

Course: IT202-008-S2025

Assignment: IT202 Milestone 1

Student: Dylan W. (dw347)

Status: Submitted | Worksheet Progress: 88.46%

Potential Grade: 7.70/10.00 (77.00%)

Received Grade: 0.00/10.00 (0.00%)

Grading Link: <https://learn.ethereallab.app/assignment/v3/IT202-008-S2025/it202-milestone-1/grading/dw347>

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`

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

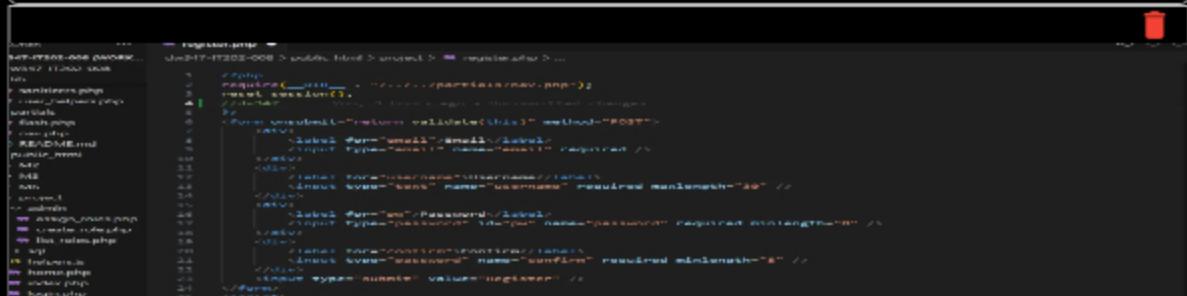
- System should allow the user to correct the error without wiping/clearing the form (for the PHP side this includes sticky-form logic to keep the username/email address entries)

☞ Image Prompt

Weight: 50%

Details:

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



code



Email: dw347@jilt.edu
Username: dw347
Password: ...

Please lengthen this text to 6 characters or more. (you are currently using 3 characters)

pw too short



The chosen email is not available

Email
Username
Password
Confirm
Register

email taken



The chosen email is not available.

Email: dw347njlt.edu
Username
Password
Confirm
Register

Please include an '@' in the email address. 'dw347njlt.edu' is missing an '@'.

email wrong



This chosen email is not available.

Email: dw347@jilt.edu
Username: dw347
Password:
Confirm:

dw347-it202-008-dev-1023681d3b09.herokuapp.com says:
Passwords do not match.

passwords dont match



A screenshot of a web browser window. The URL is <https://dw847-it202-00n-dev-f028c61dbb09.herokuapp.com/project/register.php>. The page has a header with 'Login' and 'Register' buttons. A green banner at the top says 'Successfully registered!'. Below the banner, there is some placeholder text: 'Email: _____', 'Username: _____', 'Password: _____', 'Confirm: _____', and a 'Register' button.

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

A screenshot of a code editor showing a PHP file named 'register.php'. The code contains a function named 'validateForm' which performs various validations on form inputs. It checks for empty fields, valid email format, and password complexity. It also checks if the password matches the confirmation. If any validation fails, it returns an error message. If all validations pass, it returns 'Valid'.

```
function validateForm() {
    // Check if all fields are filled
    if (form.username.value === "" || form.password.value === "" || form.confirm.value === "") {
        return "All fields are required";
    }

    // Check if email is valid
    const emailRegex = /^[^\s]+@[^\s]+\.[^\s]+$/;
    if (!emailRegex.test(form.username.value)) {
        return "Email is not valid";
    }

    // Check if password is valid
    const passwordRegex = /^(?=.*[a-zA-Z])(?=.*\d)[a-zA-Z\d]{8,16}$/;
    if (!passwordRegex.test(form.password.value)) {
        return "Password must be at least 8 characters long and contain both letters and numbers";
    }

    // Check if password matches confirmation
    if (form.password.value !== form.confirm.value) {
        return "Passwords do not match";
    }

    return "Valid";
}
```

code



A screenshot of a web browser window. The URL is <https://dw847-it202-00n-dev-f028c61dbb09.herokuapp.com/project/register.php>. The page has a header with 'Login' and 'Register' buttons. A red banner at the top says 'Username must only contain a-z, A-Z, ., or -'. Below the banner, there is a form with fields: 'Email' (placeholder: 'Email'), 'Username' (placeholder: 'Username'), 'Password' (placeholder: 'Password'), 'Confirm' (placeholder: 'Confirm'), and a 'Register' button.

user wrong



The screenshot shows a registration form on a website. The URL is <https://dw347-it202-000-dev-f023c61d3b09.herokuapp.com/project/register.php>. The form has fields for Email, Username, Password, and Confirm. An error message at the top states: "Username must only contain 3-30 characters a-z, 0-9, ., or -". A tooltip on the "Email" field says: "Please enter a valid email address". A tooltip on the "Username" field says: "Username must be between 3 and 30 characters". A tooltip on the "Password" field says: "Please lengthen this text to 8 characters or more (you are currently using 4 characters)". A tooltip on the "Confirm" field says: "Passwords do not match". A blue "OK" button is visible in the top right corner of the error message box.

email wrong

The screenshot shows a registration form on a website. The URL is <https://dw347-it202-000-dev-f023c61d3b09.herokuapp.com/project/register.php>. The form has fields for Email, Username, Password, and Confirm. An error message at the top states: "Username must only contain 3-30 characters a-z, 0-9, ., or -". A tooltip on the "Email" field says: "Please enter a valid email address". A tooltip on the "Username" field says: "Username must be between 3 and 30 characters". A tooltip on the "Password" field says: "Please lengthen this text to 8 characters or more (you are currently using 4 characters)". A tooltip on the "Confirm" field says: "Passwords do not match". A blue "OK" button is visible in the top right corner of the error message box.

user too long or short

The screenshot shows a registration form on a website. The URL is <https://dw347-it202-000-dev-f023c61d3b09.herokuapp.com/project/register.php>. The form has fields for Email, Username, Password, and Confirm. An error message at the top states: "Username must only contain 3-30 characters a-z, 0-9, ., or -". A tooltip on the "Email" field says: "Please enter a valid email address". A tooltip on the "Username" field says: "Username must be between 3 and 30 characters". A tooltip on the "Password" field says: "Please lengthen this text to 8 characters or more (you are currently using 4 characters)". A tooltip on the "Confirm" field says: "Passwords do not match". A blue "OK" button is visible in the top right corner of the error message box.

password too short

The screenshot shows a registration form on a website. The URL is <https://dw347-it202-000-dev-f023c61d3b09.herokuapp.com/project/register.php>. The form has fields for Email, Username, Password, and Confirm. An error message at the top states: "Passwords do not match". A blue "OK" button is visible in the top right corner of the error message box.

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

Combo Task:

Weight: 33.33%

Image Prompt

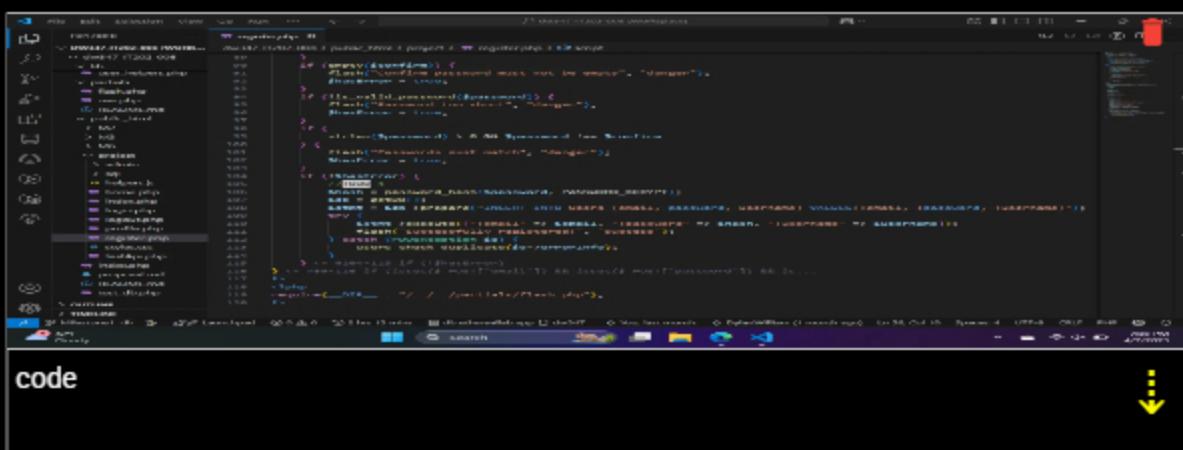
Weight: 50%

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



code



code



email cant be empty

The screenshot shows a registration form on a website. The URL is <https://dw347-i202-008-dev-f023c61d3b09.herokuapp.com/project/register.php>. The form has fields for Email, Username, Password, and Confirm. The Email field contains "dw347@njit.edu". The Password and Confirm fields both have the placeholder "*****". A yellow error bar at the top says "The chosen email is not available." Below the form, a message "user cant be empty" is displayed.

The screenshot shows a registration form on a website. The URL is <https://dw347-i202-008-dev-f023c61d3b09.herokuapp.com/project/register.php>. The form has fields for Email, Username, Password, and Confirm. The Email field contains "dw347@njit.edu". The Password and Confirm fields both have the placeholder "*****". A yellow error bar at the top says "The chosen email is not available.". Another error bar on the right says "dw347-i202-008-dev-f023c61d3b09.herokuapp.com says: Passwords do not match.". Below the form, a message "password dont match" is displayed.

The screenshot shows a registration form on a website. The URL is <https://dw347-i202-008-dev-f023c61d3b09.herokuapp.com/project/register.php>. The form has fields for Email, Username, Password, and Confirm. The Email field contains "dw347@njit.edu". The Password and Confirm fields both have the placeholder "*****". A yellow error bar at the top says "The chosen email is not available.".

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

https://github.com/dyylan2018/dw347-IT202-008-Milestone1/public_html/project/register.php		https://github.com/dyylan2018/dw347-IT202-008-prod-39957dd	
https://dw347-it202-008-project.herokuapp.com/project/register.php		https://github.com/dyylan2018/dw347-IT202-008	
https://github.com/dyylan2018/dw347-IT202-008		https://github.com/dyylan2018/dw347-IT202-008	

Saved: 4/7/2025 7:23:38 PM

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

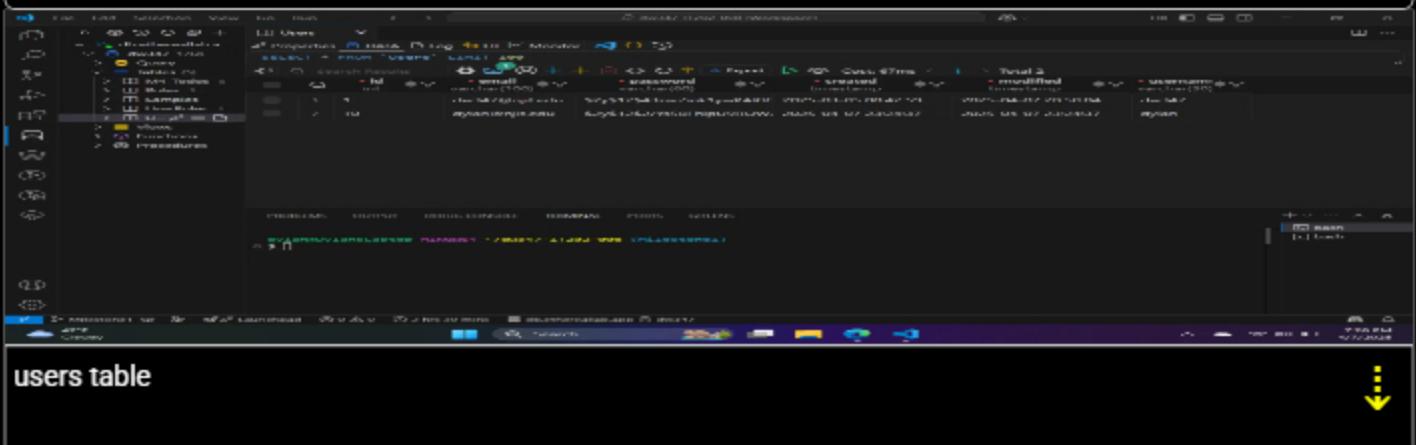
Image Prompt

Weight: 33.33%

Objective: *Database - Users table*

Details:

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



Saved: 4/7/2025 7:26:31 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)



File	Line	Content
index.php	40	<input type="text" value="dw347@njit.edu"/>
index.php	41	<input type="password" value="12345678"/>
index.php	42	<input type="submit" value="Login"/>
index.php	43	Invalid password

code



A screenshot of a web browser displaying a login form. The URL is https://dw347-it202-008-dev-f022e61d2b09.herokuapp.com/project/login.php. The form has two fields: 'Email/Username' containing 'dw347@njit.edu' and 'Password' containing '12345678'. Below the fields is a button labeled 'Login'. A red error message 'Invalid password' is displayed above the password field.

sticky form keeps user entry while PW is wrong



Saved: 4/7/2025 9:52:20 PM

Text Prompt

Weight: 50%

Details:

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

Your Response:

This login form combines client-side and server-side validation for security and usability: it uses JavaScript (`validate()`) for preliminary checks on submission (though not shown here, likely verifying email/username format and password requirements), while the `novalidate` attribute overrides default HTML5 validation to allow custom checks. The form submits via POST to securely transmit credentials, and the email input is protected against XSS attacks using `htmlspecialchars()` to escape previously entered values. HTML5 attributes (`required`, `minlength="8"`) enforce non-empty fields and an 8-character

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

A screenshot of a code editor displaying a file named `login.php`. The code contains a `validate()` function that checks if the email/username field is empty and if the password is at least 6 characters long. If either condition fails, it sets a flash message and returns `false`. Otherwise, it returns `true`.

```
function validate($form) {
    // This is implemented via JavaScript validation
    $isEmailValid = true;
    $isPwValid = true;
    $isPw = $form['password'].value;
    if (!isValidEmail($isEmail)) {
        Flash("Hey Buddy That's Wrong");
        $isEmailValid = false;
    }
    if (!isValidPassword($isPw)) {
        Flash("Hey Buddy That's Wrong");
        $isPwValid = false;
    }
    return $isEmailValid & $isPwValid;
}
```

code

A screenshot of a web browser displaying a login page at `http://dw347-it202-000-dev-t023cGId3b09.herokuapp.com/project/login.php`. The page has "Login" and "Register" buttons. A validation error message "Hey Buddy That's Wrong" is displayed above the input fields. The "Email/Username" field contains "dw347" and the "Password" field contains three dots, indicating it is too short.

password too short

Saved: 4/7/2025 9:53:01 PM

☞ Text Prompt

Weight: 50%

Details:

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

Your Response:

This JavaScript validation function checks if the password meets security requirements by calling `isValidPassword()` (a presumed helper function that verifies password strength), displaying an error message via `flash()` if validation fails, and returning `false` to prevent form submission; if valid, it returns `true`, allowing the form to proceed, ensuring only properly formatted passwords are submitted while providing user feedback.



Saved: 4/7/2025 9:53:01 PM

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

The screenshot shows a code editor with a dark theme. On the left is a file tree with files like `index.php`, `user_register.php`, `password.php`, `email.php`, and `username.php`. The main pane displays PHP code for a login page. It includes functions for validating email and password, and handling user registration. A red arrow points to the bottom right corner of the code editor.

```
function validate_email($email) {
    $email = trim($email);
    $email = filter_var($email, FILTER_SANITIZE_EMAIL);
    $email = filter_var($email, FILTER_VALIDATE_EMAIL);
    if ($email === false) {
        flash("Email is not valid");
        return false;
    }
    return true;
}

function validate_password($password) {
    $password = trim($password);
    if (empty($password)) {
        flash("Password is required");
        return false;
    }
    if (!preg_match('/^(?=.*[A-Z])(?=.*[a-z])(?=.*\d)[A-Za-z\d]{8,}$/', $password)) {
        flash("Password must be at least 8 characters long and contain at least one uppercase letter, one lowercase letter, and one digit.");
        return false;
    }
    return true;
}

function register_user($username, $email, $password) {
    $username = trim($username);
    $email = trim($email);
    $password = trim($password);
    if (empty($username) || empty($email) || empty($password)) {
        flash("All fields are required");
        return false;
    }
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        flash("Email is not valid");
        return false;
    }
    if (!validate_password($password)) {
        flash("Password is not valid");
        return false;
    }
    // Check if user exists
    if (user_exists($username)) {
        flash("User already exists");
        return false;
    }
    // Hash password
    $password_hash = password_hash($password, PASSWORD_DEFAULT);
    // Insert into database
    $query = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";
    $stmt = $db->prepare($query);
    $stmt->bind_param("sss", $username, $email, $password_hash);
    if ($stmt->execute() === false) {
        flash("Error registering user");
        return false;
    }
    $stmt->close();
    return true;
}

function user_exists($username) {
    $query = "SELECT * FROM users WHERE username = ?";
    $stmt = $db->prepare($query);
    $stmt->bind_param("s", $username);
    $stmt->execute();
    $result = $stmt->get_result();
    if ($result->num_rows > 0) {
        $stmt->close();
        return true;
    }
    $stmt->close();
    return false;
}
```

code

```
index.php
1 <?php
2 // Database connection
3 $conn = new mysqli("localhost", "root", "", "test");
4
5 if ($conn->connect_error) {
6     die("Connection failed: " . $conn->connect_error);
7 }
8
9 // Create a new role
10 $role_name = "newRole";
11 $query = "CREATE ROLE $role_name";
12 $conn->query($query);
13
14 // Grant SELECT privilege on all tables to the new role
15 $grant_query = "GRANT SELECT ON *.* TO '$role_name'";
16 $conn->query($grant_query);
17
18 // Set password for the new role
19 $password = "password123";
20 $set_password_query = "SET PASSWORD FOR '$role_name' = '$password'";
21 $conn->query($set_password_query);
22
23 // Test the new role
24 $test_query = "SELECT * FROM users WHERE id = 1";
25 $result = $conn->query($test_query);
26
27 if ($result->num_rows > 0) {
28     echo "Success: User found!";
29 } else {
30     echo "Error: User not found!";
31 }
32
33 // Close connection
34 $conn->close();
35
```

code

```
index.php
1 <?php
2 // Database connection
3 $conn = new mysqli("localhost", "root", "", "test");
4
5 if ($conn->connect_error) {
6     die("Connection failed: " . $conn->connect_error);
7 }
8
9 // Create a new role
10 $role_name = "newRole";
11 $query = "CREATE ROLE $role_name";
12 $conn->query($query);
13
14 // Grant SELECT privilege on all tables to the new role
15 $grant_query = "GRANT SELECT ON *.* TO '$role_name'";
16 $conn->query($grant_query);
17
18 // Set password for the new role
19 $password = "password123";
20 $set_password_query = "SET PASSWORD FOR '$role_name' = '$password'";
21 $conn->query($set_password_query);
22
23 // Test the new role
24 $test_query = "SELECT * FROM users WHERE id = 1";
25 $result = $conn->query($test_query);
26
27 if ($result->num_rows > 0) {
28     echo "Success: User found!";
29 } else {
30     echo "Error: User not found!";
31 }
32
33 // Close connection
34 $conn->close();
35
```

code

Login Register

Email must not be empty

Invalid username

Email/Username

Password

invalid user and email empty

Welcome, dw347

Home Profile Create Role List Roles Assign Roles Logout

Home

login successful

Login Register

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://github.com/dyylan2018/dw347-_____



UR

https://github.com/dyylan2018/dw347-_____



URL #2

https://dw347-it202-008-_____prod-39957dda50b0.herokuapp.com/project/login.php



UR

https://dw347-it202-008-_____prod-39957dda50b0.herokuapp.com/project/login.php



URL #3

https://github.com/dyylan2018/dw347-_____IT202-008-_____prod-39957dd9



UR

https://github.com/dyylan2018/dw347-_____IT202-008-_____prod-39957dd9



Saved: 4/7/2025 8:04:24 PM

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

```

    <array>
      <item><name>user</name><value>User</value></item>
      <item><name>password</name><value>password</value></item>
    </array>
  </appSettings>
</configuration>

```

heroku logs



Saved: 4/7/2025 8:11:55 PM

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

Task #1 (1 pt.) - Evidence

Combo Task:

Weight: 100%

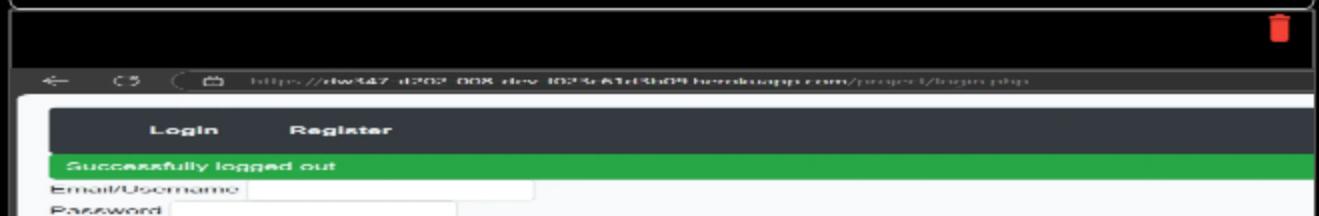
Objective: Evidence

≡, Image Prompt

Weight: 40%

Details:

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



Login

output message



```
<?php  
session_start();  
  
if (!isset($_SESSION['username'])) {  
    header('Location: login.php');  
    exit();  
}  
  
$username = $_SESSION['username'];  
  
echo "Welcome, $username!";  
  
if (isset($_POST['logout'])) {  
    session_destroy();  
    header('Location: login.php');  
    exit();  
}  
  
// More code...  
?>
```

code snippet



Saved: 4/7/2025 9:55:36 PM

Url Prompt

Weight: 20%

Details:

- Include the pull request url for this feature

URL #1

<https://github.com/dyylan2018/dw347-IT20250069>



URL

<https://github.com/dyylan2018/dw347-IT20250069>



Saved: 4/7/2025 9:55:36 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 securely terminates the user session by first destroying all session data with `session_destroy()`, invalidating the authentication state, then clearing the session cookie by setting its expiration to a past timestamp, ensuring no residual credentials remain on the client side.

Setting its expiration to a past timestamp, clearing its session cookie from the client side, and finally redirecting the user to the login page via header ("Location: login.php") to complete the logout cycle and prevent further access to protected resources.



Saved: 4/7/2025 9:55:36 PM

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

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

The screenshot shows the MySQL Workbench interface with the 'roles' table selected in the left sidebar. The table has three columns: 'id', 'name', and 'description'. There are two rows of data: one for 'Administrator' and one for 'Employee'. The 'Administrator' row has a timestamp of '2023-03-31 17:08:30' and a description of 'User can perform all database operations'. The 'Employee' row has a timestamp of '2023-04-01 17:08:30' and a description of 'User can perform basic database operations'.

	name	description
1	Administrator	2023-03-31 17:08:30 User can perform all database operations
2	Employee	2023-04-01 17:08:30 User can perform basic database operations

The screenshot shows the MySQL Workbench interface with the 'UserRoles' table selected in the left sidebar. The table has four columns: 'id', 'user_id', 'role_id', and 'start_date'. There are two rows of data: one for 'user1' with role 'Administrator' starting on '2023-01-01' and another for 'user2' with role 'Employee' starting on '2023-01-01'. Both rows have an end date of '2023-01-01'.

	user_id	role_id	start_date	end_date
1	user1	Administrator	2023-01-01	2023-01-01
2	user2	Employee	2023-01-01	2023-01-01

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

The screenshot shows a web browser window with a URL starting with "https://dwddz-it202-001-dev-fuxxerdb09.herokuapp.com/project/login.php". The page has a dark header with "Login" and "Register" buttons. Below the header is a yellow banner with the text "You must be logged in to view this page". There are two input fields labeled "Email/Username" and "Password", and a "Login" button.

must be logged in

The screenshot shows a web browser window with a URL starting with "https://dwddz-it202-001-dev-fuxxerdb09.herokuapp.com/project/login.php". The page has a dark header with "Login" and "Register" buttons. Below the header is a yellow banner with the text "You don't have permission to view this page" and "You must be logged in to view this page". There are two input fields labeled "Email/Username" and "Password", and a "Login" button.

must have access

```
#!/usr/bin/php
function is_logged_in($redirect = false, $destination = "login.php") {
    $isLoggedin = !isset($_SESSION["user"]);
    // If this triggers, the calling script must receive a reply since it's unlikely to terminates the
    // script("You must be logged in to view this page", "warning");
    die("You must be logged in to view this page", "warning");
    if ($isLoggedin) {
        return $isLoggedin;
    }
}
```

login check function

role check function



Saved: 4/7/2025 9:58:19 PM

≡ Url Prompt

Weight: 20%

Details:

- Include the pull request url for this feature

URL #1

<https://github.com/dyylan2018/dw347-IT2020D089>



UR

<https://github.com/dyylan2018/dw>



Saved: 4/7/2025 9:58:19 PM

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

Task #1 (1 pt.) - Validation

Weight: 50%

Objective: Validation

Details:

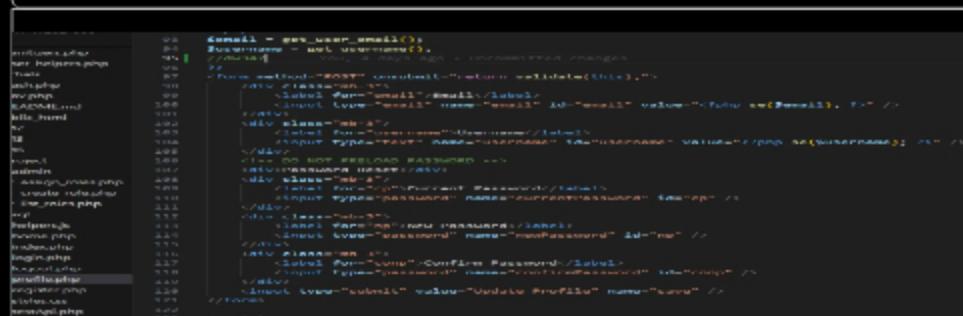
- Show the relevant code snippets for each validation layer
- Show the relevant demo of each validation layer
- Briefly explain how the validation steps work at each layer

Sub-Tasks:**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)



A screenshot of a terminal window displaying several lines of PHP code. The code includes HTML form fields and validation logic. It features multiple `<input>` tags with attributes like `name="username"`, `name="email"`, and `name="password"`. There are also `if` statements and validation messages such as "Email is required" and "Email must be valid". The code is part of a larger script, likely a registration or login form.

code snippet

Profile saved

Email: dw347@njit.edu

Username: dw347

Password Reset

Current Password

New Password

Confirm Password

profile saved output



Saved: 4/7/2025 9:13:44 PM

Text Prompt

Weight: 50%**Details:**

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

Your Response:

This form implements secure profile updates with client-side validation (via the validate() function) and server-side protection. It collects the user's email and username (pre-filled safely with se() to prevent XSS), while requiring the current password for verification and new password fields (not preloaded) for resetting. The form ensures proper password confirmation and uses POST method to securely transmit sensitive data, while the onsubmit validation prevents submission if requirements aren't met, balancing usability with security against unauthorized changes.

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

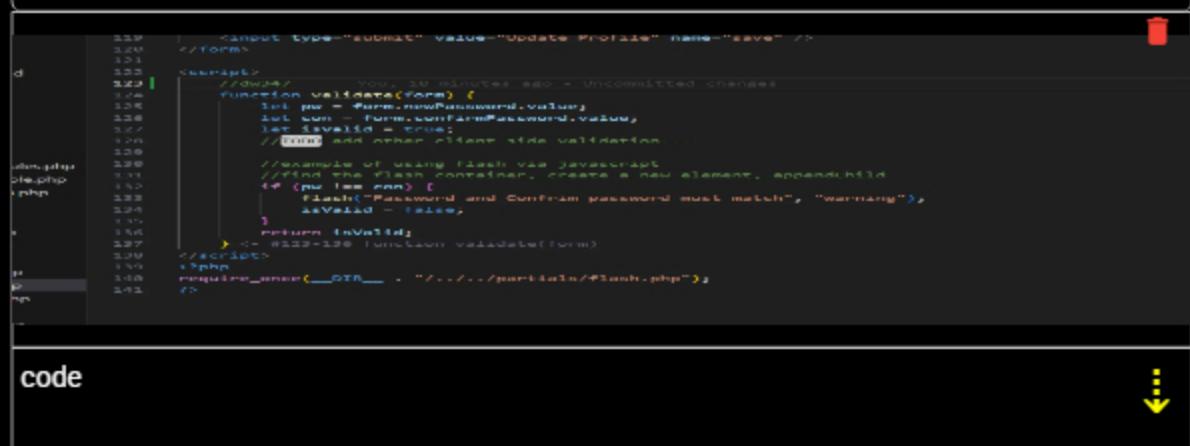
Combo Task:

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

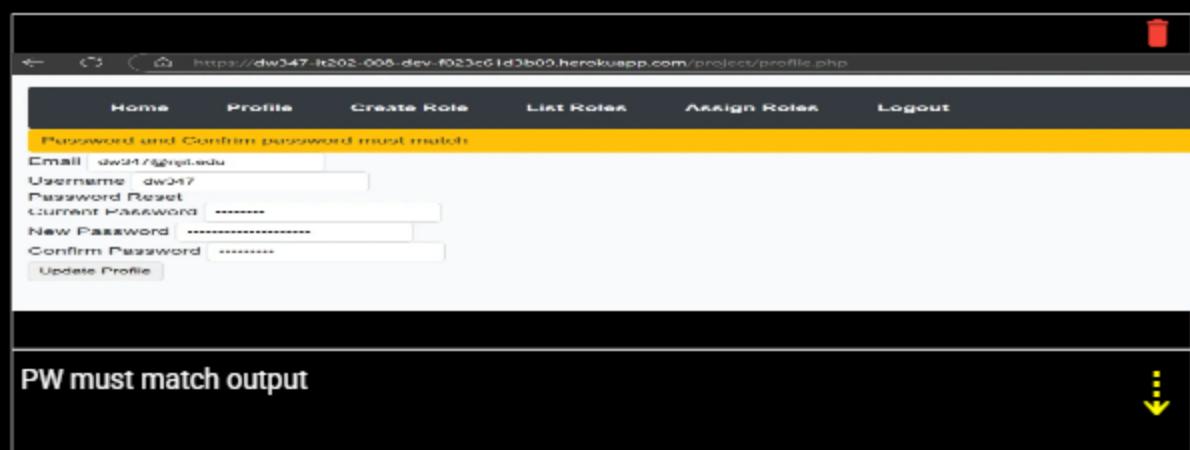


```

3.20 <input type="text" value="Update Profile" name="submit" />
3.21
3.22 // Example of using Flash var javascript
3.23 function validateForm() {
3.24     let pw = Form.password.value;
3.25     let con = Form.confirmPassword.value;
3.26     let isValid = true;
3.27     // DOH! and other client side validation...
3.28
3.29     // Example of using Flash var javascript
3.30     // If not the flash container, create a new element, append it to
3.31     // the form
3.32     if (pw != con) {
3.33         Flash("Password and Confirm password must match", "warning");
3.34         isValid = false;
3.35     }
3.36
3.37     return isValid;
3.38 }
3.39 </script>
3.40 </form>
3.41 require_once('includes/DB.php');
3.42
3.43 </body>
3.44 </html>

```

code



Home Profile Create Role List Roles Assign Roles Logout

Password and Confirm password must match

Email: dw347@gmail.com
 Username: dw347
 Password Reset
 Current Password:
 New Password:
 Confirm Password:

PW must match output

 Saved: 4/7/2025 9:23:36 PM

Text Prompt

Weight: 50%

Details:

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

Your Response:

This JavaScript validation function ensures password consistency during profile updates by comparing the new password (newPassword) and confirmation (confirmPassword) fields, returning false and triggering a warning flash message if they don't match (preventing form submission), while maintaining security by performing initial client-side checks before server-side validation processes the sensitive data. The isValid flag allows for expandable

Server-side validation processes the sensitive data. The `isValid` flag allows for expandable validation (implied by the TODO comment) to incorporate additional checks like password complexity or current password verification in future implementations.

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

A screenshot of a code editor showing a file named `profile.php. The code contains logic for validating user input. It includes a getProfile() method that performs several database queries to check for existing users based on email and username. It also includes validation logic for username, email, and password fields, including checks for length and matching confirmation. The code uses MySQL syntax for queries and PHP for conditional logic.`

code

A screenshot of a code editor showing a file named `profile.php`. This version of the code is similar to the previous one but includes additional validation logic. It checks if the `username` and `email` fields are empty. It also includes a `checkUser()` function that performs a database query to verify if a user with the given email already exists. The code uses MySQL syntax for queries and PHP for conditional logic.

code

```
    if (check_password($new_password, $current_password)) {
        // Update database
        $query = "UPDATE users SET password = '$password' WHERE id = '$id'";
        $stmt = $conn->prepare($query);
        $stmt->execute();
        $stmt->close();
        $conn->close();
        $response = [
            "status" => "success",
            "message" => "Password updated successfully."
        ];
        $error = null;
    } else {
        $response = [
            "status" => "error",
            "message" => "Current password is invalid. Please try again."
        ];
        $error = "Current password is invalid. Please try again.";
    }
    $headers = ["Content-Type: application/json"];
    $response["headers"] = $headers;
    return $response;
} else {
    $response = [
        "status" => "error",
        "message" => "New passwords don't match. Please enter the same password again."
    ];
    $error = "New passwords don't match. Please enter the same password again.";
}

```

Invalid email address

Email: dw24@gmail.com

Username: dw24

Password Reset

Current Password

New Password

Confirm Password

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://github.com/dyylan2018/dw347->



<https://github.com/dyylan2018/dw347>

IT2018Milestone1/public_html/project/profile.php

URL #2

<https://dw347-it202-008->



<https://dw347-it202-008-prod-39957dd>

Section #6: (1 pt.) Misc

Task #1 (0.33 pts.) - Github Details

Combo Task:

Weight: 33.33%

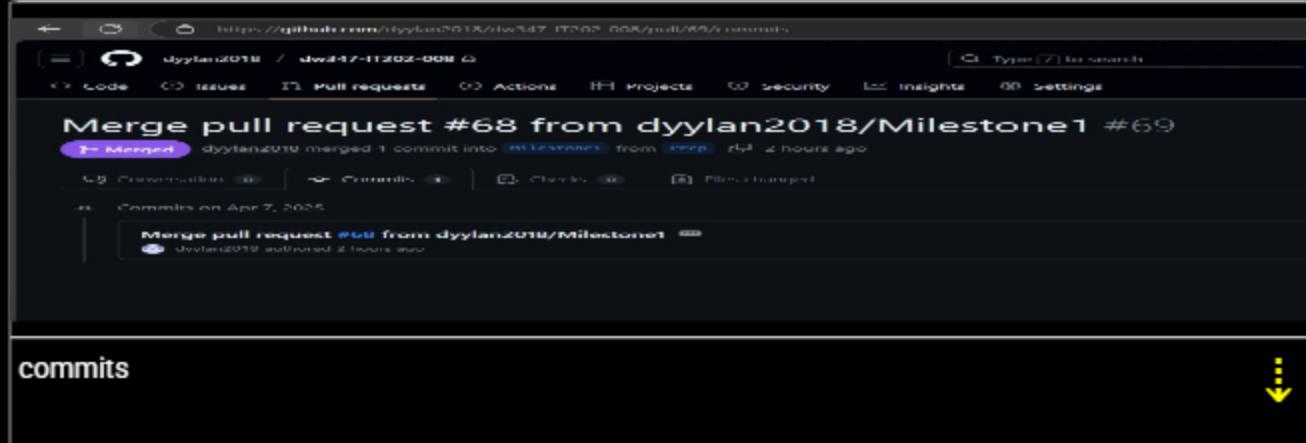
Objective: *Github Details*

☞ Image Prompt

Weight: 60%

Details:

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



Saved: 4/7/2025 9:42:35 PM

☞ Url Prompt

Weight: 40%

Details:

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

URL #1

<https://github.com/dyylan2018/dw347->

IT2024-08-09



UH

<https://github.com/dyylan2018/dw>



Saved: 4/7/2025 9:42:35 PM

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



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



Task #3 (0.33 pts.) - Reflection

Weight: 33.33%

Sub-Tasks:

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

☞ Text Prompt

Weight: 33.33%

Objective: *What did you learn?*

Details:

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

Your Response:

I learned alot about php, javascript, html scripting and how to implement all of that into a webpage.



Saved: 4/7/2025 9:44:34 PM

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

☞ 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 html scripting was the easiest for me



Saved: 4/7/2025 9:45:11 PM

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:

submitting was the hardest (sorry)



Saved: 4/7/2025 9:45:27 PM