

# Submission Worksheet

**CLICK TO GRADE**

<https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/njz5>

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

## Submissions:

Submission Selection

1 Submission [active] 4/1/2024 7:38:55 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 ▲**



ACOLLAPSE A

### Task #1 - Points: 1

## **Text: Screenshot of form on website page**

## Checklist

**\*The checkboxes are for your own tracking**

#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required)
#4	1	Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)
#5	1	Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)
#6	1	Demonstrate user-friendly message of new account being created

## Task Screenshots:

### Gallery Style: Large View

**Small      Medium      Large**

Client] Email must not be empty  
Client] Invalid email address  
Client] Username must not be empty  
Client] Invalid username  
Client] Password must not be empty  
Client] Password too short  
Client] Confirm password must not be empty  
Client] Confirming password too short

Email

Username

Password

Confirm

## Empty registration.

### **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)

The screenshot shows a web browser window with the URL <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/register.php>. The page has a teal header with 'Login Register' buttons. Below the header is a yellow error message: 'Email | Password and Confirm do not match'. The form fields are as follows:

Email	lakwmdlkwmad@alwkdm.co
Username	lakwmdlkwmwdlkam
Password	password
Confirm	wordpass

A 'Register' button is at the bottom left. At the bottom of the page, there is a red box containing the text 'Passwords do not match.'

#### 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)

The screenshot shows a web browser window with the URL <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/register.php>. The page has a teal header with 'Login Register' buttons. Below the header is a yellow error message: 'Email | Password and Confirm do not match'. The form fields are as follows:

Email	lakwmdlkwmad@alwkdm.co
Username	lakwmdlkwmwdlkam
Password	password

A 'Register' button is at the bottom left.

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)

A screenshot of a web browser window displaying a registration form. The URL in the address bar is <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/register.php>. The page has a teal header with 'Login Register' buttons. Below the header is a yellow horizontal bar containing the text 'Email is already in use. Please choose another.' The registration form includes fields for Email, Username, Password, Confirm, and a Register button. The 'Email' field is highlighted with a red border, indicating it is the current focus or has an error.

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)

A screenshot of a web browser window displaying a registration form. The URL in the address bar is <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/register.php>. The page has a teal header with 'Login Register' buttons. Below the header is a yellow horizontal bar containing the text 'Username is already in use. Please choose another.' The registration form includes fields for Email, Username, Password, Confirm, and a Register button. The 'Username' field is highlighted with a red border, indicating it is the current focus or has an error.

Login Register

Successfully registered!

Email  
Username  
Password  
Confirm  
Register

Successfully registered.

### Checklist Items (1)

#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 field

### Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

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

```

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

```

Registration form code.

### 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

```

23 <script>
24   function validate(form) {
25     //NICHOLAS ZUBRZYCKI mjt5 4/1/2024    You, 23 minutes ago + Uncommitted chan
26     //TODO: Implement JavaScript validation
27     //ensure it returns false for an error and true for success
28     let email = form.email.value;
29     let username = form.username.value;
30     let password = form.pw.value;
31     let confirm = form.confirm.value;
32     let hasError = false;
33
34     if(email.length === 0){
35       flash("[Client] Email must not be empty", "warning");
36       hasError = true;
37     }
38     email = sanitize_email(email);
39     if(!is_valid_email(email)){
40       flash("[Client] Invalid email address", "warning");
41       hasError = true;
42     }
43     if(username.length === 0){
44       flash("[Client] Username must not be empty", "warning");
45       hasError = true;
46     }
47     if(!is_valid_username(username)){
48       flash("[Client] Invalid username", "warning");
49       hasError = true;
50     }
51     if(password.length === 0){
52       flash("[Client] Password must not be empty", "warning");
53       hasError = true;
54     }
55     if(!is_valid_password(password)){
56       flash("[Client] Password too short", "warning");
57       hasError = true;
58     }
59     if(confirm.length === 0){
60       flash("[Client] Confirm password must not be empty", "warning");
61       hasError = true;
62     }
63     if(!is_valid_password(confirm)){
64       flash("[Client] Confirm password too short", "warning");
65       hasError = true;
66     }
67     if(password !== confirm){
68       flash("[Client] Password and Confirm Password do not match", "warning");
69       hasError = true;
70     }
71   }
72   return hasError;
73 </script>

```

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

```

18 | //NICHOLAS ZUBRZYCKI njzs 4/1/2024
19 | function sanitize_email(email = ""){
20 |
21 |     //Remove all characters except specified, according to PHP documentation.
22 |     return email.replace(/[^a-zA-Z0-9!#$%&'*+=?^_`{|}~@.[]]/g, '');
23 |
24 |
25 |     function is_valid_email(email = ""){
26 |
27 |         //One or more chars, @, One or more chars, ., One or more chars
28 |         let emailRegex = /^[^s@]+@[^\s@]+\.[^\s@]+$/;
29 |         if(emailRegex.test(email)){
30 |             return true;
31 |         }
32 |         return false;
33 |     }
34 |
35 |     function is_valid_username(username){
36 |
37 |         let usernameRegex = /^[a-zA-Z0-9_-]{3,16}$/;
38 |         if(usernameRegex.test(username)){
39 |             return true;
40 |         }
41 |         return false;
42 |     }
43 |
44 |     function is_valid_password(password){
45 |
46 |         return password.length >= 8;
47 |     }

```

**JavaScript validation functions in helpers.js.****Checklist Items (1)****#1 Show the JavaScript validations (include any extra files related)**

```

76 <?php
77 //TODO 2: add PHP Code
78 //NICHOLAS ZUBRZYCKI njzs 4/1/2024      You, 2 minutes ago * Uncommitted changes
79 if(isset($_POST["email"]) && isset($_POST["username"]) && isset($_POST["password"]) && isset($_POST["confirm"])){
80     $email = se($_POST, "email", "", false);
81     $username = se($_POST, "username", "", false);
82     $password = se($_POST, "password", "", false);
83     $confirm = se($_POST, "confirm", "", false);
84     //TODO 3: validate/use
85     $hasError = false;
86     if(empty($email)){
87         flash("Email must not be empty", "warning");
88         $hasError = true;
89     }
90     $email = sanitize_email($email);
91     if(!is_valid_email($email)){
92         flash("Invalid email address", "warning");
93         $hasError = true;
94     }
95     if(!is_valid_username($username)){
96         flash("Username must only be alphanumeric and can only contain - or _", "warning");
97         $hasError = true;
98     }
99     if(empty($password)){
100        flash("Password must not be empty", "warning");
101        $hasError = true;
102    }
103    if(empty($confirm)){
104        flash("Confirm password must not be empty", "warning");
105        $hasError = true;
106    }
107    if(!is_valid_password($password)){
108        flash("Password is too short", "warning");
109        $hasError = true;
110    }

```

```

108
109     if($password){
110         if(strlen($password) > 0 && $password != $confirm){
111             flash("Passwords must match", "warning");
112             $hasError = true;
113         }
114     }
115     //7000: 4: Send information to database

```

### PHP validation in register.

#### Checklist Items (1)

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

```

1  <?php
2  //NICHOLAS ZUBRZYCKI njz5 4/1/2024
3  function sanitize_email($email = ""){
4  {
5      return filter_var(trim($email), FILTER_SANITIZE_EMAIL);
6  }
7  function is_valid_email($email = ""){
8  {
9      return filter_var(trim($email), FILTER_VALIDATE_EMAIL);
10 }
11 function is_valid_username($username)
12 {
13     return preg_match('/^[\a-z0-9_-]{3,16}$/', $username);
14 }
15 function is_valid_password($password)
16 {
17     if($password){
18         return strlen($password) >= 8;
19     }
20 }

```

### PHP validation functions in sanitizers.php.

#### 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
<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

Small

Medium

Large

The screenshot shows a database interface with a sidebar on the left containing a tree view of database objects. The main area displays a table named 'Users' with the following data:

	id	email	password	created	modified	username
> 1	1	email@email.com	\$2y\$10\$HUvpLqEX/XDL	2024-03-18 02:24:02	2024-03-18 02:24:02	email
> 2	2	email2@email.com	\$2y\$10\$Z.G6QH8MsclD	2024-03-18 02:25:11	2024-03-18 02:25:11	email2
> 3	3	email3@email.com	\$2y\$10\$3Y/uO6dn7Dt7	2024-03-18 23:06:19	2024-03-18 23:06:19	email3
> 4	6	email4@email.com	\$2y\$10\$xz71hnSh5DCy	2024-03-19 03:11:12	2024-03-19 03:12:18	username4
> 5	11	heroku@heroku.com	\$2y\$10\$KnDYEmVAksQ	2024-04-01 23:46:34	2024-04-01 23:46:34	heroku

Users table.

#### 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 user loads an empty page on register.php. After inputting their information into each field (email, username, password, and confirm password), the user submits their information. First the HTML validation validates the data using form requirements. If that is successful, then the JavaScript validation validates the data by inspecting the values of input. If that is successful, then the PHP validation validates the data using the POST data. If all layers of validation confirm the data is good, then email, username, and password are inserted into the users table. The data for id, created, and modified is automatic, so the database handles that itself.

### Task #6 - Points: 1

Text: Include pull request links related to this feature

#### Details:

Should end in /pull/#

URL #1

<https://github.com/NickZub/njz5-it202-008/pull/12>

URL #2

<https://github.com/NickZub/njz5-it202-008/pull/22>

URL #3

<https://github.com/NickZub/njz5-it202-008/pull/24>

### User Login (2 pts.)

### 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	Include correct data where username is used to login
<input checked="" type="checkbox"/> #4	1	Include correct data where email is used to login
<input checked="" type="checkbox"/> #5	1	Show success login message
<input checked="" 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 checked="" type="checkbox"/> #7	1	Demonstrate user-friendly message of when an account doesn't exist
<input checked="" type="checkbox"/> #8	1	Demonstrate user-friendly message of when password doesn't match what's in the DB
<input checked="" type="checkbox"/> #9	1	Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)

## Task Screenshots:

## Gallery Style: Large View

[Small](#)[Medium](#)[Large](#)

A screenshot of a web browser window displaying a login form. The URL in the address bar is <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/login.php>. The browser interface includes tabs for dev, prod, and Localhost. The login form has two input fields: 'Email/Username' and 'Password', both of which are currently empty. Below the inputs is a 'Login' button. A yellow horizontal bar spans the width of the form, containing four error messages: 'Email/Username must not be empty', 'Email/Username Invalid username', 'Email/Password must not be empty', and 'Email/Password too short'. The overall layout is a standard web-based login interface.

Empty login.

## 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)

A screenshot of a web browser window displaying a login form. The URL in the address bar is <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/login.php>. The browser interface includes tabs for dev, prod, and Localhost. The login form has two input fields: 'Email/Username' and 'Password', both of which are currently empty. Below the inputs is a 'Login' button. A yellow horizontal bar spans the width of the form, containing four error messages: 'Email/Username must not be empty', 'Email/Username Invalid username', 'Email/Password must not be empty', and 'Email/Password too short'. The overall layout is a standard web-based login interface.

Email/Username @@@

Password @@@

Forgot Password

Login

Invalid email and password.

### 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)

A screenshot of a web browser window displaying a login form. The URL in the address bar is `https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/login.php`. The browser interface includes standard controls like back, forward, and search. Below the address bar, there are tabs for 'dev', 'prod', and 'localhost'. The main content area shows a login form with two error messages: 'Email or Username not found' and 'Password must be at least 6 characters long'. The 'Email/Username' and 'Password' fields are highlighted with red boxes, indicating they are invalid. A 'Login' button is visible below the fields.

Account doesn't exist.

### Checklist Items (1)

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

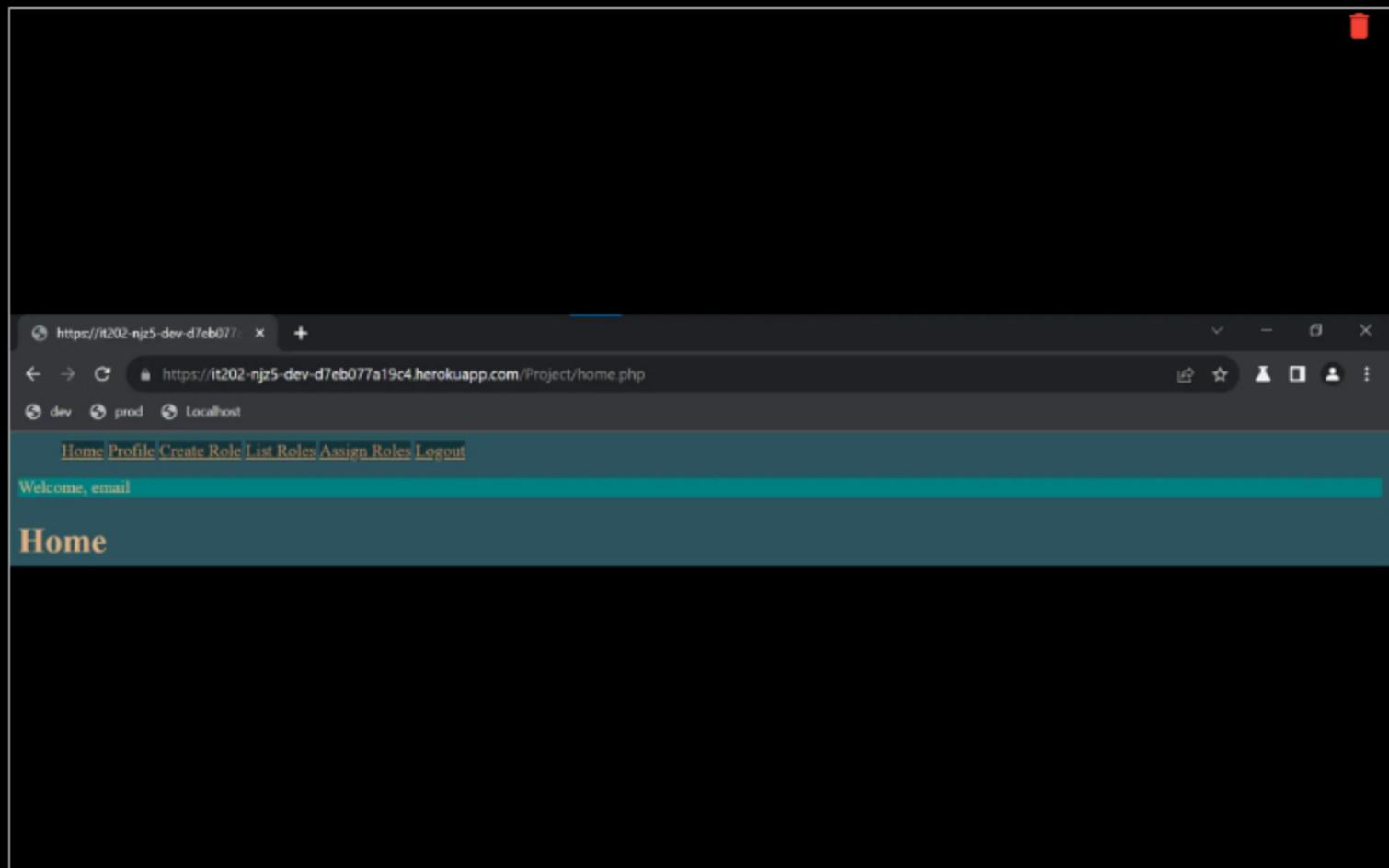
A screenshot of a web browser window showing a login form. The URL in the address bar is <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/login.php>. The page has a teal header with 'Login Register' links. Below the header is a yellow horizontal bar containing placeholder text 'Email/Password'. The main form area has two input fields: 'Email/Username' and 'Password', both with their respective labels above them. A 'Login' button is located below the password field. The entire page is set against a dark background.

Password doesn't match database entry.

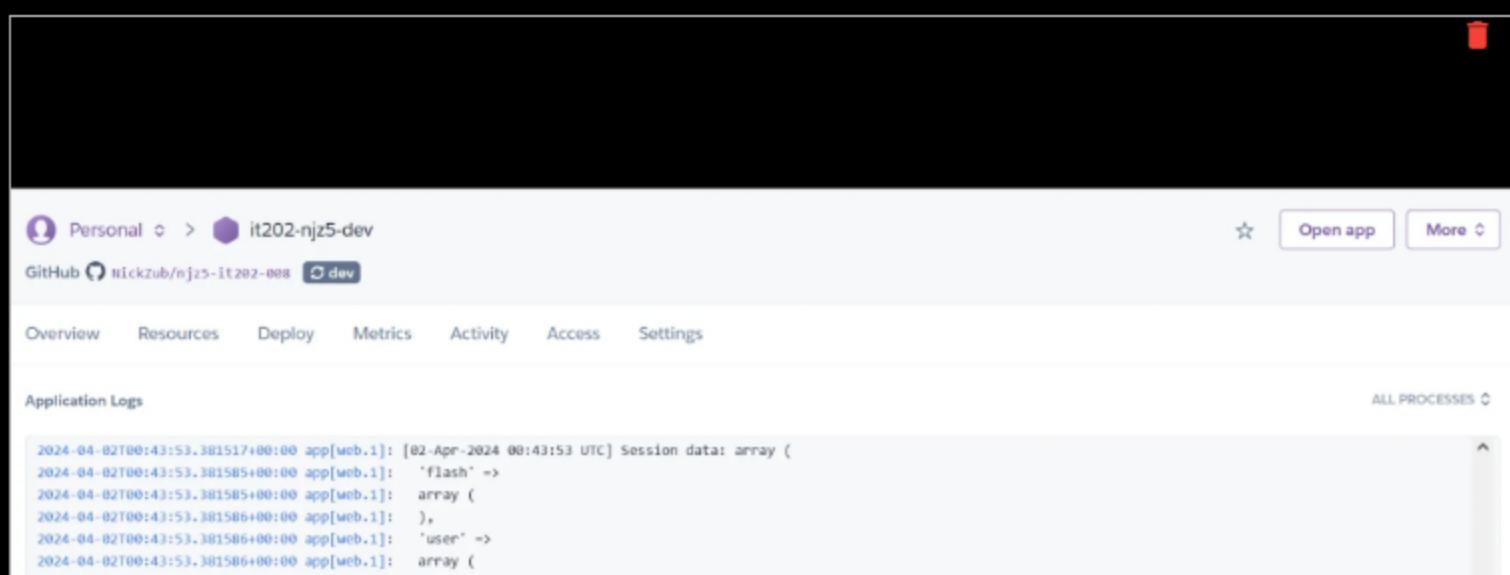
#### Checklist Items (1)

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

A screenshot of a web browser window showing a successful login message. The URL in the address bar is <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/home.php>. The page has a teal header with 'Home Profile Create Role List Roles Assign Roles Logout' links. Below the header is a teal horizontal bar with the text 'Welcome, email'. The main content area displays the word 'Home' in large, bold, black font. At the very bottom of the page, there is a small, faint watermark-like text 'Successful username login'.

**Checklist Items (2)****#3 Include correct data where username is used to login****#9 Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)**

Successful email login (looks the same).

**Checklist Items (2)****#4 Include correct data where email is used to login****#9 Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)**

```
2024-04-02T00:43:53.381587+00:00 app[web.1]:      'id' -> 2,
2024-04-02T00:43:53.381587+00:00 app[web.1]:      'email' -> 'email2@email.com',
2024-04-02T00:43:53.381588+00:00 app[web.1]:      'username' -> 'email2',
2024-04-02T00:43:53.381588+00:00 app[web.1]:      'roles' ->
2024-04-02T00:43:53.381589+00:00 app[web.1]:      array (
2024-04-02T00:43:53.381603+00:00 app[web.1]:      ),
2024-04-02T00:43:53.381613+00:00 app[web.1]:      ),
```

Autoscroll with output

Save

## Session data in server logs.

### Checklist Items (1)

#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 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



```
3 | //NICHOLAS ZUBRZYCKI njz5 4/1/2024
4 |
5 | <form onsubmit="return validate(this)" method="POST">
6 |   <div>
7 |     <label for="email">Email/Username</label>
8 |     <input type="text" id="email" name="email" required />
9 |   </div>
10 |  <div>
11 |    <label for="pw">Password</label>
12 |    <input type="password" id="pw" name="password" required minlength="8" />
13 |  </div>
```

```
14 |     <input type="submit" value="Login" />
15 | </form>
```

Login form code with combined field.

## 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)

```
16 |     <script>
17 |         function validate(form) [
18 |             //TODO 1: implement JavaScript validation
19 |             //ensure it returns false for an error and true for success
20 |             //NICHOLAS ZUBRZYCKI njz5 4/1/2024
21 |             let input = form.email.value;
22 |             let password = form.pw.value;
23 |             let hasError = false;
24 |             if(input.length === 0){
25 |                 flash("[Client] Input must not be empty", "warning");
26 |                 hasError = true;
27 |             }
28 |             if(input.includes('@')){
29 |                 input = sanitize_email(input);
30 |                 if(!is_valid_email(input)){
31 |                     flash("[Client] Invalid email address", "warning");
32 |                     hasError = true;
33 |                 }
34 |             } else{
35 |                 if(!is_valid_username(input)){
36 |                     flash("[Client] Invalid username", "warning");
37 |                     hasError = true;
38 |                 }
39 |             }
40 |             if(password.length === 0){
41 |                 flash("[Client] Password must not be empty", "warning");
42 |                 hasError = true;
43 |             }
44 |             if(!is_valid_password(password)){
45 |                 flash("[Client] Password too short", "warning");
46 |                 hasError = true;
47 |             }
48 |         return !hasError;
49 |     }
50 | </script>
```

JavaScript validation for login.

## Checklist Items (1)

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

```
18 | //NICHOLAS ZUBRZYCKI njz5 4/1/2024
19 | function sanitize_email(email = "") {
20 |     //Remove all characters except specified, according to PHP documentation.
21 |     return email.replace(/[^a-zA-Z0-9!#$%&'*+=?^_`{|}~@.[]]/g, '');
22 |
23 | }
24 |
25 | function is_valid_email(email = "") {
26 |     //One or more chars, @, One or more chars, ., One or more chars
27 |     let emailRegex = /^[^\\s@]+@[^\\s@]+\.[^\\s@]+$/;
```

```

29     if(emailRegex.test(email)){
30         return true;
31     }
32     return false;
33 }
34
35 function is_valid_username(username)
36 {
37     let usernameRegex = /^[a-zA-Z0-9_-]{3,16}$/;
38     if(usernameRegex.test(username)){
39         return true;
40     }
41     return false;
42 }
43
44 function is_valid_password(password)
45 {
46     return password.length >= 8;
47 }

```

### JavaScript validation functions in helpers.js.

#### Checklist Items (1)

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

```

51     <?php
52     //TODO 2: add PHP Code
53     //NICHOLAS ZUBRZYCKI njz5 4/1/2024
54     if (isset($_POST["email"]) && isset($_POST["password"])) {
55         $email = se($_POST, "email", "", false);
56         $password = se($_POST, "password", "", false);
57
58         //TODO 3
59         $hasError = false;
60         if (empty($email)) {
61             flash("Email must not be empty", "warning");
62             $hasError = true;
63         }
64         if (str_contains($email, "@")) {
65             //sanitize
66             $email = sanitize_email($email);
67             //validate
68             if (!is_valid_email($email)) {
69                 flash("Invalid email address", "warning");
70                 $hasError = true;
71             }
72         } else {
73             if (!is_valid_username($email)) {
74                 flash("Invalid username", "warning");
75                 $hasError = true;
76             }
77         }
78         if (empty($password)) {
79             flash("Password must not be empty", "warning");
80             $hasError = true;
81         }
82         if (!is_valid_password($password)) {
83             flash("Password too short", "warning");
84             $hasError = true;
85         }
86         if (!$hasError) {
87             //flash("Welcome, $email");
88             //TODO 4
89             $db = getDB();
90             $stmt = $db->prepare("SELECT id, email, username, password from Users
91 where email = :email or username = :email");

```

### PHP validation in login.

#### Checklist Items (1)

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

```

1     <?php
2     //NICHOLAS ZUBRZYCKI njz5 4/1/2024

```

```

3   function sanitize_email($email = "")
4   {
5       return filter_var(trim($email), FILTER_SANITIZE_EMAIL);
6   }
7   function is_valid_email($email = "")
8   {
9       return filter_var(trim($email), FILTER_VALIDATE_EMAIL);
10  }
11  function is_valid_username($username)
12  {
13      return preg_match('/^[\w\-\_]{3,16}$/', $username);
14  }
15  function is_valid_password($password)
16  {
17      if($password){
18          return strlen($password) >= 8;
19      }
20  }

```

### PHP validation functions in sanitizers.php.

#### 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
<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:

When visiting login.php, an empty page is loaded. After inputting email/username and password, the input values are validated through HTML, JavaScript (client-side), and PHP (server-side) validation. If proper data is submitted, a database query is made to search for the email or username that is inputted. If the query is successful, the passwords are verified and compared, but if the query is unsuccessful, a not found error is returned. Sessions are similar to cookies, except they are stored on the server side rather than client side. Login requires nav and nav starts the session, so the login page is able to access the session data. If proper data is submitted, then the database query result information is stored in the session. This means if a user logs in once, they are able to access other pages using that same account even after closing the tab, without having to login again.

## Task #4 - Points: 1

Text: Include pull request links related to this feature

### Details:

Should end in /pull/#

#### URL #1

<https://github.com/NickZub/njz5-it202-008/pull/13>

#### URL #2

<https://github.com/NickZub/njz5-it202-008/pull/14>

#### URL #3

<https://github.com/NickZub/njz5-it202-008/pull/16>

#### URL #4

<https://github.com/NickZub/njz5-it202-008/pull/19>

#### URL #5

<https://github.com/NickZub/njz5-it202-008/pull/21>

## 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



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

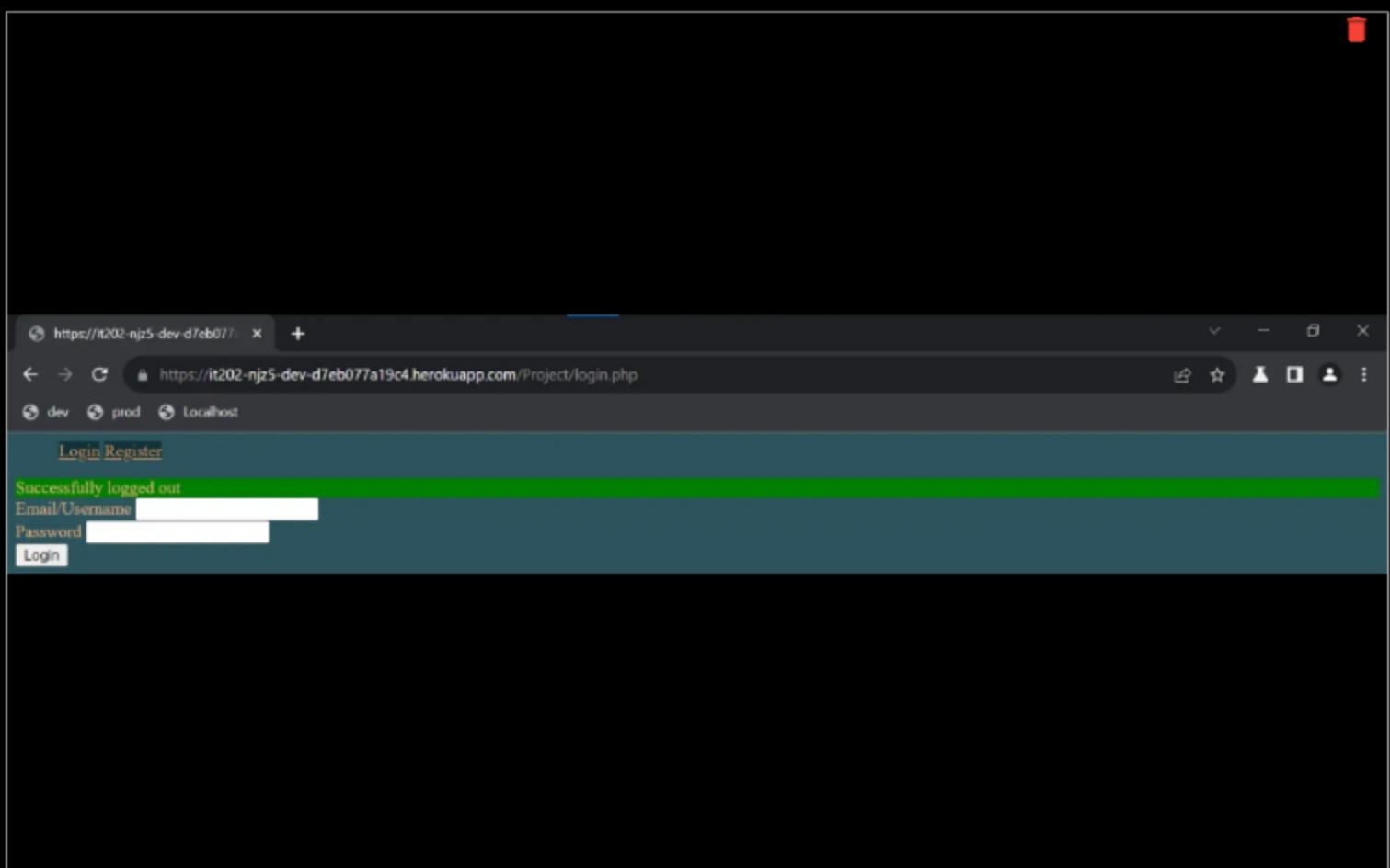
Welcome, email

## Home

Logged in navigation.

### Checklist Items (1)

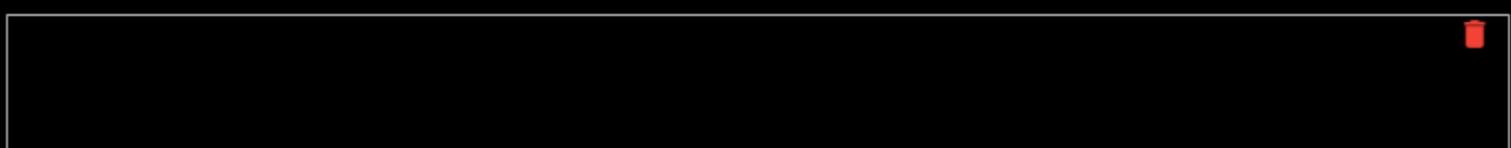
#1 Screenshot of the navigation when logged in (site)



Logout redirect with message.

### Checklist Items (1)

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



```
1 <?php
2 session_start();
3 require(__DIR__ . "/../../lib/functions.php");
4 reset_session();
5 //NICHOLAS ZUBRZYCKI njz5 4/1/2024
6 flash("Successfully logged out", "success");
7 header("Location: login.php");
8
```

Logout.php code.

#### Checklist Items (1)

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

```
1 <?php
2 function reset_session()
3 {
4     session_unset();
5     session_destroy();
6     session_start();
7 }
8 //NICHOLAS ZUBRZYCKI njz5 4/1/2024
```

Reset session function.

#### 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

#### ● Details:

Should end in /pull/#

URL #1

<https://github.com/NickZub/njz5-it202-008/pull/13>

URL #2

<https://github.com/NickZub/njz5-it202-008/pull/19>

### Basic Security Rules and Roles (2 pts.)

▲ COLLAPSE ▲

### Task #1 - Points: 1

Text: Authentication Screenshots

#### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
<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



```
7 | //NICHOLAS ZUBRZYCKI njz5 4/1/2024
8 | function is_logged_in($redirect = false, $destination = "login.php")
9 | {
```

```
10     $isLoggedIn = isset($_SESSION["user"]);
11     if ($redirect && !$isLoggedIn) {
12         //if this triggers, the calling script won't receive a reply since die()/exit() terminates it
13         flash("You must be logged in to view this page", "warning");
14         die(header("Location: $destination"));
15     }
16 }
17 }
```

Function to check if a user is logged in.

#### Checklist Items (1)

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

```
1  <?php
2  require_once(__DIR__ . "/../../partials/nav.php");
3  is_logged_in(true);
4 | //NICHOLAS ZUBRZYCKI njz5 4/1/2024
5  ?>
```

Use login function in profile.php.

#### Checklist Items (2)

#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)

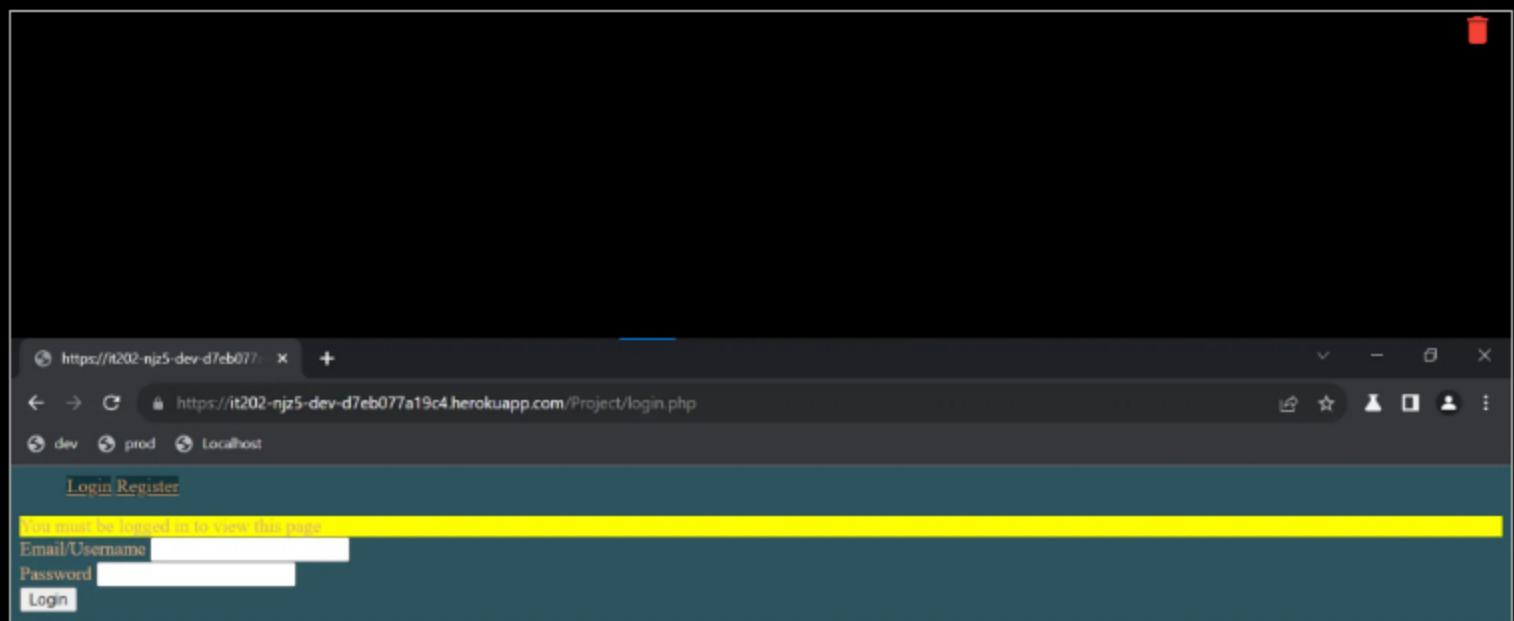
```
1 <?php
2 require(__DIR__."/../../partials/nav.php");
3 ?>
4 <h1>Home</h1>
5 <?php
6 if(is_logged_in(true)){
7     error_log("Session data: " . var_export($_SESSION, true));
8 }
9 require(__DIR__ . "/../../partials/flash.php");
10 //NICHOLAS ZUBRZYCKI njz5 4/1/2024
11 ?>
```

Use login function in home.php.

#### Checklist Items (2)

#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 to access a login-protected page while logged out.

## Checklist Items (1)

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

### Task #2 - Points: 1

Text: Authorization Screenshots

#### Checklist

\*The checkboxes are for your own tracking

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

#### Task Screenshots:

Gallery Style: Large View

Small      Medium      Large

```
17 | //NICHOLAS ZUBRZYCKI njz5 4/1/2024
18 | function has_role($role)
19 | {
20 |     if (is_logged_in() && isset($_SESSION["user"]["roles"])) {
21 |         foreach ($_SESSION["user"]["roles"] as $r) {
22 |             if ($r["name"] === $role) {
23 |                 return true;
24 |             }
25 |         }
26 |     }
27 |     return false;
28 | }
```

Function to check for a specific role.

## Checklist Items (1)

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

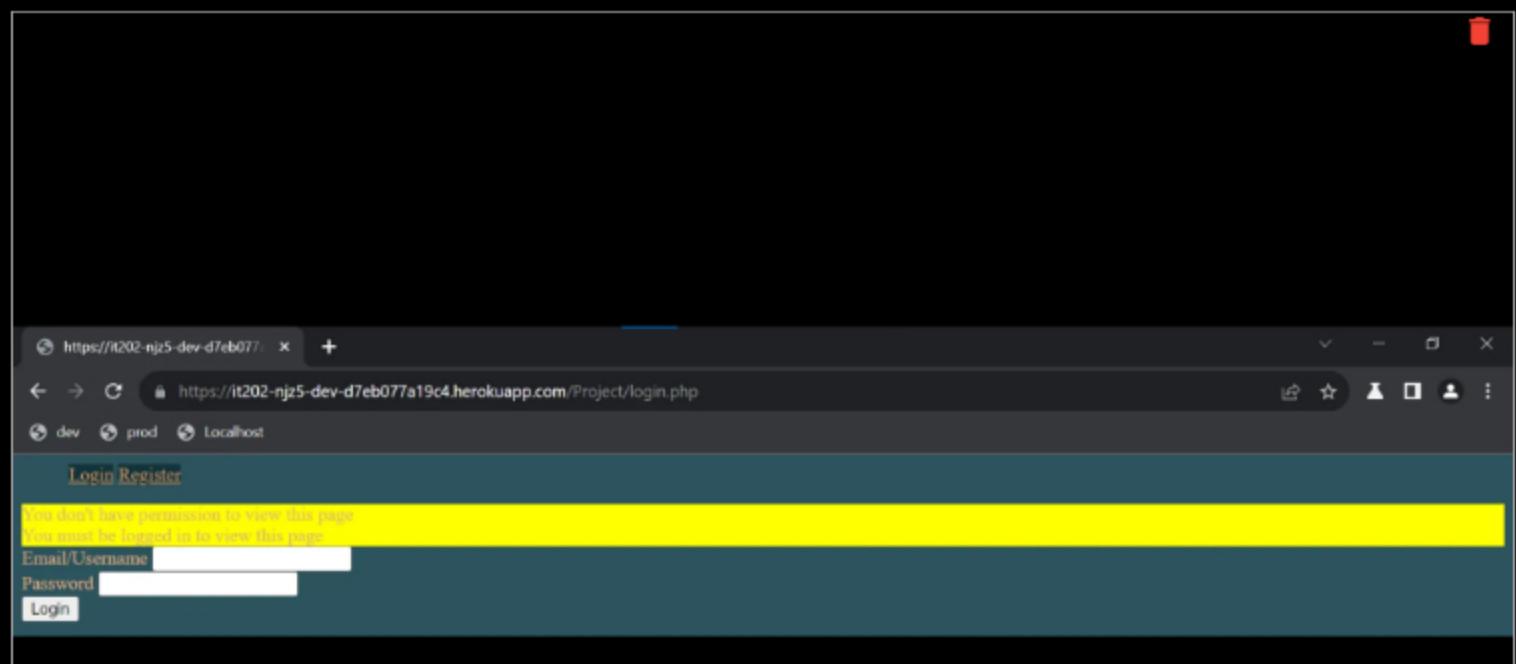
```
3  require(__DIR__ . "/../../../../partials/nav.php");
4 //NICHOLAS ZUBRZYCKI njz5 4/1/2024      You, 23 hours ago • Uncommit
5 if (!has_role("Admin")) {
6     flash("You don't have permission to view this page", "warning");
7     exit(header("Location: $BASE_PATH" . "/home.php"));
8 }
```

Role check function being used. It is used on all the admin pages that regular users do not have access to.

## 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)



Trying to access a role-protected page while logged out.

### 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
<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 uid)

### Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

The screenshot shows the MySQL Workbench interface with the following details:

- Left Panel:** Shows the database structure with 'db.ethereallab.app@330...' selected. It lists 'information\_schema', 'njz5 128k', 'Tables (3)', 'Roles 1', 'UserRoles 1', 'Users 3', 'Views', 'Procedures', and 'Functions'.
- Properties Tab:** Selected tab, showing the table structure for 'UserRoles'.
- SQL Tab:** Contains the query: `SELECT * FROM `UserRoles` LIMIT 100`.
- Search Results Tab:** Shows the results of the query:

	id	user_id	role_id	is_active	created	modified
>	1	2	1	1	2024-03-30 02:24:04	2024-03-30 02:24:04

## UserRoles table.

### 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)

The screenshot shows the MySQL Workbench interface. On the left, the database tree shows 'db.ethereallab.app@3306' expanded, with 'Tables (3)' selected. Under 'Tables (3)', 'Roles' is highlighted. The main pane displays the 'Roles' table structure with the following data:

	Q	P	id	name	description	is_active	created	modified
	Q	P	int	varchar(2)	varchar(100)	tinyint(1)	timestamp	timestamp
	>	1	1	Admin		1	2024-03-30 02:21:08	2024-04-01 02:04:49

## Roles table.

### 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)

[^COLLAPSE ^](#)

#### TASK #4 - POINTS: 1

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

##### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
<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 UserRoles table is used to connect the users table and roles table together, resulting in the association between users and the respective roles they have. This allows you to easily identify and query what roles a specific user has. Roles.is\_active is a global toggle for the role and affects every single user; UserRoles.is\_active is a local toggle for the role and affects only that single user.



[^COLLAPSE ^](#)

#### Task #5 - Points: 1

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

##### i Details:

Should end in /pull/#

##### URL #1

<https://github.com/NickZub/njz5-it202-008/pull/20>



##### User Profile (2 pts.)

[^COLLAPSE ^](#)



[^COLLAPSE ^](#)

#### Task #1 - Points: 1

**Text:** View Profile Website Page

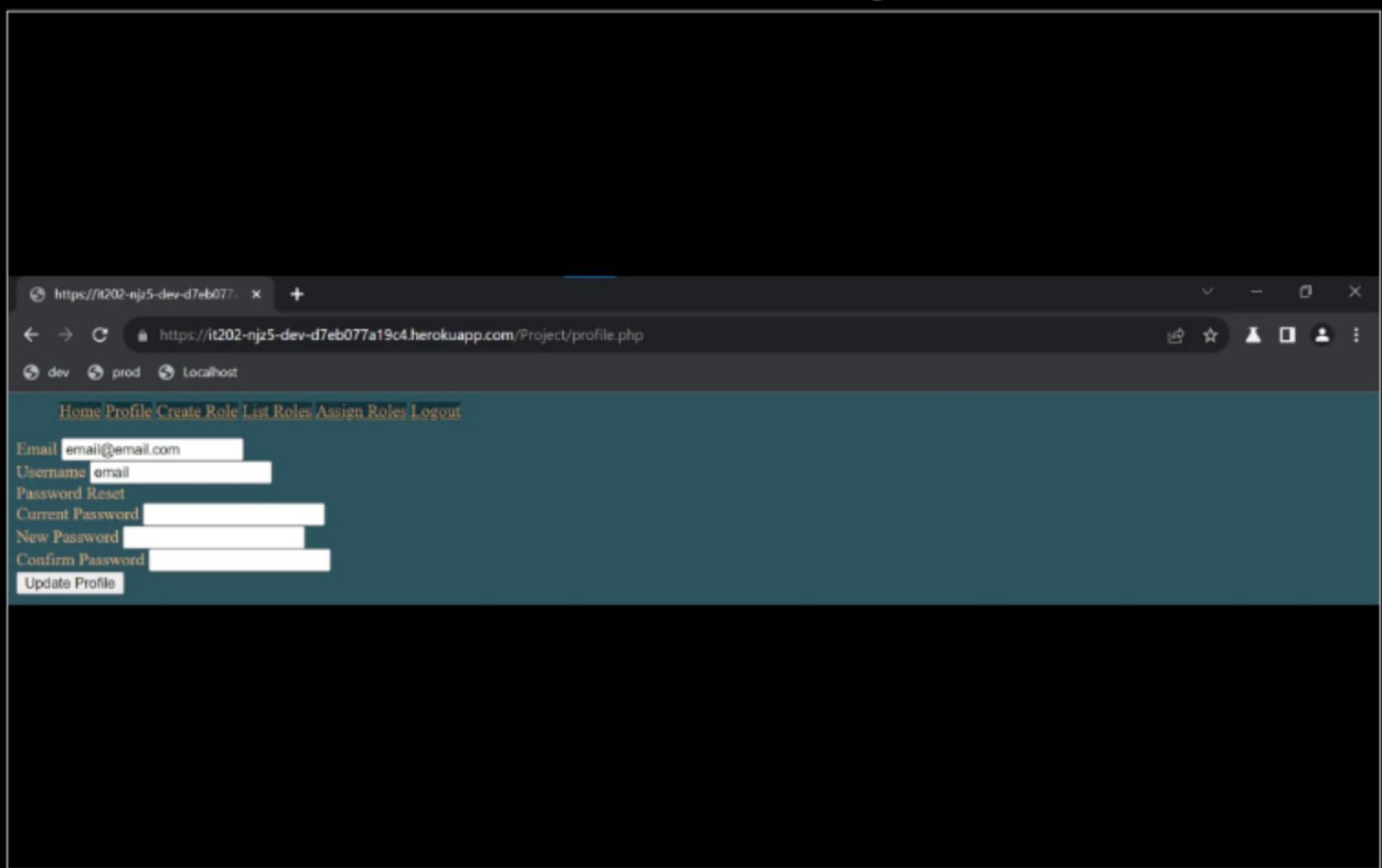
##### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
<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



### 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:

The functions `get_username` and `get_user_email` in `user_helpers.php` are used to retrieve the values of `email` and

username from the session data. After retrieving these values in profile.php, the value attributes for the email and username fields in the form are populated using the safer echo of their values. This preloads their data so you can see it when the form loads.

### 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

The screenshot shows a web browser window with a dark theme. At the top, there is a navigation bar with icons for back, forward, and refresh, followed by the URL: <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php>. Below the URL, there are three tabs labeled 'dev', 'prod', and 'localhost'. The main content area displays a profile edit form. The 'Email' field contains 'heroku2@heroku.com'. The 'Username' field contains 'heroku2'. Below these are fields for 'Password Reset', 'Current Password', 'New Password', and 'Confirm Password', all of which are currently empty. At the bottom of the form is a 'Update Profile' button.

Before username change.

Checklist Items (1)

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

The screenshot shows a web browser window with the URL <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php>. The page has a teal header with links for Home, Profile, and Logout. A green success message bar at the top says "Profile saved". Below it, there are input fields for Email (heroku2@heroku.com), Username (heroku), and Password Reset. There are also fields for Current Password, New Password, and Confirm Password, followed by a "Update Profile" button.

After username change.

Checklist Items (1)

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

The screenshot shows the same web browser window after a username change. The URL remains the same: <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php>. The page structure is identical to the previous screenshot, but the "Username" field now contains "heroku2", indicating the successful update.

New Password

Confirm Password

[Update Profile](#)

Before email change.

Checklist Items (1)

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

https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php

← → C https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php

dev prod Localhost

[Home](#) [Profile](#) [Logout](#)

Profile saved

Email heroku@heroku.com

Username heroku

Password Reset

Current Password

New Password

Confirm Password

[Update Profile](#)

After email change.

Checklist Items (1)

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

https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php

← → C https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php

dev prod Localhost

[Home](#) [Profile](#) [Logout](#)

Profile saved

Profile saved

Password reset

Email heroku@heroku.com

Username heroku

Password Reset

Current Password

New Password

Confirm Password

Update Profile

Successfully updated password.

Checklist Items (1)

#5 Demonstrate the success message of updating password

The screenshot shows a web browser window with the URL <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php>. The page displays a profile update form with several validation errors highlighted in yellow:

- [Client] Invalid email address
- [Client] Invalid username

The form fields are as follows:

- Email: @@ (highlighted in yellow)
- Username: @@ (highlighted in yellow)
- Current Password
- New Password
- Confirm Password

Below the form is a success message:

JavaScript email and username 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 the URL <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php>. The page displays a profile update form with validation messages:

- Email: Please enter a valid email address.
- Username: Please enter a valid username.
- Current Password: Please enter a valid password.
- New Password: Please enter a valid password.
- Confirm Password: Please enter a valid password.

Below the form is a success message:

JavaScript email and username validation.

[Home](#) [Profile](#) [Logout](#)

[Client] New password and Confirm password must match

[Client] New password too short

[Client] Confirm password too short

Email

Username

Password Reset

Current Password

New Password

Confirm Password

JavaScript password validation.

#### Checklist Items (1)

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

 <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/>  

    <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php>

 dev  prod  Localhost

[Home](#) [Profile](#) [Logout](#)

The chosen email is not available.

Email

Username

Password Reset

Current Password

New Password

Confirm Password

PHP 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)

in use, current password doesn't match what's in the DB)

The screenshot shows a web browser window with the URL <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php>. The page displays a profile update form. At the top, there are navigation links for Home, Profile, and Logout. A yellow error bar at the top states "The chosen username is not available." Below it, the Email field contains "heroku@heroku.com". The Username field contains "heroku". The Password Reset, Current Password, New Password, and Confirm Password fields are empty. A red "Update Profile" button is visible. A green message bar at the bottom states "PHP 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)

The screenshot shows a web browser window with the URL <https://it202-njz5-dev-d7eb077a19c4.herokuapp.com/Project/profile.php>. The page displays a profile update form. At the top, there are navigation links for Home, Profile, and Logout. A green message bar at the top states "Profile saved". A yellow error bar below it states "Current password is invalid". Below it, the Email field contains "heroku@heroku.com". The Username field contains "heroku". The Password Reset, Current Password, New Password, and Confirm Password fields are empty. A red "Update Profile" button is visible.

### 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)

#### 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:

1. Username and email are checked for validity according to HTML, JavaScript, then PHP validations. If each passes, then an update statement with the new email and username parameters is queried against the database to change the current data. It uses the get\_user\_id() function to only change the data for that respective user.
2. If any value is entered for the passwords, the current password and new passwords are checked for validity using HTML, JavaScript, and PHP validations. If those pass, then the password verify function checks to see if the current password field value matches the password value in the database for that user id, which is done on the PHP server side. If it does, then the password is changed using an update statement.

#### Task #5 - Points: 1

Text: Include pull request links related to this feature

#### i Details:

Should end in /pull/#

URL #1

<https://github.com/NickZub/njz5-it202-008/pull/34>

URL #2

<https://github.com/NickZub/njz5-it202-008/pull/35>

URL #3

<https://github.com/NickZub/njz5-it202-008/pull/36>

URL #4

<https://github.com/NickZub/njz5-it202-008/pull/16>

URL #5

<https://github.com/NickZub/njz5-it202-008/pull/11>

Misc (1 pt.)

[^COLLAPSE ^](#)

Task #1 - Points: 1

Text: Screenshot of wakatime

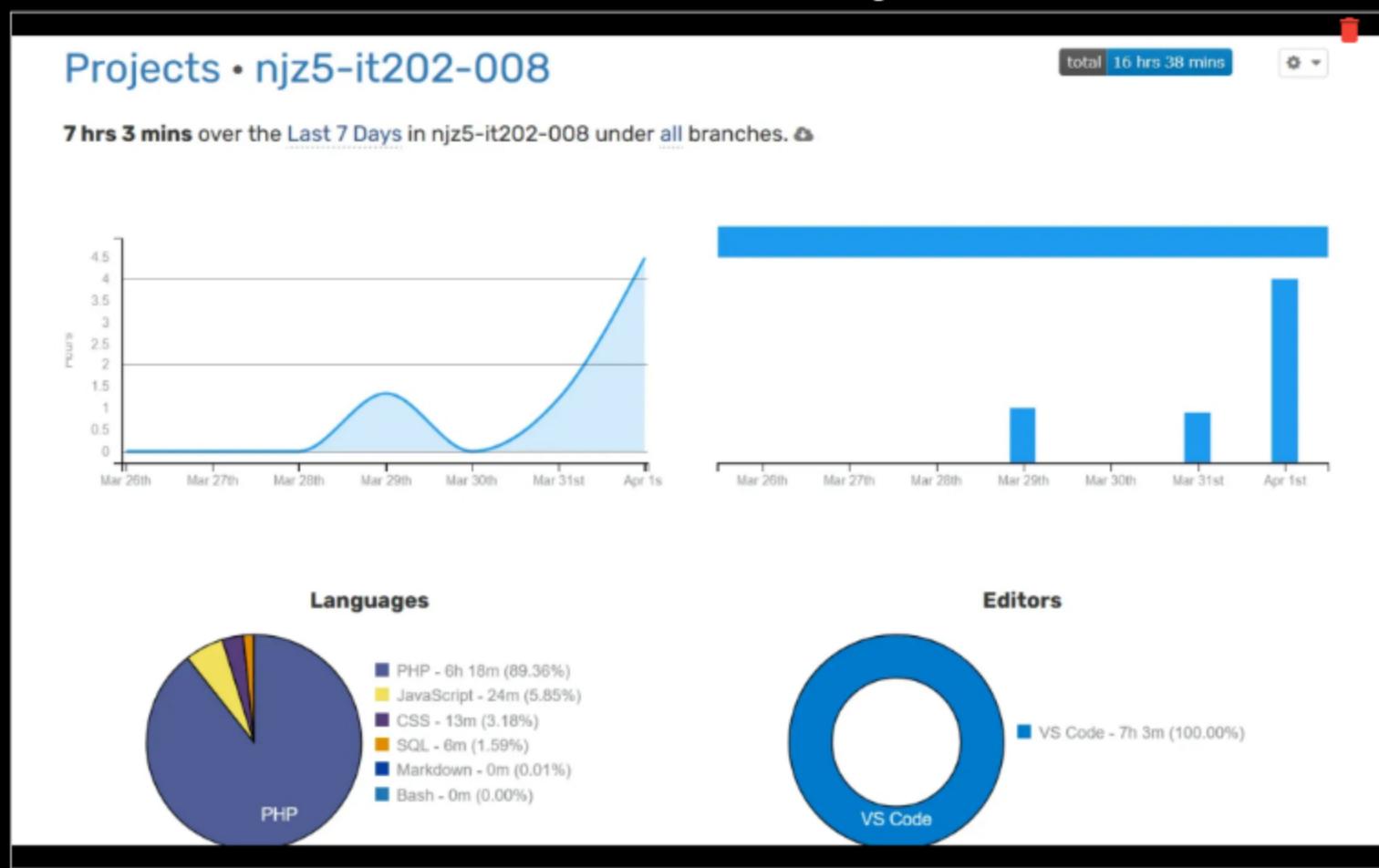
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



Waka.



## login.php

1 hr 51 mins

## profile.php

1 hr 41 mins



## Waka 2.

### Files

1 hr 51 mins	public_html/Project/login.php
1 hr 41 mins	...ic_html/Project/profile.php
33 mins	...c_html/Project/register.php
20 mins	lib/functions.php
19 mins	partials/nav.php
14 mins	lib/sanitizers.php
14 mins	lib/duplicate_user_details.php
14 mins	lib/sanitizers.js
13 mins	public_html/Project/home.php
13 mins	public_html/Project/styles.css
12 mins	lib/user_helpers.php
8 mins	public_html/Project/logout.php
8 mins	public_html/Project/helpers.js
6 mins	lib/reset_session.php
5 mins	public_html/index.php
5 mins	...roject/admin/list_roles.php
4 mins	.../005_insert_roles_admin.sql
2 mins	...ject/admin/create_role.php
2 mins	...ject/admin/assign_roles.php
2 mins	...html/Project/sanitizers.js
1 min	partials/flash.php
1 min	lib/get_url.php
1 min	..._create_table_userroles.sql
32 secs	.../003_create_table_roles.sql
7 secs	lib/db.php
5 secs	lib/config.php

### Branches

2 hrs 47 mins	Milestone1
2 hrs 25 mins	MS1-Feat-JavaScriptValidation
58 mins	MS1-InClass-Housekeeping
51 mins	MS1-Feat-UserRoles

## Waka 3.

### 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

NickZub / Projects / njz5-it202-008

Type / to search

Add status update

View 1 New view

Filter by keyword or by field

Discard Save

**Todo** 0

This item hasn't been started

+ Add item

**In Progress** 0

This is actively being worked on

+ Add item

**Done** 9

This has been completed

njz5-it202-008 #25  
MS1 - User will be able to register a new account

njz5-it202-008 #26  
MS1 - User will be able to login to their account

njz5-it202-008 #27  
MS1 - User will be able to logout

+ Add item

Project board.

Column	Items
Todo	0 items
In Progress	0 items
Done	3 items: njz5-it202-008 #25: MS1 - User will be able to register a new account njz5-it202-008 #26: MS1 - User will be able to login to their account njz5-it202-008 #27: MS1 - User will be able to logout

Task #3 - Points: 1

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

URL #1

<https://github.com/users/NickZub/projects/2/views/1>

Task #4 - Points: 1

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

URL #1

<https://it202-njz5-prod-bedd6fae57b2.herokuapp.com/Project/login.php>

End of Assignment