

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/lm457>

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

Submissions:

Submission Selection

1 Submission [active] 3/31/2024 5:38:43 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
<input checked="" type="checkbox"/> #1	1	Heroku dev url should be present in the address bar
<input checked="" type="checkbox"/> #2	1	Should have thoughtful CSS applied
<input checked="" type="checkbox"/> #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)
<input checked="" type="checkbox"/> #4	1	Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)
<input checked="" type="checkbox"/> #5	1	Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)
<input checked="" type="checkbox"/> #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 multiple tabs open. The active tab displays a registration form titled 'Register' at the top. The form includes fields for Email, Username, Password, and Confirm, each with a blue placeholder text box. Below the fields is a blue 'Register' button. The browser's address bar shows the URL: https://it202-lm457-dev-7b44497d46f0.herokuapp.com/project/register.php. The browser interface includes standard navigation buttons (back, forward, search) and a toolbar with icons for refresh, stop, and other functions.

The initial register page deployed on Heroku dev containing the fields required to make an account (email, username, pw).

Checklist Items (2)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

The screenshot shows a web browser window with multiple tabs. The active tab is a registration form at <https://it202-lm457-dev-7b44497d46f0.herokuapp.com/project/register.php>. The form has four fields: Email, Username, Password, and Confirm. The Email field contains "testcase@" and has a red border with the error message "[JS] Invalid email address". The other three fields contain "testcase1", "abababab", and "abababab" respectively. A "Register" button is at the bottom. The developer tools' Elements panel is open, showing the HTML structure of the page. The Network panel shows several requests made by the page. The Console panel is also visible.

JS Email Validation.

Checklist Items (1)

#3 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)

This screenshot is similar to the previous one, showing the same registration form at the same URL. The Email field now contains " testcase1@gmail.com" and has a red border with the error message "[JS] Username must only contain 3-16 characters a-z, 0-9, _ or -". The other fields and the developer tools are identical to the previous screenshot.

A screenshot of the Chrome DevTools Network panel. At the top, there's a header bar with tabs for 'Console' and 'What's new'. Below the header, a message says 'Highlights from the Chrome 122 update'. The main content area shows a warning titled 'Third-party cookie phaseout warnings in Network and Application'. The warning text states: 'The Network and Application panels now show you warnings next to cookies affected by the third-party cookie restrictions from Tracking'. To the right of the text is a small thumbnail image of a video player with a play button.

JS Username Validation.

Checklist Items (1)

#3 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)

The screenshot shows a web browser window with multiple tabs open. The active tab is a registration page at <https://it202-lmv57-dev-7b44497d46f0.herokuapp.com/project/register.php>. The page displays a registration form with four fields: Email, Username, Password, and Confirm. The 'Password' field contains 'abab' and has a red error message above it: 'JS] Password too short'. The 'Confirm' field also contains 'abab'. Below the form are two buttons: 'Login' and 'Register'.

The browser's developer tools are open, specifically the 'Console' tab. It shows a series of JavaScript logs and changes made to the DOM elements. The logs include:

```
[{"name": "Email", "value": "testcase1@gmail.com"}, {"name": "Username", "value": "testcase1"}, {"name": "Password", "value": "abab"}, {"name": "Confirm", "value": "abab"}]
```

Below these logs, there is a list of changes:

- Removed required from element email VM123:49
- Changed type to text for element email VM123:49
- Removed required from element username VM123:32
- Removed minlength from element username VM123:32
- Removed required from element password VM123:32
- Removed minlength from element password VM123:32
- Changed type to text for element password VM123:49
- Removed required from element confirme VM123:32
- Removed minlength from element confirme VM123:32
- Changed type to text for element confirme VM123:49

At the bottom of the developer tools, there is a message about third-party cookie phaseout warnings:

Third-party cookie phaseout warnings in Network and Application

The Network and Application panels now show you warnings next to cookies affected by the third-party cookie restrictions from the Tracking

JS Password Validation.

Checklist Items (1)

#3 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)

A screenshot of a web browser window. The address bar shows the URL: https://it202-lm457-dev-7b44497d46f0.herokuapp.com/project/register.php. The page displays a registration form with three input fields: Email, Username, and Password. The Email field has the value 'testcase1@gmail.com', the Username field has 'testcase1', and the Password field has 'ababababa'. Above the form, a red error message box contains the text 'Passwords must match'. Below the form, the developer tools console is open, showing a warning message: 'Warning: [No name] has been blocked until user reloads page.'

JS PW and Confirm PW Matching Validation.

Checklist Items (1)

#3 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

Removed required from element email VM138:32
Changed type to text for element email VM138:49
Removed required from element username VM138:32
Removed maxlength from element username VM138:32
Removed required from element password VM138:32
Removed minlength from element password VM138:32
Changed type to text for element password VM138:49
Removed required from element confirm VM138:32
Removed minlength from element confirm VM138:32
Changed type to text for element confirm VM138:49

undefined

Console What's new X
Highlights from the Chrome 122 update

Third-party cookie phaseout warnings in Network and Application
The Network and Application panels now show you warnings next to cookies affected by the third-party cookie restrictions from Tracking

JS All fields need to be filled validation.

Checklist Items (1)

#3 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

Removed required from element email VM151:32
Changed type to text for element email VM151:49
Removed required from element username VM151:32
Removed maxlength from element username VM151:32
Removed required from element password VM151:32
Removed minlength from element password VM151:32
Changed type to text for element password VM151:49
Removed required from element confirm VM151:32
Removed minlength from element confirm VM151:32
Changed type to text for element confirm VM151:49

undefined

Console What's new X
Highlights from the Chrome 122 update

Third-party cookie phaseout warnings in Network and Application
The Network and Application panels now show you warnings next to cookies affected by the third-party cookie restrictions from Tracking

JS All fields need to be filled validation.

Checklist Items (1)

#3 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

The chosen email is not available.

Email

Username

Password

Confirm

Email already in use error.

Checklist Items (1)

#4 Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)

The chosen username is not available.

Email

Username

Password

Confirm

Username already in use error

Checklist Items (1)

#5 Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)

User message after successful account registration.

Checklist Items (1)

#6 Demonstrate user-friendly message of new account being created

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

```
<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="8" />
</div>
<div>
    <label for="confirm">Confirm</label>
    <input type="password" name="confirm" required minlength="8" />
</div>
<input type="submit" value="Register" />
</form>

```

Register 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
<input checked="" type="checkbox"/> #1	1	Show the JavaScript validations (include any extra files related)
<input checked="" type="checkbox"/> #2	1	Show the PHP validations (include any lib content)
<input checked="" type="checkbox"/> #3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```

<script>
    function validate(form) {
        // [15] 1: Implement JavaScript validation
        // ensure it returns false for an error and true for success
        var email = form.email.value;
        var username = form.username.value;
        var password = form.password.value;
        var confirm = form.confirm.value;

        var hasError = false;

        function is_valid_email(email) {
            const emailRegEx = /^[a-zA-Z0-9._-]{3,16}@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}+$/;
            return emailRegEx.test(email);
        }

        function is_valid_username(username) {
            const usernameRegEx = /^[a-zA-Z_]{3,16}$/;
            return usernameRegEx.test(username);
        }

        if (email === "") {
            flash("[" + 15 + "] Email must not be empty", "danger");
            hasError = true;
        } else if (!is_valid_email(email)) {
            flash("[" + 15 + "] Invalid email address", "danger");
            hasError = true;
        }
    }

```

//UCID: LM657
//Date: 3/31/2024

```

        hasError = true;
    }

    if (username === "") {
        flash(["[JS] Username must not be empty", "danger"]);
        hasError = true;
    } else if (!is_valid_username(username)) {
        flash(["[JS] Username must only contain 3-16 characters a-z, A-Z, _, or -, "danger"]);
        hasError = true;
    }

    if (password === "") {
        flash(["[JS] Password must not be empty", "danger"]);
        hasError = true;
    } else if (password.length < 8) {
        flash(["[JS] Password too short", "danger"]);
        hasError = true;
    }

    if (password !== confirm) {
        flash(["[JS] Passwords must match", "danger"]);
        hasError = true;
    }

    return !hasError;
} //-- #25-75 function validate(form)

```

JS Validation for the register page.

Checklist Items (2)

#1 Show the JavaScript validations (include any extra files related)

#3 Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

```

<?php
// [OK] 2: add PHP Code
if (isset($_POST['email']) && isset($_POST['password']) && isset($_POST['confirm']) && isset($_POST['username'])) {
    $email = se($_POST, "email", "", false);
    $password = se($_POST, "password", "", false);
    $confirm = se($_POST, "confirm", "", false);
    $username = se($_POST, "username", "", false);
    // [OK]
    $hasError = false;
    if (empty($email)) {
        #flash["Email must not be empty", "danger"];
        $hasError = true;
    }
    //sanitize
    $email = sanitize_email($email);
    //validate
    if (!is_valid_email($email)) {
        #flash["Invalid email address", "danger"];
        $hasError = true;
    }
    if (!is_valid_username($username)) {
        #flash["Username must only contain 3-16 characters a-z, A-Z, _, or -, "danger"];
        //CID: UN457
        //Date: 3/31/2024
        $hasError = true;
    }
    if (empty($password)) {
        #flash["Password must not be empty", "danger"];
        $hasError = true;
    }
    if (empty($confirm)) {
        #flash["Confirm password must not be empty", "danger"];
        $hasError = true;
    }
    if (!is_valid_password($password)) {
        #flash["Password too short", "danger"];
        $hasError = true;
    }
    if (
        strlen($password) > 0 && $password !== $confirm
    ) {
        #flash["Passwords must match", "danger"];
        $hasError = true;
    }
    if (!$hasError) {
        // [OK] 4
        $hash = password_hash($password, PASSWORD_BCRYPT);
        $db = getDB();
        $stmt = $db->prepare("INSERT INTO users (email, password, username) VALUES(:email, :password, :username)");
        try {
            $stmt->execute([":email" => $email, ":password" => $hash, ":username" => $username]);
            #flash["Successfully registered!", "success"];
        } catch (PDOException $e) {
            users_check_duplicate($e->errorInfo);
        } catch (Exception $e) {
            #flash["An unexpected error occurred, please try again", "danger"];
        }
    }
} //-- #25-133 if (!$hasError)
//-- #25-134 if (isset($_POST['email']) && isset($_POST['password']) && is...

```

PHP Validation for the register page.

Checklist Items (2)

#2 Show the PHP validations (include any lib content)

#3 Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

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

The screenshot shows a MySQL Workbench interface with the 'Users' table selected. The table has columns: id, email, password, created, modified, and username. The data is as follows:

	id	email	password	created	modified	username
1	1	whtrv@gmail.com	\$2y\$10\$UWvwNtB7HdYGr	2024-03-27 17:14:33	2024-03-27 17:41:15	likhitb2
2	2	kappaline27@gmail.com	\$2y\$10\$tcp9X3zS2f-karO!	2024-03-30 20:46:26	2024-03-31 02:57:21	sat
3	9	lmv57@njit.edu	\$2y\$10\$h7ggH6lKSHmMP	2024-03-31 02:55:12	2024-03-31 02:55:12	likhitj
4	10	testcase1@gmail.com	\$2y\$10\$wA6lhFmyIdyvcau	2024-03-31 04:25:48	2024-03-31 04:25:48	test1
5	14	tester2@gmail.com	\$2y\$10\$IOG7tuSaPRUYDL	2024-03-31 21:38:10	2024-03-31 21:38:10	tester2

The interface includes a left sidebar with database and table navigation, a top menu bar, and a bottom terminal window showing a GitHub commit message.

Users table screenshot.

Checklist Items (3)

#1 Password should be hashed

#2 Should have email, password, username (unique), created, modified, and id fields

#3 Ensure left panel or database name is present (should contain your ucid)

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

i Details:

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

Response:

The submission form is written in simple HTML and it utilizes browser validation attributes within the elements, specifically input elements. These attributes ensure certain input requirements like mandatory completion of all fields, certain password length, or email format are met. JavaScript provides additional validation and primarily serves the purpose of improving user experiences by preventing the submission of invalid values by catching them on the client side, thus reducing the burden on the program which otherwise would have to resort to server-side validation. Once the page is refreshed after submitting the form, PHP retrieves the values from the `$_POST` magic variable to perform a final validation, prioritizing less resource-intensive checks (non-DB), and it is implemented to add an additional layer of security that upholds data integrity. Lastly, SQL is used to insert user data into the database, where all of the data is entered into the appropriate columns as given by the user, except the password column, which is populated by the value obtained by hashing the password during our PHP validation. The Users table in which the data is stored is explicitly built in a manner to prevent duplicate entries in the email and username columns, therefore a try-catch block of code is used to handle exceptions in PHP as well which notifies users if their email or username is already in use.

Task #6 - Points: 1

Text: Include pull request links related to this feature

i Details:

Should end in /pull/#

URL #1

<https://github.com/Kith07/spring2024-IT202/pull/18>

URL #2

<https://github.com/Kith07/spring2024-IT202/pull/19>

User Login (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of form on website page

Task #2 - Points: 1

▼ EXPAND ▼

TASK #2 - POINTS: 1

Text: Screenshot of the form code



▼ EXPAND ▼

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



▼ EXPAND ▼

Task #4 - Points: 1

Text: Include pull request links related to this feature



User Logout (1 pt.)

▲ COLLAPSE ▲



▲ COLLAPSE ▲

Task #1 - Points: 1

Text: Capture the following screenshots

Checklist

*The checkboxes are for your own tracking

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

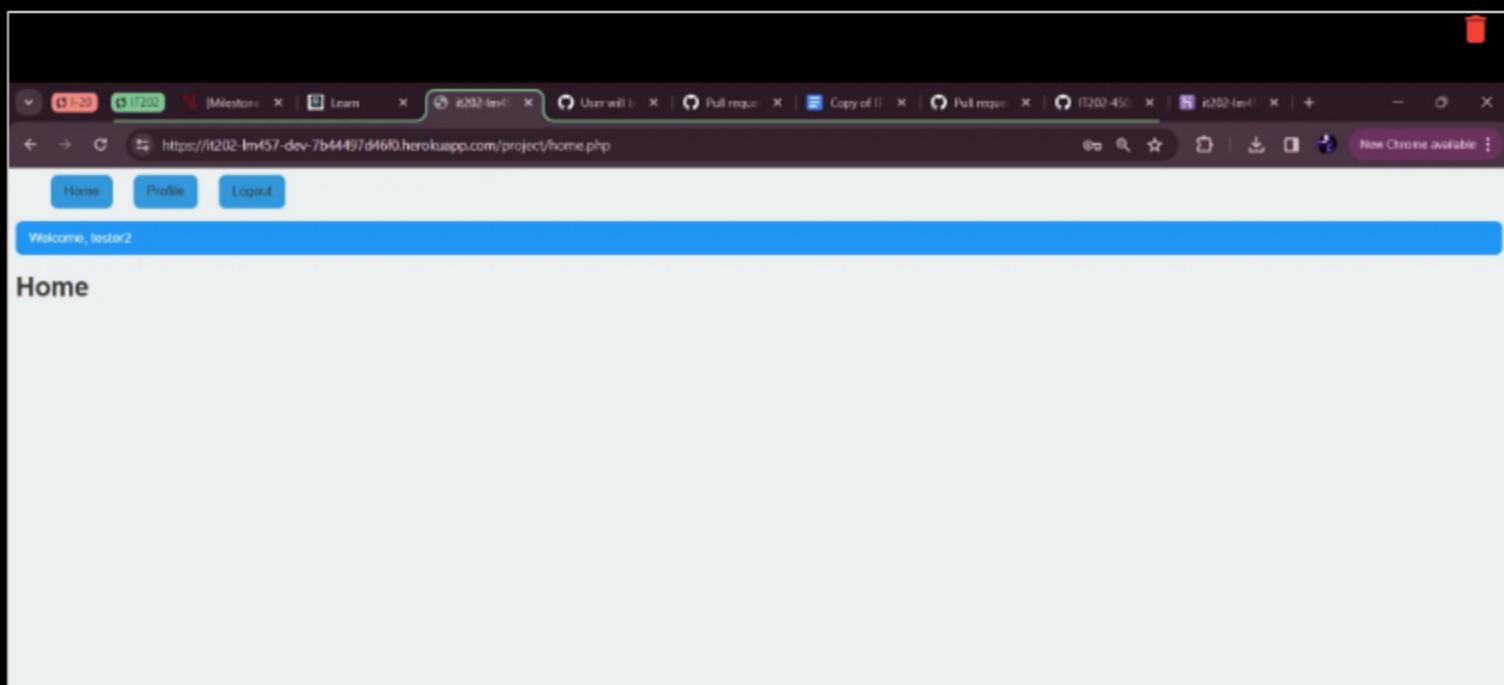
Task Screenshots:

Gallery Style: Large View

Small

Medium

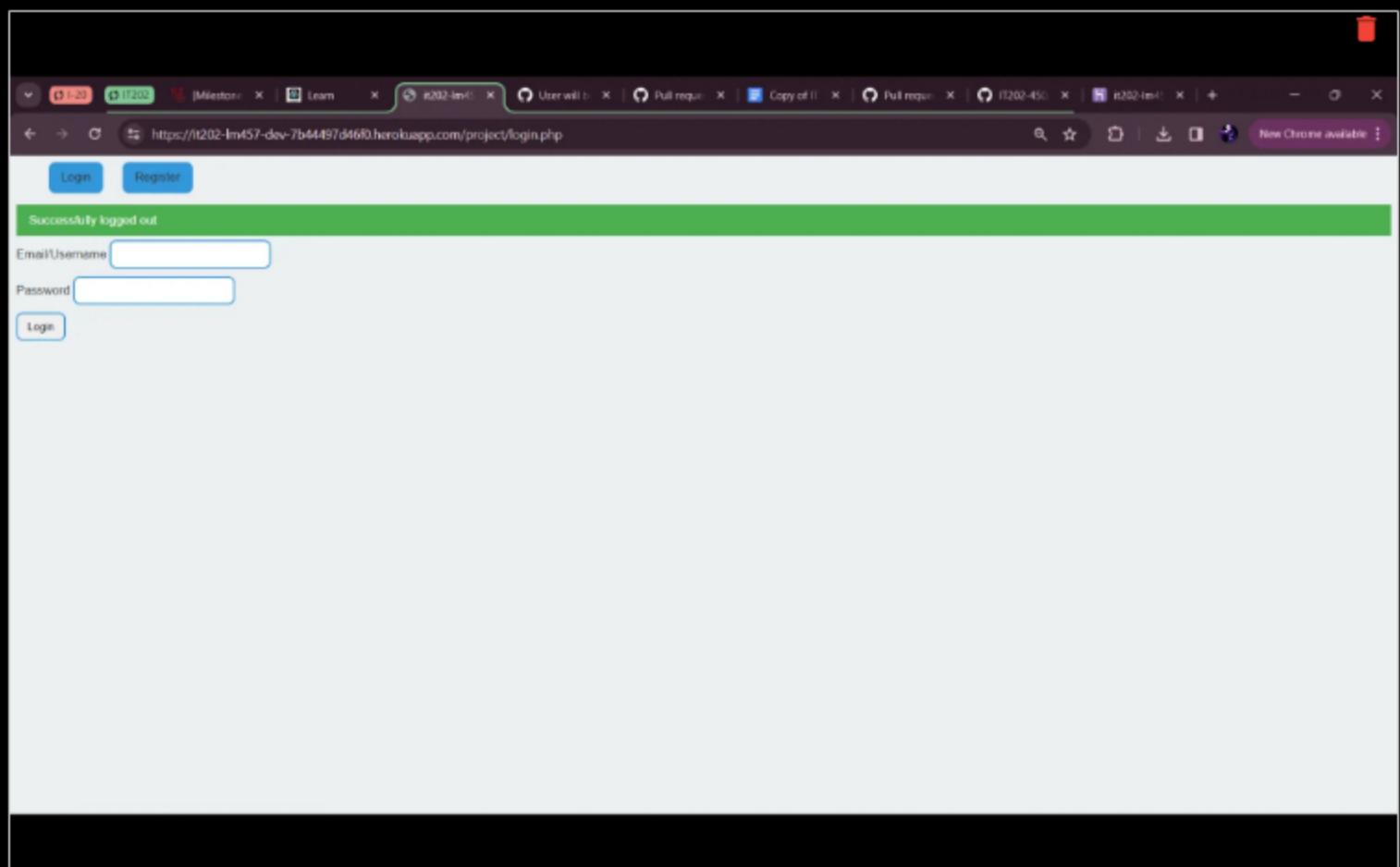
Large



Navigation to home page after logging in.

Checklist Items (1)

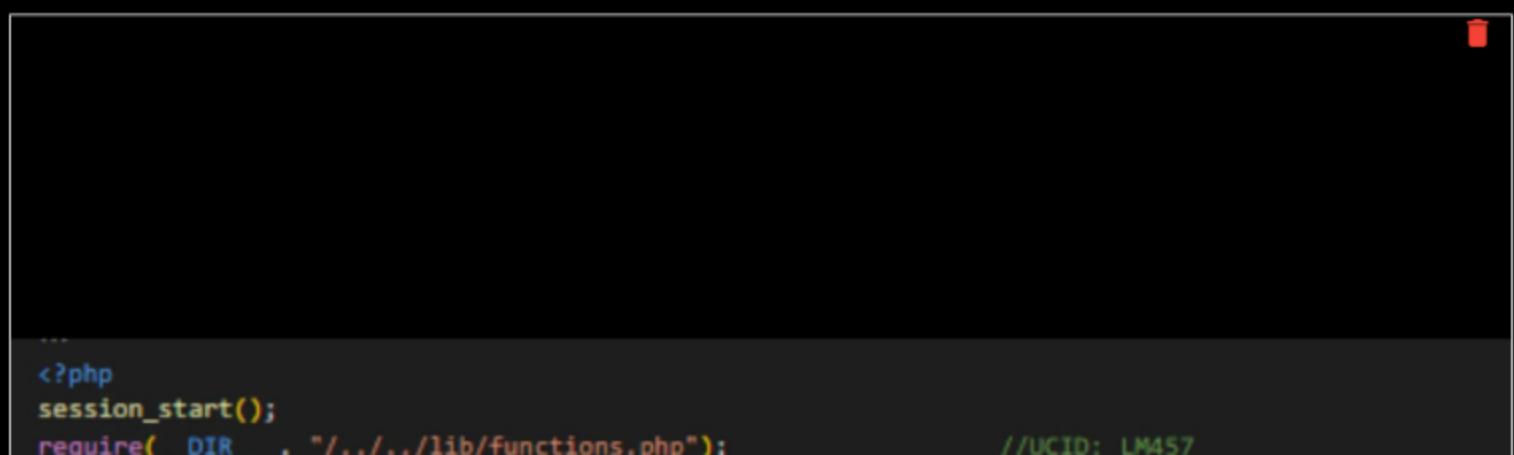
#1 Screenshot of the navigation when logged in (site)



Redirect to login page after logging out with user-friendly logout message.

Checklist Items (1)

#2 Screenshot of the redirect to login with the user-friendly logged-out message (site)



```
reset_session(); //Date: 3/31/2024

flash("Successfully logged out", "success");
header("Location: login.php");
```

Logout.php code showcasing how the session is reset once a user logs out.

Checklist Items (1)

#3 Screenshot of the logout-related code showing the session is destroyed (code). Ensure ucid/date comment is present.

Task #2 - Points: 1

Text: Include pull request links related to this feature

ⓘ Details:

Should end in /pull/#

URL #1

<https://github.com/Kith07/spring2024-IT202/pull/20>

URL #2

<https://github.com/Kith07/spring2024-IT202/pull/22>

Basic Security Rules and Roles (2 pts.)

[^COLLAPSE ^](#)

Task #1 - Points: 1

Text: Authentication Screenshots

[^EXPAND ^](#)

Task #2 - Points: 1

Text: Authorization Screenshots

[^EXPAND ^](#)

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

[^EXPAND ^](#)

▼ EXPAND ▼

Task #4 - Points: 1

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

▼ EXPAND ▼

Task #5 - Points: 1

Text: Include pull request links related to this feature

^ COLLAPSE ^

User Profile (2 pts.)

^ COLLAPSE ^

Task #1 - Points: 1

Text: View Profile Website Page

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input checked="" type="checkbox"/> #1	1	Heroku dev url should be present in the address bar
<input checked="" type="checkbox"/> #2	1	Should have thoughtful CSS applied
<input checked="" type="checkbox"/> #3	1	Show the profile form correctly populated on page load (username, email)
<input checked="" type="checkbox"/> #4	1	Should have the following fields: username, email, current password, new password, confirm password (or similar)

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

The screenshot shows a web browser window with the following details:

- Address Bar:** https://it202-lm457-dev-7b4497d46f0.herokuapp.com/project/profile.php
- Header:** Home, Profile, Create Role, List Roles, Assign Roles, Logout
- Form Fields:**
 - Email: kappaiku27@gmail.com
 - Username: sai
 - Current Password: (empty)
 - New Password: (empty)
 - Confirm Password: (empty)
- Buttons:** Update Profile

Profile page on initial page load with all of the specified fields and corrected populated data.

Checklist Items (4)

#1 Heroku dev url should be present in the address bar

#2 Should have thoughtful CSS applied

#3 Show the profile form correctly populated on page load (username, email)

#4 Should have the following fields: username, email, current password, new password, confirm password (or similar)

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

i Details:

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

Response:

When a user is logged in, the `$_SESSION` magic variable stores user info such as the email, username, and any assigned roles. Therefore, when a user attempts to access the profile page, PHP uses two of the defined functions `get_user_email` and `get_username` which retrieve the relevant user data from the current `$_SESSION` variable in order to prepopulate the respective form fields.

Task #3 - Points: 1

Text: Edit Profile Website Page

Checklist

*The checkboxes are for your own tracking

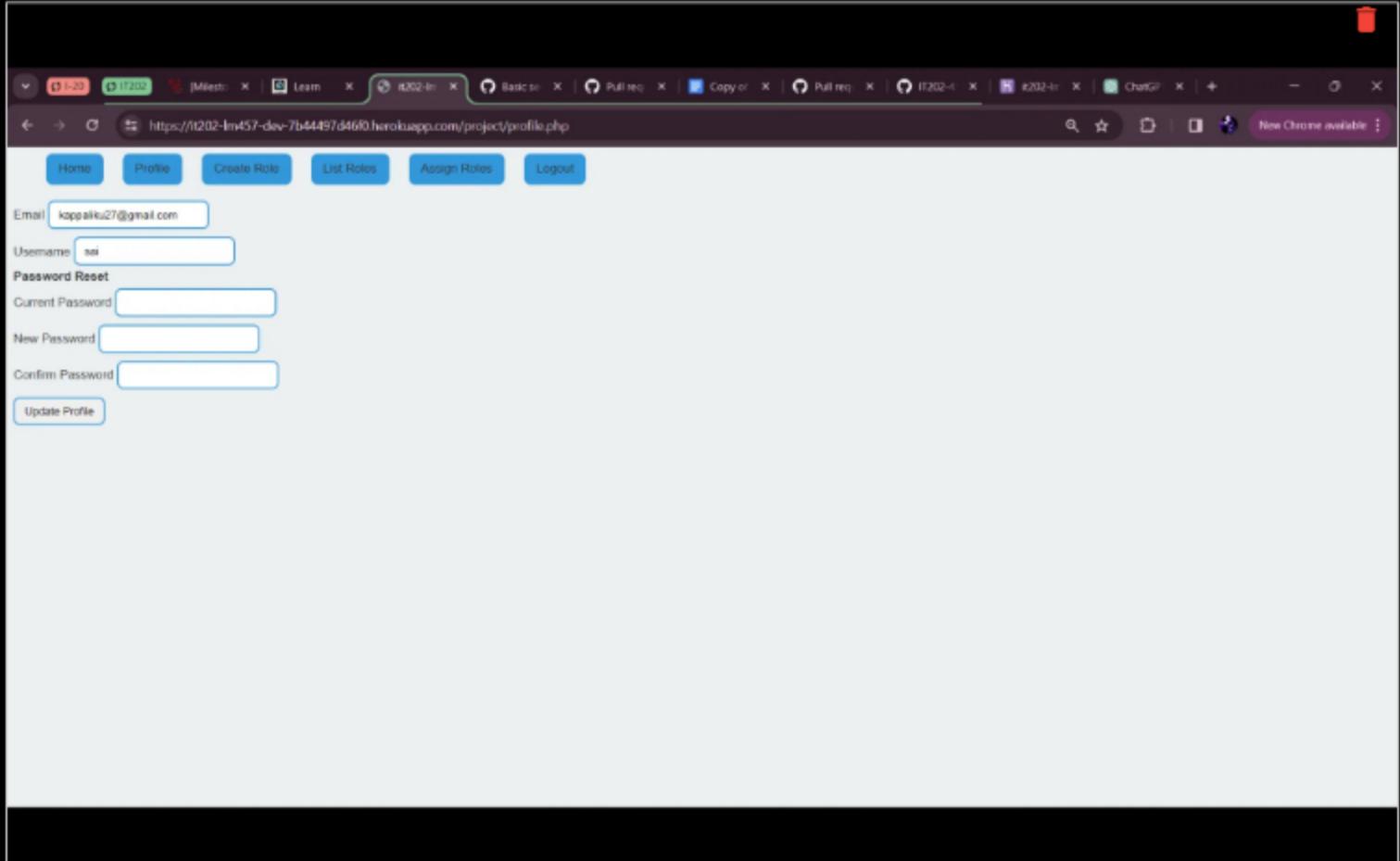
#	Points	Details
<input checked="" type="checkbox"/> #1	1	Heroku dev url should be present in the address bar
<input checked="" type="checkbox"/> #2	1	Should have thoughtful CSS applied
<input checked="" type="checkbox"/> #3	1	Demonstrate with before and after of a username change (including success message)
<input checked="" type="checkbox"/> #4	1	Demonstrate with a before and after of an email change (including success message)
<input checked="" type="checkbox"/> #5	1	Demonstrate the success message of updating password

#6	1	Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)
#7	1	Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)

Task Screenshots:

Gallery Style: Large View

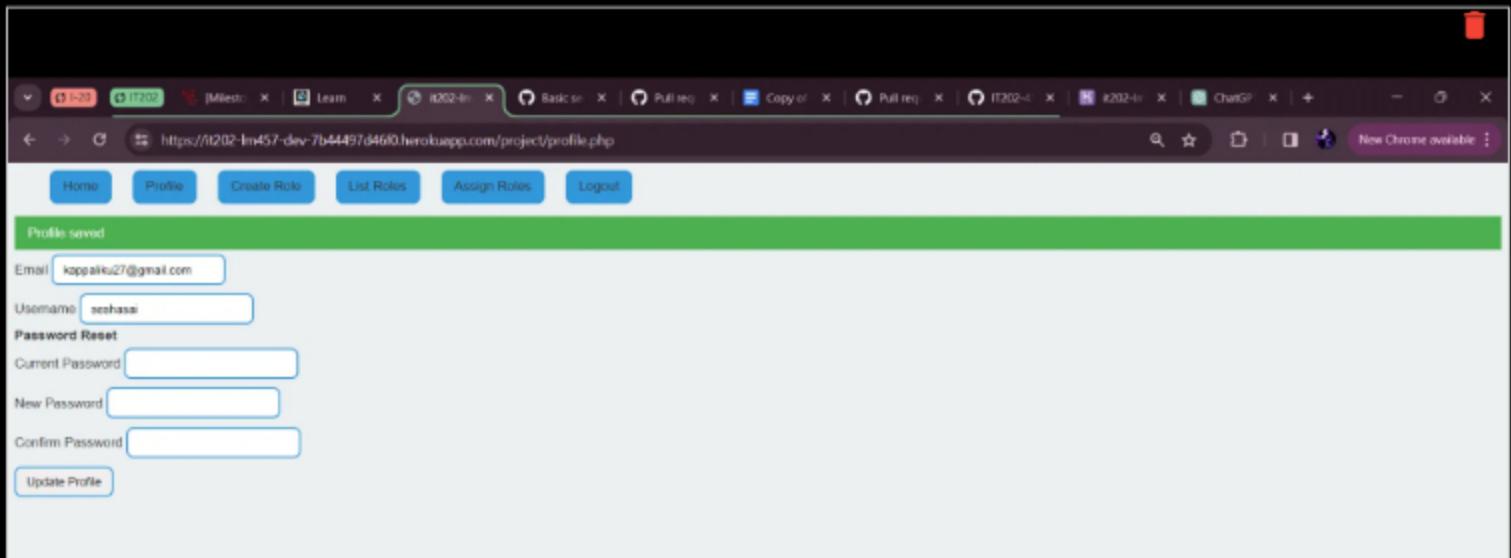
Small Medium Large



Initial username (Before change).

Checklist Items (1)

#3 Demonstrate with before and after of a username change (including success message)



After username change with success message.

Checklist Items (1)

#3 Demonstrate with before and after of a username change (including success message)

The screenshot shows a web application's profile editing interface. The URL in the address bar is <https://it202-lm457-dev-7b44497d46f0.herokuapp.com/project/profile.php>. The page features a navigation bar with links for Home, Profile, Create Role, List Roles, Assign Roles, and Logout. The Profile link is currently selected. Below the navigation, there are several input fields: Email (containing kappaiku27@gmail.com), Username (containing seshasai), and a Password Reset section with three fields: Current Password, New Password, and Confirm Password. At the bottom of the form is a blue 'Update Profile' button.

Before email change.

Checklist Items (1)

#4 Demonstrate with a before and after of an email change (including success message)

The screenshot shows a web application's profile editing interface, similar to the previous one but with a success message. The URL in the address bar is <https://it202-lm457-dev-7b44497d46f0.herokuapp.com/project/profile.php>. The page features a navigation bar with links for Home, Profile, Create Role, List Roles, Assign Roles, and Logout. The Profile link is currently selected. A prominent green success message 'Profile saved' is displayed above the input fields. Below it, there are input fields for Email (containing holad@gmail.com) and Username (containing seshasai). A 'Password Reset' section with three password fields is also present. The overall layout is identical to the previous screenshot but includes the success message.

Current Password

New Password

Confirm Password

After email change with success message.

Checklist Items (1)

#4 Demonstrate with a before and after of an email change (including success message)

Profile saved

Password reset

Email

Username

Current Password

New Password

Confirm Password

Success messages after updating the password.

Checklist Items (1)

#5 Demonstrate the success message of updating password

Invalid email JS validation.

The screenshot shows a web application's profile update page. The URL is https://it202-lm457-dev-7b44497d46f0.herokuapp.com/project/profile.php. The page has a navigation bar with links: Home, Profile, Create Role, List Roles, Assign Roles, and Logout. A yellow error message box at the top left says "Email [Email] Invalid email address". Below it is a form with four input fields: Email (containing "hola@gm"), Username ("sehasai"), Current Password, and New Password. A "Password Reset" section is also present. At the bottom is an "Update Profile" button. In the top right corner, the Chrome developer tools' Console tab is open, displaying JavaScript code related to the validation of the email field. The code includes logic for changing the type of the input element to 'text' if validation fails, and an alert message about HTML validation being disabled. The developer tools also show a "Highlights from the Chrome 122 update" banner.

Invalid email JS validation.

Checklist Items (1)

#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)

The screenshot shows a web application interface for managing user profiles. The main page has a header with links for Home, Profile, Create Role, List Roles, Assign Roles, and Logout. A yellow banner at the top displays the message "JS] Invalid username". Below this, there are input fields for Email (containing "hola@gmail.com"), Username (containing "Seshasai"), and Password Reset sections with fields for Current Password, New Password, and Confirm Password. A "Update Profile" button is located below the password fields. In the bottom right corner of the browser window, the Chrome DevTools Console tab is open, showing a log of JavaScript console entries related to element type changes and validation messages.

```
JS] Invalid username
Email: hola@gmail.com
Username: Seshasai
Password Reset
Current Password: [REDACTED]
New Password: [REDACTED]
Confirm Password: [REDACTED]
Update Profile

[JS]
top
Elements
Console
Sources
Network
Performance
Memory
Default levels
3 issues
3 hidden

console.log("Changed type to text for element ${inp.name} || ["No name"]");
}
}
)
alert("HTML Validation has been disabled until page reload");
})();
Removed required from element email
Removed required from element username
Removed minlength from element currentPassword
Removed minlength from element newPassword
Removed minlength from element confirmPassword
Changed type to text for element confirmPassword
< undefined
)

Console What's new X
Highlights from the Chrome 122 update

Third-party cookie phaseout warnings in Network and Application
The Network and Application panels now show you warnings next to cookies affected by the third-party cookie restrictions from Tracking Protection.
new
```

Checklist Items (1)

#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)

The screenshot shows a web browser window with a profile update form. The URL is <https://it202-lm457-dev-7b44497d46f0.herokuapp.com/project/profile.php>. The form includes fields for Email, Username, Current Password, New Password, and Confirm Password. An error message "JS] New password is too short." is displayed above the New Password field. The developer tools' Console tab shows several log entries related to element types and validation rules being removed or changed.

```
inp.type = "text";
console.log("Changed type to text for element ${inp.name} || ${[No name]}");
}
}
}
alert("HTML Validation has been disabled until page reload");
})();
Removed required from element email
Changed type to text for element email
Removed required from element username
Removed minlength from element currentPassword
Changed type to text for element currentPassword
Removed minlength from element newPassword
Changed type to text for element newPassword
Removed minlength from element confirmPassword
Changed type to text for element confirmPassword
e undefined
)
}

Console What's new X
Highlights from the Chrome 122 update

Third-party cookie phaseout warnings in Network and Application
The Network and Application panels now show you warnings next to cookies affected by the third-party cookie restrictions from Tracking Protection.
new
```

Invalid (short) password JS validation.

Checklist Items (1)

#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)

The screenshot shows a web browser window with a profile update form. The URL is <https://it202-lm457-dev-7b44497d46f0.herokuapp.com/project/profile.php>. The form includes fields for Email, Username, Current Password, New Password, and Confirm Password. An error message "JS] Password and Confirm password must match." is displayed above the New Password field. The developer tools' Console tab shows several log entries related to element types and validation rules being removed or changed.

```
inp.type = "text";
console.log("Changed type to text for element ${inp.name} || ${[No name]}");
}
}
}
alert("HTML Validation has been disabled until page reload");
})();
Removed required from element email
Changed type to text for element email
Removed required from element username
Removed minlength from element currentPassword
Changed type to text for element currentPassword
Removed minlength from element newPassword
Changed type to text for element newPassword
Removed minlength from element confirmPassword
Changed type to text for element confirmPassword
e undefined
)
}

Console What's new X
Highlights from the Chrome 122 update

Third-party cookie phaseout warnings in Network and Application
new
```



Password mismatch JS validation

Checklist Items (1)

#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)

The chosen email is not available.

Email: hola@gmail.com

Username: seshasai

Current Password:

New Password:

Confirm Password:

Update Profile

Error message after attempting to change email to an existing email.

Checklist Items (1)

#7 Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)

The chosen username is not available.

Email: hola@gmail.com

Username: seshasai

Current Password:

New Password:

Confirm Password:

Update Profile

Error after attempting to use an existing username.

Checklist Items (1)

#7 Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)

The screenshot shows a web browser window with multiple tabs open. The active tab is for a profile update page at the URL <https://it202-lm457-dev-7b44497d46f0.herokuapp.com/project/profile.php>. The page has a navigation bar with links for Home, Profile, Create Role, List Roles, Assign Roles, and Logout. A green success message banner at the top says "Profile saved". Below it, a yellow error message banner says "Current password is invalid". The main form contains fields for Email (with value "hole@gmail.com"), Username (with value "sehsusai"), and Password Reset (with three input fields for Current Password, New Password, and Confirm Password). At the bottom of the form is a blue "Update Profile" button.

Error message while using the incorrect current password associated with the account while attempting to reset the password.

Checklist Items (1)

#7 Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)

Task #4 - Points: 1

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

#	Points	Details
<input checked="" type="checkbox"/> #1	1	Updating Username/Email
<input checked="" type="checkbox"/> #2	1	Updating password
<input checked="" type="checkbox"/> #3	1	Don't just show code, translate things to plain English

Response:

After validating the user data concerning email and username and once the form is submitted, PHP obtains the form data by retrieving it from the `$_POST` magic variable. After the server-side validation is done to ensure proper formatting of the email and username, using SQL, the newly provided information overrides the old data in the Users table, where the accounts are matched based on the ID value which remains constant. Then, SQL is once again used to select the fresh data from the updated table to populate the email and username form fields once the profile page reloads after a successful update.

As for password changes, PHP checks to see if such a request has been placed at the time of the form's submission. If it is, it retrieves the inputted user data using the `$_POST` magic variable and performs a server-side validation to see if the new password is appropriately formatted (length) and both the new password and confirm password entries match up. Once verified, the new password is hashed using the same salt value as the current password in order to see if they're the same. If so, using SQL, the newly provided hashed password overrides the existing hashed password in the Users table, where once again the user ID is used to match the accounts in the database. Thus, the account's password has been changed and the user is prompted with a success message regarding password reset.

Task #5 - Points: 1

Text: Include pull request links related to this feature

① Details:

Should end in /pull/#

URL #1

<https://github.com/Kith07/spring2024-IT202/pull/23>

URL #2

<https://github.com/Kith07/spring2024-IT202/pull/24>

URL #3

<https://github.com/Kith07/spring2024-IT202/pull/40>

Misc (1 pt.)

[^COLLAPSE ^](#)

Task #1 - Points: 1

Text: Screenshot of wakatime

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



Weekly code stats for 2024-03-11 until 2024-03-17

1 hr 48 mins

total

1 hr 48 mins

daily average

Categories:

Coding 1 hr 48 mins

Projects:

spring2024-IT202 1 hr 48 mins

Languages:

PHP 1 hr 37 mins

SQL 7 mins

JavaScript 1 min

CSS 1 min

Editors:

VS Code 1 hr 48 mins

Operating Systems:

Windows 1 hr 48 mins

Machines:

Mintu 1 hr 48 mins

Initial file setup.



Weekly code stats for 2024-05-25 until 2024-05-31

8 hrs 20 mins

total

2 hrs 5 mins

daily average

Categories:

Coding 8 hrs 20 mins

Projects:

spring2024-IT202 8 hrs 20 mins

Languages:

PHP 6 hrs 17 mins

CSS 1 hr 14 mins

JavaScript 33 mins

SQL 15 mins

Editors:

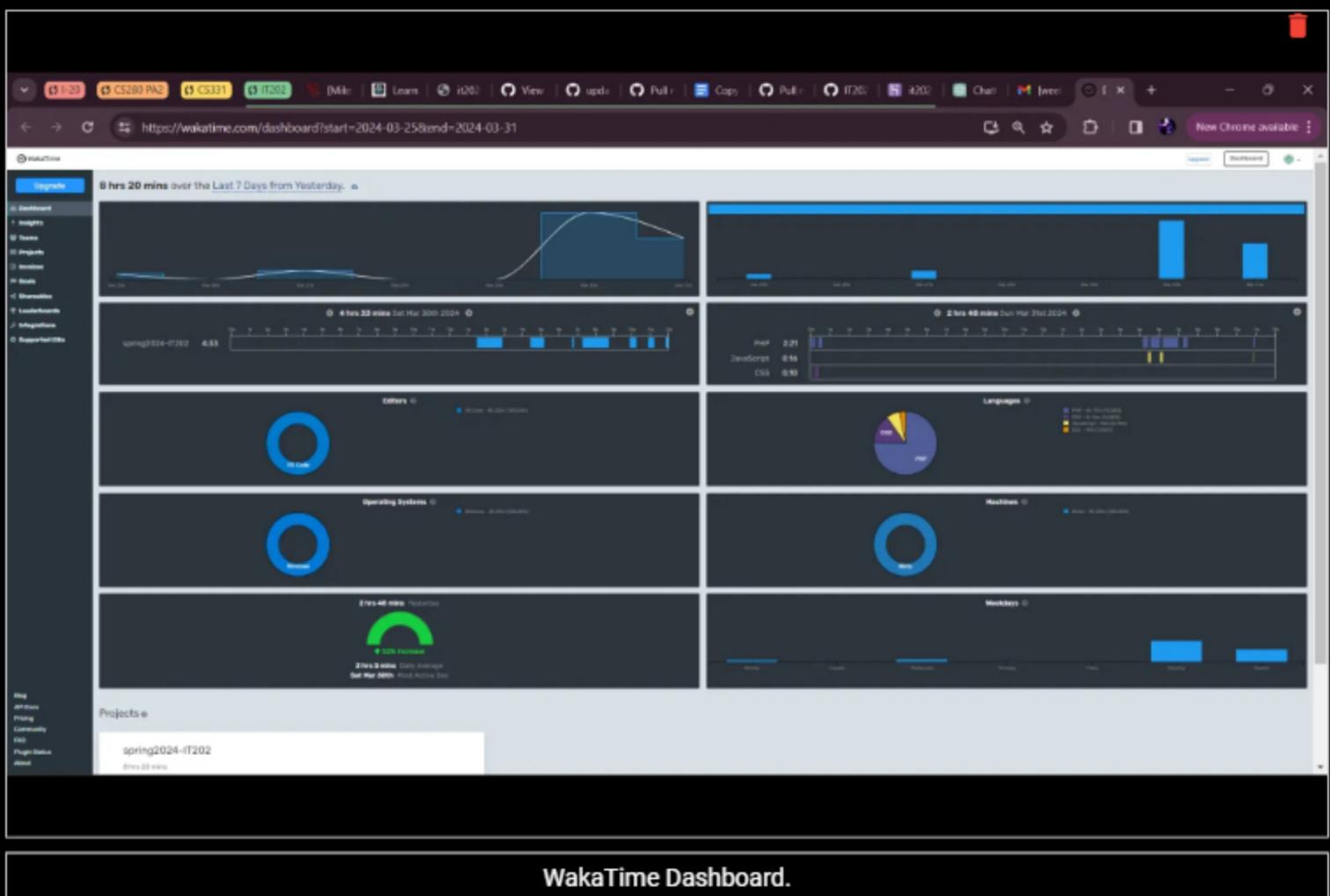
VS Code 8 hrs 20 mins

Operating Systems:

Windows 8 hrs 20 mins

Machines:

Mintu 8 hrs 20 mins



WakaTime Dashboard.

Task #2 - Points: 1

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

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

The screenshot shows a GitHub project board for "IT 202 - Travel App Project B". The board has three columns: Todo, In Progress, and Done.

- Todo:** This item hasn't been started.
- In Progress:** This is actively being worked on.
- Done:** This has been completed.
 - spring2024-IT202 #31: User will be able to register a new account.
 - spring2024-IT202 #32: User will be able to login to their account.

(given they enter the correct credentials)

spring2024-IT202 #33

User will be able to logout

spring2024-IT202 #34

Basic security rules implemented

spring2024-IT202 #35

Basic Roles implemented

+ Add item

+ Add item

+ Add item

Github repo Project board screenshot for Milestone1.

Task #3 - Points: 1

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

URL #1

<https://github.com/users/Kith07/projects/1/views/1>

Task #4 - Points: 1

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

URL #1

<https://it202-lm457-prod-cf97d0c34365.herokuapp.com/project/login.php>

End of Assignment