

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/na569>

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

Submissions:

Submission Selection

1 Submission [active] 4/1/2024 1:20:44 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)
- 1. Consider styling all forms/inputs, data output, navigation, etc
- 2. Implement JavaScript validation on Register, Logout, and Profile (include "[Client]" in the output messages to differentiate between server-side validations)
- 3.
- 4.
- 5.

Instructions:

6.

- 7. Make sure you're in Milestone1 with the latest changes pulled
- 8. Ensure Milestone1 has been deployed to heroku dev
- 9. Gather the requested evidence and fill in the explanations per each prompt
- 10. 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 web browser window with the following details:

- Address Bar:** na569-dev-9569cb3ae6c9.herokuapp.com/project/register.php
- Content Area:**
 - Email:** asd (highlighted with a red border)
 - Username:** asd
 - Password:** (redacted)
 - Confirm:** (redacted)
- Buttons:** A "Register" button at the bottom.

A yellow horizontal bar spans the width of the content area, containing the text "[Client] Invalid email address".

At the bottom of the page, there is a footer section with the text "email validation".

#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 `na569-dev-9569cb3ae6c9.herokuapp.com/project/register.php`. The page displays a registration form with two tabs: "Login" and "Register". A yellow error message at the top states: "[Client] Username must only contain 3-16 characters a-z, 0-9, _, or -". Below the message are four input fields: "Email" containing `asd@test.com`, "Username" containing `as`, "Password" containing `.....`, and "Confirm" containing `.....`. A blue "Register" button is located below the inputs. At the bottom of the page, there is a dark bar with the text "username validation" and a "Checklist Items (1)" section containing the same validation rule.

[Client] Username must only contain 3-16 characters a-z, 0-9, _, or -

Email

Username

Password

Confirm

Register

username validation

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 `na569-dev-9569cb3ae6c9.herokuapp.com/project/register.php`. The page displays a registration form with two tabs: "Login" and "Register". A yellow error message at the top states: "[Client] Password must be at least 8 characters". Below the message are three input fields: "Email" containing `ads@test.com`, "Username" containing `asd`, and "Password" (partially visible). A blue "Register" button is located below the inputs.

[Client] Password must be at least 8 characters

Email

Username

Password ...

Confirm ...

Register

password must be at least 8 characters

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 na569-dev-9569cb3ae6c9.herokuapp.com/project/register.php. The page has two navigation links: "Login" and "Register". Below them is a yellow error message: "[Client] Passwords must match". The registration form consists of four input fields: "Email" (value: asd@teset.com), "Username" (value: asd), "Password" (value:), and "Confirm" (value:). A red validation error message "passwords must match" is displayed below the "Confirm" field. At the bottom, there is a "Register" button.

[Client] Passwords must match

Email asd@teset.com

Username asd

Password

Confirm

Register

passwords must 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 na569-dev-9569cb3ae6c9.herokuapp.com/project/register.php. The page has two navigation links: "Login" and "Register". Below them is a yellow error message: "[Client] Email cannot be empty". Another yellow error message "[Client] Username cannot be empty" is partially visible at the bottom. The registration form fields are identical to the previous screenshot: "Email" (value: asd@teset.com), "Username" (value: asd), "Password" (value:), and "Confirm" (value:).

[Client] Email cannot be empty

[Client] Username cannot be empty

[Client] Username cannot be empty
[Client] Password cannot be empty
[Client] Confirm password cannot be empty
[Client] Invalid email address
[Client] Username must only contain a-z, 0-9, _, or -
[Client] Password must be at least 8 characters

Email

Username

Password

Confirm

each field required

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 chosen email is not available.

Email

Username

Password

Confirm

email not available

Checklist Items (1)

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

Login Register

The chosen username is not available.

Email

Username

Password

Confirm

Register

username not available

Checklist Items (1)

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

Login Register

Successfully registered!

Email

Username

Password

Confirm

Register

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

```
<form onsubmit="return validate(this)" method="POST">      You, last month • adding and
  <div>
    <label for="email">Email</label>
    <input type="text" name="email" value=<?php se($email); ?>" required />
  </div>
  <div>
    <label for="username">Username</label>
    <input type="text" name="username" value=<?php se($username); ?>" required />
  </div>
  <div>
    <label for="pw">Password</label>
    <input type="password" id="pw" name="password" required />
  </div>
  <div>
    <label for="confirm">Confirm</label>
    <input type="password" name="confirm" required />
  </div>
  <input type="submit" value="Register" />
</form>
<script>
```

Screenshot of form code for register.php

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>
  //na569, 4/1/24
  function validate(form) {
    //TODO 1: implement JavaScript validation
    //ensure it returns false for an error and true for success

    let email = form.email.value;
    let username = form.username.value; You, 2 days ago * javascript client side validation
    let password = form.password.value;
    let confirmPassword = form.confirm.value;
    let isValid = true;

    let validEmail = /^[a-zA-Z0-9_.\+-]+@[a-zA-Z.-]+\.(a-zA-Z){2,6}$/;
    let validUsername = /^[a-zA-Z0-9_-]{3,16}$/;

    if (email.length < 1) {
      flash("[client] Email cannot be empty", "warning");
      isValid = false;
    }
    if (username.length < 1) {
      flash("[client] Username cannot be empty", "warning");
      isValid = false;
    }
    if (password.length < 1) {
      flash("[client] Password cannot be empty", "warning");
      isValid = false;
    }
    if (confirmPassword.length < 1) {
      flash("[client] Confirm password cannot be empty", "warning");
      isValid = false;
    }
    if (!validEmail.test(email)) {
      flash("[client] Invalid email address", "warning");
      isValid = false;
    }
    if (!validUsername.test(username)) {
      flash("[client] Invalid username", "warning");
      isValid = false;
    }
  }
}
```

js validation part 1

Checklist Items (1)

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

```
<script>
  function validate(form) {
    isValid = false;
  }
  if (username.length < 1) {
    flash("[Client] Username cannot be empty", "warning");
    isValid = false;
  }
  if (password.length < 1) {
    flash("[Client] Password cannot be empty", "warning");
    isValid = false;
  }
  if (confirmPassword.length < 1) {
    flash("[Client] Confirm password cannot be empty", "warning");
    isValid = false;
  }
}
```

```

if (!isValidEmail.test(email)) {
    flash("[Client] Invalid email address", "warning");
    isValid = false;
}
if (!isValidUsername.test(username)) {
    flash("[Client] Username must only contain a-z, 0-9, _, or -", "warning");
    isValid = false;
}
if (password.length < 8) {
    flash("[Client] Password must be at least 8 characters", "warning");
    isValid = false;
}
if (password !== confirmPassword) {
    flash("[Client] Passwords must match", "warning");
    isValid = false;
}
return isValid;
//na569,4/1/24      You, 1 second ago • Uncommitted changes

]} <- #28-76 function validate(form)
</script>

```

js validation part 2

Checklist Items (1)

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

```

//validate, na569, 4/1/24
if (!is_valid_email($email)) {
    flash("Invalid email address", "danger");
    $hasError = true;
}
if (!is_valid_username($username)) {
    flash("Username must only contain a-z, 0-9, _, or -", "danger");
    $hasError = true;
}
if (empty($password)) {
    flash("password must not be empty", "danger");
    $hasError = true;
}
if (empty($confirm)) {
    flash("Confirm password must not be empty", "danger");
    $hasError = true;
}
if (!is_valid_password($password)) {
    flash("Password too short", "danger");
    $hasError = true;
}
if (
    strlen($password) > 0 && $password !== $confirm
) {
    flash("Passwords must match", "danger");
    $hasError = true;
}

```

php validation part 1

Checklist Items (2)

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

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

```

        $hasError = true;
    }
    if (!$hasError) {
        //Todo: 4
        $hash = password_hash($password, PASSWORD_BCRYPT);
        $db = getDB();
        $stmt = $db->prepare("INSERT INTO Users (email, password, username) VALUES(:email, :password, :username)");
        try {
            $stmt->execute([":email" => $email, ":password" => $hash, ":username" => $username]);
            flash("Successfully registered!", "success");
        } catch (PDOException $e) {
            users_check_duplicate($e->errorInfo);
        }
    } //na569, 4/1/24 <- #120-131 if (!$hasError)
} <- #80-132 if (isset($_POST["email"]) && isset($_POST["password"])) && is...
?>
<?php

```

php validation part 2

Checklist Items (2)

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

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

Task #4 - Points: 1

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

Checklist

*The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large

Properties Data Process

SELECT * FROM `Users` LIMIT 100

Search results Cost: 95ms Total 7

	Q	id	email	password	created	modified	username
	Q	id	email	password	created	modified	username
>	1	na569@njit.edu	\$2y\$10\$qtu0f0wZXsazx6Gt	2024-02-26 16:07:59	2024-02-26 16:07:59	na569	
>	2	naf@test.com	\$2y\$10\$ioz6K0hYCsWx6wC	2024-02-26 16:09:04	2024-02-26 16:09:04	naf	
>	3	nafisa2004@gmail.com	\$2y\$10\$cGbCGo9e0dNbgT	2024-03-25 01:32:21	2024-03-25 01:32:21	nafisa37	
>	4	naf2@test.com	\$2y\$10\$TbS5g9eBvEadzLEx	2024-03-25 18:20:48	2024-03-25 18:20:48	naf2	
>	5	df@test.com	\$2y\$10\$8gBzAnGtkes3LqGI	2024-03-29 15:17:28	2024-03-29 15:17:28	123	
>	6	df2@gmail.com	\$2y\$10\$wGlnfl3wfQ02R8	2024-04-01 04:41:45	2024-04-01 05:05:50	test2	
>	7	naf@test2.com	\$2y\$10\$25Rv5dQggJ6v4p	2024-04-01 17:15:48	2024-04-01 17:15:48	testing	

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 registration page loads and a form is displayed where users can input their email, username, password, and confirm password. When they submit the form, the validate JS functions does client-side validation. Warnings are displayed if any of the input is invalid. If everything is valid, the form data is sent to the server. On the server side, PHP does server-side validation to make sure everything is correct as well. Then, the password is hashed and the user data gets saved to the database.

Task #6 - Points: 1

i Details:

Should end in /pull/#

URL #1

<https://github.com/nafisa37/na569-it202-008/pull/19>**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

 [←](#) [→](#) [⌂](#)[na569-dev-9569cb3ae6c9.herokuapp.com/project/home.php](#)[Home](#)[Profile](#)[Create Role](#)[List Roles](#)[Assign Roles](#)[Logout](#)

Welcome, na569

Home

logged in successfully

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

#5 Show success login message

[Client] Invalid username or password

Email/Username as

Password

Login

invalid username or password

Checklist Items (1)

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

Login Register

[Client] Email or username cannot be empty

[Client] Password cannot be empty

[Client] Invalid username or password

Email/Username

Password

Login

each field required

Checklist Items (1)

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

Login Register

Email/Username not found

Email/Username

asd@test.com

Password

.....

Login

account doesn't exist

Checklist Items (1)

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

The screenshot shows a web browser window with the URL na569-dev-9569cb3ae6c9.herokuapp.com/project/login.php. The page has a header with 'Login' and 'Register' buttons. Below the header, there is a dark grey error message box containing the text 'Invalid password'. Below this, there are two input fields: 'Email/Username' and 'Password', both of which are empty. A 'Login' button is located below the password field. At the bottom of the page, there is a horizontal bar with the text 'invalid password'.

Checklist Items (1)

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

The screenshot shows a web browser window with the URL na569-dev-9569cb3ae6c9.herokuapp.com/project/login.php. The page has a header with 'Login' and 'Register' buttons. Below the header, there are two input fields: 'Email/Username' containing 'na569' and 'Password' containing several dots ('.....'). A 'Login' button is located below the password field. At the bottom of the page, there is a horizontal bar with the text 'invalid password'.

correct sign in info

Checklist Items (2)

#3 Include correct data where username is used to login

#4 Include correct data where email is used to login

```
[Mon Apr  1 14:38:33 2024] [::1]:61142 Accepted
[Mon Apr  1 14:38:33 2024] [::1]:61143 Accepted
[Mon Apr  1 14:38:33 2024] [::1]:61142 [302]: POST /project/login.php
[Mon Apr  1 14:38:33 2024] [::1]:61142 Closing
[Mon Apr  1 14:38:33 2024] Session data: array (
    'user' =>
    array (
        'id' => 1,
        'email' => 'na569@njit.edu',
        'username' => 'na569',
        'roles' =>
        array (
            0 =>
            array (
                'name' => 'Admin',
            ),
        ),
    ),
    'flash' =>
    array (
        0 =>
        array (
            'text' => 'Welcome, na569',
            'color' => 'info',
        ),
    ),
)
[Mon Apr  1 14:38:33 2024] [::1]:61143 [200]: GET /project/home.php
[Mon Apr  1 14:38:33 2024] [::1]:61143 Closing
[Mon Apr  1 14:38:33 2024] [::1]:61145 Accepted
[Mon Apr  1 14:38:33 2024] [::1]:61146 Accepted
[Mon Apr  1 14:38:33 2024] [::1]:61145 [200]: GET /project/styles.css
```

session data set

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
-------	--------	-------

```
<script>
    function validate(form) [
        //TODO 1: implement Javascript validation
        //ensure it returns false for an error and true for success

        //TODO update clientside validation to check if it should
        //valid email or username
        //na569, 4/2/24

        let emailUser = form.email.value;
        let password = form.password.value;
        let isValid = true;

        let validEmail = /^[a-zA-Z0-9_\.\+\-]+@[a-zA-Z\.\-]+\.(a-zA-Z\.){2,6}$/;
        let validUsername = /^[a-zA-Z0-9_-]{3,16}$/;

        if (emailUser.length < 1) {
            flash("[Client] Email or username cannot be empty", "warning");
            isValid = false;
        }
        if (password.length < 1) {
            flash("[Client] Password cannot be empty", "warning");
            isValid = false;
        }

        if (!validEmail.test(emailUser)) {
            if (!validUsername.test(emailUser)) {
                flash("[Client] Invalid username or password", "warning");
                isValid = false;
            }
        } <- #41-46 if (!validEmail.test(emailUser))
        return isValid;

    ] <- #18-49 function validate(form)
</script>
```

combined field for username and email

Checklist Items (3)

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

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

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

```
//TODO 3
$hasError = false;
if (empty($email)) {
    flash("Email must not be empty");
    $hasError = true;
} //na569, 4/1/24
if (str_contains($email, "@")) {
    //sanitize
    //$email = filter_var($email, FILTER_SANITIZE_EMAIL);
    $email = sanitize_email($email);
```

```

$email = sanitize_email($email);
//validate
/*if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    flash("Invalid email address");
    $hasError = true;
}*/
if (!is_valid_email($email)) {
    flash("Invalid email address");
    $hasError = true;
}
} else {
    if (!is_valid_username($email)) {
        flash("Invalid username");
        $hasError = true;
    }
} <- #76-81 else
if (empty($password)) {
    flash("Password must not be empty");
    $hasError = true;
}
if (!is_valid_password($password)) {
    flash("Password too short");
    $hasError = true;
}

```

php validation

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:

The login page loads and a form is there for users to input their username/email and password. Javascript client-side validation checks to make sure they fit the required criteria and displays warnings if not. If the data is valid, PHP does client-side validation. The database then fetches the user details and stores them in the session. The user is taken to the home page. When a user logs in successfully, their details and user roles are stored in the \$_SESSION variables. The session data is stored on the server and is able to be accessed across different PHP scripts.

Task #4 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

<https://github.com/nafisa37/na569-it202-008/pull/41>

User Logout (1 pt.)

[^COLLAPSE ^](#)

Task #1 - Points: 1

Text: Capture the following screenshots

Checklist

*The checkboxes are for your own tracking

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



← → ⌂ na569-dev-9569cb3ae6c9.herokuapp.com/project/home.php

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

Welcome, na569

Home

home page logged in

Checklist Items (1)

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

The screenshot shows a web browser window with the URL `na569-dev-9569cb3ae6c9.herokuapp.com/project/login.php`. The page has a header with 'Login' and 'Register' buttons. A green banner at the top displays the message 'Successfully logged out'. Below the banner are input fields for 'Email/Username' and 'Password', and a 'Login' button.

Successfully logged out

Email/Username

Password

Login

logged out page

Checklist Items (1)

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

The screenshot shows a terminal or code editor displaying PHP code for a file named `logout.php`. The code includes session management and a flash message for successful logout, followed by a header redirect.

```
html > project > 🐱 logout.php
...
//na569, 4.1.24
<?php
session_start();
require(__DIR__ . "/../../lib/functions.php");
reset_session();

flash("Successfully logged out", "success");
header("Location: login.php");
```

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.

Task #2 - Points: 1

Text: Include pull request links related to this feature

① Details:

Should end in /pull/#

URL #1

<https://github.com/nafisa37/na569-it202-008/pull/26>

Basic Security Rules and Roles (2 pts.)

▲ COLLAPSE ▲

Task #1 - Points: 1

Text: Authentication Screenshots

Task #2 - Points: 1

Text: Authorization Screenshots

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

Task #4 - Points: 1

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

Task #5 - Points: 1

Text: Include pull request links related to this feature

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**

← → ⌂ na569-dev-9569cb3ae6c9.herokuapp.com/project/profile.php

Home Profile Create Role List Roles Assign Roles Logout

Email na569@njit.edu

Username na569

Password Reset

Current Password

New Password

Confirm Password

Update Profile

profile page

Checklist Items (4)

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

#2 Should have thoughtful CSS applied

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

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

Task #2 - Points: 1

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

Details:

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

Response:

When the profile page is visited, the page checks to make sure that the user is logged in. It uses the session data. After making sure the user is verified, `get_user_email()` and `get_username()` are called to retrieve data that is populated into the input fields. The `se()` function allows the user to view and update their information as needed.

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)

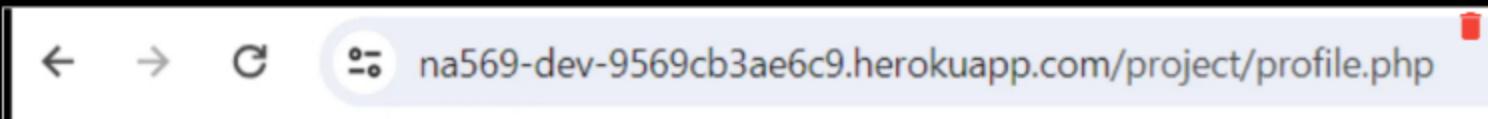
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



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

Email

Username

Password Reset

Current Password

New Password

Confirm Password

[Update Profile](#)

before email change

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)

The screenshot shows a web browser window with the URL `na569-dev-9569cb3ae6c9.herokuapp.com/project/profile.php`. The page has a header with links for Home, Profile, and Logout. A green success message bar at the top says "Profile saved". Below it is a form with fields for Email (`naf@test3.com`) and Username (`testing`). Below the form is a "Password Reset" section with fields for Current Password, New Password, and Confirm Password. At the bottom is a "Update Profile" button.

← → G na569-dev-9569cb3ae6c9.herokuapp.com/project/profile.php

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

Profile saved

Email

Username

Password Reset

Current Password

New Password

Confirm Password

[Update Profile](#)

Update Profile

after email change

Checklist Items (1)

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

← → ⌂ na569-dev-9569cb3ae6c9.herokuapp.com/project/profile.php

Profile

Email

Username

Password Reset

Current Password

New Password

Confirm Password

Update Profile

before username change

Checklist Items (1)

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

← → ⌂ na569-dev-9569cb3ae6c9.herokuapp.com/project/profile.php

Profile

Profile saved

Email

Username

Password Reset

Current Password

New Password

Confirm Password

after username change

Checklist Items (1)

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

← → ⌂ na569-dev-9569cb3ae6c9.herokuapp.com/project/profile.php

Home Profile Logout

Profile saved

Password reset

Email

Username

Password Reset

Current Password

New Password

Confirm Password

password reset

Checklist Items (1)

#5 Demonstrate the success message of updating password

← → ⌂ na569-dev-9569cb3ae6c9.herokuapp.com/project/profile.php

Home Profile Logout

The chosen email is not available.

Email

Username

Username testing1

Password Reset

Current Password

New Password

Confirm Password

email already 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 `na569-dev-9569cb3ae6c9.herokuapp.com/project/profile.php`. The page has a navigation bar with links for Home, Profile, and Logout. Below the navigation, there is a yellow error message box containing the text "The chosen username is not available.". The main form fields are: Email (`naf@test3.com`), Username (`testing1`), Current Password, New Password, and Confirm Password. At the bottom of the form is a button labeled "Update Profile".

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)

This screenshot is identical to the one above, showing the same web browser interface with the URL `na569-dev-9569cb3ae6c9.herokuapp.com/project/profile.php`. It displays the same validation error message "username in use" and the same form fields for updating the profile.

Profile saved

Current password is invalid

Email

Username

Password Reset

Current Password

New Password

Confirm Password

current password doesn't match

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:

When the form is submitted, the code validates the information entered. If the data is valid, the SQL update query updates the information in the database. The password validity is also checked and updated. After submitting, the new profile data is fetched from the database and stored in the session.

Task #5 - Points: 1

Text: Include pull request links related to this feature

i Details:

URL #1

<https://github.com/nafisa37/na569-it202-008/pull/24>

Misc (1 pt.)

[^COLLAPSE ^](#)

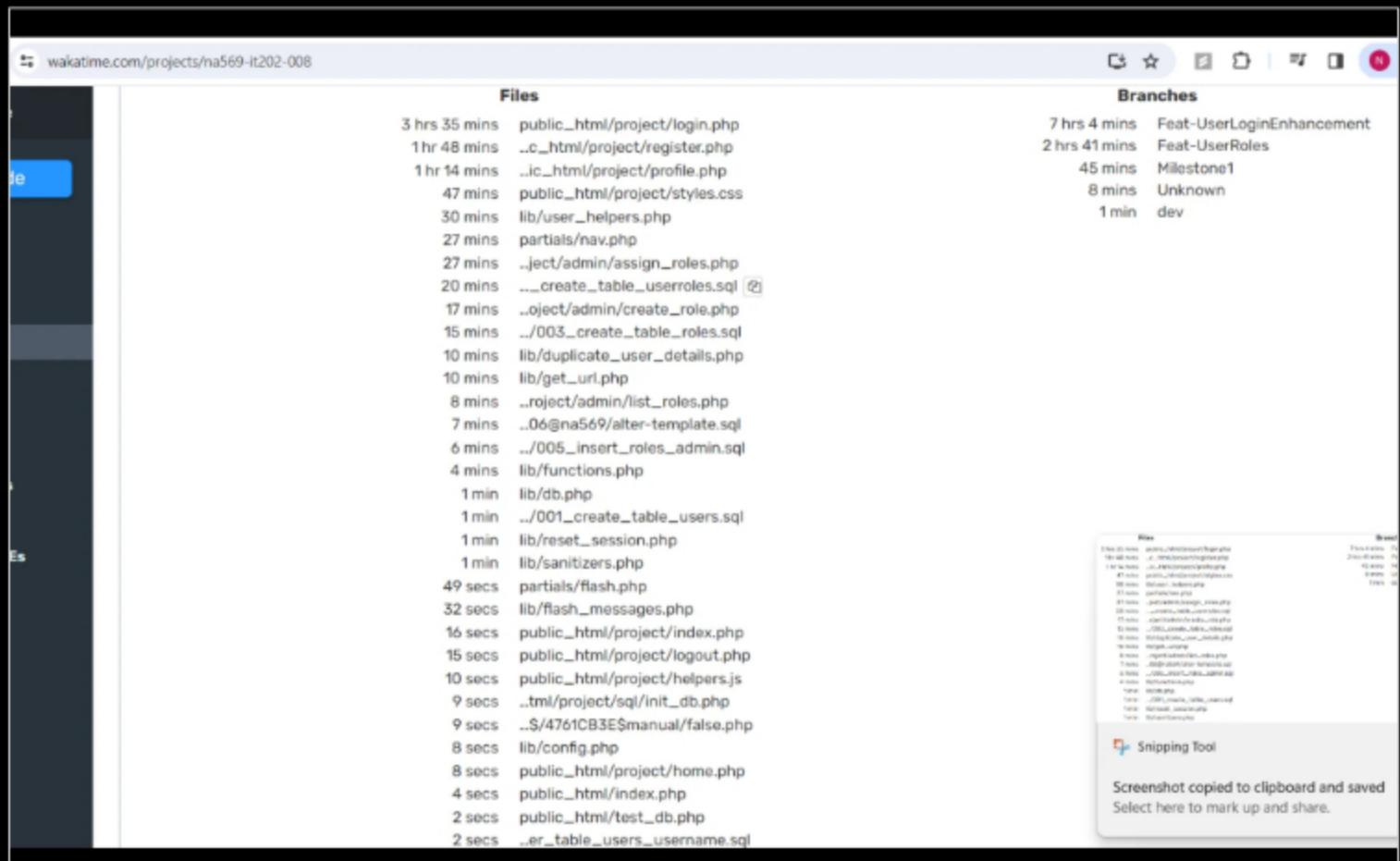
Task #1 - Points: 1

Text: Screenshot of wakatime

Task Screenshots:

Gallery Style: Large View

Small Medium Large



wakatime screenshot

Task #2 - Points: 1

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

Task Screenshots:

Small

Medium

Large

Nafisa's Board

View 1 + New view

Filter by keyword or by field

Todo	In Progress	Done
0	0	9
This item hasn't been started	This is actively being worked on	<ul style="list-style-type: none"> na569-it202-008 #29 MS1-User will be able to register a new account na569-it202-008 #32 MS1-User will be able to login to their account (given they enter the correct credentials) na569-it202-008 #28 MS1-User will be able to logout na569-it202-008 #33 MS1-Basic security rules implemented na569-it202-008 #34 MS1-Basic Roles implemented
+ Add item	+ Add item	+ Add item

screenshot of 9 tasks for Milestone 1

●
^COLLAPSE ^
Task #3 - Points: 1
Text: Provide a direct link to the project board on GitHub
URL #1
<https://github.com/users/nafisa37/projects/1/views/1>
●
^COLLAPSE ^
Task #4 - Points: 1
Text: Provide a direct link to the login page from your prod instance
URL #1
<https://na569-prod-74aadc581478.herokuapp.com/project/login.php>