

Submission Worksheet

Submission Data

Course: IT202-450-M2025

Assignment: IT202 Milestone 1

Student: Mark Y. (may23)

Status: Submitted | **Worksheet Progress:** 99%

Potential Grade: 9.95/10.00 (99.50%)

Received Grade: 0.00/10.00 (0.00%)

Started: 7/7/2025 12:06:12 AM

Updated: 7/8/2025 9:41:48 PM

Grading Link: <https://learn.ethereallab.app/assignment/v3/IT202-450-M2025/it202-milestone-1/grading/may23>

View Link: <https://learn.ethereallab.app/assignment/v3/IT202-450-M2025/it202-milestone-1/view/may23>

Instructions

- Overview Link: <https://youtu.be/IDtqdaLwcVg>

1. Refer to Milestone1 of this doc:

<https://docs.google.com/document/d/1XE96a8DQ52Vp49XACBDTNCq0xYDt3kF29cO88EWVwfo/view>

2. Ensure you read all instructions and objectives before starting.

3. Ensure you've gone through each lesson related to this Milestone

4. Switch to the Milestone1 branch

1. git checkout Milestone1 (ensure proper starting branch)
2. git pull origin Milestone1 (ensure history is up to date)

5. Fill out the below worksheet

- Ensure there's a comment with your UCID, date, and brief summary of the snippet in each screenshot
- Ensure proper styling is applied to each page
- Ensure there are no visible technical errors; only user-friendly messages are allowed

6. Once finished, click "Submit and Export"

7. Locally add the generated PDF to a folder of your choosing inside your repository folder and move it to Github

1. git add .
2. git commit -m "adding PDF"
3. git push origin Milestone1
4. On Github merge the pull request from Milestone1 to dev
5. On Github create a pull request from dev to prod and immediately merge. (This will trigger the prod deploy to make the heroku prod links work)

8. Upload the same PDF to Canvas

9. Sync Local

10. git checkout dev

11. git pull origin dev

Section #1: (2 pts.) Feature: User Will Be Able To Register A New Account

≡ Task #1 (0.67 pts.) - Validation

Progress: 100%

Details:

- Show the relevant code snippets for each validation layer
- Show the relevant demo of each validation layer
- Briefly explain how the validation steps work at each layer

≡ Task #1 (0.33 pts.) - HTML Form/Validation

Progress: 100%

Details:

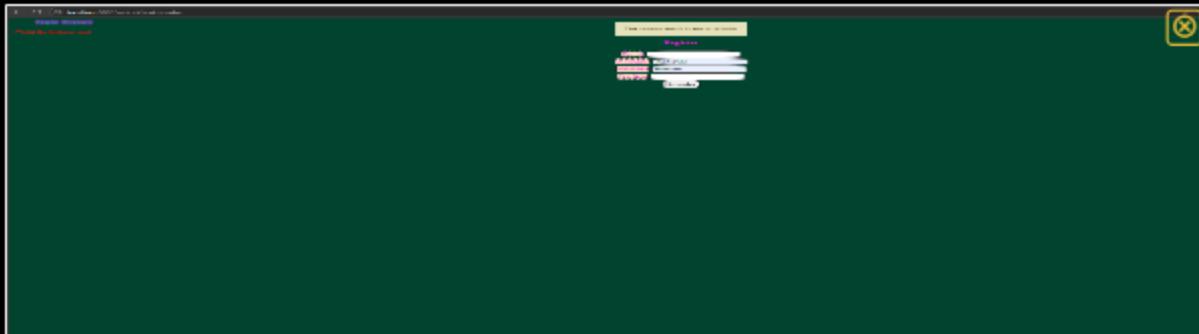
- System should allow the user to correct the error without wiping/clearing the form (for the PHP side this includes sticky-form logic to keep the username/email address entries)

❑ Part 1:

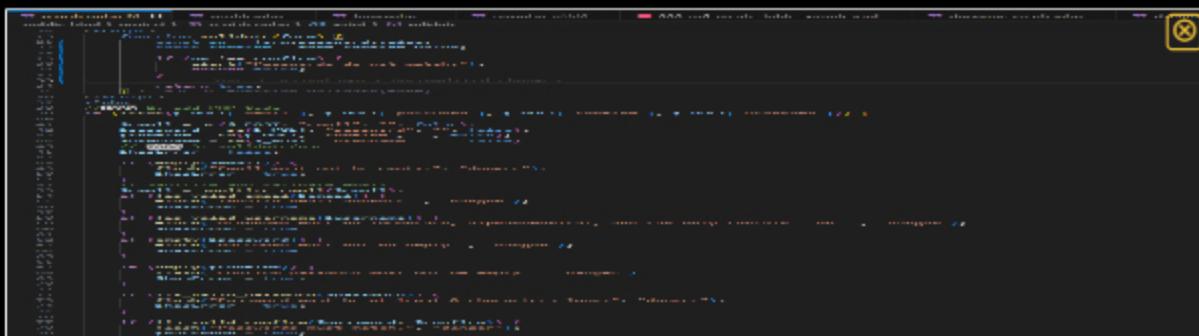
Progress: 100%

Details:

- Show the code related to the register page's form (HTML validation)
- Show examples of each validation message (you may be able to capture this in one or few screenshots)



validation message that I already created an account with that email



validation code



Saved: 7/7/2025 12:09:42 AM

Part 2:

Progress: 100%

Details:

- Briefly explain the validation step (i.e, what you chose, how they solve the requirement)

Your Response:

The validation step ensures that user input is accurate and secure before processing. On the client side, HTML5 attributes like required, type="email", and minlength check for basic input rules, while JavaScript adds a check to confirm that the password and confirmation match. On the server side, PHP re-validates all inputs to prevent bypassing client checks, sanitizes the email, verifies the username format, checks password strength and matching, and handles duplicate entries or database errors safely.



Saved: 7/7/2025 12:09:42 AM

Task #2 (0.33 pts.) - JS Validation (validate() function)

Progress: 100%

Part 1:

Progress: 100%

Details:

- Show the code related to the register page's validate() function
- Show examples of each validation message [username format, email format, password format, password doesn't match confirm](you may be able to capture this in one or few screenshots)
- Note: You may need to use dev tools to temporarily apply novalidate to the form tag

```
POST /register.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4453.102 Safari/537.36
Content-Length: 102
Host: localhost:8080
Connection: keep-alive
Referer: http://localhost:8080/register.html

username=JohnDoe&password=12345678&confirm_password=12345678&email=johndoe@example.com&checkbox=1

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 102
Date: Mon, 07 Jul 2025 12:09:42 GMT
Server: Apache/2.4.41 (Ubuntu)
Set-Cookie: PHPSESSID=1234567890; expires=Thu, 07-Jul-2025 12:09:42 UTC; path=/; HttpOnly; Secure
Content-Encoding: gzip

{
    "username": "The username must be between 3 and 20 characters long.", 
    "password": "The password must be between 6 and 20 characters long.", 
    "confirm_password": "The password and confirmation password must match.", 
    "email": "The email address is invalid.", 
    "checkbox": "The checkbox must be checked to proceed."
}
```

validation code

The chosen email is not available.

Register

Email: may23@njit.edu

Please include an '@' in the email address. 'may23njit.edu' is missing an '@'.

Confirm: *****

Responsible

email domain

Username must be lowercase, alphanumeric, and can only contain _ or -.

Register

Email: may23@njit.edu

Username: may23@njit.edu

Password: *****

Confirm: *****

Responsible

username format

localhost:3000 says:

Passwords do not match.

OK

Email: may23@njit.edu

Username: marko132

Password: *****

Confirm: *****

Responsible

password does not match

The chosen email is not available.

Register

Email: may23@njit.edu

Username: marko132

Please lengthen this text to 6 characters or more (you are currently using 2 characters).

Responsible

password format



Part 2:

Progress: 100%

Details:

- Briefly explain the validation step (i.e, what you chose, how they solve the requirement)

Your Response:

The validation step uses HTML5 and JavaScript to ensure user input meets requirements before submission. HTML5 checks for required fields, email format, and password length. JavaScript adds custom checks for username format and password confirmation. This prevents invalid data and improves security and user experience.



Saved: 7/7/2025 12:16:54 AM

≡ Task #3 (0.33 pts.) - PHP Validation (steps before the DB call)

Progress: 100%

Part 1:

Progress: 100%

Details:

- Show the code related to the register page's PHP validation
- Show examples of each validation message [username format, email format, password format, invalid password, password doesn't match confirm, username taken, email taken] (you may be able to capture this in one or few screenshots)
- Note: You may need to use dev tools to temporarily apply `novalidate` to the form tag and temporarily override the validate() function or use the provided helper script in the instructions
- Include the outputs related to username/email already in use

A screenshot of a code editor displaying PHP code. The code includes logic for validating user input, such as checking if an email is available and ensuring passwords match. It also shows the use of session variables and database queries for user registration.

PHP validation

The chosen email is not available

Register

Email:

Username:

Password:

Please lengthen this text to 8 characters or more (you are currently using 2 characters).

password format

A screenshot of a registration dialog box. The title bar says "localhost:2000 says". The message area says "Passwords do not match." A blue "OK" button is at the bottom right. Below the message are four input fields: Email (may23@njit.edu), Username (marko132), Password (redacted), and Confirm (redacted). A "Register" button is at the bottom.

password does not match

A screenshot of a registration dialog box. The title bar says "localhost:2000 says". The message area says "Passwords do not match." A blue "OK" button is at the bottom right. Below the message are four input fields: Email (may23@njit.edu), Username (marko132), Password (redacted), and Confirm (redacted). A "Register" button is at the bottom.

username format

A screenshot of a registration dialog box. The title bar says "localhost:2000 says". The message area says "The chosen email is not available." A blue "OK" button is at the bottom right. Below the message are four input fields: Email (may23@njit.edu), Username (marko132), Password (redacted), and Confirm (redacted). A "Register" button is at the bottom. An orange error icon is next to the "Email" field.

email domain

A screenshot of a registration dialog box. The title bar says "localhost:2000 says". The message area says "Please include an '@' in the email address. 'may23@njit.edu' is missing an '@'." A blue "OK" button is at the bottom right. Below the message are four input fields: Email (may23@njit.edu), Username (marko132), Password (redacted), and Confirm (redacted). A "Register" button is at the bottom. An orange error icon is next to the "Email" field.

Part 2:

Progress: 100%

Details:

- Briefly explain the validation step (i.e., what you chose, how they solve the requirement)

Your Response:

Your Response:

The PHP validation code ensures all user inputs are safe and meet the system's rules. It checks if fields are filled, uses helper functions to validate email and username format, ensures the password is strong and matches the confirmation, and finally attempts to insert the user into the database. If a duplicate email or username is detected, the `users_check_duplicate()` helper displays a specific error. This two-level validation approach protects the backend from bad input and enhances user feedback.



Saved: 7/7/2025 12:21:43 AM

✉ Task #2 (0.67 pts.) - Related URLs

Progress: 100%

Details:

- Include the direct link to this file from the Milestone branch (should end in .php)
- Include the heroku prod link to this file (Just grab the base prod url and manually enter the path to the file)
- Include the related pull request link for this feature

URL #1

<https://github.com/Mark-5555/may23->



URL

<https://github.com/Mark-555>



IT202-450/prod/public_html/project/register.php

URL #2

<https://may23-it202-450-prod->

a1263d847137.herokuapp.com/project/register.php



URL

<https://may23-it202-450-prod->



URL #3

<https://github.com/Mark-5555/may23->

IT202-450/pull/14



URL

<https://github.com/Mark-555>



Saved: 7/8/2025 6:33:43 PM

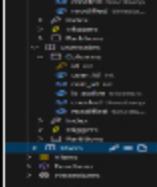
✉ Task #3 (0.67 pts.) - Database - Users table

Progress: 100%

Details:

- Add a screenshot of the database view from vs code showing a valid registered user (preferably a recent one during evidence capturing)





valid user



Saved: 7/7/2025 12:44:20 AM

Section #2: (2 pts.) Feature: User Will Be Able To Login To Their Account

Progress: 100%

≡ Task #1 (0.67 pts.) - Validation

Progress: 100%

Details:

- Show the relevant code snippets for each validation layer
- Show the relevant demo of each validation layer
- Briefly explain how the validation steps work at each layer

≡ Task #1 (0.33 pts.) - HTML Form/Validation

Progress: 100%

Details:

- System should allow the user to correct the error without wiping/clearing the form (for the PHP side this includes sticky-form logic to keep the username/email address entries)

❑ Part 1:

Progress: 100%

Details:

- Show the code related to the login page's form (HTML validation)
- Show examples of each validation message (you may be able to capture this in one or few screenshots)

```
<h3>Login</h3>
<form onsubmit="return validate(this)" method="POST">
  <div>
    <label for="email">Email or Username</label>
    <input type="text" name="email" required="required" />
```



```
<input id="email" type="text" name="email" required />
</div>
<div>
    <label for="pw">Password</label>
    <input type="password" id="pw" name="password" required minlength="8" />
</div>
<input type="submit" value="Login" />
</form>
```

HTML code

Invalid login attempt

empty field

Saved: 7/8/2025 6:25:24 PM

=, Part 2:

Progress: 100%

Details:

- Briefly explain the validation step (i.e., what you chose, how they solve the requirement)

Your Response:

The form uses built-in HTML validation attributes like `required` to ensure fields aren't left blank, and `minlength="8"` on the password field to enforce a minimum password length. These validations are automatically handled by the browser before submission, preventing obvious empty or too-short inputs.

Saved: 7/8/2025 6:25:24 PM

≡ Task #2 (0.33 pts.) - JS Validation (validate() function)

Progress: 100%

☒ Part 1:

Progress: 100%

Details:

- Show the code related to the login page's validate() function
- Show examples of each validation message [username format, email format, password format] (you may be able to capture this in one or few screenshots)
- Note: You may need to use dev tools to temporarily apply `novalidate` to the form tag

```
<script>
  function validate(form) {
    //TODO 1: implement JavaScript validation (you'll do this on your own towards the end of Milestone1)
    //ensure it returns false for an error and true for success

    return true;
} <- #17-22 function validate(form)
</script>
```

JS code



Saved: 7/8/2025 6:25:34 PM

☒ Part 2:

Progress: 100%

Details:

- Briefly explain the validation step (i.e, what you chose, how they solve the requirement)

Your Response:

A placeholder validate(form) function is included in the



Saved: 7/8/2025 6:25:34 PM

≡ Task #3 (0.33 pts.) - PHP Validation (steps before the DB call)

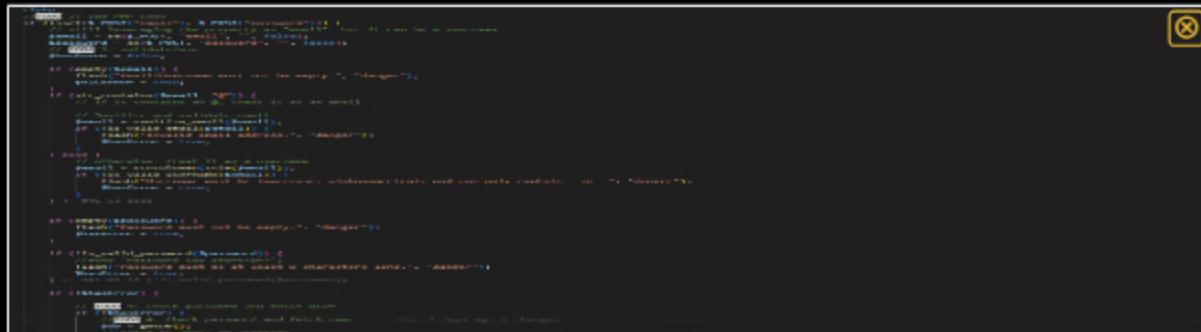
Progress: 100%

☒ Part 1:

Progress: 100%

Details:

- Show the code related to the login page's PHP validation
- Show examples of each validation message [username format, email format, password format, invalid password] (you may be able to capture this in one or few screenshots)
- Note: You may need to use dev tools to temporarily apply `novalidate` to the form tag and temporarily override the validate() function or use the provided helper script in the instructions
- Include the outputs related to user doesn't exist and php-side password mismatch



A screenshot of a code editor displaying PHP code. The code includes form fields for email and password, and a validate() function that checks for empty fields and password length. It also includes logic for handling user existence and password mismatch.

```
<?php
// Validate form inputs
if (empty($_POST['email'])) {
    $error[] = "Email is required";
} elseif (!filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
    $error[] = "Email is not valid";
}

if (empty($_POST['password'])) {
    $error[] = "Password is required";
} elseif (strlen($_POST['password']) < 8) {
    $error[] = "Password must be at least 8 characters long";
}

// Check if user exists
if (empty($error)) {
    $user = User::findUserByEmail($_POST['email']);
    if ($user === null) {
        $error[] = "User does not exist";
    } elseif (!password_verify($_POST['password'], $user->password)) {
        $error[] = "Incorrect password";
    }
}

// Output errors
if (!empty($error)) {
    echo json_encode(['errors' => $error]);
}
?>
```

PHP code



Saved: 7/8/2025 6:26:23 PM

Progress: 100%

Details:

- Briefly explain the validation step (i.e, what you chose, how they solve the requirement)

Your Response:

On form submission, PHP re-validates all inputs for security and accuracy. It checks whether the email/username is empty, determines if the input is a valid email or username based on its format, and verifies that the password is non-empty and meets the length requirement. This ensures no invalid or malicious data is processed and provides specific error messages to guide the user while preserving entered values (sticky form logic).



Saved: 7/8/2025 6:26:23 PM

Progress: 100%

Task #2 (0.67 pts.) - Related URLs

Details:

Details:

- Include the direct link to this file from the Milestone branch (should end in .php)
- Include the heroku prod link to this file (Just grab the base prod url and manually enter the path to the file)
- Include the related pull request link for this feature

URL #1

[https://github.com/Mark-5555/may23-](https://github.com/Mark-5555/may23-IT202-450-prod/public_html/project/login.php)

URL

[https://github.com/Mark-555](https://github.com/Mark-5555/may23-IT202-450-prod/public_html/project/login.php)

IT202-450/prod/public_html/project/login.php

URL #2

<https://may23-it202-450-prod-a1263d847137.herokuapp.com/project/login.php>

URL

[https://may23-it202-450-prod](https://may23-it202-450-prod-a1263d847137.herokuapp.com/project/login.php)

URL #3

<https://github.com/Mark-5555/may23-IT202-450/pull/15>

URL

[https://github.com/Mark-555](https://github.com/Mark-5555/may23-IT202-450/pull/15)

Saved: 7/8/2025 6:33:17 PM

☒ Task #3 (0.67 pts.) - User Session Evidence

Progress: 100%

Details:

- From the heroku logs (Dashboard -> More button -> view Logs), capture
- It should show the id, username, email, and roles

A screenshot of a browser window displaying the Heroku logs for a specific application. The logs are presented in a table format with columns for timestamp, log level, and message. The messages contain user session data, including user IDs, usernames, emails, and roles. The browser interface includes a navigation bar with tabs like Home, Log, and Settings.

Heroku logs



Saved: 7/8/2025 6:59:53 PM

Section #3: (1 pt.) Feature: User Will Be Able To Logout

≡ Task #1 (1 pt.) - Evidence

☒ Part 1:

Details:

- Show the successful logout message
- Show the code snippet of the logout page

```
You, 2 days ago | 1 author (You)  
<?php  
// require functions.php to pull in flash()  
require(__DIR__ . "/../../lib/functions.php");  
reset_session(); // clear session data and start a new session  
flash("You have been logged out", "success");  
header("Location: $BASE_PATH/login.php"); // redirect back to login
```

logout code

You have been logged out

Login

Email or Username:

Password:

Login

logout message



Saved: 7/8/2025 7:09:45 PM

☞ Part 2:

Details:

- Include the pull request url for this feature

URL #1

<https://github.com/Mark-5555/may23-IT2024505>



<https://github.com/Mark-5555/may23-IT2024505>



Saved: 7/8/2025 7:09:45 PM

Part 3:

Progress: 100%

Details:

- Briefly explain how the logout process works and how the user is officially logged off

Your Response:

When a user clicks the logout button (from the nav.php navigation bar), they are sent to logout.php, which contains a call to session_destroy() and session_unset(). These functions clear all session data, effectively removing the user's authentication information. After clearing the session, the script redirects the user to the login page (login.php) using header("Location: login.php"). This officially logs the user off the system, removing their session from the server.



Saved: 7/8/2025 7:09:45 PM

Section #4: (2 pts.) Feature: Security Rules

Progress: 100%

Task #1 (1 pt.) - Database Tables

Progress: 100%

Details:

- From the vs code mysql tool, show both the Roles and UserRoles tables

Roles table

user roles table



Saved: 7/8/2025 7:11:22 PM

≡ Task #2 (1 pt.) - Demo Security Rules

Progress: 100%

❑ Part 1:

Progress: 100%

Details:

- Show the message related to needing to be logged in (i.e., try to manual)
- Show the message related to not having the proper permission/role (i.e.,
- Include a snippet of the login check function
- Include a snippet of the role check function

A screenshot of a code editor displaying PHP code. The code includes functions for checking user roles and permissions, such as `checkRole()` and `checkPermission()`. It also shows a snippet of a login check function.

```
function checkRole($role) {
    // Implementation of role check
}

function checkPermission($permission) {
    // Implementation of permission check
}

function loginCheck() {
    // Implementation of login check
}
```

login check function

A screenshot of a web browser window showing a login page. The URL is `localhost:2000/project/login.php`. A message box displays "You must be logged in to view this page". Below the message, there is a "Login" button and a "Logout" button.

You must be logged in to view this page

Login

Logout

login requirement message



Saved: 7/8/2025 9:11:16 PM

☞ Part 2:

Details:

- Include the pull request url for this feature

URL #1

<https://github.com/Mark-5555/may23-IT2024500>



URL

<https://github.com/Mark-5555/may23-IT2024500>



Saved: 7/8/2025 9:11:16 PM

≡ Part 3:**Details:**

- Briefly explain how the login check code works
- Briefly explain how the role check code works

Your Response:

The login check uses the function `is_logged_in()`, which checks if `$_SESSION["user"]` is set. If `is_logged_in(true)` is used, it automatically redirects to the login page and flashes a warning message if the user is not logged in. This protects private pages from unauthorized access by unauthenticated users.

The role check uses the function `has_role("RoleName")`, which looks inside `$_SESSION["user"]["roles"]` for a role matching the given name. These roles are loaded during login from the Roles and UserRoles database tables. If the role is not found, the page can redirect the user and flash a "no permission" message, protecting role-restricted content.



Saved: 7/8/2025 9:11:16 PM

Section #5: (2 pts.) Feature: User Profile**≡ Task #1 (1 pt.) - Validation****Details:**

- Show the relevant code snippets for each validation layer
- Show the relevant demo of each validation layer
- Briefly explain how the validation steps work at each layer

☰ Task #1 (0.33 pts.) - HTML Form/Validation

Progress: 100%

Details:

- System should allow the user to correct the error without wiping/clearing the form (for the PHP side this includes sticky-form logic to keep the username/email address entries)

Part 1:

Progress: 100%

Details:

- Show the code related to the profile page's form (HTML validation)
 - Show examples of each validation message (you may be able to capture this in one or few screenshots)

HTML profile code

Usernames must be lowercase, alphanumeric, and can only contain _ or -

Profile
Display Name: markmcmillan200491@gmail.com
Username: markm2004
Personal Bio:
Current Password: *****
New Password: *****
Confirm Password: *****
Update Profile

invalid username

A screenshot of a software application window titled "Email account test - test account". The window displays a message stating "Tested account successfully connected to the server." Below this, it shows the account information: "Account: mail.ru@gmail.com (Gmail)" and "Username: mailru123". A large green checkmark icon is present next to the account name. At the bottom right, there is a button labeled "Close" and another labeled "Update Profile".

email must not be empty

The screenshot shows a dark-themed web page with a validation error message at the top: "email must not be empty". Below the message, there is a form with several input fields, including "Email", "New Password", "Confirm Password", and "Update Profile".

invalid email

The screenshot shows a dark-themed web page with a validation error message at the top: "Password and Confirm password must match". Below the message, there is a form with fields for "Email", "Old Password", "New Password", and "Confirm Password".

passwords don't match

Saved: 7/8/2025 9:24:27 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the validation step (i.e., what you chose, how they solve the requirement)

Your Response:

The HTML form includes standard input fields for email, username, and password updates. It sets type="email" and type="password" where appropriate to trigger basic browser validation (e.g., valid email format, non-empty passwords). It also uses the onsubmit="return validate(this);;" attribute to call a JavaScript function before submitting, allowing additional client-side checks.

Saved: 7/8/2025 9:24:27 PM

☰ Task #2 (0.33 pts.) - JS Validation (validate() function)

Progress: 100%

Part 1:

Details:

- Show the code related to the profile page's validate() function
- Show examples of each validation message [username format, email format, password format, password doesn't match confirm] (you may be able to capture this in one or few screenshots)
- Note: You may need to use dev tools to temporarily apply `novalidate` to the form tag

```
function validateForm() {
    let form = document.querySelector('form');
    let errors = document.querySelector('.errors');
    let invalid = false;

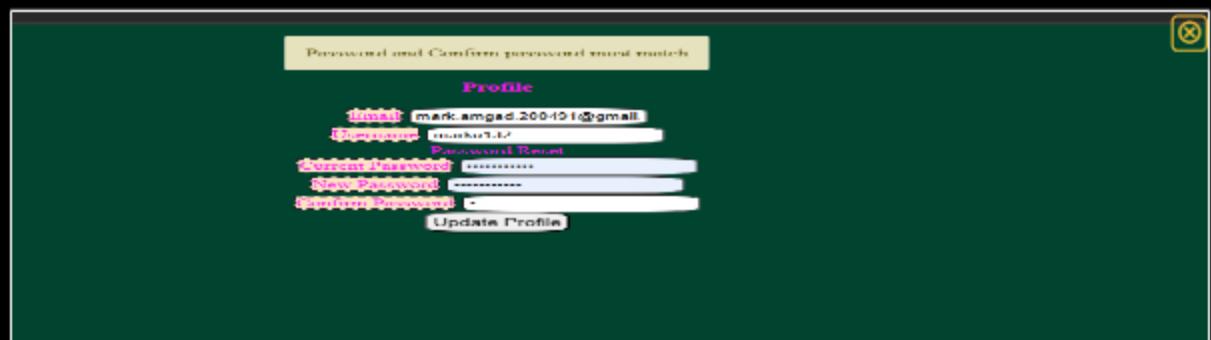
    // Add other classes here validateEmail, ...
    // Example of adding class to the JavaScript
    // and then concatenating it to a string and applying it
    // to the element. Extract the class name to a function later
    // @param [String] [0] // index of validation example
    let addClass = (className) => {
        let element = document.querySelector(`[data-validation-index=${0}]`);
        let div = document.createElement('div');
        div.className = className;
        element.appendChild(div);
        let innerDiv = document.createElement('div');
        innerDiv.className = 'error';
        div.appendChild(innerDiv);
        innerDiv.textContent = `Please enter a ${className} value`;
    };

    // Apply this function to password classes since we have two
    // different classes - 'password' and 'confirm'
    // and they both must match
    validateEmail();
    validatePassword();
    validateConfirm();

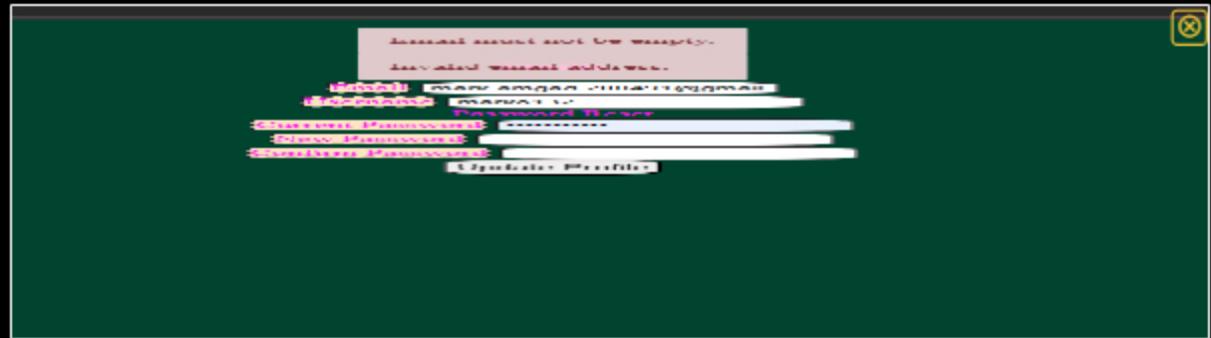
    // returning false will prevent the form from submitting
    return invalid;
}

// novalidate form tag will disable this
```

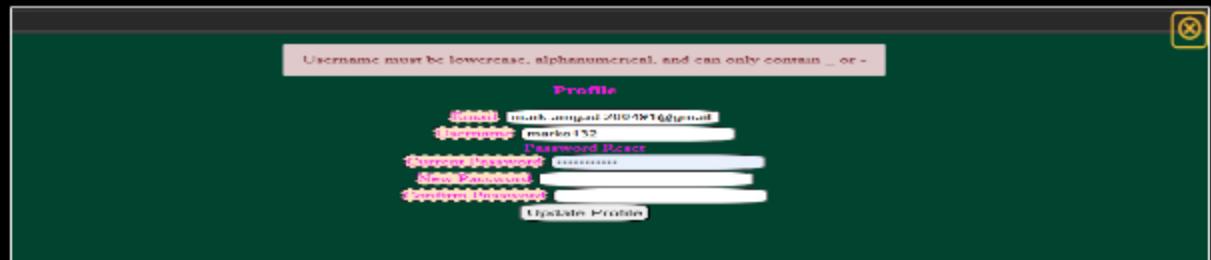
JS code



passwords don't match



invalid email



invalid username

Password must be at least 8 characters long.

Profile

First Name: markmengod 200497 Edmond
Last Name: markmengod 42
Email Address: markmengod@ed.ac.uk
Current Password:
New Password:
Confirm Password:

Update Profile

password format



Saved: 7/8/2025 9:24:39 PM

Part 2:

Progress: 100%

Details:

- Briefly explain the validation step (i.e. what you chose, how they solve the requirement)

Your Response:

The validate(form) function runs in the browser before the form submits. It checks if the new password and confirm password fields match. If they don't, it dynamically creates a warning message using DOM manipulation and inserts it into the #flash container. If there's a mismatch, the function returns false, preventing the form from submitting.



Saved: 7/8/2025 9:24:39 PM

☰ Task #3 (0.33 pts.) - PHP Validation (steps before the DB call)

Progress: 100%

Part 1:

Progress: 100%

Details:

- Show the code related to the profile page's PHP validation
- Show examples of each validation message [username format, email format, password format, invalid password, password doesn't match confirm, username taken, email taken] (you may be able to capture this in one or few screenshots)
- Note: You may need to use dev tools to temporarily apply `data-validate="true"` to the form

- Note: You may need to use dev tools to temporarily apply `novalidate` to the form tag and temporarily override the validate() function or use the provided helper script in the instructions

The screenshot shows the browser's developer tools with the "Elements" tab selected. It displays the HTML structure of a registration form. The form includes fields for Email, Username, Password, and Confirm Password, each with its corresponding validation message displayed above it.

php code

The screenshot shows a registration page with a "Profile" section. The "Username" field contains "marko132" and has a validation message "Username must be lowercase, alphanumeric, and can only contain . - _". Below the form is a button labeled "invalid username".

invalid username

The screenshot shows a registration page with a "Profile" section. The "Password" field contains "marko132" and has a validation message "Password must be at least 8 characters long.". Below the form is a button labeled "password format".

password format

The screenshot shows a registration page with a "Profile" section. The "Confirm Password" field contains a different value than the "Password" field, and has a validation message "Password and Confirm password must match". Below the form is a button labeled "passwords don't match".

passwords don't match

The screenshot shows a registration page with a "Register" button. Above the button is a validation message "The chosen email is not available".

Email:	
Username:	marko132
Password:	*****
Confirm:	*****
<input type="button" value="Register"/>	

email taken

The chosen username is not available.

<input type="button" value="Register"/>	
---	--



username taken



Saved: 7/8/2025 9:29:00 PM

☒ Part 2:

Progress: 100%

Details:

- Briefly explain the validation step (i.e, what you chose, how they solve the requirement)

Your Response:

PHP validation checks that the email isn't empty, follows a valid format, and that the username uses only allowed characters. It also validates the new password length and ensures the current password is correct before allowing a change. These checks prevent invalid or malicious input and ensure that only valid, authorized updates are processed.



Saved: 7/8/2025 9:29:00 PM

☒ Task #2 (1 pt.) - Related URLs

Progress: 100%

Details:

- Include the direct link to this file from the Milestone branch (should end in .php)
- Include the heroku prod link to this file (Just grab the base prod url and manually enter the path to the file)
- Include the related pull request link for this feature

URL #1
<https://github.com/Mark-5555/may23-IT202-450>

URL #2
<https://may23-it202-450-dev-70e1264ec6c5.herokuapp.com/project/profile.php>

URL #3
<https://github.com/Mark-5555/may23-IT202-450/pull/18>



Saved: 7/8/2025 9:34:43 PM

Section #6: (1 pt.) Misc

Progress: 95%

≡ Task #1 (0.33 pts.) - Github Details

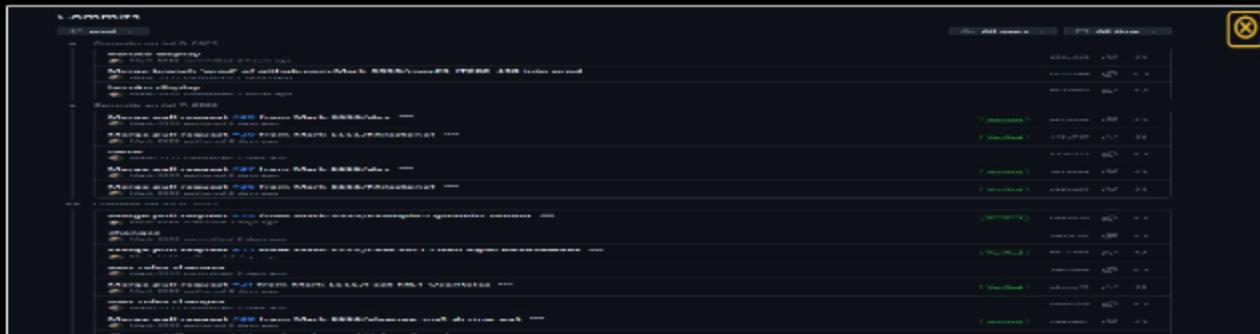
Progress: 87%

▣ Part 1:

Progress: 75%

Details:

From the Commits tab of the Pull Request screenshot the commit history



Missing Caption



Saved: 7/8/2025 9:36:21 PM

⌚ Part 2:

Progress: 100%

Details:

Include the link to the Pull Request (should end in /pull/#)

URL #1

<https://github.com/Mark-5555/may23->

IT202-4501it/827b49eb59ac88a945c5680866cc87e0cf7e55a6



LH

<https://github.com/Mark-5555/may23->



Saved: 7/8/2025 9:36:21 PM

▣ Task #2 (0.33 pts.) - WakaTime - Activity

Progress: 100%

Details:

- Visit the WakaTime.com Dashboard
- Click **Projects** and find your repository
- Capture the overall time at the top that includes the repository name
- Capture the individual time at the bottom that includes the file time
- Note: The duration isn't relevant for the grade and the visual graphs aren't necessary



bottom



top



Saved: 7/8/2025 9:37:42 PM

≡ Task #3 (0.33 pts.) - Reflection

Progress: 100%

≡, Task #1 (0.33 pts.) - What did you learn?

Details:

Briefly answer the question (at least a few decent sentences)

Your Response:

Wow, a lot of things:

- I learned how to use PHP for backend development, like handling form submissions, validating inputs, updating the database, managing user sessions, and showing success or error messages.
- I learned how to use JavaScript to do simple checks on the browser side before sending data to the server, like making sure the password and confirm password match, and even how to display warning messages on the page.
- I learned how to use HTML to create web forms using things like [redacted], [redacted], and attributes like type="email" or required to do basic input validation.
- I learned how to check if a user is logged in and how to protect pages so only logged-in users can access them.
- I also learned how to check if a user has a specific role (like "Admin") before showing certain pages or actions, and how to show a message if they don't have permission.
- I learned how to store flash messages in the session and display them after redirects to tell the user what happened.
- I learned how to securely update user info and passwords, making sure the current password is correct and the new one is strong enough.
- I even built a simple admin page to assign roles to users through a form on the website, which updates the database without needing to write SQL manually.



Saved: 7/8/2025 9:41:03 PM

=, Task #2 (0.33 pts.) - What was the easiest part of the assignment?

Details:

Briefly answer the question (at least a few decent sentences)

Your Response:

HTML coding was pretty simple.



Saved: 7/8/2025 9:41:22 PM

=> Task #3 (0.33 pts.) - What was the hardest part of the assignment?

Progress: 100%

Details:

Briefly answer the question (at least a few decent sentences)

Your Response:

wiring things together via php and JS.



Saved: 7/8/2025 9:41:48 PM