

# Submission Worksheet

**CLICK TO GRADE**

<https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/tlj3>

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

## Submissions:

Submission Selection

1 Submission [active] 4/1/2024 6:43:32 PM

## Instructions

[^ COLLAPSE ^](#)

## Prereqs:

Go through each lesson from "Project Setup and SQL" to "User Login Enhancement" and follow the branching names while gathering the code

Merge each into Milestone1 branch

Mark the related GitHub Issues items as "done"

Implement your own custom CSS (something much different than the default "ugly" CSS given as an example)

Consider styling all forms/inputs, data output, navigation, etc

Implement JavaScript validation on Register, Logout, and Profile (include "[Client]" in the output messages to differentiate between server-side validations)

## Instructions:

Make sure you're in Milestone1 with the latest changes pulled

Ensure Milestone1 has been deployed to heroku dev

Gather the requested evidence and fill in the explanations per each prompt

Save the submission and generate the output PDF

Put the output PDF into your local repository folder

add/commit/push it to GitHub

Merge Milestone1 into dev

Locally checkout dev and pull the changes

Create and merge a pull request from dev to prod to deploy Milestone1 to prod

Upload this output PDF to Canvas

**Branch name:** Milestone1

Tasks: 26 Points: 10.00



User Registration (2 pts.)

[^ COLLAPSE ^](#)

[COLLAPSE ^](#)

## Task #1 - Points: 1

Text: Screenshot of form on website page

### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required)
#4	1	Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)
#5	1	Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)
#6	1	Demonstrate user-friendly message of new account being created

### Task Screenshots:

#### Gallery Style: Large View

Small

Medium

Large

The screenshot shows a web browser window with a registration form titled "Login/Register". The URL in the address bar is `tjj3-dev-b3a210191069.herokuapp.com/Project//register.php`. The browser's developer tools are open, specifically the Elements tab, showing the HTML structure and CSS styles for the body element. The form has four input fields: Email, Username, Password, and Confirm. The Email field is highlighted with a yellow background and has the error message "Email must not be empty". The developer tools also show the console log with the message "Changed type to text for element" followed by a timestamp.

## Screenshot shows email is empty

### Checklist Items (0)

Email must not be empty  
Username must not be empty  
Email (e)  
Username (e)  
Password (e)  
Confirm (e)  
Register

HTML body

body { background-color: cornflowerblue; color: rgb(5, 0, 153); }

body { display: block; margin: 8px; }

margin: 8px; border: 1px solid black;

Console What's new

Default levels | 6 Issues: 3 3 2 hidden

confirm  
Changed type to text for element [VH44:47](#)  
confirm  
↳ undefined

## Screenshot shows username is empty

### Checklist Items (0)

Email must not be empty  
Username must not be empty  
Password must not be empty  
Email (e)  
Username (e)  
Password (e)  
Confirm (e)  
Register

HTML body

body { background-color: cornflowerblue; color: rgb(5, 0, 153); }

body { display: block; margin: 8px; }

margin: 8px; border: 1px solid black;

Console What's new

Default levels | 6 Issues: 3 3 2 hidden

confirm  
Changed type to text for element [VH44:47](#)  
confirm  
↳ undefined

## Screenshot shows password is empty

### Checklist Items (0)

A screenshot of a web browser displaying a registration form. The URL is <https://tij3-dev-b3a210191069.herokuapp.com/Project//register.php>. The page title is "Login/Register". The registration form includes fields for Email, Username, Password, and Confirm. The Password and Confirm fields are currently empty. A yellow box highlights the entire registration area. The browser's developer tools are open, showing the HTML structure and CSS styles for the body element.

## Screenshot shows password needs to be longer

### Checklist Items (0)

A screenshot of a web browser displaying a registration form. The URL is <https://tij3-dev-b3a210191069.herokuapp.com/Project//register.php>. The page title is "Login/Register". The registration form includes fields for Email, Username, Password, and Confirm. The Password and Confirm fields contain the value "nt123456". A yellow box highlights the entire registration area. The browser's developer tools are open, showing the HTML structure and CSS styles for the body element. An error message "Passwords do not match" is visible near the bottom of the registration form.

## Screenshot shows that the password doesn't match confirm password

### Checklist Items (0)

A screenshot of a web browser displaying a registration form. The URL is <https://tj3-dev-b3a210191069.herokuapp.com/Project//register.php>. The page title is "Login Register". The form has four input fields: Email, Username, Password, and Confirm. All fields have red borders indicating they are required. The "Email" field contains "tj3@ yahoo.com". The "Username" field contains "tj3". The "Password" field contains "tj3". The "Confirm" field contains "tj3". Below the form, a yellow box highlights the validation error message: "Email must not be empty". The browser's developer tools are open, showing the CSS styles for the body element and the JavaScript console with an error message: "Changed type to text for element VM44:42".

## Screenshot shows invalid email

### Checklist Items (0)

A screenshot of a web browser displaying a registration form. The URL is <https://tj3-dev-b3a210191069.herokuapp.com/Project//register.php>. The page title is "Login Register". The form has four input fields: Email, Username, Password, and Confirm. All fields have red borders indicating they are required. The "Email" field contains "tj3@ yahoo.com". The "Username" field contains "tj3". The "Password" field contains "tj3". The "Confirm" field contains "tj3". Below the form, a yellow box highlights the validation error message: "Email must not be empty". The browser's developer tools are open, showing the CSS styles for the body element and the JavaScript console with an error message: "Changed type to text for element VM44:42".

## Screenshot shows invalid username

### Checklist Items (0)

The screenshot shows a web browser window with a registration form. The URL is [tj3-dev-b3a210191069.herokuapp.com/Project//register.php](http://tj3-dev-b3a210191069.herokuapp.com/Project//register.php). The form has fields for Email, Username, Password, and Confirm. An error message "The chosen email is not available." is displayed above the Email field. The browser's developer tools are open, showing the CSS styles for the body element and the element containing the error message.

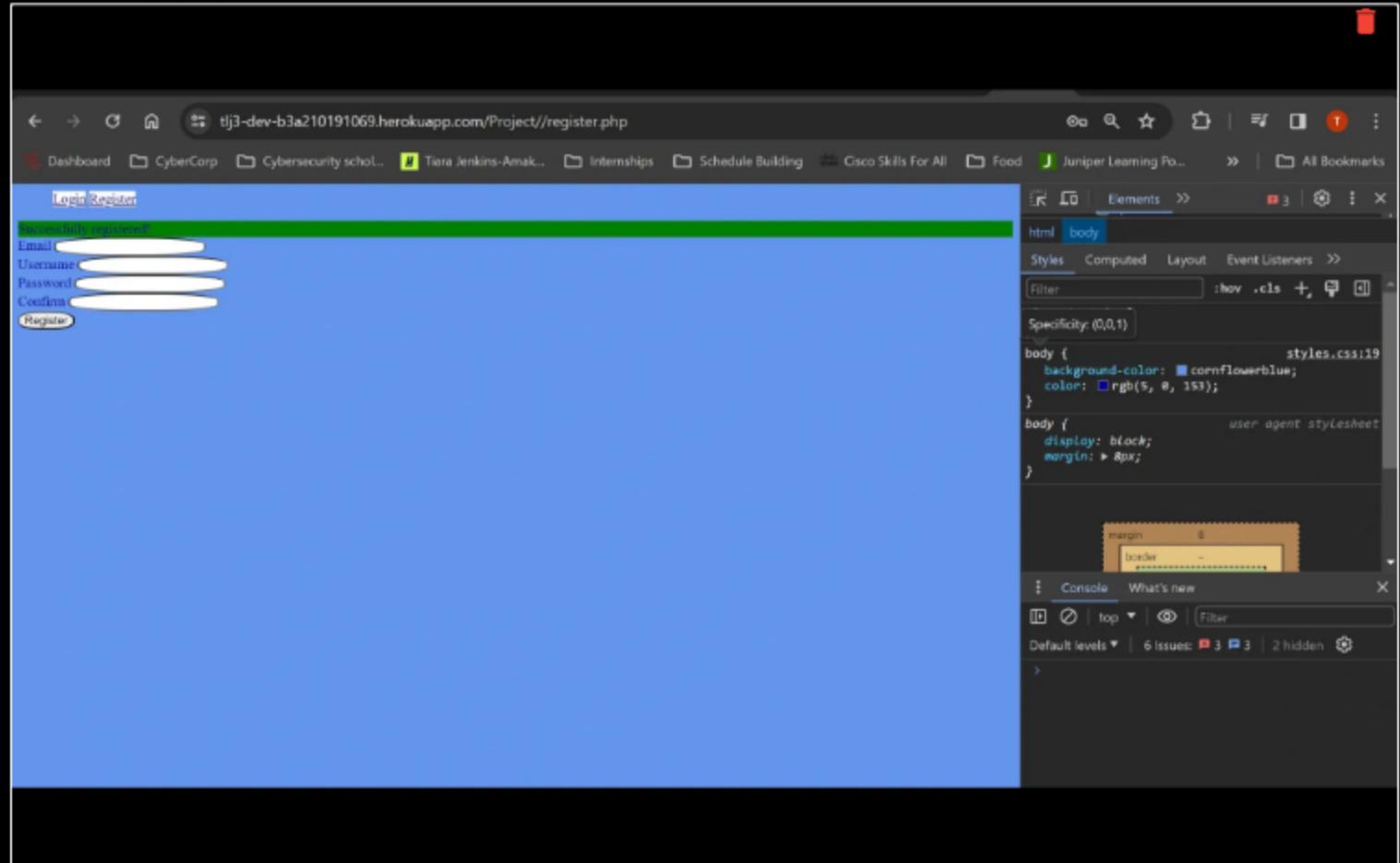
Screenshot shows that the email chosen isn't available

### Checklist Items (0)

The screenshot shows a web browser window with a registration form. The URL is [tj3-dev-b3a210191069.herokuapp.com/Project//register.php](http://tj3-dev-b3a210191069.herokuapp.com/Project//register.php). The form has fields for Email, Username, Password, and Confirm. An error message "The chosen username is not available." is displayed above the Username field. The browser's developer tools are open, showing the CSS styles for the body element and the element containing the error message.

Screenshot shows that the chosen username isn't available

Checklist Items (0)



Screenshot shows a successful registration

Checklist Items (0)

Task #2 - Points: 1

Text: Screenshot of the form code

**Details:**

Should have appropriate input types for the field

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

A screenshot of a code editor showing a PHP file named validate.php. The code includes client-side validation using JavaScript and server-side validation using PHP. The validation logic checks for email, username, password, and confirmation fields.

```
1 <?php
2 require(__DIR__ . "/../../partials/nav.php");
3 reset_session();
4 ?>
5 <form onsubmit="return validate(this)" method="POST">
6   <div>
7     <label for="email">Email</label>
8     <input type="email" name="email" required />
9   </div>
10  <div>
11    <label for="username">Username</label>
12    <input type="text" name="username" required maxlength="30" />
13  </div>
14  <div>
15    <label for="pw">Password</label>
16    <input type="password" id="pw" name="password" required minlength="8" />
17  </div>
18  <div>
19    <label for="confirm">Confirm</label>
20    <input type="password" name="confirm" required minlength="8" />
21  </div>
22  <input type="submit" value="Register" />
23 </form>
24 <script>
25   function validate(form) {
26     //TODO 1: implement JavaScript validation
27     //ensure it returns false for an error and true for success
28     let email = form.email.value;
29     let username = form.username.value;
```

Screenshot includes the form code.

### Task #3 - Points: 1

Text: Screenshot of the client-side and server-side validation code

#### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
#1	1	Show the JavaScript validations (include any extra files related)
#2	1	Show the PHP validations (include any lib content)
#3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

#### Task Screenshots:

##### Gallery Style: Large View

Small      Medium      Large

A screenshot of a code editor showing a PHP file named validate.php. The code includes client-side validation using JavaScript and server-side validation using PHP. The validation logic checks for email, username, password, and confirmation fields.

```
1 <?php
2 require(__DIR__ . "/../../partials/nav.php");
3 reset_session();
4 ?>
5 <form onsubmit="return validate(this)" method="POST">
6   <div>
7     <label for="email">Email</label>
8     <input type="email" name="email" required />
9   </div>
10  <div>
11    <label for="username">Username</label>
12    <input type="text" name="username" required maxlength="30" />
13  </div>
14  <div>
15    <label for="pw">Password</label>
16    <input type="password" id="pw" name="password" required minlength="8" />
17  </div>
18  <div>
19    <label for="confirm">Confirm</label>
20    <input type="password" name="confirm" required minlength="8" />
21  </div>
22  <input type="submit" value="Register" />
23 </form>
24 <script>
25   function validate(form) {
26     //TODO 1: implement JavaScript validation
27     //ensure it returns false for an error and true for success
28     let email = form.email.value;
29     let username = form.username.value;
```

```
31 let password = form.password.value;
32 let confirm = form.confirm.value;
33 let emailRegex = /^[^\\s@]+@[^\\s@]+\.[^\\s@]+$/;
34 let usernameRegex = /^[a-zA-Z0-9_]{3,20}$/;
35
36 if (email === "") {
37   flash("Email must not be empty", "warning");
38   return false;
39 }
40
41 if (username === "") {
42   flash("Username must not be empty", "warning");
43   return false;
44 }
45
46 if (password === "") {
47   flash("Password must not be empty", "warning");
48   return false;
49 }
50
51 if (!emailRegex.test(email) && !usernameRegex.test(username)) {
52   flash("Invalid email address or username", "warning");
53   return false;
54 }
55
56 You, 2 hours ago + Uncommitted changes
57
58
59
60
61
62
63
64
65
66 } <- #25-66 function validate(form)
67
```

You, 2 hours ago  
IT202 tij3  
7:12 PM 4/1/2024

## Part 1 of client side validation code.

### Checklist Items (0)

```
24 <script>
25   function validate(form) {
26
27     if (username === "") {
28       flash("Username must not be empty", "warning");
29       return false;
30     }
31
32     if (password === "") {
33       flash("Password must not be empty", "warning");
34       return false;
35     }
36
37     if (!emailRegex.test(email) && !usernameRegex.test(username)) {
38       flash("Invalid email address or username", "warning");
39       return false;
40     }
41
42     if (password.length < 8) {
43       flash("Password must be at least 8 characters long", "warning");
44       return false;
45     }
46
47     if (password !== confirm) {
48       flash("Passwords do not match", "warning");
49       return false;
50     }
51
52     return true;
53   } <- #25-66 function validate(form)
54
```

You, 2 hours ago + Uncommitted changes  
IT202 tij3  
7:12 PM 4/1/2024

## Part 2 Client Side Validation

### Checklist Items (0)

```
68 <?php
69 //TODO: 2: add PHP Code
70 if (isset($_POST["email"]) && isset($_POST["password"]) && isset($_POST["confirm"]) && isset($_POST["username"])){
71   $email = se($_POST, "email", "", false);
72   $password = se($_POST, "password", "", false);
73 }
```

```
73 > sql
74 & dont use
75 & helpers.js
76 & home.php
77 & index.php
78 & login.php M
79 & logout.php
80 & profile.php M
81 & register.php M
82 # styles.css
83 & index.php
84 & php.ini
85 & README.md
86 & test_db.php
87 & .gitignore
88 & .htaccess
89 & composer.json
90 & composer.lock
91 & Profile
92 > OUTLINE
93 > TIMELINE
94 & Feat-UserLoginEnhancement* & 0 0 0 0 0 0 Connect 3 hrs 29 mins IT202 tja You, 2 hours ago Ln 59, Col 10 Spaces: 4 UTF-8 CRLF PHP Go Live 7:13 PM 4/1/2024
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
```

## Part 1 of Server Side Validation

### Checklist Items (0)

```
public_html > Project > register.php > script > validate
96 if (!empty($confirm)) {
97     flash("Confirm password must not be empty", "danger");
98     $hasError = true;
99 }
100 if (!is_valid_password($password)) {
101     flash("Password too short", "danger");
102     $hasError = true;
103 }
104 if (
105     strlen($password) > 0 && $password !== $confirm
106 ) {
107     flash("Passwords must match", "danger");
108     $hasError = true;
109 }
110 if (!$hasError) {
111     //TODO: 4
112     $hash = password_hash($password, PASSWORD_BCRYPT);
113     $db = getDB();
114     $stmt = $db->prepare("INSERT INTO Users (email, password, username) VALUES(:email, :password, :username)");
115     try {
116         $stmt->execute([':email' => $email, ':password' => $hash, ':username' => $username]);
117         flash("Successfully registered!", "success");
118     } catch (PDOException $e) {
119         users_check_duplicate($e->errorInfo);
120     }
121 } <- #110-121 IF (!$hasError)
122 } <- #70-122 IF (isset($_POST["email"]) && isset($_POST["password"]) && is...
123 ?>
124 </php>
```

## Part 2 of Server Side Validation

### Checklist Items (0)

```

EXPLORER    ... roles.php functions.php ✘ get_url.php nav.php create_role.php 003_create_table_roles.sql 004 ...
File Edit Selection View Go Run ... ← → tij3-it202-008
lib > functions.php > ...
lib > functions.php > ...
You, 5 days ago | 1 author (You)
1 <?php
2 //TODO 1: require db.php
3 require_once(__DIR__ . "/db.php");
4 //This is going to be a helper for redirecting to our base project path since it's nested in another folder
5 //This MUST match the folder name exactly
6 $BASE_PATH = '/Project/';
7 //we moved the flash require higher so following functions have access to it
8 //TODO 4: Flash Message Helpers You, last week + Milestone 1 Housekeeping
9 require(__DIR__ . "/flash_messages.php");
10
11 //require safer_echo.php
12 require(__DIR__ . "/safer_echo.php");
13 //TODO 2: filter helpers

```

### Function.php that was part of the user registration process.

#### Checklist Items (0)

#### Task #4 - Points: 1

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

#### Checklist

\*The checkboxes are for your own tracking

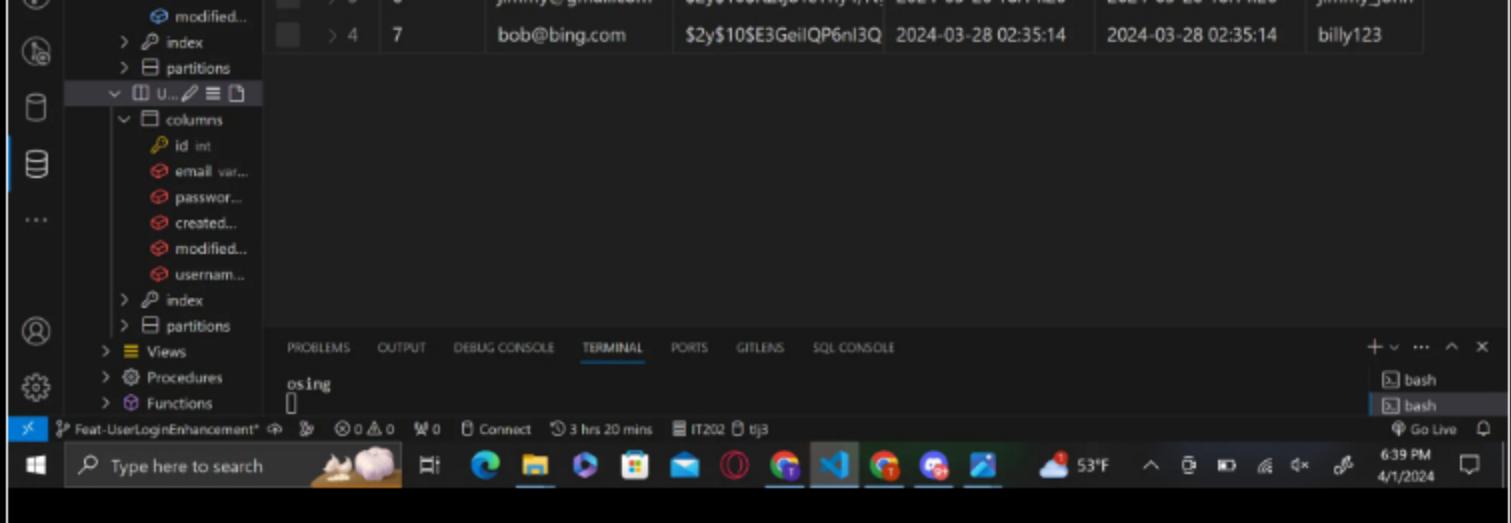
#	Points	Details
#1	1	Password should be hashed
#2	1	Should have email, password, username (unique), created, modified, and id fields
#3	1	Ensure left panel or database name is present (should contain your ucid)

#### Task Screenshots:

#### Gallery Style: Large View

Small      Medium      Large

	id	email	password	created	modified	username
1	3	johnny@gmail.com	\$2y\$10\$N5UI2qfAtplC	2024-03-18 18:45:41	2024-03-18 20:52:04	johnny_boy
2	5	bob@google.com	\$2y\$10\$Sx/Oy0bHSWrd	2024-03-18 20:53:04	2024-03-18 20:53:04	bobby
3	6	jimmy@nmail.com	\$2v\$10\$RZtioYeYhw4/Nr	2024-03-20 18:14:26	2024-03-20 18:14:26	jimmy_john



This screenshot includes the User table, with the email, username, Id, password(hashed), created, and modified

## Checklist Items (0)

### Task #5 - Points: 1

**Text:** Explain the registration logic in a step-by-step manner from when the page loads to when the data is saved to the DB

#### ⓘ Details:

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

## Response:

When the page loads, the user is left with a form that contains an email, username, password, and confirm password field. The user will fill those out while following the requirements and then submits it. The form will be sent to the server via an HTTP POST request. When its received, it goes through the server side validation to check for anything that doesn't meet the requirements (empty fields, valid email/user/password format, strong password, etc.). Once it passes the validation, the server interacts with the db to store the information from the user's form (including the username, email, password, id, etc.). Before the password is saved, it gets hashed using a hashing algorithm to ensure that the password isn't stored in plain text for enhanced security. Once the data is saved to the db, the server responds to the user with a success message.

### Task #6 - Points: 1

**Text:** Include pull request links related to this feature

#### ⓘ Details:

Should end in /pull/#

## URL #1

<https://github.com/Tiara-Jenks/tli3-it202-008/pull/13>

User Login (2 pts.)

[^ COLLAPSE ^](#)

Task #1 - Points: 1

Text: Screenshot of form on website page

Checklist

\*The checkboxes are for your own tracking

#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Include correct data where username is used to login
#4	1	Include correct data where email is used to login
#5	1	Show success login message
#6	1	Demonstrate JavaScript validation for each field [can be combined] (username format, email format, password format, and each field being required)
#7	1	Demonstrate user-friendly message of when an account doesn't exist
#8	1	Demonstrate user-friendly message of when password doesn't match what's in the DB
#9	1	Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)
#10	1	Demonstrate session data being set (captured from server logs)

Task Screenshots:

Gallery Style: Large View

Small      Medium      Large



← → ⌂ ⌂ tlj3-dev-b3a210191069.herokuapp.com/Project//login.php

Dashboard CyberCorp Cybersecurity schol... Tiara Jenkins-Amak... Internships S...

[Login](#) [Register](#)

Email/Username bob@aol.com

Password \*\*\*\*\*

Login

screenshot of the correct data where email is used to login

Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL: tlj3-dev-b3a210191069.herokuapp.com/Project//login.php. The page content includes a "Login" link and a "Register" link. Below them are two input fields: "Email/Username" containing "billy456" and "Password" containing "\*\*\*\*\*". A "Login" button is located below the password field.

screenshot of the correct data where username is being used to login

Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL: tlj3-dev-b3a210191069.herokuapp.com/Project//home.php. The page content includes a navigation bar with links: Dashboard, CyberCorp, Cybersecurity school, Tiara Jenkins-Amak..., Internships, and Schedule Build. Below the navigation bar, there is a user profile section with a yellow square icon, the name "Tiara Jenkins-Amak...", and a "Logout" button.

Welcome, billy456

# Home

login success message

## Checklist Items (0)

The screenshot shows a web browser window with the URL [tlj3-dev-b3a210191069.herokuapp.com/Project//login.php](https://tlj3-dev-b3a210191069.herokuapp.com/Project//login.php). The page displays a login form with two error messages at the top: "Email Username and password must not be empty". The "Email Username" field contains "Qmnothy@gmail.com" and the "Password" field contains "123123123". The "Login" button is visible below the fields. On the right side of the browser, the developer tools' Elements tab is open, showing the HTML structure and the CSS styles applied to the body element. The CSS rules include a background color of cornflowerblue and a margin of 8px.

email and username are empty

## Checklist Items (0)

The screenshot shows a web browser window with the URL [tlj3-dev-b3a210191069.herokuapp.com/Project//login.php](https://tlj3-dev-b3a210191069.herokuapp.com/Project//login.php). The page displays a login form with two error messages at the top: "Email Username and password must not be empty". The "Email Username" field contains "Qmnothy@gmail.com" and the "Password" field contains "123123123". The "Login" button is visible below the fields. On the right side of the browser, the developer tools' Elements tab is open, showing the HTML structure and the CSS styles applied to the body element. The CSS rules include a background color of cornflowerblue and a margin of 8px.

```
body { background-color: #cornflowerblue; color: #rgb(5, 0, 153); }
body { display: block; margin: 0 8px; }

margin: 0;
border: 1px solid black;

margin: 0;
border: 1px solid black;

margin: 0;
border: 1px solid black;

margin: 0;
border: 1px solid black;
```

Console What's new X

Default levels ▾ 3 Issues: 1 2 1 hidden Filter

password

Changed type to text for element Y492:47 password

6 undefined >

password is empty

## Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL `tj3-dev-b3a210191069.herokuapp.com/Project//login.php`. The page title is "Login/Register". There are two input fields: "Email/Username" and "Password", both of which have been redacted. Below the inputs is a "Login" button. In the bottom right corner of the browser window, there is a red trash can icon. The browser's developer tools are open, specifically the "Console" tab, which displays the error message "password is empty". The "Elements" tab shows the HTML structure of the page.

password is empty

## Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL `tj3-dev-b3a210191069.herokuapp.com/Project//login.php`. The page title is "Login/Register". There are two input fields: "Email/Username" and "Password", both of which have been redacted. Below the inputs is a "Login" button. In the bottom right corner of the browser window, there is a red trash can icon. The browser's developer tools are open, specifically the "Console" tab, which displays the error message "email not found". The "Elements" tab shows the HTML structure of the page.

email not found

The screenshot shows a browser window with a login form and its corresponding developer tools. The browser's address bar contains the URL 'http://www.bing.com'. The login form includes fields for 'Email/Username' (bob@bing.com) and 'Password' (bob), and a 'Login' button.

The developer tools are open, specifically the Elements and Elements Inspector panes. In the Elements pane, the body element is selected, showing the following CSS rules:

```
body { background-color: # cornflowerblue; color: #rgb(5, 0, 153); }  
body { display: block; margin: 0px; }
```

The Elements Inspector shows the current style for the 'margin' property of the body element, which is set to '0'. It also shows the 'border' property as 'none'.

The bottom pane, titled 'Console', displays the following message:

```
password  
Changed type to text for element VM185:47  
password  
↳ undefined  
↳
```

**password doesn't meet requirements**

### Checklist Items (0)

The screenshot shows a web browser window with the URL `tj3-dev-b3a210191069.herokuapp.com/Project//login.php`. The page displays an error message: "Invalid password". Below the message are two input fields for "Email/Username" and "Password", both of which are redacted with white rectangles. A "Login" button is located below the fields. To the right of the page, the browser's developer tools are open, specifically the Elements and Styles panels. The Styles panel shows the CSS rule for the body element:

```
body {  
background-color: #cornflowerblue;  
color: #rgb(5, 0, 253);  
}  
body {  
display: block;  
margin: 8px;  
}
```

The browser's address bar and various tabs are visible at the top. The developer tools also show the DevTools menu and a small preview of the current element.

**password didn't match the one in the db**

### Checklist Items (0)

[Login](#) [Register](#)

Invalid email address or username

Email/Username bob.com

Password •••••

[Login](#)

email and username are invalid as it doesn't meet the requirements

Checklist Items (0)

The screenshot shows the developer tools application tab open in a browser. On the left, there's a sidebar with 'Application' selected, showing 'Manifest', 'Service workers', and 'Storage'. Under 'Storage', 'Session storage' and 'Cookies' are expanded. A cookie named 'PHPSESSID' is listed under 'Cookies'. The main panel displays a table of cookies with columns for Name, Value, Domain, Path, Expires, Size, HttpOnly, Secure, SameSite, Part of, and Priority. Below the table, a message says 'Select a cookie to preview its value'. At the bottom, there's a 'Issues' section with three items: 'An element doesn't have an autocomplete attribute', 'Incorrect use of <label for=FORM\_ELEMENT>', and 'Page layout may be unexpected due to Quirks Mode'. There are also checkboxes for 'Group by kind' and 'Include third-party cookie issues'.

session data being captured (professor told me about the error\_logs from home.php in application, but those are the only errors that I saw. If this isn't correct, please tell me in the notes on canvas or anywhere else)

Checklist Items (0)

This screenshot is identical to the one above, showing the developer tools application tab with session storage and cookies. It includes the same sidebar, table of cookies, message to select a cookie, and the 'Issues' section with the same three items and checkboxes.

The screenshot shows the Chrome DevTools Network tab. A POST request to 'https://tij3-dev-b3a210' has been made. In the 'Headers' section, there is a 'Cookie' header with the value 'PHPSESSID=t7g00qe0dirkh1r5al3frorj5hnrvun'. In the 'Body' section, there is a JSON payload: {"username": "Bob.com", "password": "\*\*\*\*\*"}.

**Network**

Name	Value	Do...	Path	Expi...	Size	Htt...	Sec...	Sam...	Part...	Prio...
PHPSESSID	t7g00qe0dirkh1r5al3frorj5hnrvun	.tij3...	/Pro...	202...	41	✓	✓	Lax		Me...

**Application**

- Manifest
- Service workers
- Storage

**Storage**

- Local storage
- Session storage
- IndexedDB
- Web SQL
- Cookies
  - https://tij3-dev-b3a210
  - Private state tokens
  - Interest groups
  - Shared storage

Select a cookie to preview its value

Console What's new Issues

Default levels ▾ 3 Issues: 1 2 1 hidden

this is a part 2 to the session data being set. I saw this in the Nav and login part 2 video on Learn and just posted this here to be on the safe side. Again, if this isn't correct, please tell me.

## Checklist Items (0)

### Task #2 - Points: 1

**Text:** Screenshot of the form code

#### Checklist

\*The checkboxes are for your own tracking

#	Points	Details
#1	1	Should have proper input types for the fields (note in the caption if you're using two fields for Username/Email or a combined field)
#2	1	Show JavaScript validations (include any extra files related)
#3	1	Show PHP validations (include any lib content)
#4	1	Include uid/date code comments in all screenshots (one per screenshot is sufficient)

#### Task Screenshots:

Gallery Style: Large View

Small      Medium      Large



```
login.php X register.php helpers.js profile.php 005_insert_roles_admin.sql list_roles.php assi 40 0 0 0 0 ...  
public_html > Project > login.php > ...  
You 1 hour ago | 1 author (You)  
1 <?php  
2 require(__DIR__ . "/../../partials/nav.php");  
3 ?>  
4 <form onsubmit="return validate(this)" method="POST">  
5     <div>  
6         <label for="email">Email/Username</label>  
7         <input type="text" name="email" required />  
8     </div>  
9     <div>  
10        <label for="pw">Password</label>  
11        <input type="password" id="pw" name="password" required minlength="8" />  
12    </div>  
13    <input type="submit" value="Login" />  
14 </form>  
15 <script>
```

Screenshot of form code (used a combined field for email/username)

## Checklist Items (0)

```
public_html > Project > login.php > ...  
15     <script>  
16         function validate(form) {  
17             //TODO update clientside validation to check if it should  
18             //valid email or username  
19  
20             let emailOrUsername = form.email.value;  
21             let password = form.password.value;  
22             let emailRegex = /^[^@\s]+@[^\s]+\.\w+$/;  
23             let usernameRegex = /^[a-zA-Z0-9_]{3,20}$/;  
24  
25             if (emailOrUsername === "" || password === "") {  
26                 flash("Email/Username and password must not be empty", "warning");  
27                 return false;  
28             }  
29  
30             if (!emailRegex.test(emailOrUsername) && !usernameRegex.test(emailOrUsername)) {  
31                 flash("Invalid email address or username", "warning");  
32                 return false;  
33             }  
34  
35             if (password.length < 8) {  
36                 flash("Password must be at least 8 characters long", "warning");  
37                 return false;  
38             }  
39  
40             if (password.length > 45) {  
41                 flash("Password must be less than 45 characters long", "warning");  
42                 return false;  
43             }  
44  
45             return true;  
46         } </script>
```

Screenshot of client side validation

## Checklist Items (0)

```
public_html > Project > login.php > ...  
48 <?php
```

```
public_html
    > Project
        > admin
        > sql
        > dont use
        > helpers.js
        > home.php
        > index.php
        > login.php
        > logout.php
        > profile.php
        > register.php
        # styles.css
    > index.php
    > php.ini
    README.md
    test_db.php
    .gitignore
    .htaccess
    composer.json
    composer.lock
    Profile
    > OUTLINE
    > TIMELINE
```

```
49 //TODO 2: add PHP Code
50 if (isset($_POST["email"]) && isset($_POST["password"])) {
51     $email = se($_POST, "email", "", false);
52     $password = se($_POST, "password", "", false);
53
54     //TODO 3
55     $hasError = false;
56     if (empty($email)) {
57         flash("Email must not be empty");
58         $hasError = true;
59     }
60     if (str_contains($email, "@")) {
61         //sanitize
62         //$email = filter_var($email, FILTER_SANITIZE_EMAIL);
63         $email = sanitize_email($email);
64         //validate
65         /*if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
66             flash("Invalid email address");
67             $hasError = true;
68         }*/
69         if (!is_valid_email($email)) {
70             flash("Invalid email address");
71             $hasError = true;
72         }
73     } else {
74         if (!is_valid_username($email)) {
75             flash("Invalid username");
76             $hasError = true;
77         }
78     }
79 } <- #73-78 else
80 if (empty($password)) {
81     flash("password must not be empty");
82 }
83 if (!is_valid_password($password)) {
84     flash("Password too short");
85     $hasError = true;
86 }
87 if (!$hasError) {
88     //flash("Welcome, $email");
89     //TODO 4
90     $db = getDB();
91     $stmt = $db->prepare("SELECT id, email, username, password from Users
92     where email = :email or username = :email");
93     try {
94         $r = $stmt->execute([":email" => $email]);
95         if ($r) {
96             $user = $stmt->fetch(PDO::FETCH_ASSOC);
97             if ($user) {
98                 $hash = $user["password"];
99                 unset($user["password"]);
100                if (password_verify($password, $hash)) {
101                    //flash("Welcome $email");
102                    $_SESSION["user"] = $user; //sets our session data from db
103                    //lookup potential roles
104                    $stmt = $db->prepare("SELECT Roles.name FROM Roles
105                     JOIN UserRoles on Roles.id = UserRoles.role_id
106                     where UserRoles.user_id = :user_id and Roles.is_active = 1 and UserRoles.is_active = 1")
107                    $stmt->execute([":user_id" => $user["id"]]);
```

Screenshot of server side part1

## Checklist Items (0)

```
TLJ3-IT202-008
public_html > Project > login.php > ...
    > M4
    > Project
        > admin
        > sql
        > dont use
        > helpers.js
        > home.php
        > index.php
        > login.php
        > logout.php
        > profile.php
        > register.php
        # styles.css
    > index.php
    > php.ini
    README.md
    test_db.php
    .gitignore
    .htaccess
    composer.json
    composer.lock
    Profile
    > OUTLINE
    > TIMELINE
```

```
78 } <- #73-78 else
79 if (empty($password)) {
80     flash("password must not be empty");
81 }
82 if (!is_valid_password($password)) {
83     flash("Password too short");
84     $hasError = true;
85 }
86 if (!$hasError) {
87     //flash("Welcome, $email");
88     //TODO 4
89     $db = getDB();
90     $stmt = $db->prepare("SELECT id, email, username, password from Users
91     where email = :email or username = :email");
92     try {
93         $r = $stmt->execute([":email" => $email]);
94         if ($r) {
95             $user = $stmt->fetch(PDO::FETCH_ASSOC);
96             if ($user) {
97                 $hash = $user["password"];
98                 unset($user["password"]);
99                 if (password_verify($password, $hash)) {
100                     //flash("Welcome $email");
101                     $_SESSION["user"] = $user; //sets our session data from db
102                     //lookup potential roles
103                     $stmt = $db->prepare("SELECT Roles.name FROM Roles
104                      JOIN UserRoles on Roles.id = UserRoles.role_id
105                      where UserRoles.user_id = :user_id and Roles.is_active = 1 and UserRoles.is_active = 1")
106                     $stmt->execute([":user_id" => $user["id"]]);
```

Screenshot of server side part2

## Checklist Items (0)

```

107     $stmt->execute([":user_id" => $user["id"]]);
108     $roles = $stmt->fetchAll(PDO::FETCH_ASSOC); //fetch all since we'll want multiple
109     //save roles or empty array
110     if ($roles) {
111         $_SESSION["user"]["roles"] = $roles; //at least 1 role
112     } else {
113         $_SESSION["user"]["roles"] = []; //no roles
114     }
115     flash("Welcome, " . get_username());
116     die(header("Location: home.php"));
117     } else {
118         flash("Invalid password");
119     }
120     } else {
121         flash("Email not found");
122     }
123     } <- #97-123 if ($user)
124     } catch (Exception $e) {
125         flash("<pre>" . var_export($e, true) . "</pre>");
126     }
127 } <- #93-127 try
128 } <- #91-128 $stmt = $db->prepare
129 ?>
130 </php
131 require(__DIR__ . "/../../partials/flash.php"); <- #50-131 if (isset($_POST["email"]) && isset($_POST["passw

```

Screenshot of server side part3

## Checklist Items (0)

### Task #3 - Points: 1

**Text:** Explain the login logic step-by-step from when the page loads to when the data is fetched from the DB and stored in the session

Checklist		*The checkboxes are for your own tracking
#	Points	Details
#1	1	Don't just show code, translate things to plain English
#2	1	Explain how the session works and why/how it's used

## Response:

When the page is loaded, the user is left with a form containing email/username and password fields. The user has to enter their credentials and submit the form. This data is sent to the server via HTTP POST request. Once received, it goes through the server side validation to ensure there's no empty fields, ensure the username/email exists within the db, and to ensure the password matches the one in the db. If the credentials pass the validation, the server interacts with the db to get the user's data based on the credentials provided (querying the users table to retrieve the user's records). Once the records are received, the server verifies the password submitted against the hashed password in the db, ensuring that they match. If this turns out successful, the server creates a new session for the user. The creation of a session involved generating a unique session ID and associating it with the data from the user (session ID is stored as a cookie).

▲ COLLAPSE ▲

#### Task #4 - Points: 1

**Text:** Include pull request links related to this feature

i **Details:**

Should end in /pull/#

**URL #1**

<https://github.com/TiaraJenks/tlj3-it202-008/pull/14>

▲ COLLAPSE ▲

User Logout (1 pt.)

▲ COLLAPSE ▲

#### Task #1 - Points: 1

**Text:** Capture the following screenshots

**Checklist**

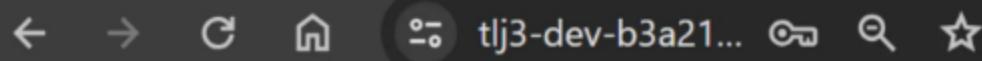
\*The checkboxes are for your own tracking

#	Points	Details
#1	1	Screenshot of the navigation when logged in (site)
#2	1	Screenshot of the redirect to login with the user-friendly logged-out message (site)
#3	1	Screenshot of the logout-related code showing the session is destroyed (code). Ensure ucid/date comment is present.

**Task Screenshots:**

Gallery Style: Large View

Small      Medium      Large



[Dashboard](#) [CyberCorp](#) [Cybersecurity schol...](#)

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

Welcome, billy456

**Home**

## Navigation when logged in

Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL "tlj3-dev-b3a210191...". Below the address bar is a navigation bar with icons for back, forward, search, and a star. The main content area has a dark header with a logo and the text "Dashboard", "CyberCorp", and "Cybersecurity schol...". Below the header is a blue bar with "Login" and "Register" buttons. A green banner at the top of the page says "Successfully logged out". Below the banner are fields for "Email/Username" and "Password", both of which are redacted with white ovals. A "Login" button is located below the password field.

Logged out on site with user-friendly message

Checklist Items (0)

A screenshot of a file explorer interface. On the left, there is a sidebar with various icons: a folder, a magnifying glass, a gear, a circular arrow, a file, and a square. The main area shows a project structure under "public\_html > Project > logout.php". The "logout.php" file is open in a code editor. The code in the editor is:

```
You, last week | 1 author (You)
<?php      You, last month • Basic navigation
session_start();
require(__DIR__ . "/../../../../lib/functions.php");
reset_session();
flash("Successfully logged out", "success");
header("Location: login.php");
```

login.php  
logout.php  
profile.php

logout-related code showing the session is destroyed

Checklist Items (0)

Task #2 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

<https://github.com/TiaraJenks/tlj3-it202-008/pull/14>

Basic Security Rules and Roles (2 pts.)

Task #1 - Points: 1

Text: Authentication Screenshots

Task #2 - Points: 1

Text: Authorization Screenshots

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

Task #4 - Points: 1

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

Task #5 - Points: 1

Text: Include pull request links related to this feature

User Profile (2 pts.)



[^ COLLAPSE ^](#)

Task #1 - Points: 1

Text: View Profile Website Page



[▼ EXPAND ▼](#)

Task #2 - Points: 1

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



[▼ EXPAND ▼](#)

Task #3 - Points: 1

Text: Edit Profile Website Page



[▼ EXPAND ▼](#)

Task #4 - Points: 1

Text: Explain the logic step-by-step of how the data is checked and saved for the following scenarios



[▼ EXPAND ▼](#)

Task #5 - Points: 1

Text: Include pull request links related to this feature



[▼ EXPAND ▼](#)

Misc (1 pt.)



[^ COLLAPSE ^](#)

Task #1 - Points: 1

Text: Screenshot of wakatime



[▼ EXPAND ▼](#)

Task #2 - Points: 1

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



[▼ EXPAND ▼](#)

Task #3 - Points: 1

Text: Provide a direct link to the project board on GitHub



[▼ EXPAND ▼](#)

Task #4 - Points: 1

▼ EXPAND ▼

TASK #4 - POINTS: 1

Text: Provide a direct link to the login page from your prod instance

End of Assignment