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

Progress: 100%

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:

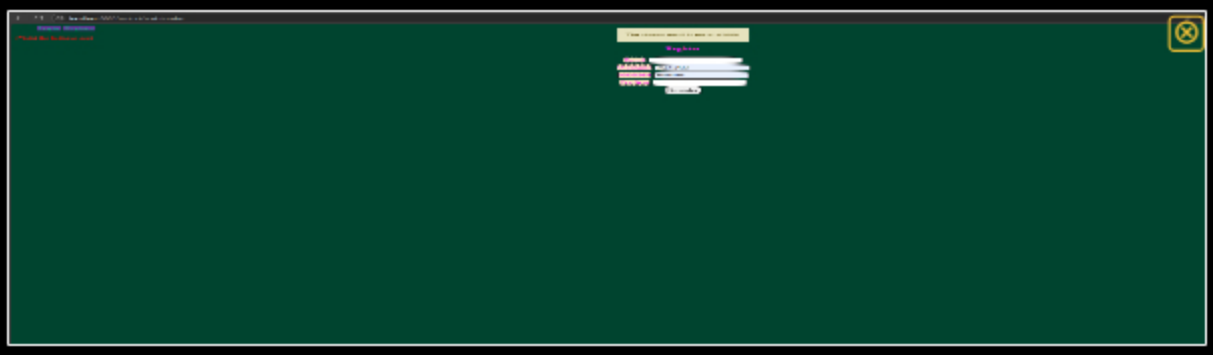
- System should allow the user to correct the error without wiping/clearing the form (for the PHP side this includes sticky-form logic to keep the username/email address entries)

Part 1:

Progress: 100%

Details:

- Show the code related to the 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%

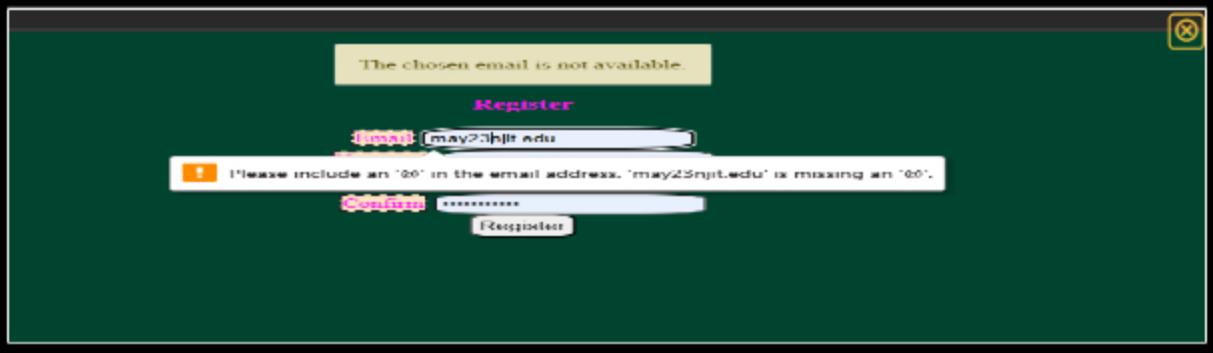
Part 1:

Progress: 100%

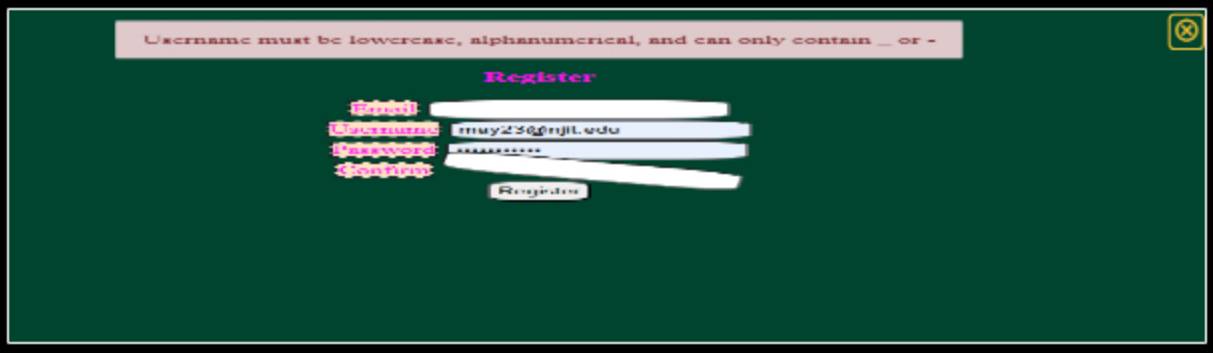
Details:

- 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

validation code

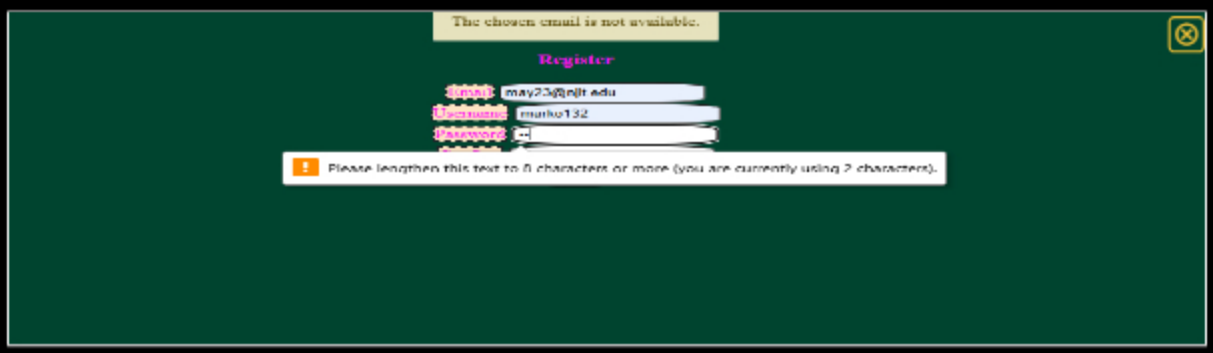


email domain



username format

password does not match



password format

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

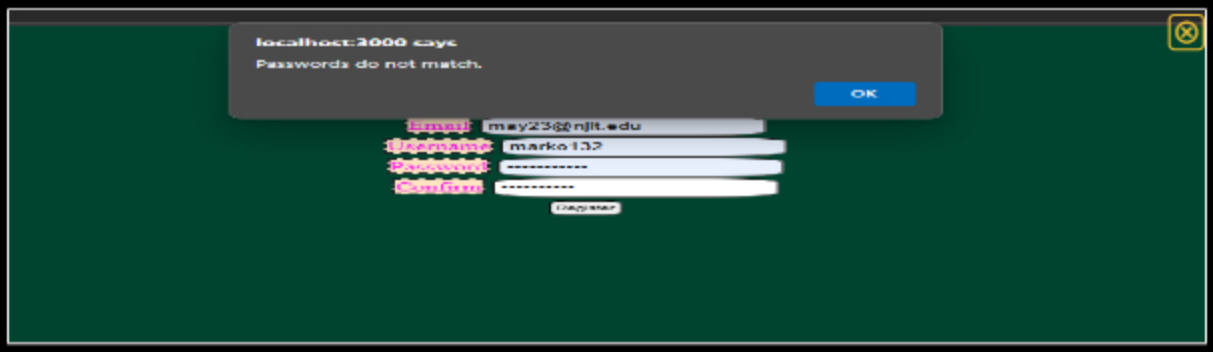
Progress: 100%

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

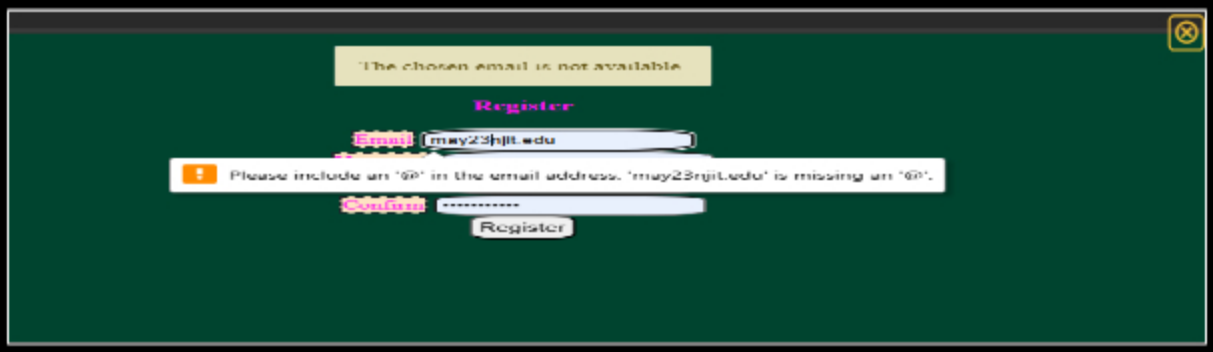
PHP validation

password format



password does not match

username format



email domain

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

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-IT202-450/blob/prod/public_html/project/register.php
URL
https://github.com/Mark-555

URL #2
https://may23-it202-450-prod-a1263d847137.herokuapp.com/project/register.php
URL
https://may23-it202-450-proc

URL #3
https://github.com/Mark-5555/may23-IT202-450/pull/14
URL
https://github.com/Mark-555

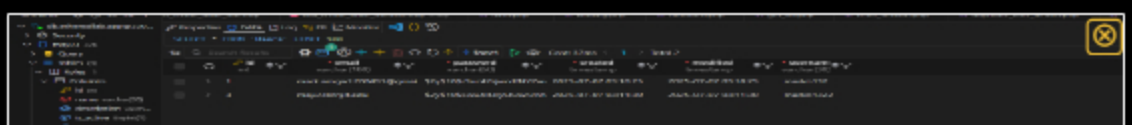
Saved: 7/8/2025 6:33:43 PM

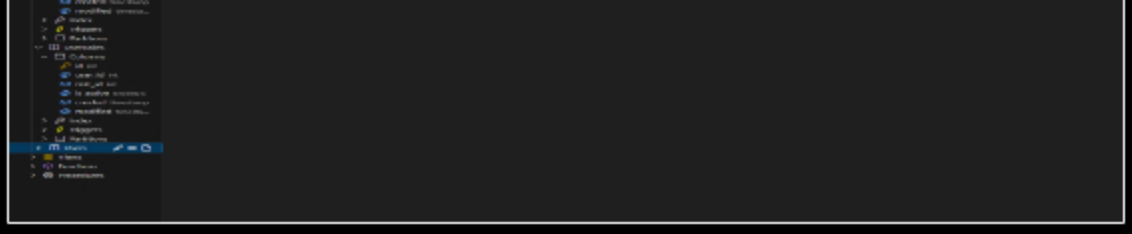
Task #3 (0.67 pts.) - Database - Users table

Progress: 100%

Details:

- Add a screenshot of the database view from vs code showing a valid registered user (preferrably 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 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

PHP code

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

Part 2:

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

Task #2 (0.67 pts.) - Related URLs

Progress: 100%

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-IT202-450/
https://may23-it202-450-prod/public_html/project/login.php
URL
https://github.com/Mark-555

URL #2
https://may23-it202-450-prod-a1263d847137.herokuapp.com/project/login.php
URL
https://may23-it202-450-prod

URL #3
https://github.com/Mark-5555/may23-IT202-450/pull/15
URL
https://github.com/Mark-555

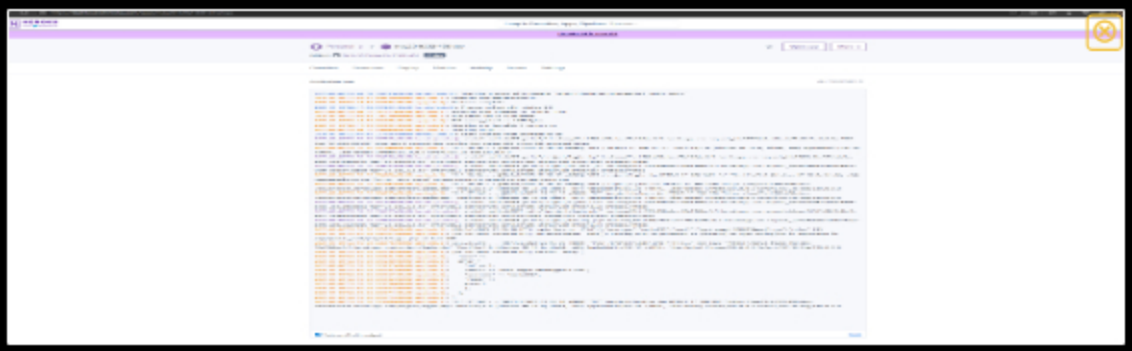
Saved: 7/8/2025 6:33:17 PM

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

Progress: 100%

Details:

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



Heroku logs

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

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

To Logout

Progress: 100%

Task #1 (1 pt.) - Evidence

Progress: 100%

Part 1:

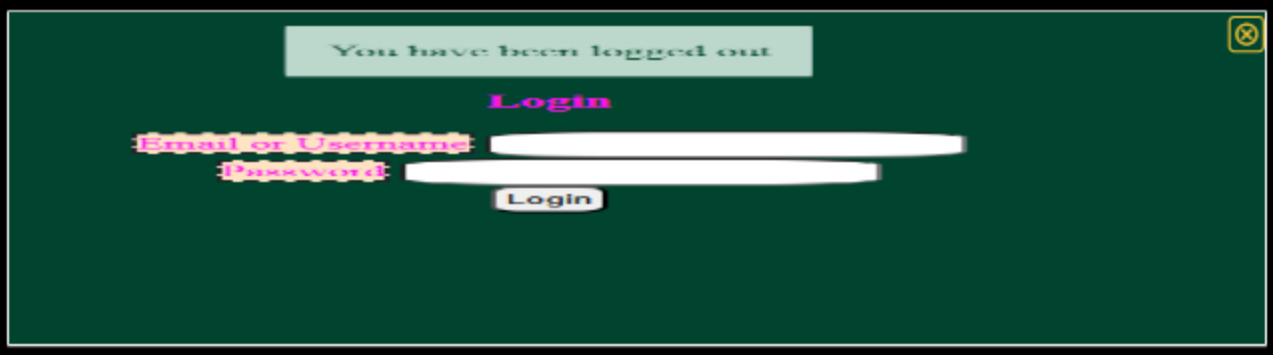
Progress: 100%

Details:

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

```php
<?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



logout message

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

Part 2:

Progress: 100%

Details:

- Include the pull request url for this feature

URL #1
https://github.com/Mark-5555/may23-IT202-450/5

URL
https://github.com/Mark-5555/ma

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

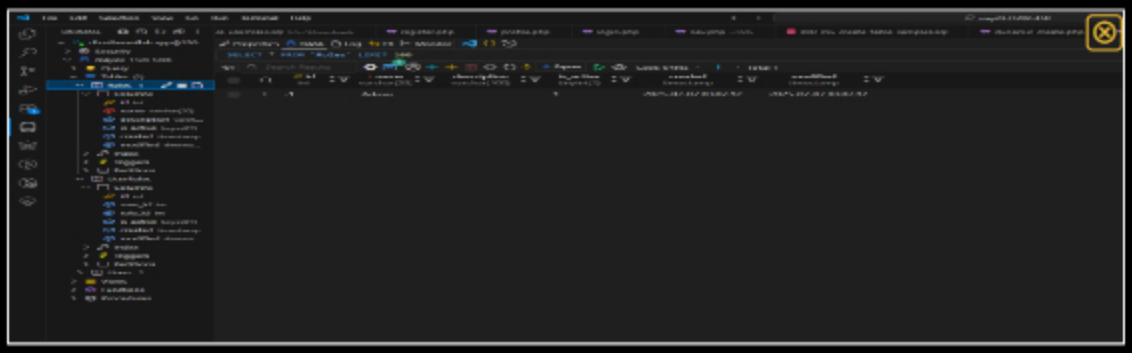
Progress: 100%

Task #1 (1 pt.) - Database Tables

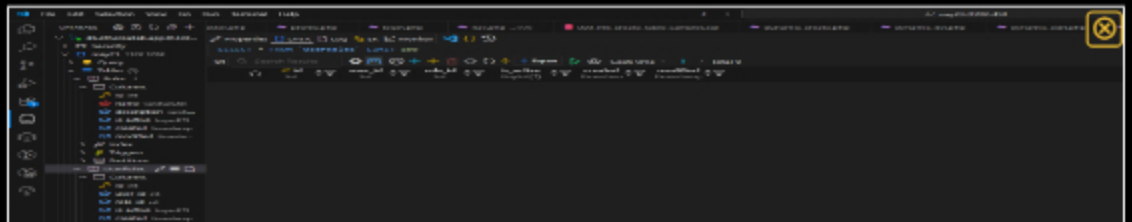
Progress: 100%

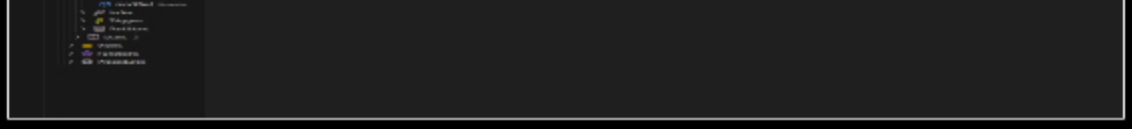
Details:

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



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

login check function

login requirment message

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

Part 2:

Progress: 100%

Details:

- Include the pull request url for this feature

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

URL
https://github.com/Mark-5555/ma

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

Part 3:

Progress: 100%

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

Progress: 100%

Task #1 (1 pt.) - 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:

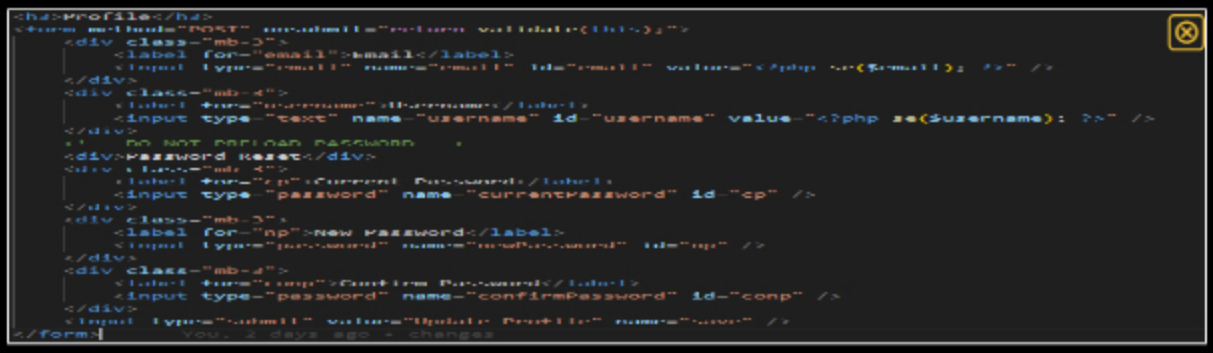
- System should allow the user to correct the error without wiping/clearing the form (for the PHP side this includes sticky-form logic to keep the username/email address entries)

Part 1:

Progress: 100%

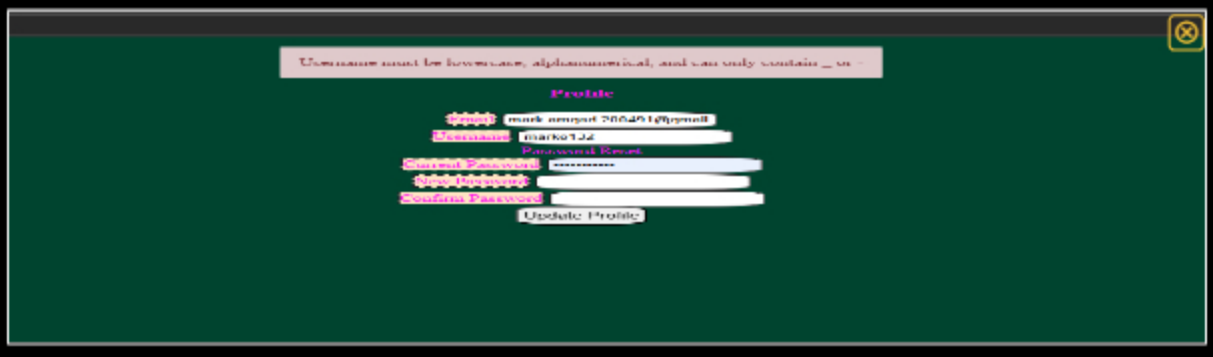
Details:

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

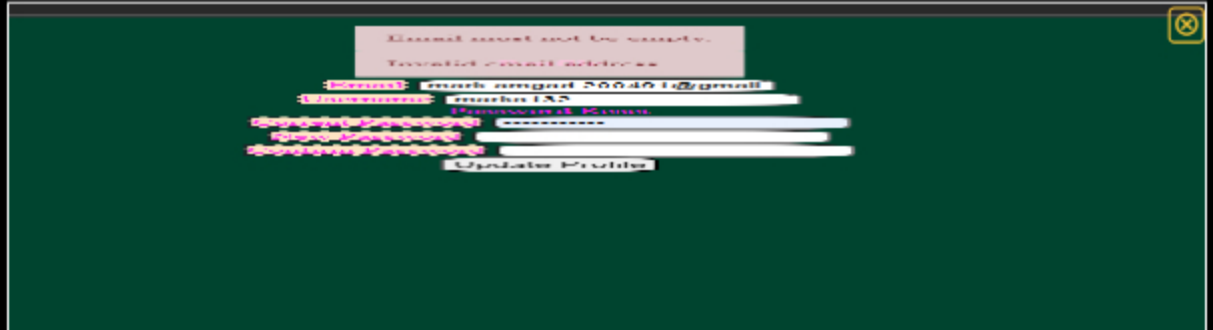


```
<h2>Profile</h2>
<form method="POST" onsubmit="return validate();">
    <div class="mb-3">
        <label for="email">Email</label>
        <input type="email" name="email" id="email" value="<?php se($email); ?>" />
    </div>
    <div class="mb-4">
        <label for="username">Username</label>
        <input type="text" name="username" id="username" value="<?php se($username); ?>" />
    </div>
    <!-- DO NOT PRELOAD PASSWORD -->
    <div>Password Reset</div>
    <div class="mb-4">
        <label for="cp">Current Password</label>
        <input type="password" name="currentPassword" id="cp" />
    </div>
    <div class="mb-3">
        <label for="np">New Password</label>
        <input type="password" name="newPassword" id="np" />
    </div>
    <div class="mb-4">
        <label for="conp">Confirm Password</label>
        <input type="password" name="confirmPassword" id="conp" />
    </div>
    <input type="submit" value="Update Profile" name="save" />
</form>
```

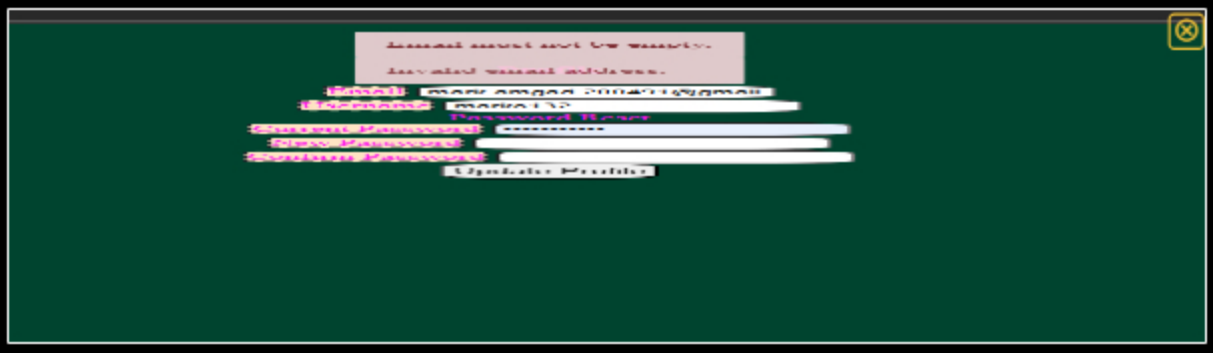
HTML profile code



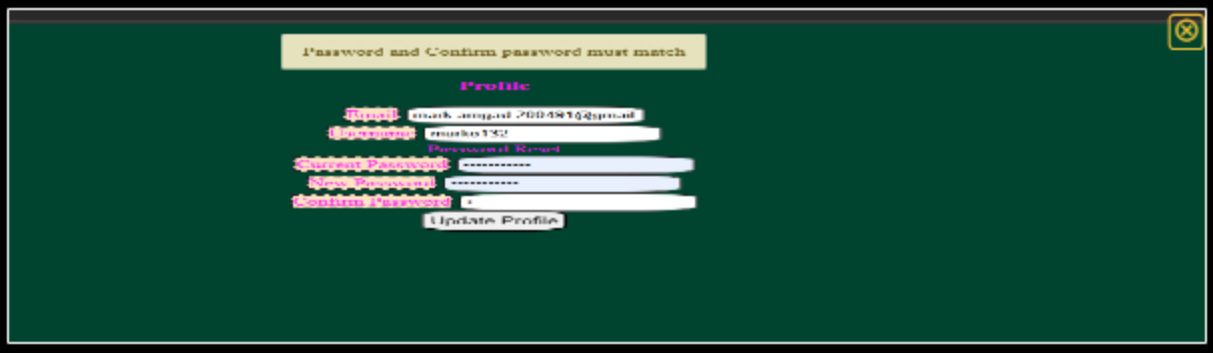
invalid username



email must not be empty



invalid email



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:

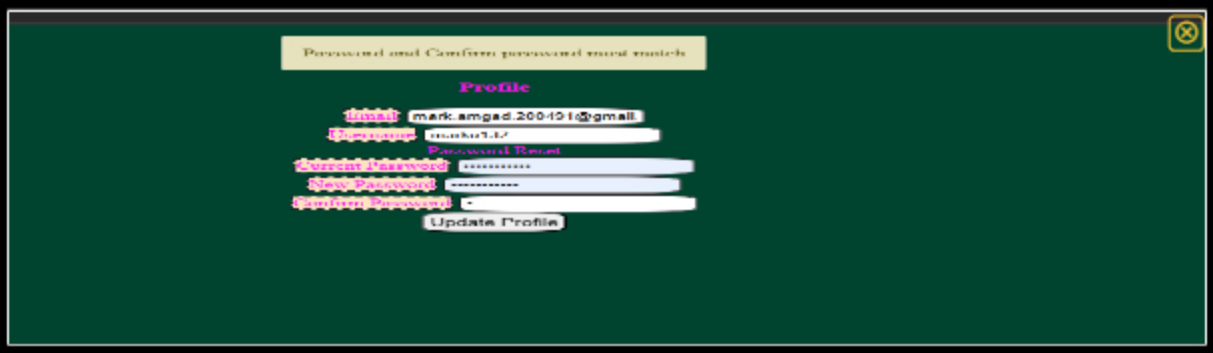
Progress: 100%

Details:

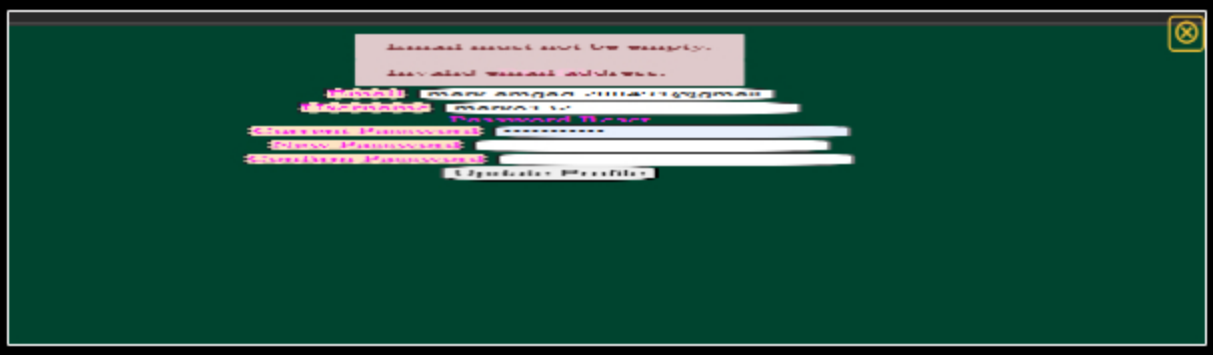
- Show the code related to the profile page's validate() function
- Show examples of each validation message [username format, email format, password format, password doesn't match confirm] (you may be able to capture this in one or few screenshots)
- Note: You may need to use dev tools to temporarily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

```
<script>
    function validate(form) {
        let pw = form.newPassword.value;
        let con = form.confirmPassword.value;
        let isValid = true;
        // TODO: add other client side validation....

        //example of using Flash via Javascript
        //Find the flash container, create a new element, append it to
        // it. We'll abstract this into some fn a function later
        if (pw !== con) { // later JS validation example
            //Find the container
            let flash = document.getElementById("flash");
            //create a div (or whatever wrapper we want)
            let outerDiv  document.createElement("div");
            outerDiv.className = "row justify-content-center";
            let innerDiv = document.createElement("div");

            //apply the css (these are bootstrap classes which we'll learn later)
            innerDiv.className = "alert alert-warning";
            //set the contents
            innerDiv.innerText = "Password and Confirm password must match";

            outerDiv.appendChild(innerDiv);      you, 2 days ago • changes
            //add the element to the DOM (if we don't it merely exists in memory)
            flash.appendChild(outerDiv);
            isValid = false;
        } // end if (pw !== con)
        // returning false will prevent the form from submitting
        return isValid;
    } // end function validate(form)
</script>
```

JS code

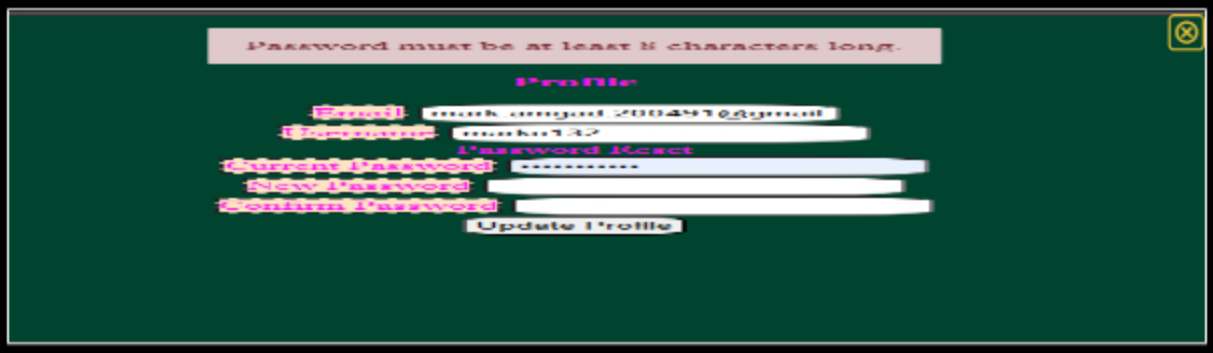


passwords don't match



invalid email

invalid username



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:

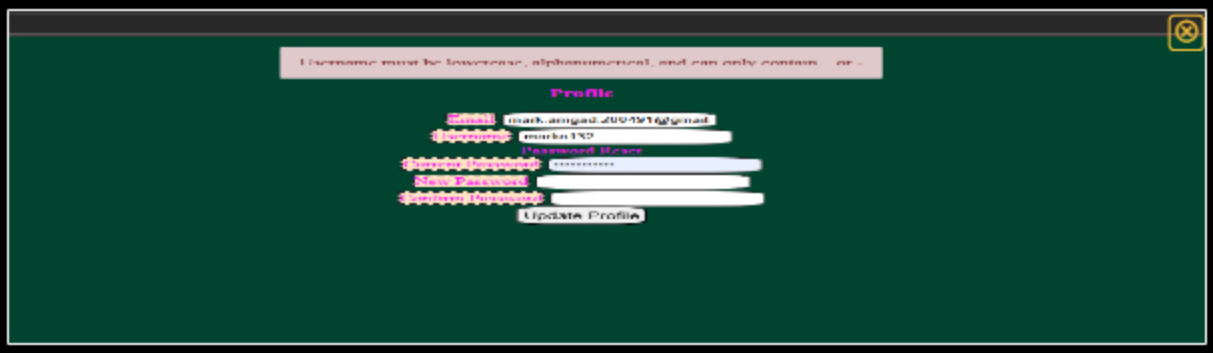
Progress: 100%

Details:

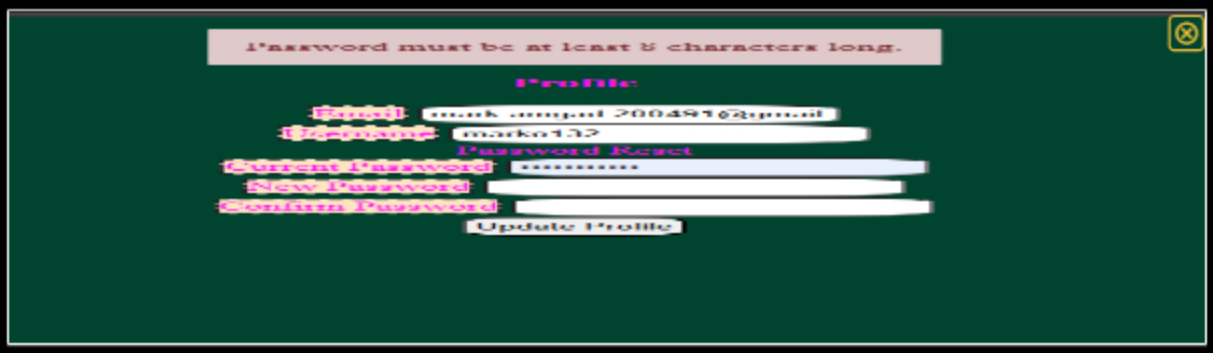
- Show the code related to the profile page's PHP validation
- Show examples of each validation message [username format, email format, password format, invalid password, password doesn't match confirm, username taken, email taken] (you may be able to capture this in one or few screenshots)
- Note: You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

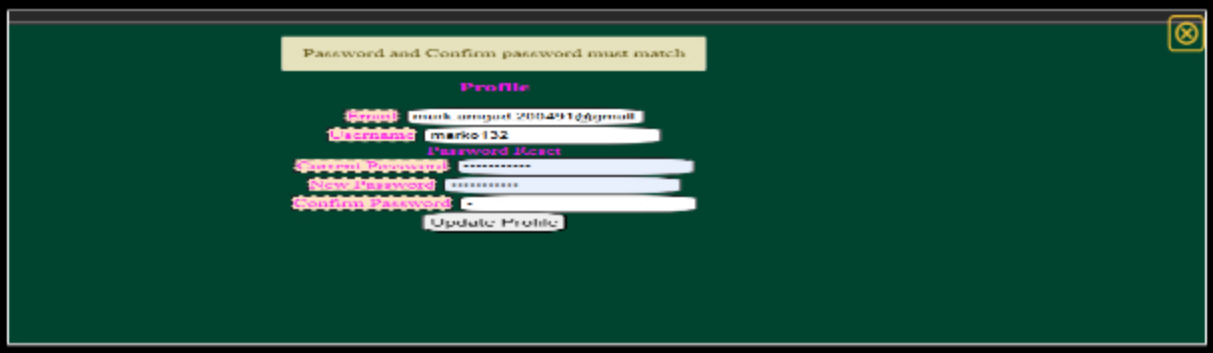
php code



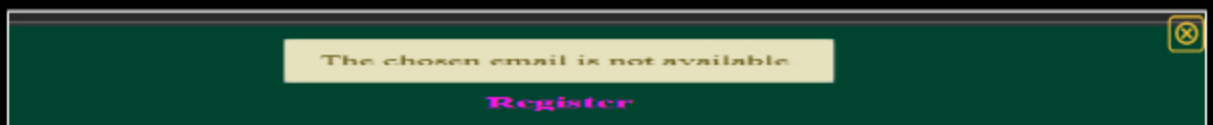
invalid username



password format



passwords don't match



email taken

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/blob/prod/public_html/project/profile.php

URL
https://github.com/Mark-555

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

URL
https://may23-it202-450-dev-

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

URL
https://github.com/Mark-555

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-IT202-450/commit/827b49eb59ac88a945c5680866cc87e0cf7e55a6

URL

https://github.com/Mark-5555/ma

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

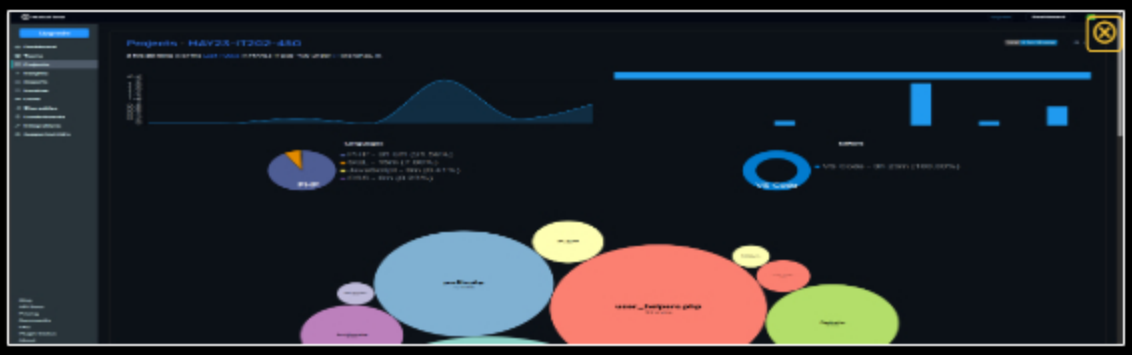
Task #2 (0.33 pts.) - WakaTime - Activity

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?

Progress: 100%

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 , , 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?

Progress: 100%

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