Course: IT202-008-S2025
Assignment: IT202 Milestone 1
Student: Gori H. (gs658)
Status: Submitted | Worksheet Progress: 100%
Potential Grade: 10.00/10.00 (100.00%)
Received Grade: 0.00/10.00 (0.00%)
Grading Link: https://learn.ethereallab.app/assignment/v3/IT202-008-S2025/it202-milestone-1/grading/gs658

Instructions

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`

100%

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

100%

Task #1 (0.67 pts.) - Validation

Weight: *33.33%*
Objective: *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

Sub-Tasks:

100% Task #1 (0.33 pts.) - HTML Form/Validation

Combo Task:

Weight: *33.33%*
Objective: *HTML Form/Validation*

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)

Image Prompt

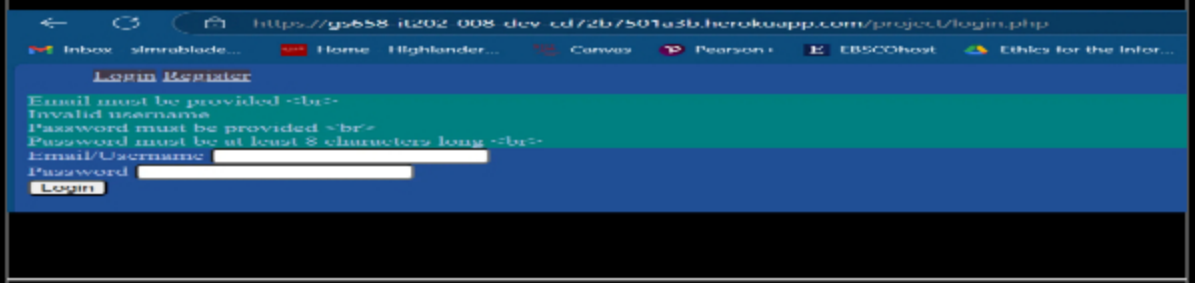
Weight: *50%*

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)

```
<form onsubmit="return validate(this)" method="POST">
    <div>
        <label for="email">Email</label>
        <input type="email" name="email" required />
    </div>
    <div>
        <label for "username">Username</label>
        <input type "text" name "username" required maxlength "30" />
    </div>
    <div>
        <label for="pw">Password</label>
        <input type="password" id="pw" name="password" required minlength="x" />
    </div>
    <div>
        <label for="confirm">Confirm</label>
        <input type="password" name="confirm" required minlength="8" />
    </div>
    <input type="submit" value="Register" />
</form>
```

HTML code



validation test

Saved: 4/13/2025 5:36:19 PM

Text Prompt

Weight: *50%*

Details:

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

Your Response:

For the HTML validation, I used built-in HTML attributes such as required, so the field needs to be filled before form submission, type="email" so that the email input is in a valid format, maxlength="30" to limit the username length to 30 characters, and minlength="8" so that passwords are at least 8 characters. These attributes all provide client-side validation.

Saved: 4/13/2025 5:36:19 PM

100%

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

Combo Task:

Weight: *33.33%*
Objective: *JS Validation (validate() function)*

Image Prompt

Weight: *50%*

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

```
<script>
    function isValidPassword(pw) {
        return pw.length >= 8;
    }
    function validate(form) {
        //TODO 1: implement JavaScript validation
        //ensure it returns false for an error and true for success
        let isValid = true;
        if(!isValidPassword(form.password.value)){
            isValid = false;
            flash("Password must be at least x characters long", "danger");
        }
        return isValid;
    }
</script>
```

JS code

pw validation

Saved: 4/13/2025 5:29:09 PM

Text Prompt

Weight: *50%*

Details:

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

Your Response:

The validate() function is connected to the form's onsubmit event; it performs client-side validation before the form data is sent to the server. IsValidPassword() checks for pw length, at least 8 characters long. If this fails, the form submission is prevented, and the user is notified about the error. This code ensures that user input is valid before sending data to the server.

100%

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

Combo Task:

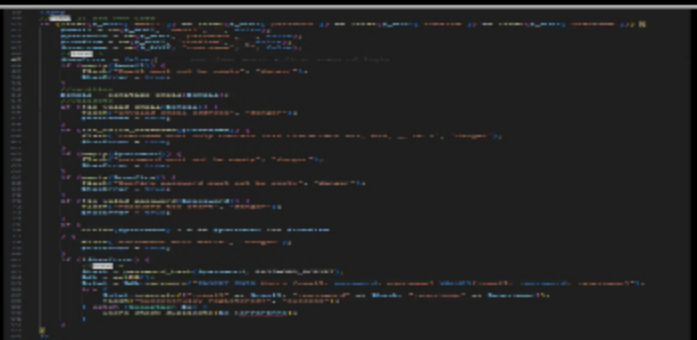
Weight: *33.33%*
Objective: *PHP Validation (steps before the DB call)*

Image Prompt

Weight: *50%*

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 code

user/pw validation

registration success

email taken

invalid email address

100%

Task #2 (0.67 pts.) - Related URLs

Url Prompt

Weight: *33.33%*
Objective: *Related URLs*

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://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/register.php

URL
https://gs658-it202-008-prod-9640930

URL #2
https://github.com/goribanu/gs658-IT202-008/Milestone1/public_html/project/register.php

URL
https://github.com/goribanu/gs658-IT

URL #3
https://github.com/goribanu/gs658-IT202-008/pull/4

URL
https://github.com/goribanu/gs658-IT

Saved: 4/13/2025 6:32:00 PM

100%

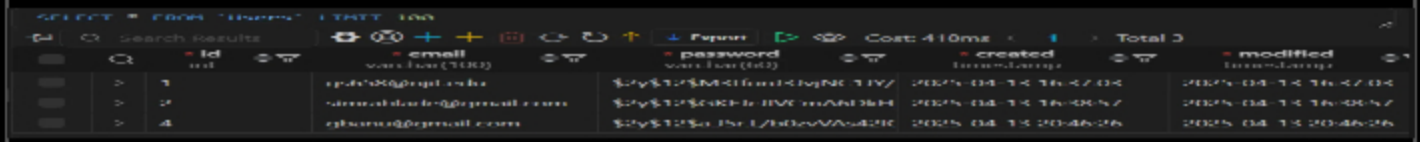
Task #3 (0.67 pts.) - Database - Users table

Image Prompt

Weight: *33.33%*
Objective: *Database - Users table*

Details:

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



showing user

Saved: 4/13/2025 6:09:52 PM

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

Task #1 (0.67 pts.) - Validation

Weight: *33.33%*
Objective: *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

Sub-Tasks:

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

Combo Task:

Weight: *33.33%*
Objective: *HTML Form/Validation*

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)

Image Prompt

Weight: *50%*

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)

```html
<form onsubmit="return validate(this)" method="POST">
    <div>
        <label for="email">Email/username</label>
        <input type="text" name="email" required />
    </div>
    <div>
        <label for="pw">Password</label>
        <input type="password" id="pw" name="password" required minlength="8" />
    </div>
    <input type="submit" value="Login" />
</form>
```

html code

Saved: 4/13/2025 6:39:21 PM

Text Prompt

Weight: *50%*

Details:

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

Your Response:

I used the required attribute for both input fields so that users can't submit an empty form. The password input also has a minlength = "8" constraint so users enter at least 8 characters before submission. This helps catch basic user errors in the browser.

Saved: 4/13/2025 6:39:21 PM

100%

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

Combo Task:

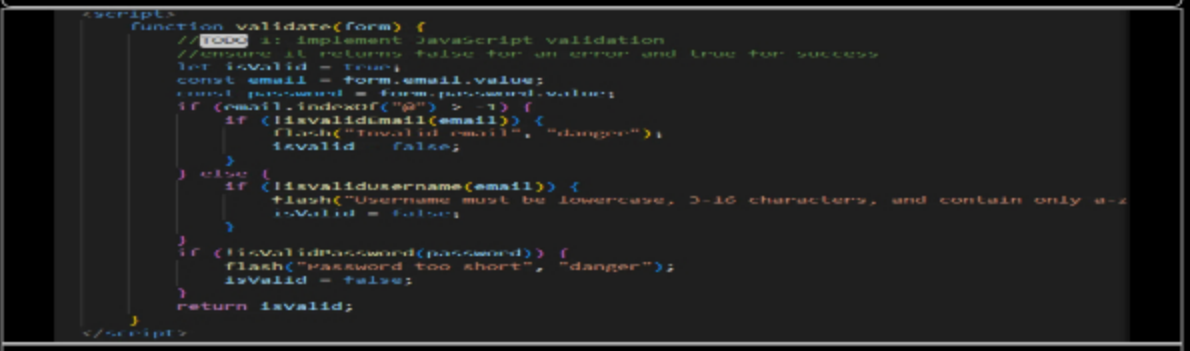
Weight: *33.33%*
Objective: *JS Validation (validate() function)*

Image Prompt

Weight: *50%*

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



js code

Saved: 4/13/2025 6:41:12 PM

Text Prompt

Weight: *50%*

Details:

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

Your Response:

I used a validate(form) function to check input values before submission of the form. The input needs to include "@" to be treated as an email and passed to isValidEmail(). It uses isValidUsername() for username checks. The password is also validated using isValidPassword(). For any fails, an error is shown and form submission is blocked by returning false.

Saved: 4/13/2025 6:41:12 PM

100%

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

Combo Task:

Weight: *33.33%*
Objective: *PHP Validation (steps before the DB call)*

Image Prompt

Weight: *50%*

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: 4/13/2025 6:43:57 PM

Text Prompt

Weight: *50%*

Details:

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

Your Response:

After form submission, it sanitizes and validates both email/username and password. It checks if the email is formatted correctly or if the username follows length and character restrictions using is_valid_email() and is_valid_username(), password is also checked. If all checks pass, the database is queried and login proceeds.

Saved: 4/13/2025 6:43:57 PM

100%

Task #2 (0.67 pts.) - Related URLs

Url Prompt

Weight: *33.33%*
Objective: *Related URLs*

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://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/login.php
URL
https://gs658-it202-008-prod-9640930

URL #2
https://github.com/goribanu/gs658-IT202-008/pull/5
URL
https://github.com/goribanu/gs658-IT

URL #3
https://github.com/goribanu/gs658-IT202-008/Milestone1/public_html/project/login.php
URL
https://github.com/goribanu/gs658-IT

Saved: 4/13/2025 6:45:45 PM

100%

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

Image Prompt

Weight: *33.33%*
Objective: *User Session Evidence*

Details:

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

logs snippet

Saved: 4/13/2025 6:52:02 PM

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

Task #1 (1 pt.) - Evidence

Combo Task:

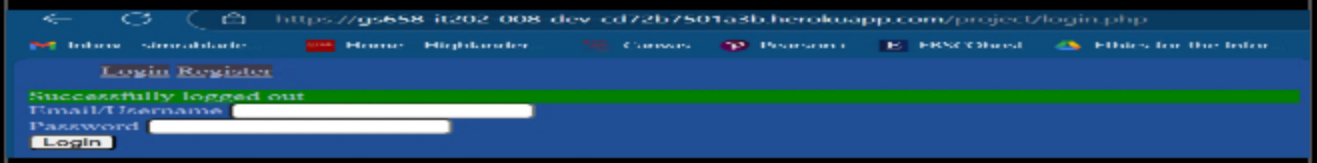
Weight: *100%*
Objective: *Evidence*

Image Prompt

Weight: *40%*

Details:

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



successfully logged out msg

logout code

Saved: 4/13/2025 6:55:59 PM

Url Prompt

Weight: *20%*

Details:

- Include the pull request url for this feature

URL #1
https://github.com/goribanu/gs658-IT202-008/

URL
https://github.com/goribanu/gs658

Saved: 4/13/2025 6:55:59 PM

Text Prompt

Weight: *40%*

Details:

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

Your Response:

The logout process starts with session_start() in order to access the current session. Then, reset_session() clears all session data and regenerates the session ID to log the user out. Flash() displays a success message, and header("Location: login.php") redirects the user back to the login page.

Saved: 4/13/2025 6:55:59 PM

100% Section #4: (2 pts.) Feature: Security Rules

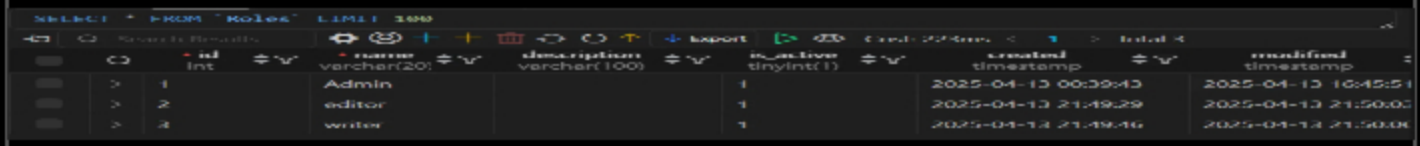
100% Task #1 (1 pt.) - Database Tables

Image Prompt

Weight: *50%*
Objective: *Database Tables*

Details:

```
- From the vs code mysql tool, show both the Roles and UserRoles tables with data in them
```



roles table

id	user_id	role_id	is_active	created	modified
3	2	1	1	2025-04-13 16:48:00	2025-04-13 16:48:00
4	1	2	1	2025-04-13 23:00:16	2025-04-13 23:00:16

100%

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

Combo Task:

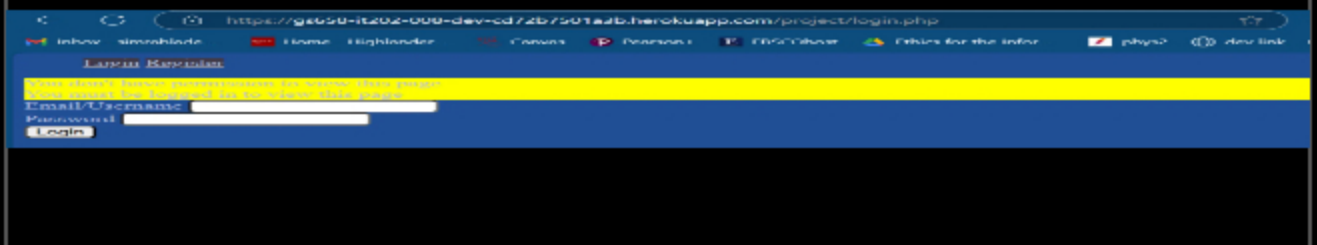
Weight: *50%*
Objective: *Demo Security Rules*

Image Prompt

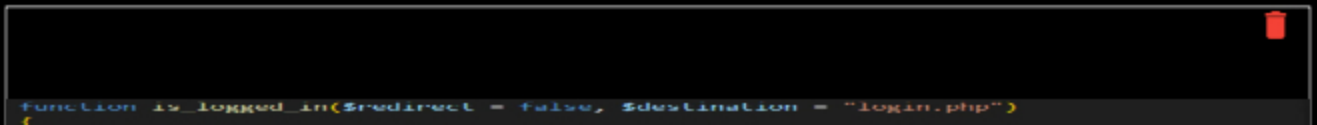
Weight: *40%*

Details:

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



need to be logged in/no access



```php
    $isLoggedIn = isset($_SESSION["user"]);
    if ($redirect && !$isLoggedIn) {
        //if this triggers, the calling script won't receive a reply since die()/exit()
        flash("You must be logged in to view this page", "warning");
        die(header("location: $destination"));
    }
    return $isLoggedIn;
}
```

login check function

```php
function has_role($role)
{
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {
        foreach ($_SESSION["user"]["roles"] as $r) {
            if ($r["name"] === $role) {
                return true;
            }
        }
    }
    return false;
}
```

role check function

Saved: 4/13/2025 7:22:28 PM

Url Prompt

Weight: *20%*

Details:

- Include the pull request url for this feature

URL #1
https://github.com/goribanu/gs658-IT202-001/1

URL
https://github.com/goribanu/gs658

Saved: 4/13/2025 7:22:28 PM

Text Prompt

Weight: *40%*

Details:

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

Your Response:

The is_logged_in() function checks if a user is logged in by verifying if $_SESSION["user"] is set. If the $redirect parameter is true and the user is not logged in, it generates a warning message and redirects the user back to the login page.

The has_role() function checks if a user already logged in has a specific role by looping through the roles in $_SESSION["user"]["roles"]. If a match is found, it returns true, else it returns false if the user does not have permission.

100%

Section #5: (2 pts.) Feature: User Profile

100%

Task #1 (1 pt.) - Validation

Weight: *50%*
Objective: *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

Sub-Tasks:

100%

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

Combo Task:

Weight: *33.33%*
Objective: *HTML Form/Validation*

Details:

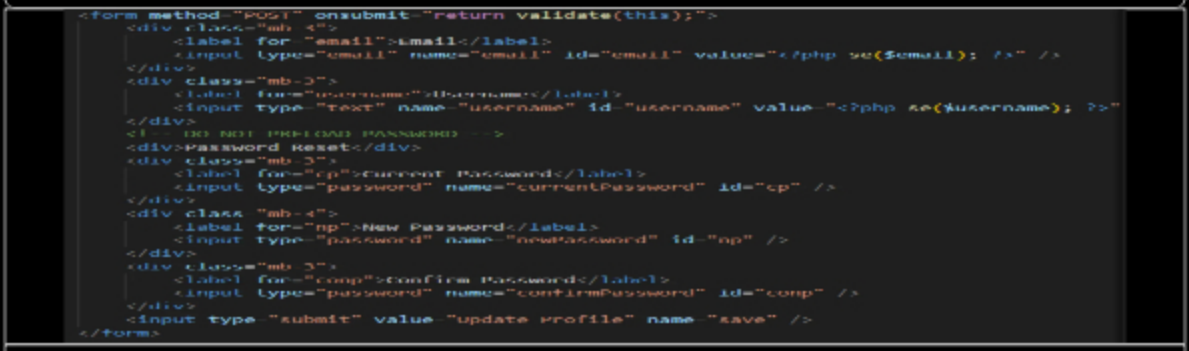
- System should allow the user to correct the error without wiping/clearing the form (for the PHP side this includes sticky-form logic to keep the username/email address entries)

Image Prompt

Weight: *50%*

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)



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

html code

Saved: 4/13/2025 7:46:07 PM

Text Prompt

Weight: *50%*

Details:

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

Your Response:

The form uses [ ] fields to collect user data such as emails, usernames, and passwords. The submit button triggers the validation process.

Saved: 4/13/2025 7:46:07 PM

100%

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

Combo Task:

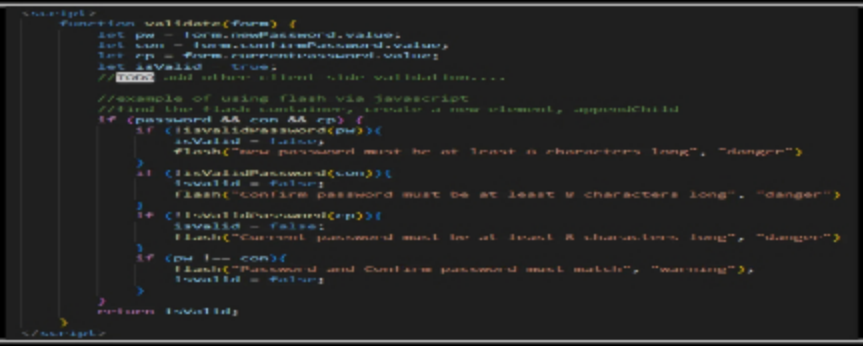
Weight: *33.33%*
Objective: *JS Validation (validate() function)*

Image Prompt

Weight: *50%*

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



js code

Saved: 4/13/2025 7:47:52 PM

Text Prompt

Weight: *50%*

Details:

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

Your Response:

The JS validate() function checks if the password fields meet the minimum 8-character requirement and ensure that the new pw and confirmed pw match. If the validation fails, it shows error messages using the flash() function and prevents form submission, returning false.

Saved: 4/13/2025 7:47:52 PM

100% Task #3 (0.33 pts.) - PHP Validation (steps before the DB call

Combo Task:

Weight: *33.33%*
Objective: *PHP Validation (steps before the DB call)*

Image Prompt

Weight: *50%*

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 `novalidate` to the form tag and temporarily override the validate() function or use the provided helper script in the instructions

php code

Saved: 4/13/2025 7:49:37 PM

Text Prompt

Weight: *50%*

Details:

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

requirement)

Your Response:

For profile updates on the server-side, the PHP code updates the user's email and username in the database using UPDATE SQL queries. Errors like duplicates are handled using flash() with specific messages. For pw updates, the code validates the current password by comparing it to the stored hash and ensures the new passwords match. If it's valid, password_hash() updates the pw with a hashed version.

Saved: 4/13/2025 7:49:37 PM

100%

Task #2 (1 pt.) - Related URLs

Url Prompt

Weight: *50%*
Objective: *Related URLs*

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://gs658-it202-008-prod-96409305ed03.herokuapp.com/project/profile.php

URL
https://gs658-it202-008-prod-9640930

URL #2
https://github.com/goribanu/gs658-IT202-008/pull/9

URL
https://github.com/goribanu/gs658-IT2

100%

Section #6: (1 pt.) Misc

100%

Task #1 (0.33 pts.) - Github Details

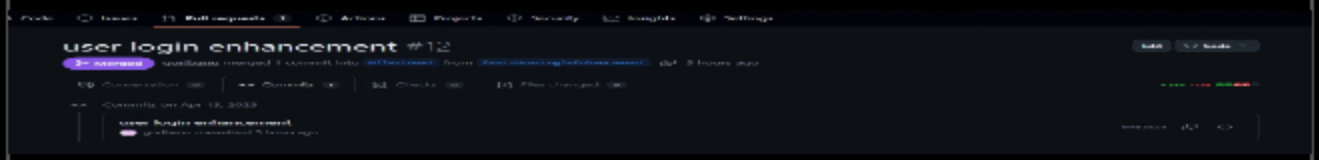
Combo Task:

Weight: *33.33%*
Objective: *Github Details*

Image Prompt

Weight: *60%*

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



commit history

Saved: 4/13/2025 7:50:15 PM

Url Prompt

Weight: *40%*

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

URL #1
https://github.com/goribanu/gs658-
IT202-001/2

URL
https://github.com/goribanu/gs658

Saved: 4/13/2025 7:50:15 PM

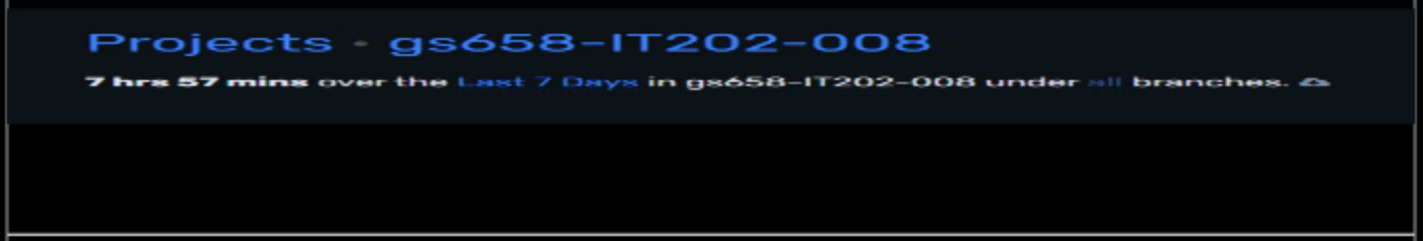
100% Task #2 (0.33 pts.) - WakaTime - Activity

Image Prompt

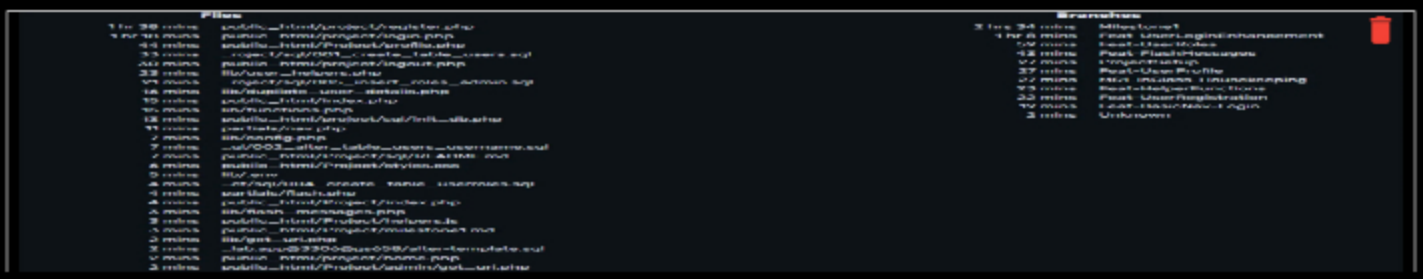
Weight: *33.33%*
Objective: *WakaTime - Activity*

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



overall wakatime activity



Task #3 (0.33 pts.) - Reflection

Weight: *33.33%*
Objective: *Reflection*

Sub-Tasks:

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

Text Prompt

Weight: *33.33%*
Objective: *What did you learn?*

Objective: *What did you learn?*

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

Your Response:

I learned how about user authorization and registration, aspects such as verifying to see if passwords match, usernames aren't taken, usernames are valid, emails aren't taken, emails are valid, etc. I also learned how to assign roles using admin perms and generating tables, altering tables, and using id to locate and give a user certain perms.

Saved: 4/13/2025 7:26:38 PM

100%

Task #2 (0.33 pts.) - What was the easiest part of the assignr

Text Prompt

Weight: *33.33%*
Objective: *What was the easiest part of the assignment?*

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

Your Response:

The easiest part of the assignment was just copying over the files since we already did 98% of the code in class. I just copy + pasted and then added on a little bit from there, such as changing up the css styles and modifying some code here and there.

Saved: 4/13/2025 7:25:03 PM

100%

Task #3 (0.33 pts.) - What was the hardest part of the assign

Text Prompt

Weight: *33.33%*
Objective: *What was the hardest part of the assignment?*

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

Your Response:

The hardest part of the assignment for me was catching up with everything. I was behind a lot so I struggled with all these errors in a small time frame, so I'll try and be on time as much as I can next time and keep up with everything.

Saved: 4/13/2025 7:23:52 PM