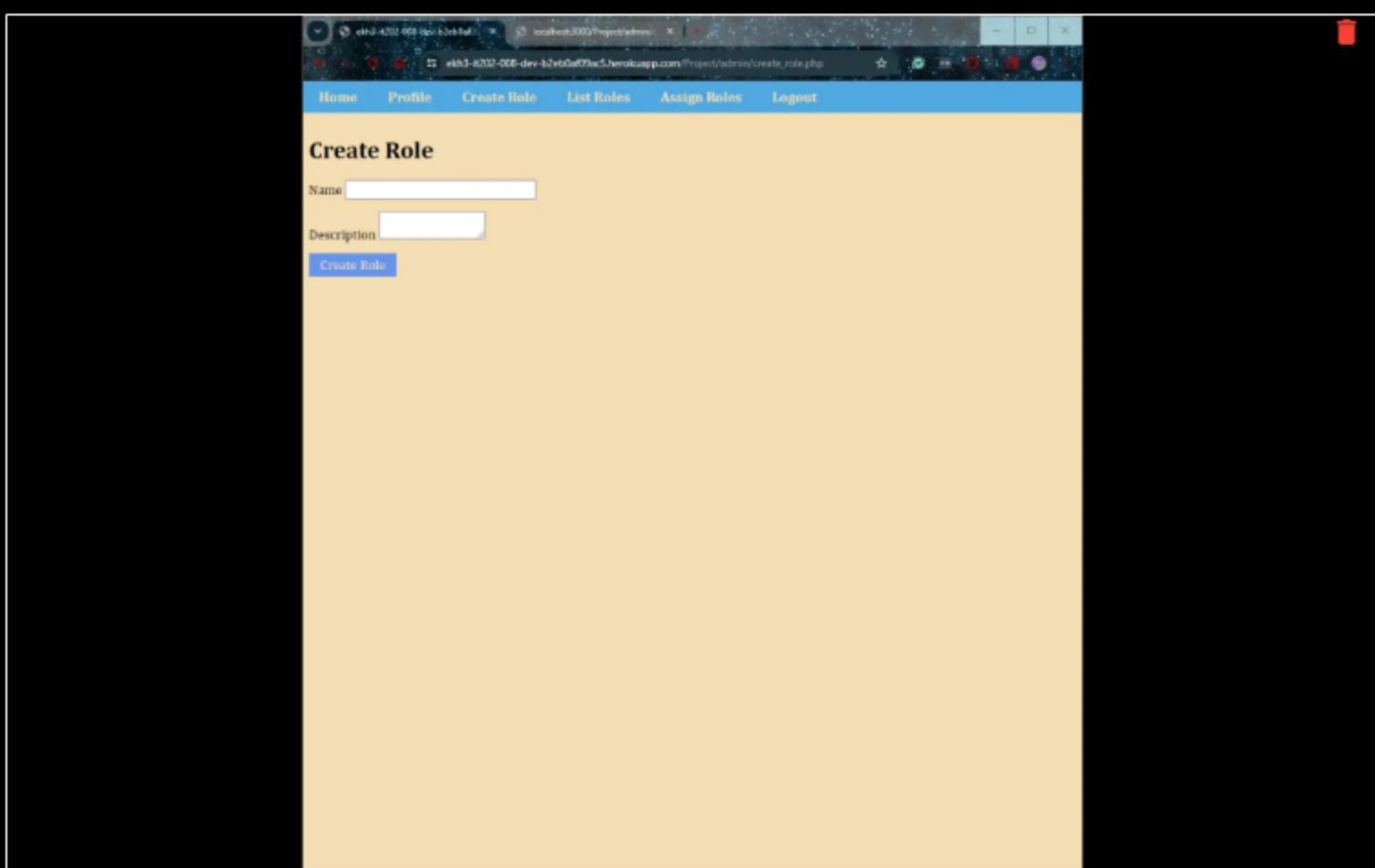
Function for has\_role() on right; on left, create\_role.php, assign\_roles.php, and create\_role.php use has\_role("Admin")

### Checklist Items (3)

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

#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)



Screenshot showing has\_role() in action on site

### Checklist Items (1)

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



COLLAPSE

## Task #3 - Points: 1

**Text: Screenshots of UserRoles and Roles Tables**

### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
<input checked="" type="checkbox"/> #1	1	At least one valid and enabled User->Role reference (UserRoles table)
<input checked="" type="checkbox"/> #2	1	UserRoles Table should have id, user_id, role_id, is_active, created, and modified columns
<input checked="" type="checkbox"/> #3	1	Roles Table should have id, name, description, is_active, modified, and created columns
<input checked="" type="checkbox"/> #4	1	At least one valid and enabled Role (Roles table)
<input checked="" type="checkbox"/> #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

The screenshot shows a database interface with two main panes. The left pane displays the 'UserRoles' table with the following data:

ID	User ID	Role ID	Is Active	Created	Modified
1	5	-1	1	2024-03-20 17:47:50	2024-03-20 17:47:53
2	2	4	0	2024-03-20 18:18:20	2024-04-01 02:51:13
3	4	4	0	2024-04-01 02:51:22	2024-04-01 02:51:13
4	8	7	1	2024-04-01 02:53:31	2024-04-01 02:53:31

The right pane displays the 'Roles' table with the following data:

ID	Name	Description	Is Active	Created	Modified
1	Admin	Administrator role	1	2024-03-28 17:02:01	2024-03-28 18:10:15
2	Cool	Cool role here	1	2024-03-28 18:16:36	2024-04-01 02:28:29
3	Ruler	He's a ruler for a ruler	0	2024-04-01 02:32:13	2024-04-01 02:28:29

Below the panes, a terminal window shows command-line logs:

```

Mon Apr 1 23:06:10 2024] [1:1]:574000 Closing
Mon Apr 1 23:06:10 2024] [1:1]:574000 Closing
Mon Apr 1 23:06:10 2024] [1:1]:574000 Accepted
Mon Apr 1 23:06:10 2024] [1:1]:574000 [load]: GET /project/index/create_role.php
Mon Apr 1 23:06:10 2024] [1:1]:574000 Closing
Mon Apr 1 23:06:10 2024] [1:1]:574000 Accepted
Mon Apr 1 23:06:10 2024] [1:1]:574000 [load]: GET /project/styles.css
Mon Apr 1 23:06:10 2024] [1:1]:574000 Closing
Mon Apr 1 23:06:10 2024] [1:1]:574000 Accepted
Mon Apr 1 23:06:10 2024] [1:1]:574000 [load]: GET /project/helpers.js
Mon Apr 1 23:06:10 2024] [1:1]:574000 Closing

```

UserRoles table (left), Roles table (right) - Screenshot

### Checklist Items (5)

- #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

#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)

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:

UserRoles table is used to build the relation of the Users from its respective table to the Roles table. As multiple users can have multiple roles that can be toggled, having it as its relational table would be easier to manage where the ID of the user is related to the ID of the role. In addition, using IDs instead of names allows for the roles to be edited without needing to alter the entire system.

As people can toggle their roles, it's easier to soft delete roles by disabling the relation of the user and the role, making it inactive; this would act as a UserRoles.is\_active. In contrast the Roles.is\_active would disable the role entirely, so regardless of the active status for a user, the role cannot be used; thus Roles.is\_active is a macro switch for a role while UserRoles.is\_active is per user and role.

Task #5 - Points: 1

Text: Include pull request links related to this feature

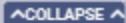
 Details:

Should end in /pull/#

URL #1

<https://github.com/WokFriedE/ekh3-it202-008/pull/24>

 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 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	Show the profile form correctly populated on page load (username, email)
<input checked="" type="checkbox"/> #4	1	Should have the following fields: username, email, current password, new password, confirm password (or similar)

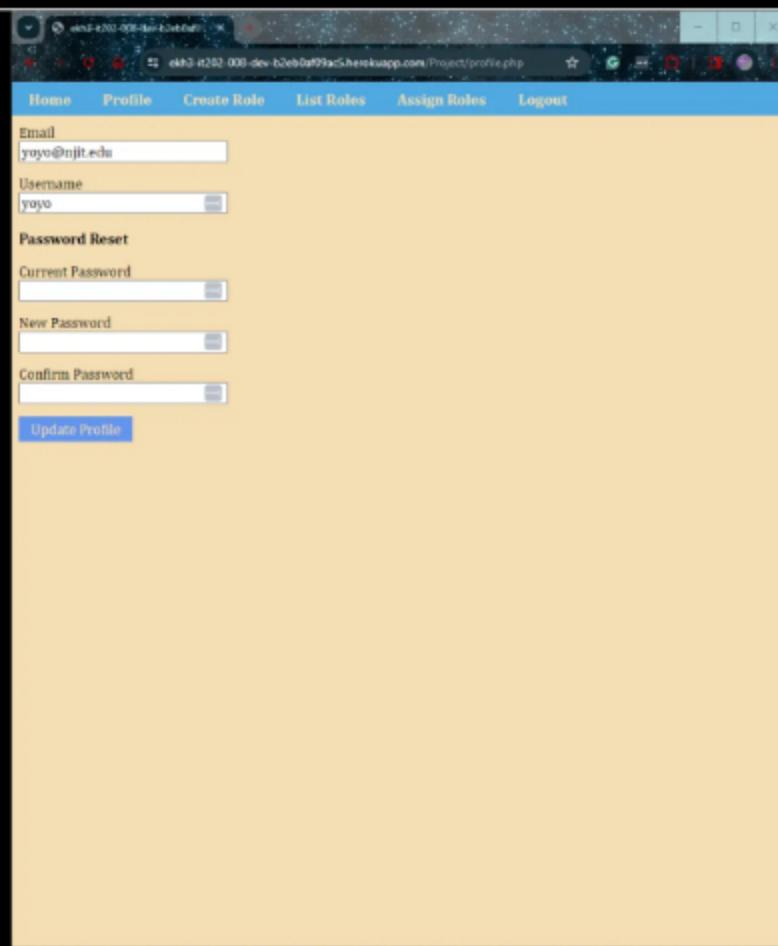
### Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



Profile screenshot with fields and auto populated items

### 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

#### i Details:

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

Response:

When the user clicks on the profile page, it would check if the user is logged in using the `is_logged_in()` function from a PHP helper function. It would check if "user" is set to see if someone is logged in. On load, the form would generate with the values from `$email` and `$username` in their respective fields. The result is that they stay "sticky."

### Task #3 - Points: 1

Text: Edit 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	Demonstrate with before and after of a username change (including success message)
<input type="checkbox"/> #4	1	Demonstrate with a before and after of an email change (including success message)
<input type="checkbox"/> #5	1	Demonstrate the success message of updating password
<input type="checkbox"/> #6	1	Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)
<input type="checkbox"/> #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)

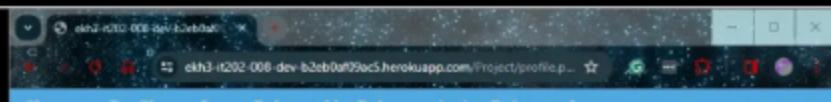
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



Home   Profile   Create Role   List Roles   Assign Roles   Logout

Email  
yoyo@njit.edu

Username  
yoyo

Password Reset

Current Password

New Password

Confirm Password

Profile before updates (both were reset to this before further updates made)

#### 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)

Profile saved

Home   Profile   Create Role   List Roles   Assign Roles   Logout

Email  
yoyo1@njit.edu

Username  
yoyo

Password Reset

Current Password

New Password

Confirm Password

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

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

Profile saved

Email  
yoyo@njit.edu

Username  
yoyo1

Password Reset

Current Password

New Password

Confirm Password

Update Profile

Profile after username change (yoyo -> yoyo1)

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

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

Profile saved

Password reset

Email  
yoyo@njit.edu

Username  
yoyo

Password Reset

Current Password

New Password

Confirm Password

### Profile after password change

#### 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

A screenshot of a web browser window titled "ekh3-it202-008-dev-b2eb0af29ac5.herokuapp.com/Project/profile.php". The browser has a blue header bar with tabs and a toolbar. The main content area shows a profile update form with several validation error messages displayed above the input fields:

- [Client] Username must only contain 3-30 characters a-z, 0-9, ., or -
- [Client] Email is not valid
- [Client] Password must be 8 characters, have 1 lowercase, 1 number, and 1 special character
- [Client] Password and Confirm password must match

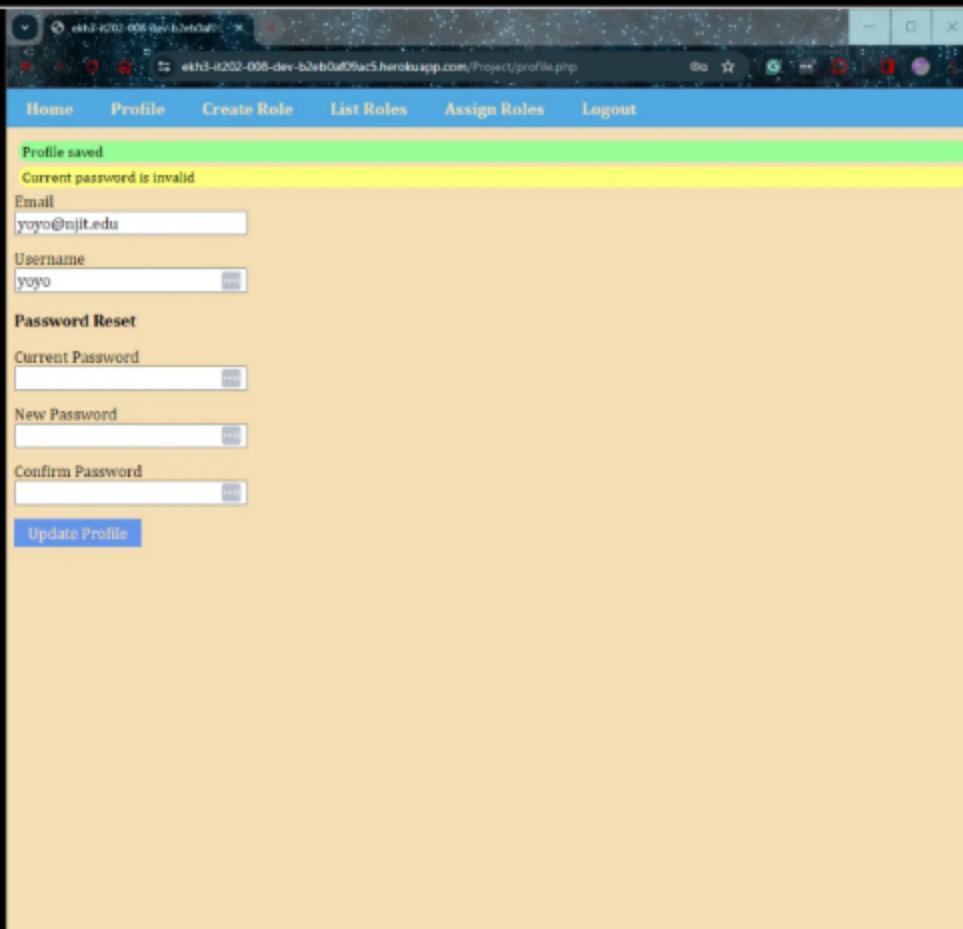
The form fields include:

- Email:
- Username:
- Current Password:
- New Password:
- Confirm Password:

### Profile JS validation

#### Checklist Items (1)

## #6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)

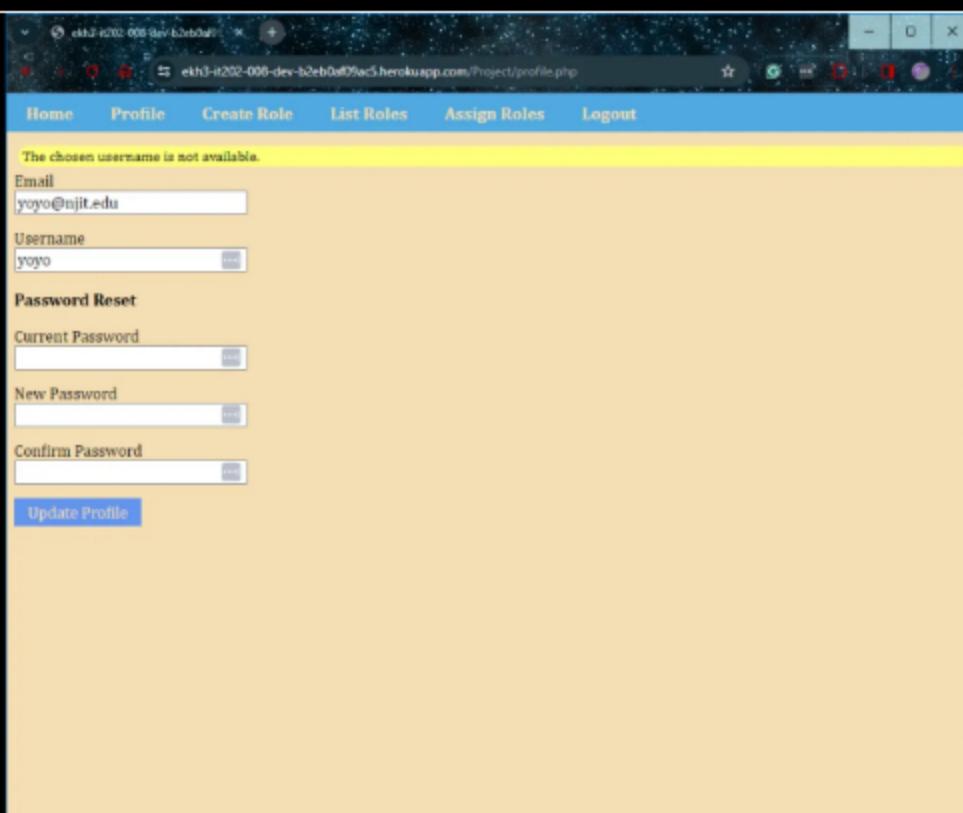


The screenshot shows a web browser window with a profile update form. At the top, there is a navigation bar with links: Home, Profile, Create Role, List Roles, Assign Roles, and Logout. Below the navigation bar, a green banner displays the message "Profile saved". A yellow banner below it shows a validation error: "Current password is invalid". The form contains fields for Email (yoyo@njit.edu), Username (yoyo), and three password fields: Current Password, New Password, and Confirm Password. A blue "Update Profile" button is at the bottom.

PHP current password invalid

## Checklist Items (1)

### #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)

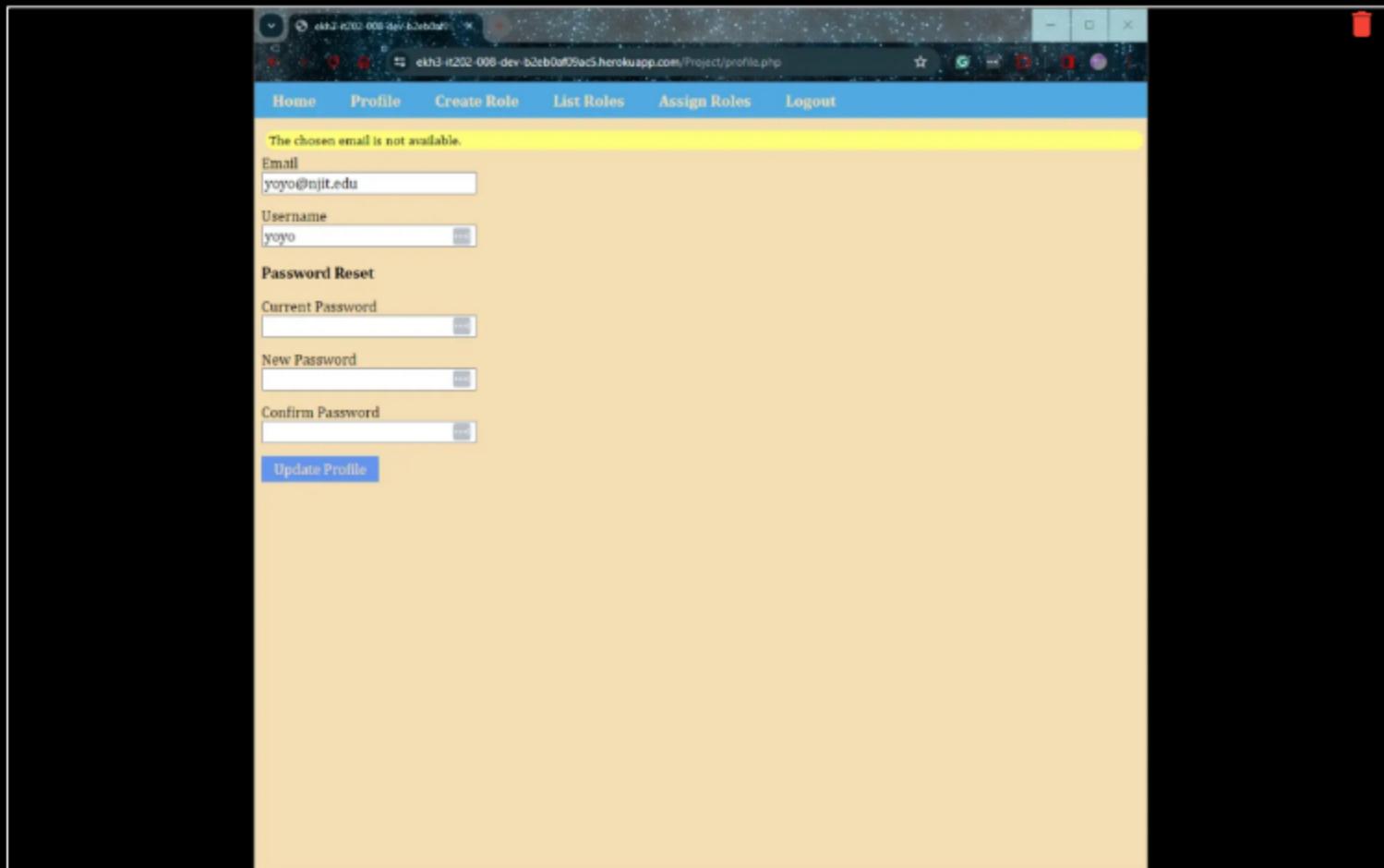


The screenshot shows a web browser window with a profile update form. At the top, there is a navigation bar with links: Home, Profile, Create Role, List Roles, Assign Roles, and Logout. A yellow banner at the top displays the message "The chosen username is not available.". The form contains fields for Email (yoyo@njit.edu), Username (yoyo), and three password fields: Current Password, New Password, and Confirm Password. A blue "Update Profile" button is at the bottom.

## Profile - PHP invalid username (in use)

### **Checklist Items (1)**

#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)



## Profile - invalid email (in use)

### **Checklist Items (1)**

#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)

```

149 if (pw != con) {
150 flash("[Client] Password and Confirm password must match", "warning");
151 isValid = false;
152 }
153 return isValid;
154 }
155 </script>
156 </?php
157 require_once(__DIR__ . "/../../partials/_flash.php");
158 ?>

```

```

22 flash("[Client] Passwords and confirm password must match", "warning");
23 return false;
24 }
25 }
26
27 function verifyPassword(pw) {
28 let passwdPattern = /^(?=.*[a-z])(?=.*\d)(?=.*[$!%#@&])[A-Za-z\d$!%#@&]{8,}$/;
29 if (!passwdPattern.test(pw)) {
30 flash("[Client] Password must be 8 characters, have 1 lowercase, 1 number, and 1 special character", "warning");
31 return false;
32 }
33 return true;
34 }

```

### Profile validation code for JS (if needed)

#### Checklist Items (0)

##### 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:

For updating the email and username, the process is roughly the same. When the user enters the profile page the current values are presented. If edited, and there is nothing in the password fields, there would be a JS validation for the username and/or email using JS regex.test(). If everything seems in check it would submit to the PHP side, otherwise it would send flashes and prevent submission. The PHP would handle its validations with a similar regex and then pass it to an update query. If there is a duplicate error for either, there would be a caught exception that would be sent to the users\_check\_duplicate() exception statement and provide the user with a flash that it cannot be used. It would then update the saved session info so it persists.

For the password, it would check if the current password field is filled, otherwise, it would ignore the rest or prompt the user to fill in the current password if the rest of the password fields are filled. If all three fields are filled, then the confirmation will be checked with the new password to make sure they match. If they are not, then prompt the user using a flash, otherwise send it to PHP to validate. The PHP would then check if the current password matches the hash it has stored, if it does it makes the change. If it does not match the stored password it would alert the user.

##### Task #5 - Points: 1

**Text:** Include pull request links related to this feature

#### Details:

Should end in /pull/#

URL #1

<https://github.com/WokFriedE/ekh3-it202-008/pull/20>

**Misc (1 pt.)**

ACOLLAPSE ^

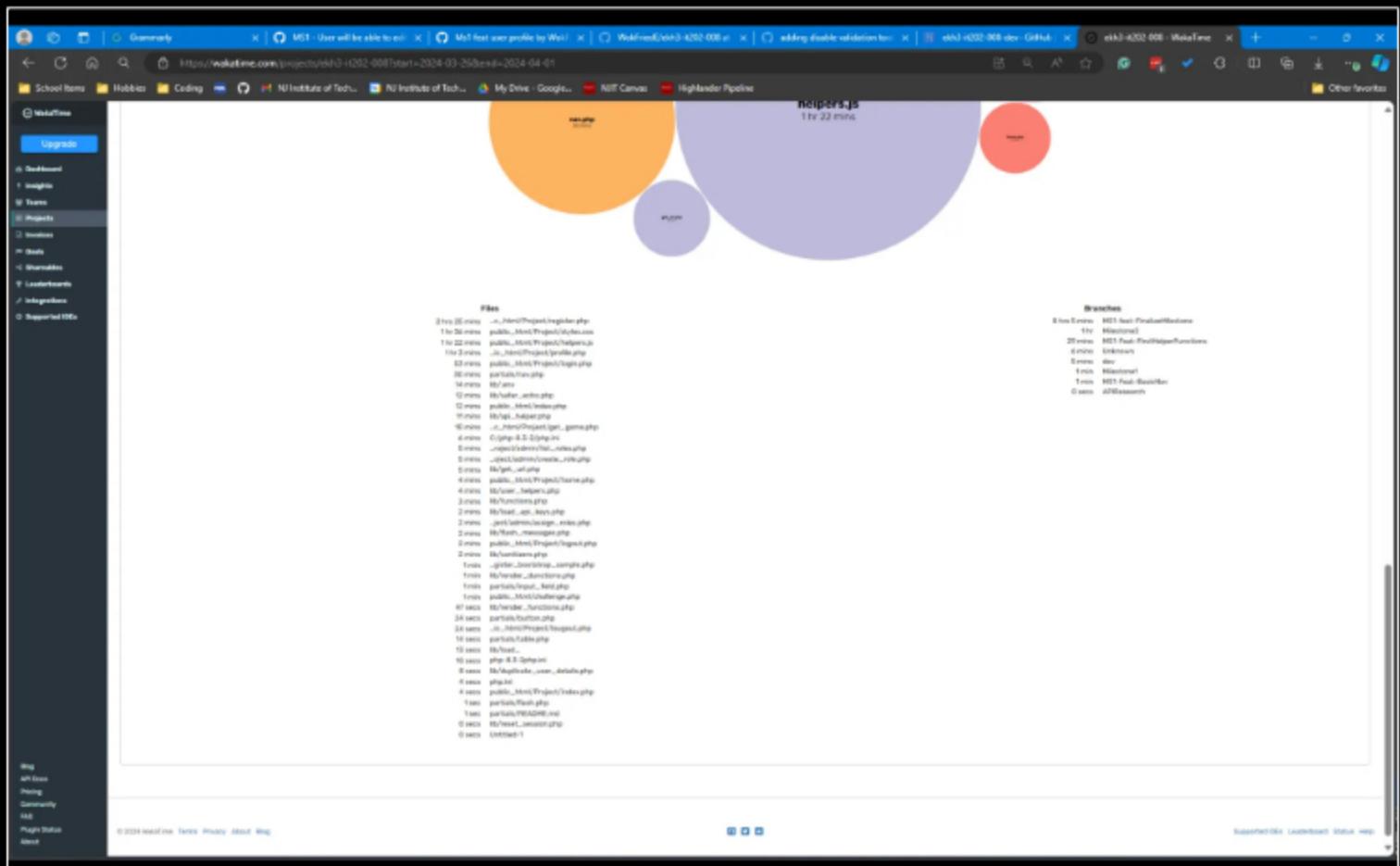
### Task #1 - Points: 1

## Text: Screenshot of wakatime

## Task Screenshots:

### Gallery Style: Large View

**Small      Medium      Large**



Wakatime (pardon the time, left my computer on)

### 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 has one item: 'This items hasn't been started'. The 'In Progress' column has one item: 'This is actively being worked on'. The 'Done' column contains seven items, each with a circular status indicator:

- WokFriedE - Ethan IT202 Application #52 MS1 - Any output messages/errors should be "user friendly" [MS1]
- WokFriedE - Ethan IT202 Application #31 MS1 - Site should have basic styles/theme applied; everything should be styled [MS1]
- WokFriedE - Ethan IT202 Application #26 User can logout [MS1]
- WokFriedE - Ethan IT202 Application #30 MS1 - Basic Roles implemented [MS1]
- WokFriedE - Ethan IT202 Application #27 MS1 - User will be able to register a new account [MS1]
- WokFriedE - Ethan IT202 Application #34 MS1 - User will be able to edit their profile [MS1]
- WokFriedE - Ethan IT202 Application #33 [MS1]

At the bottom of each column are '+ Add Item' buttons.

### Github project board

#### Task #3 - Points: 1

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

**URL #1**

<https://github.com/users/WokFriedE/projects/5/views/1>

#### Task #4 - Points: 1

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

**URL #1**

<https://ekh3-it202-008-prod-e9bbcd10cf36.herokuapp.com/Project/login.php>

End of Assignment