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

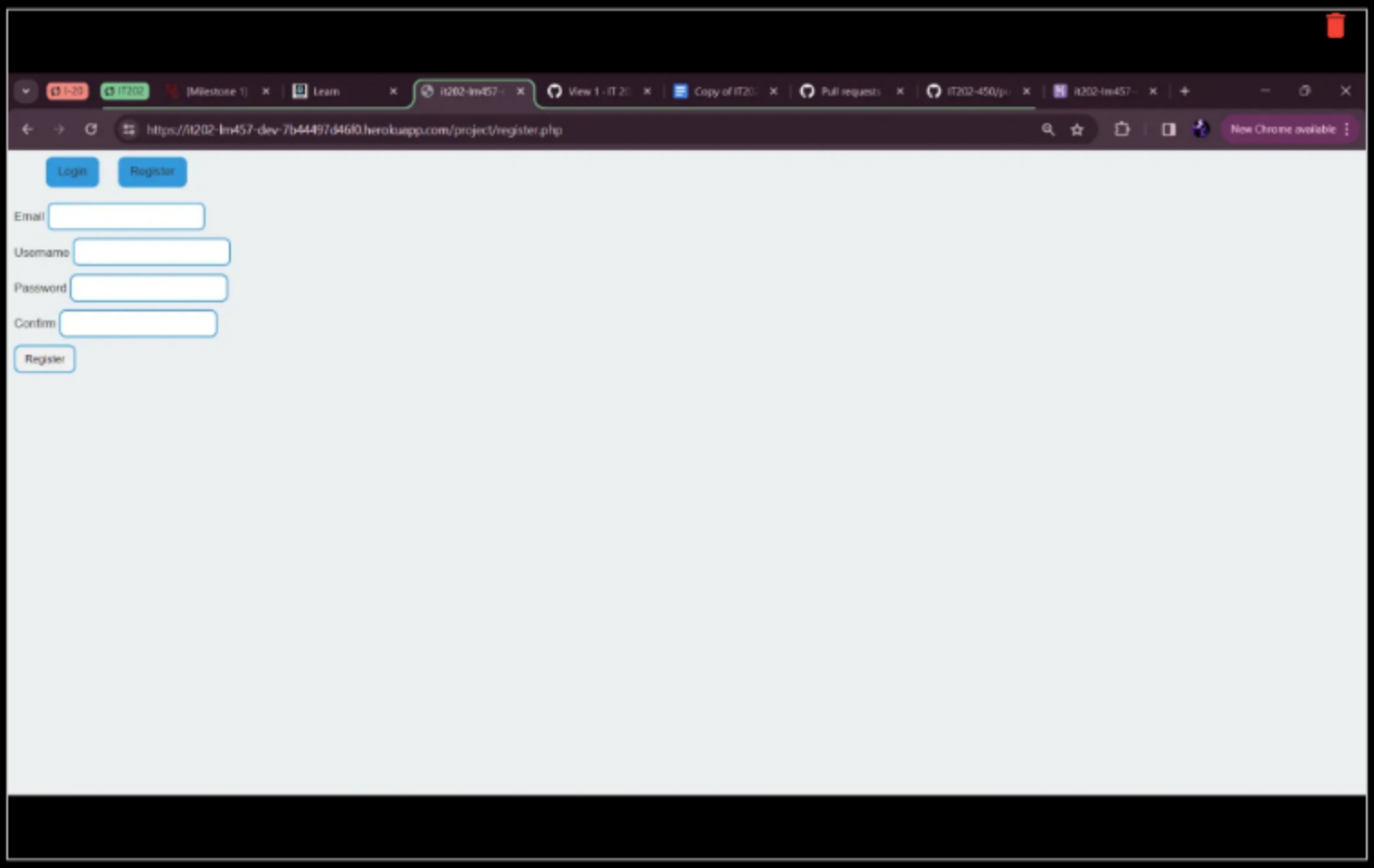
*The checkboxes are for your own tracking

#	Points	Details
#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

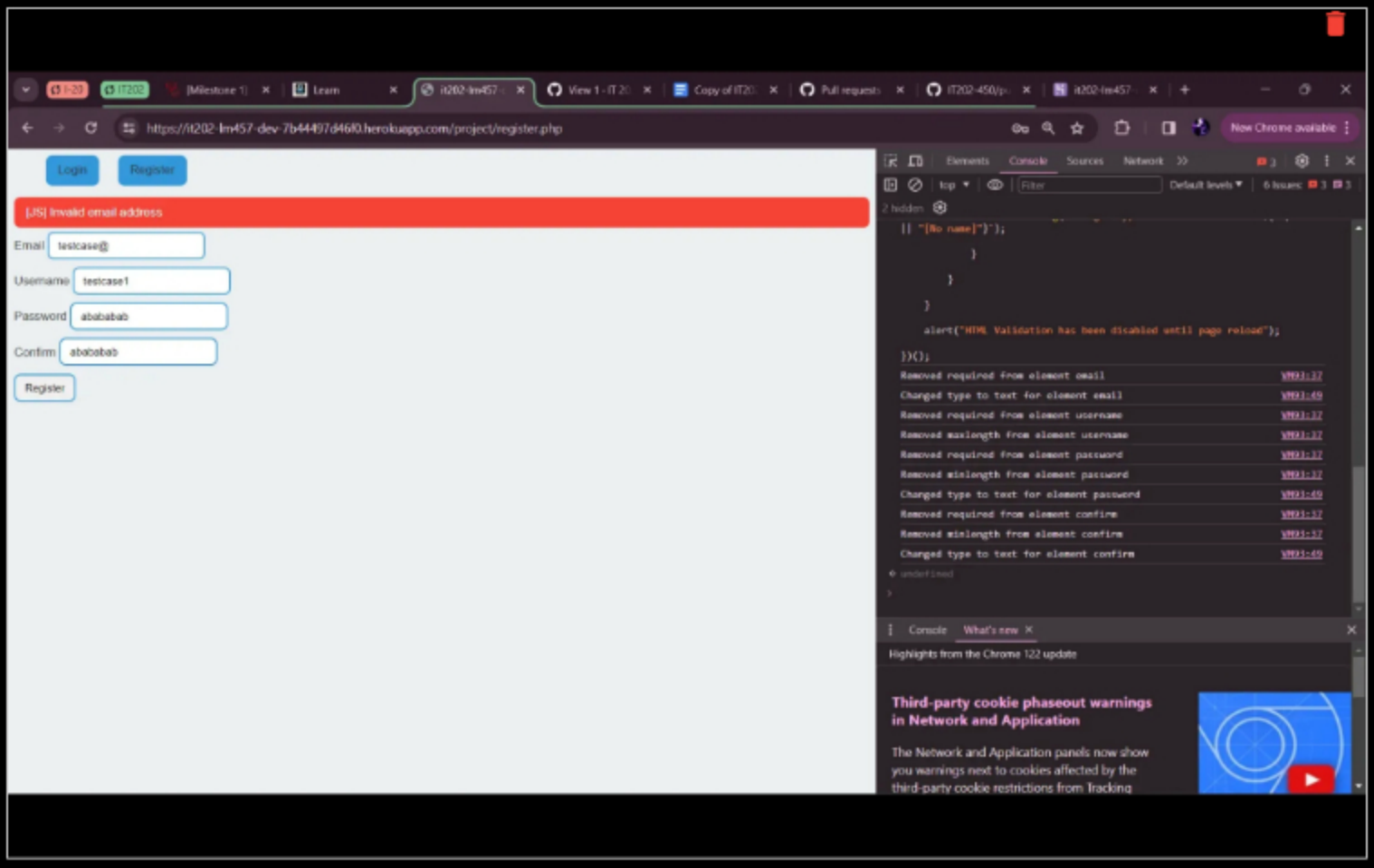


The initial register page deployed on Heroku dev containing the fields required to make an account (email, username, pw).

Checklist Items (2)

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

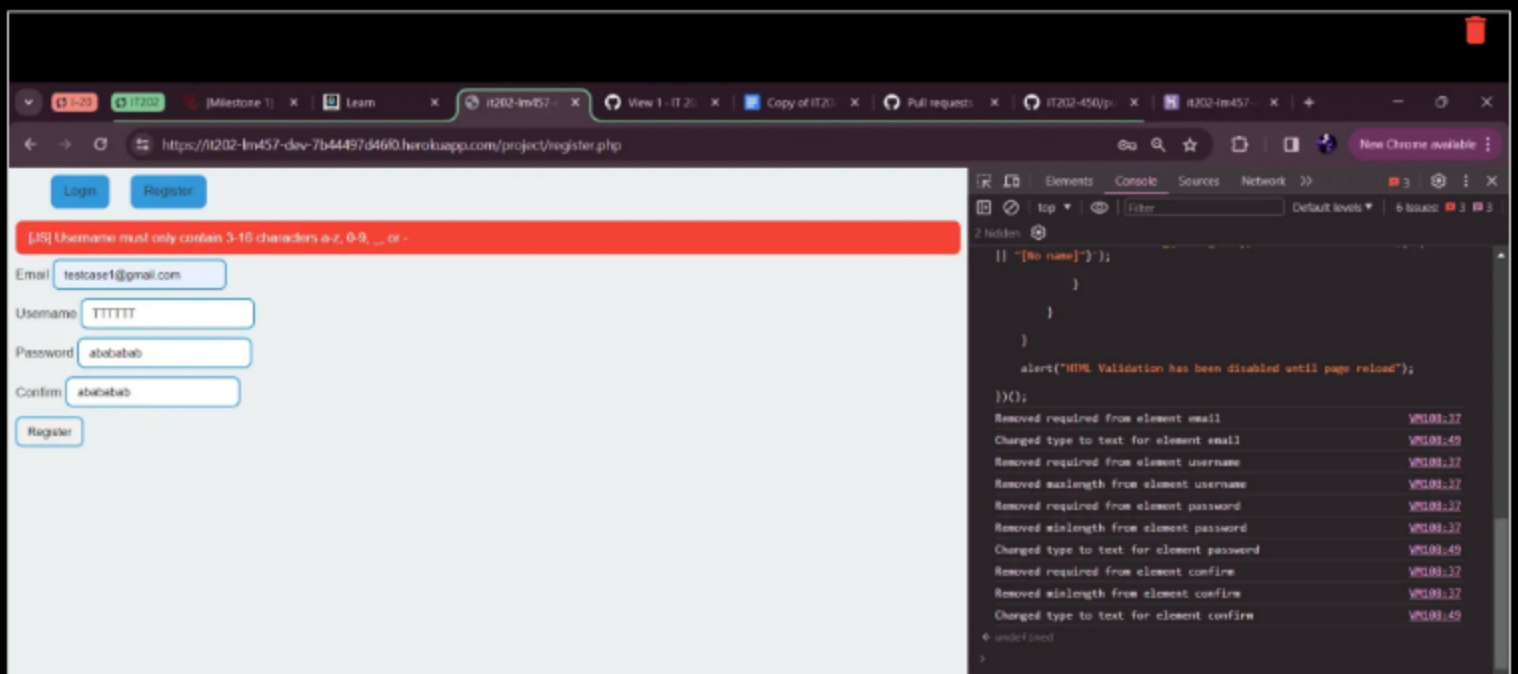
#2 Should have thoughtful CSS applied

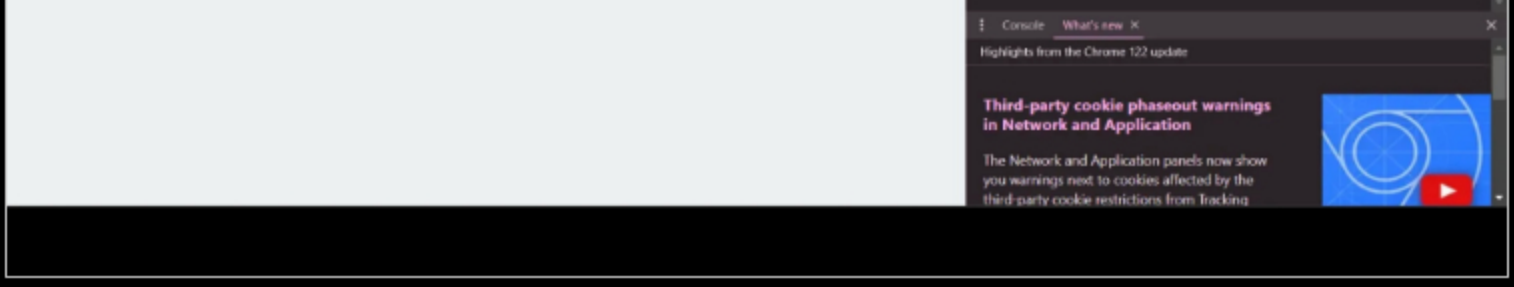


JS Email Validation.

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

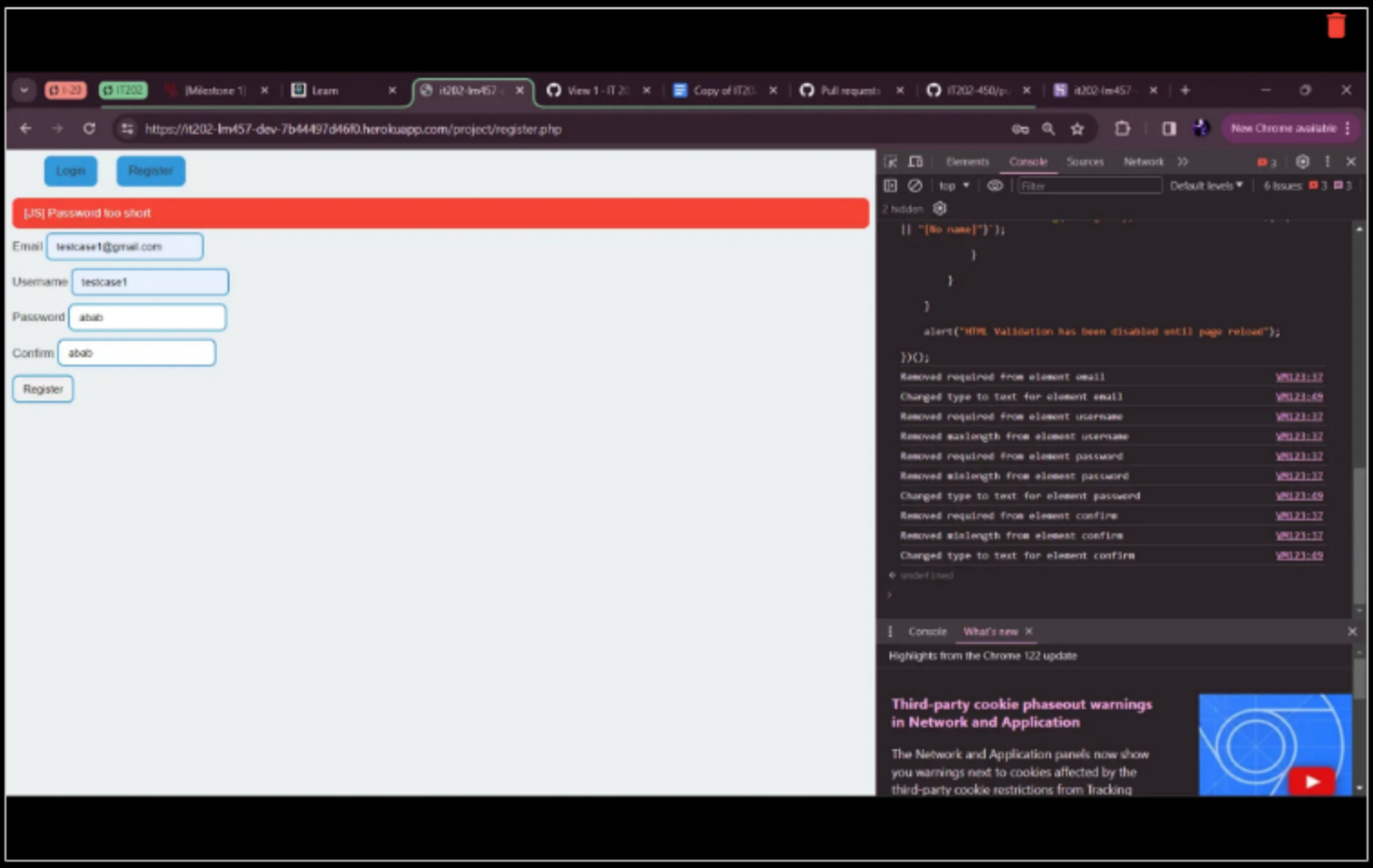




JS Username Validation.

Checklist Items (1)

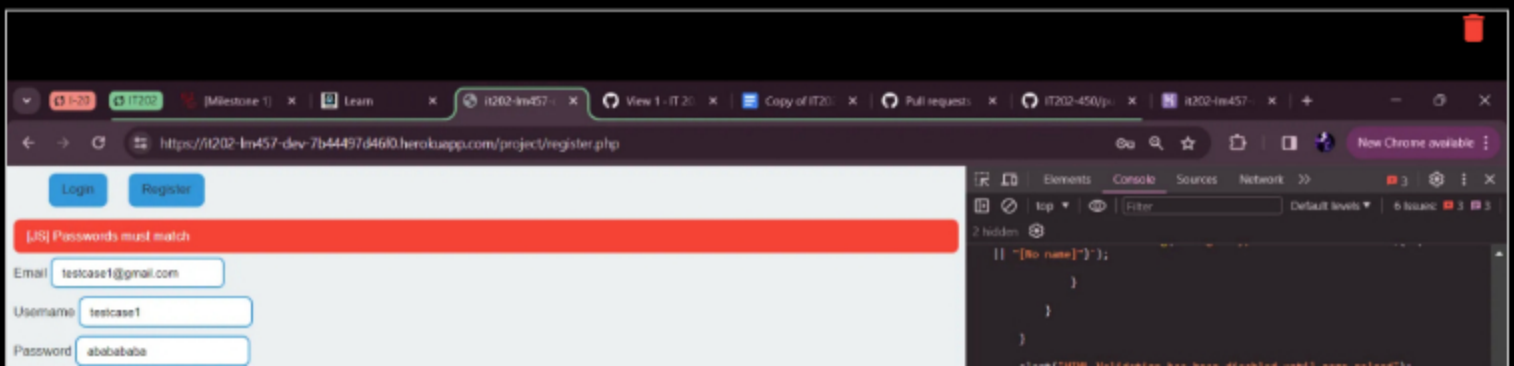
#3 Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required

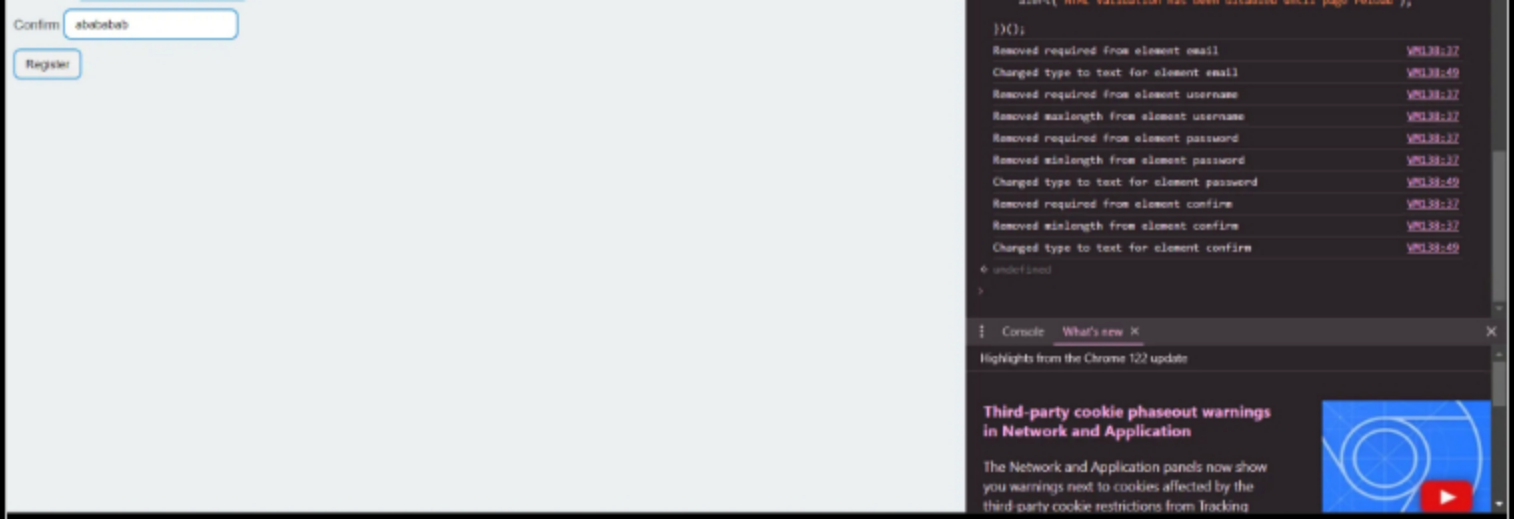


JS Password Validation.

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

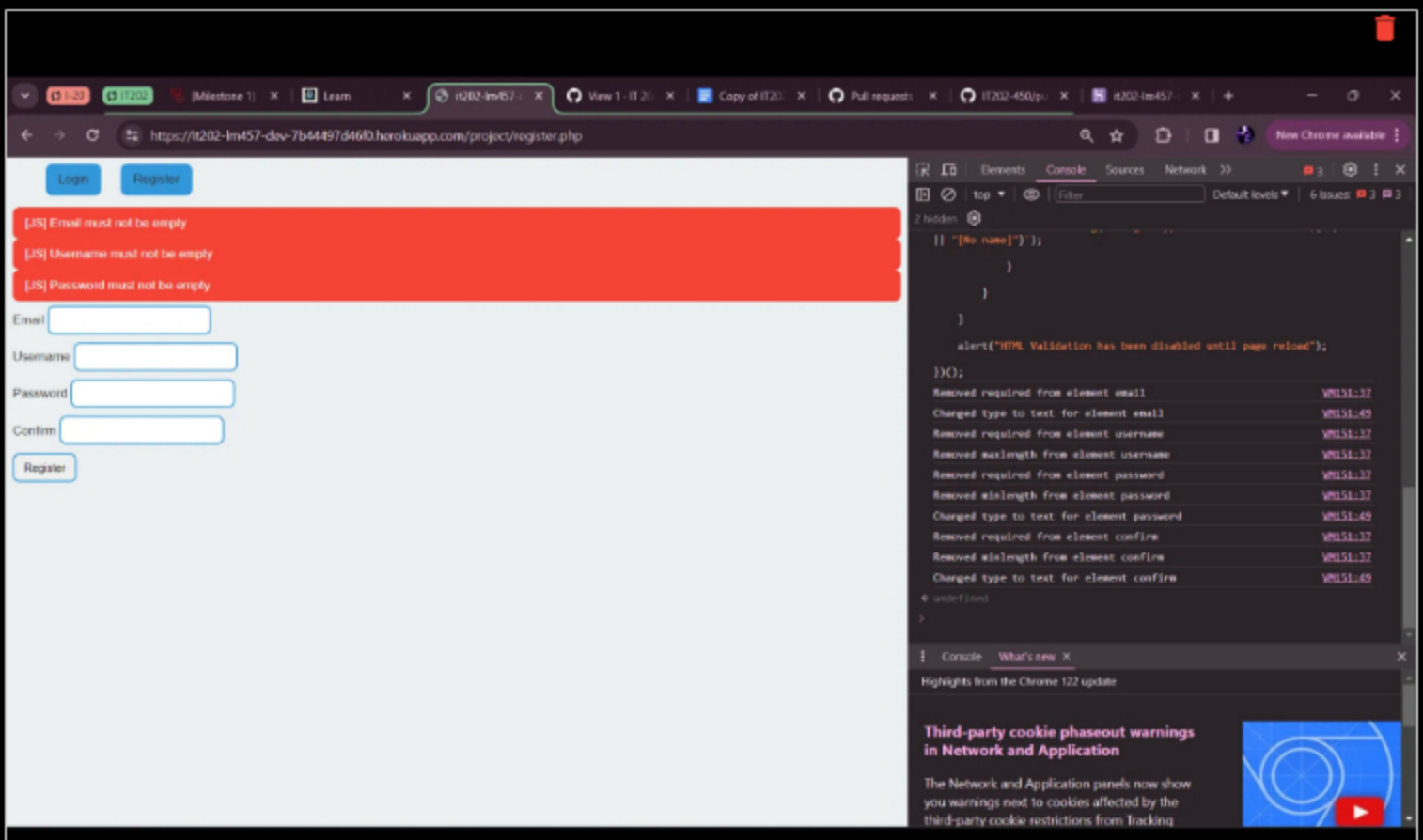




JS PW and Confirm PW Matching Validation.

Checklist Items (1)

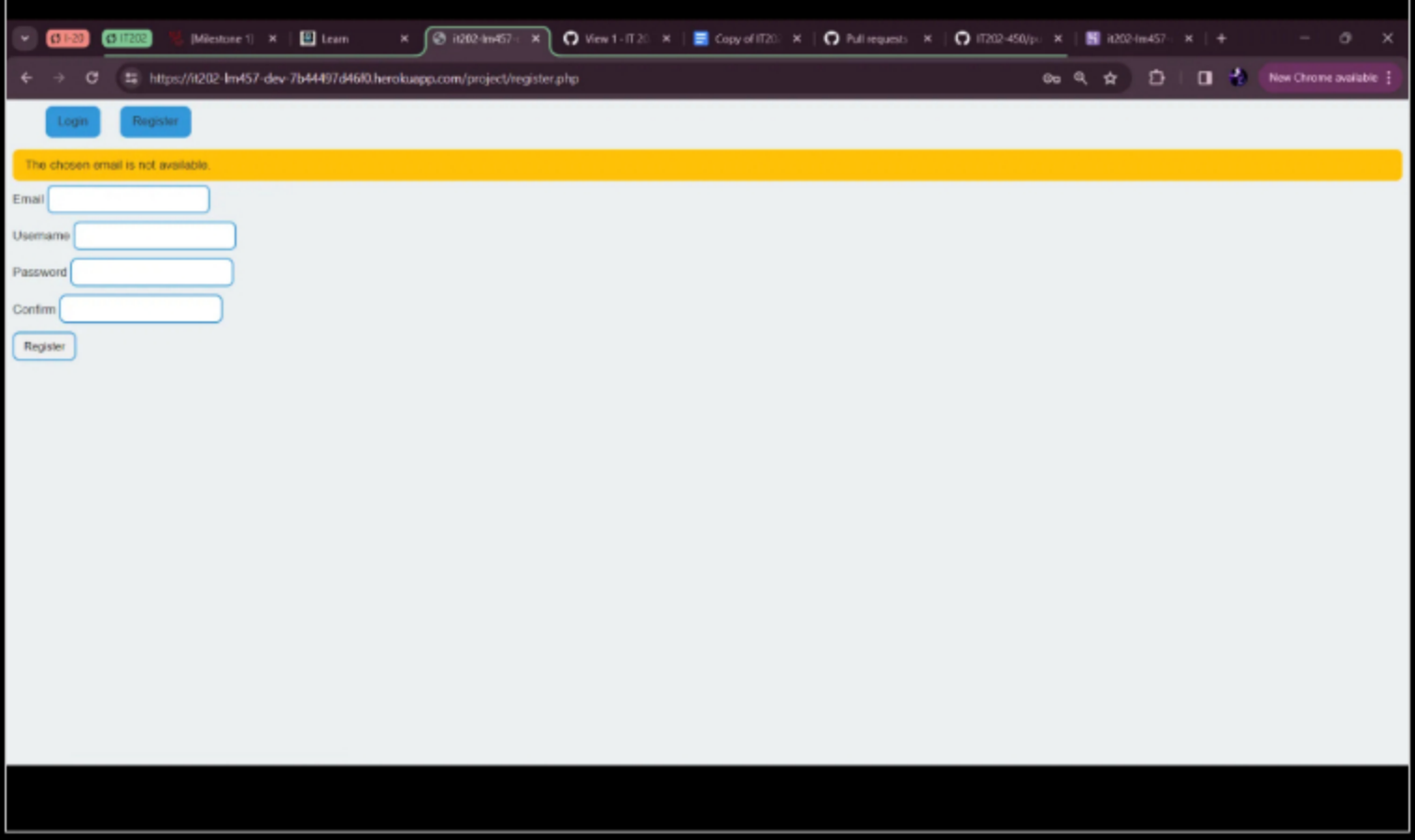
#3 Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required



JS All fields need to be filled validation.

Checklist Items (1)

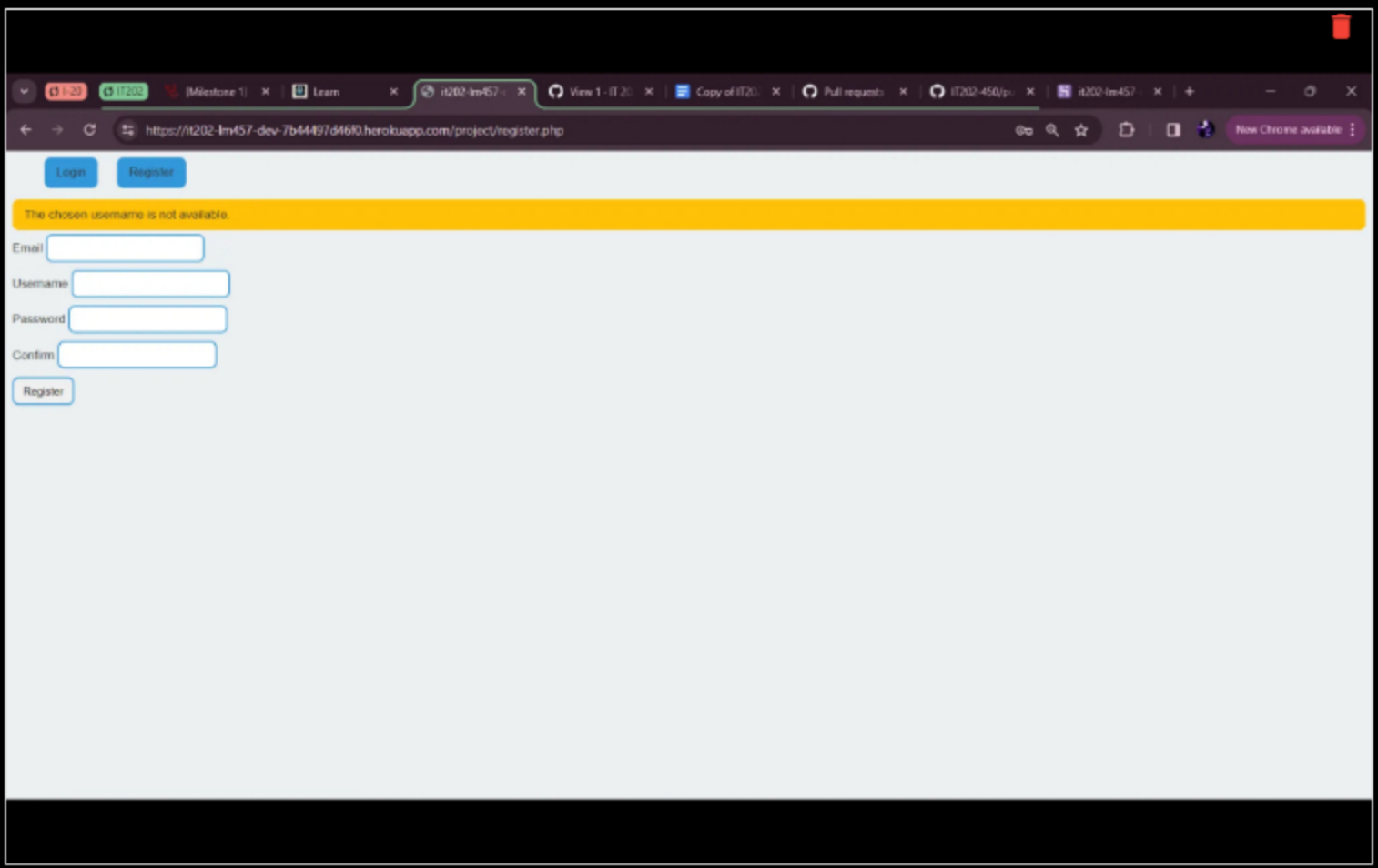
#3 Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required



Email already in use error.

Checklist Items (1)

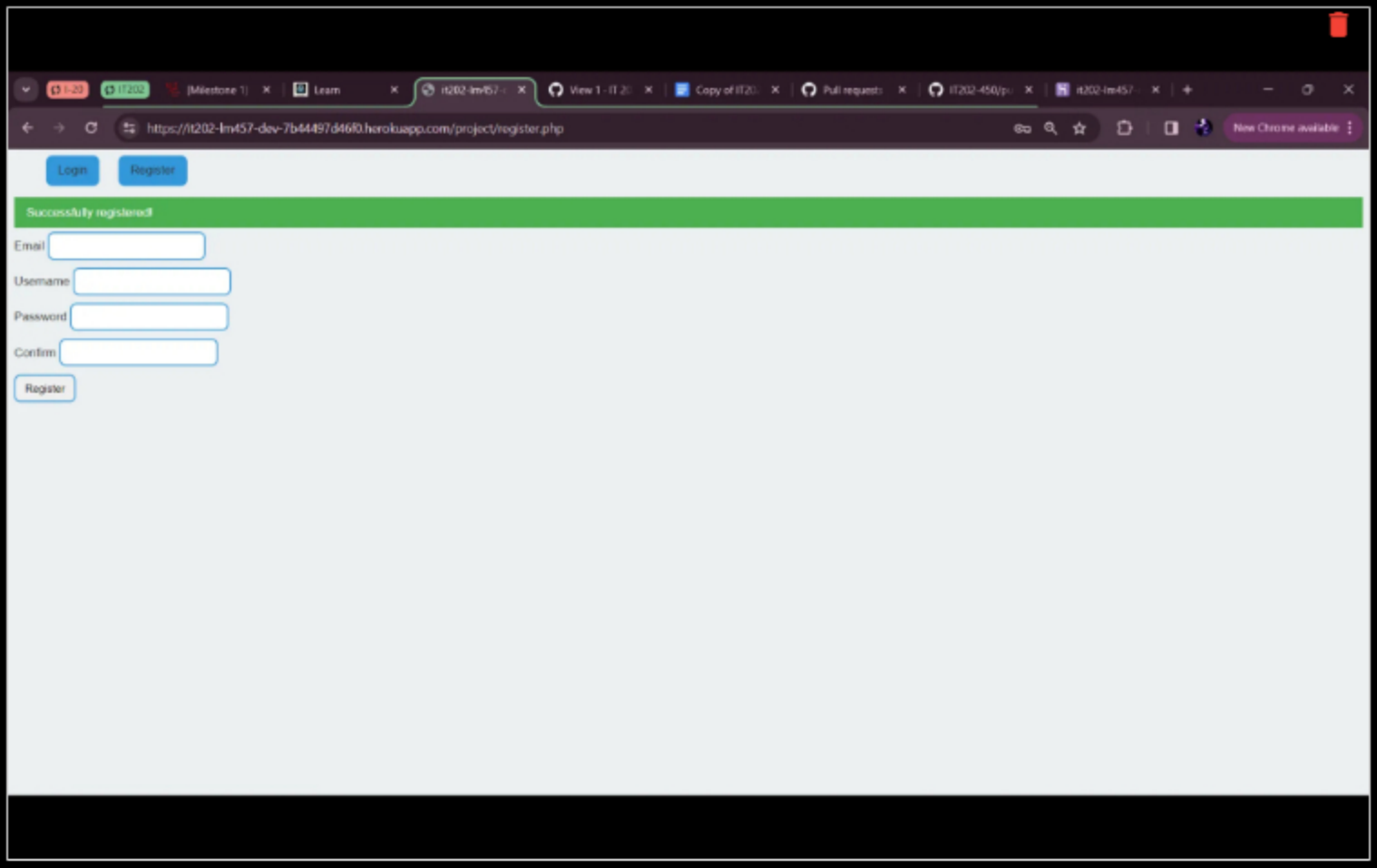
#4 Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)



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

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the form code

Details:
Should have appropriate input types for the fiel

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
<form onsubmit="return validate(this)" method="POST">
```

```
    <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.

COLLAPSE

Task #3 - Points: 1

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

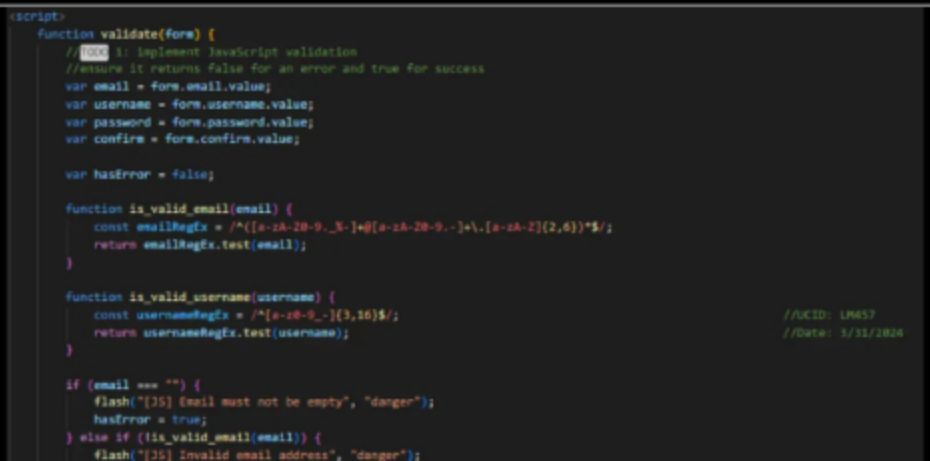
Checklist — *The checkboxes are for your own tracking

#	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



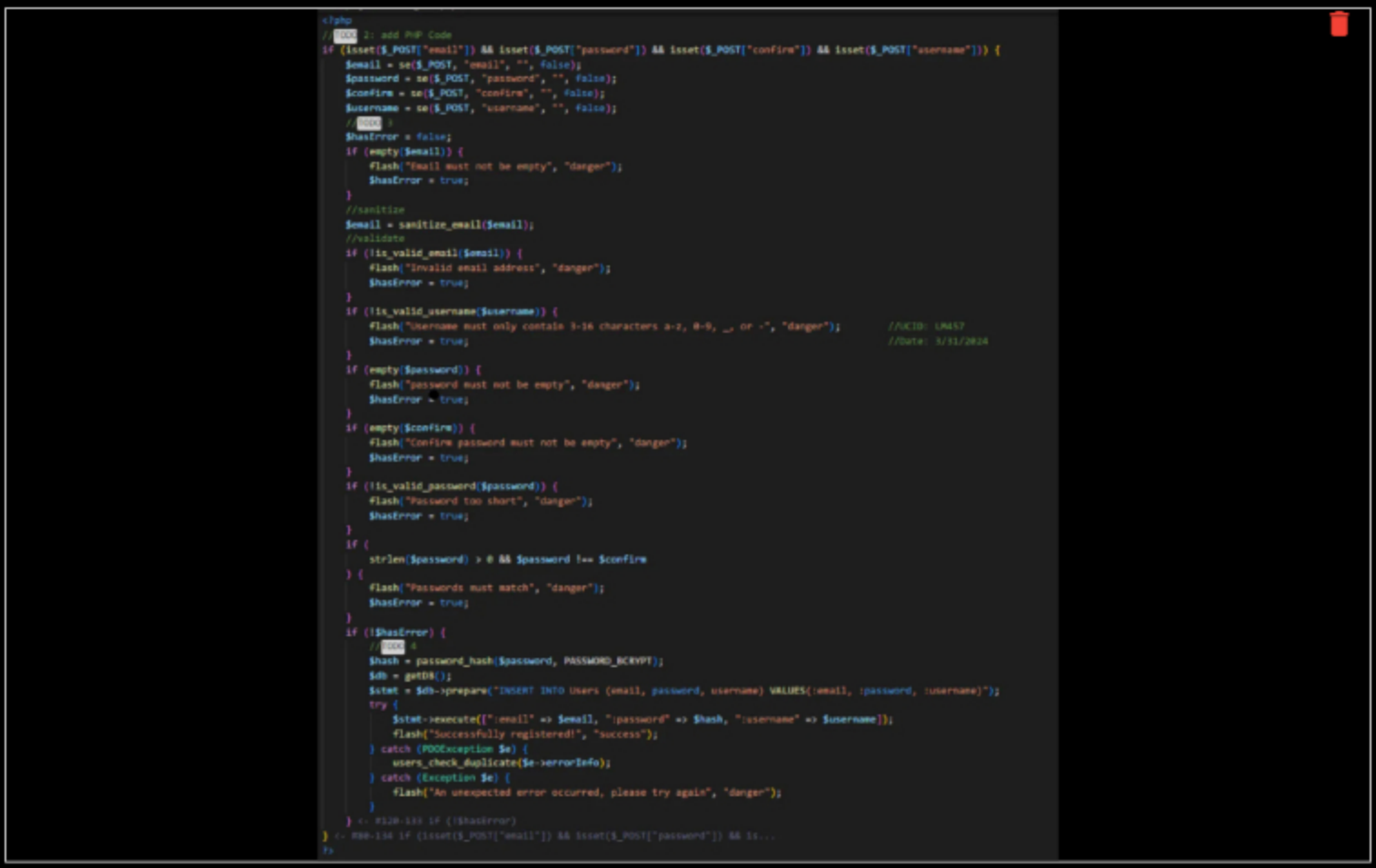
```
        hasError = true;
    }

    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, 0-9, _, 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)
</script>
```

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
// TODO 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);
    // TODO 3
    $hasError = false;
    if (empty($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, 0-9, _, or -", "danger");        //UCID: LN457
        $hasError = true;                                                                       //Date: 3/11/2024
    }
    if (empty($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) {
        // TODO 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 $e) {
            flash("An unexpected error occurred, please try again", "danger");
        }
    } <- #120-133 if (!$hasError)
} <- #80-134 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)

COLLAPSE

Task #4 - Points: 1

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

Checklist

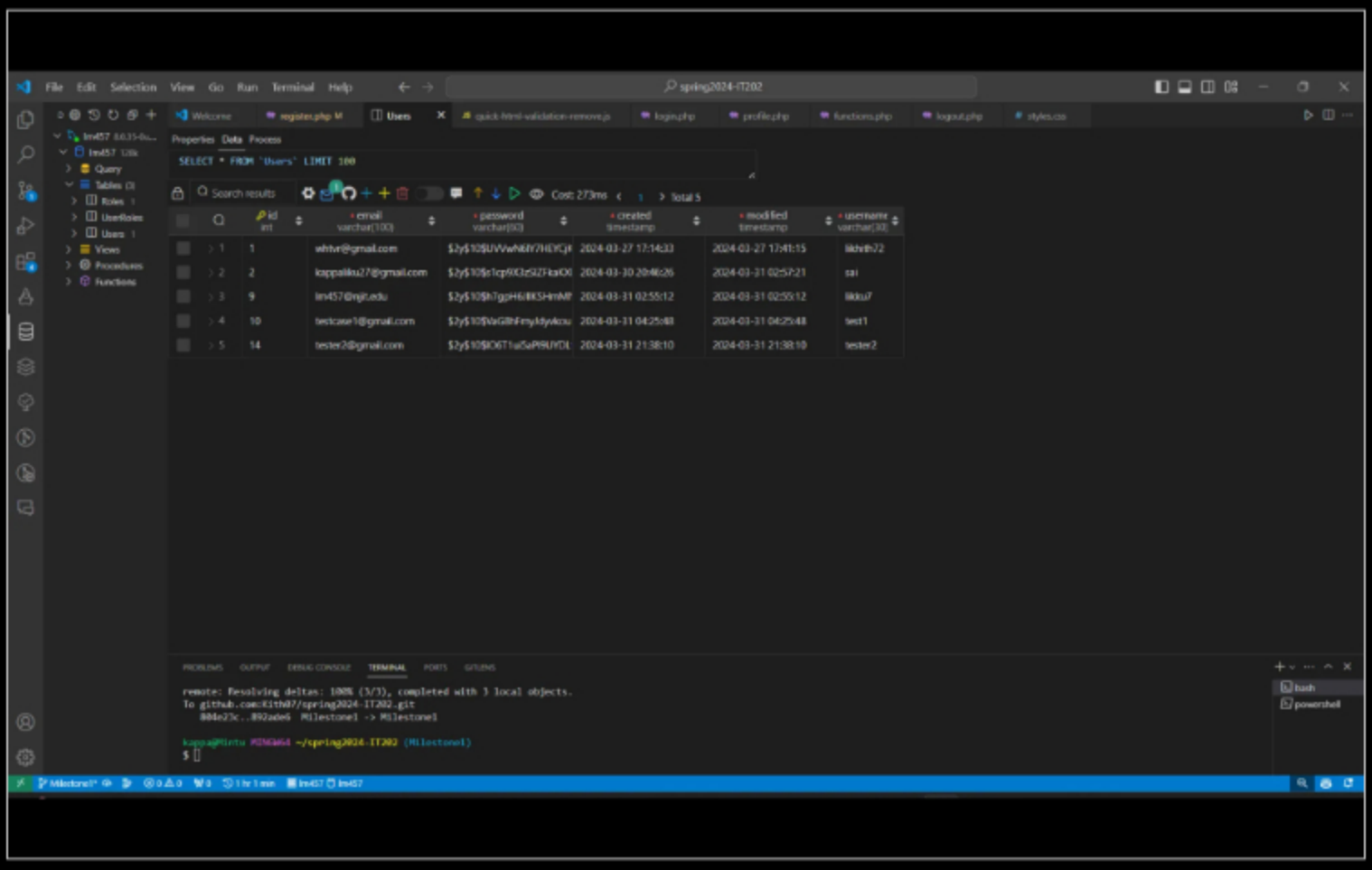
*The checkboxes are for your own tracking

#	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



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)

COLLAPSE

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:

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.

COLLAPSE

Task #6 - 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/18

URL #2
https://github.com/Kith07/spring2024-IT202/pull/19

User Login (2 pts.)

COLLAPSE

EXPAND

Task #1 - Points: 1

Text: Screenshot of form on website page

Task #2 - Points: 1

EXPAND

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

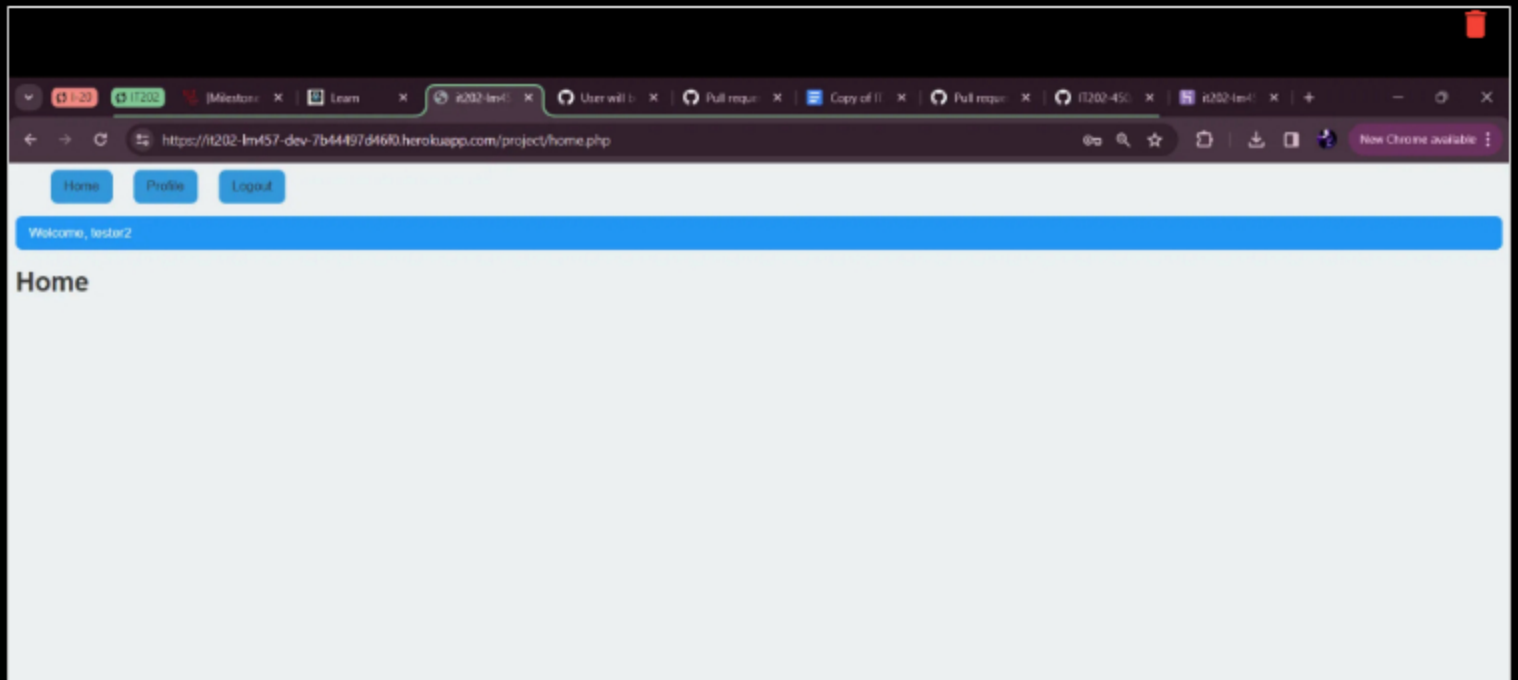
*The checkboxes are for your own tracking

#	Points	Details
#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

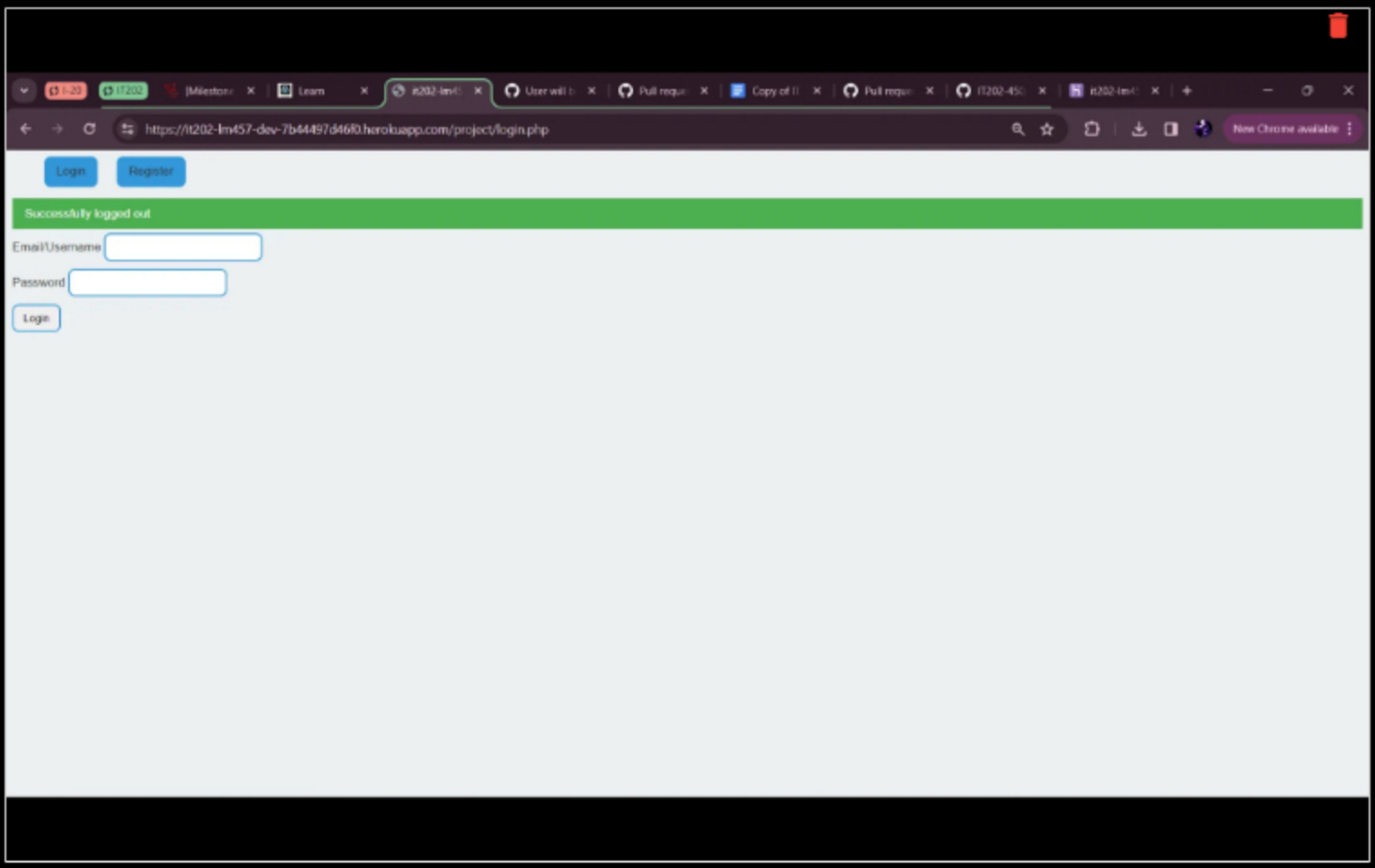
Small Medium Large



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)

```
<?php
session_start();
require(__DIR__ . "/../../lib/functions.php");                    //UCID: LM457
```

```php
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.

COLLAPSE

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

EXPAND

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

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

User Profile (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: View Profile Website Page

Checklist

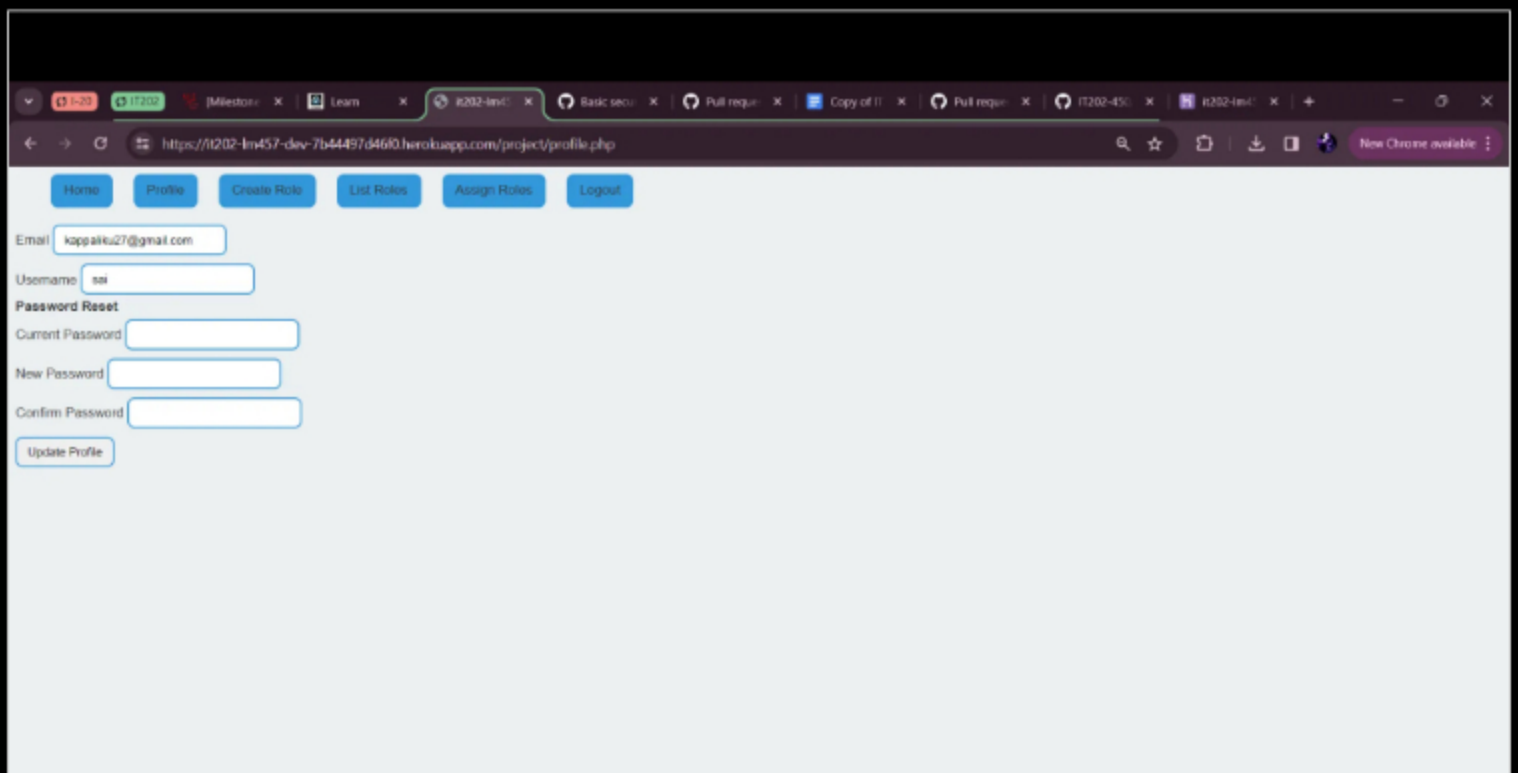
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Show the profile form correctly populated on page load (username, email)
#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



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)

COLLAPSE

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

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.

COLLAPSE

Task #3 - Points: 1

Text: Edit Profile Website Page

Checklist *The checkboxes are for your own tracking

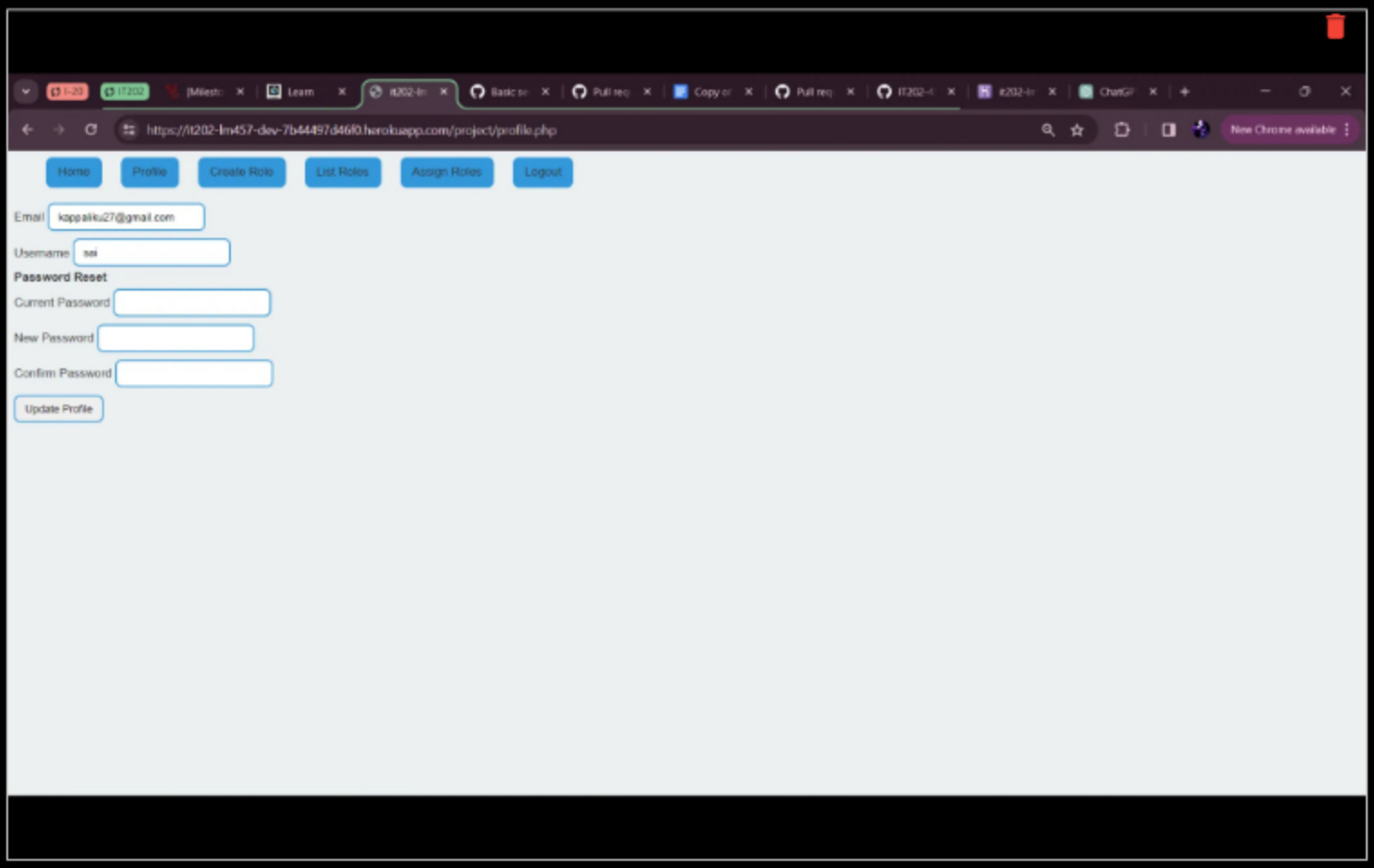
#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Demonstrate with before and after of a username change (including success message)
#4	1	Demonstrate with a before and after of an email change (including success message)
#5	1	Demonstrate 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

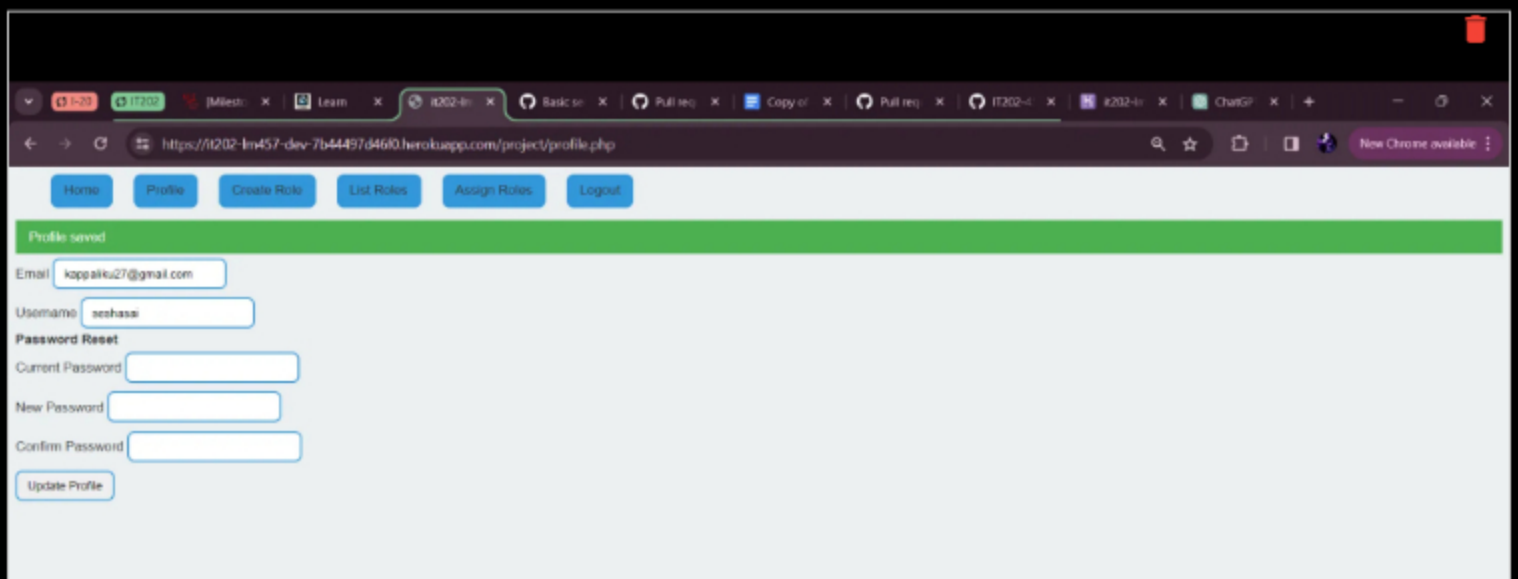
Small Medium Large



Initial username (Before change).

Checklist Items (1)

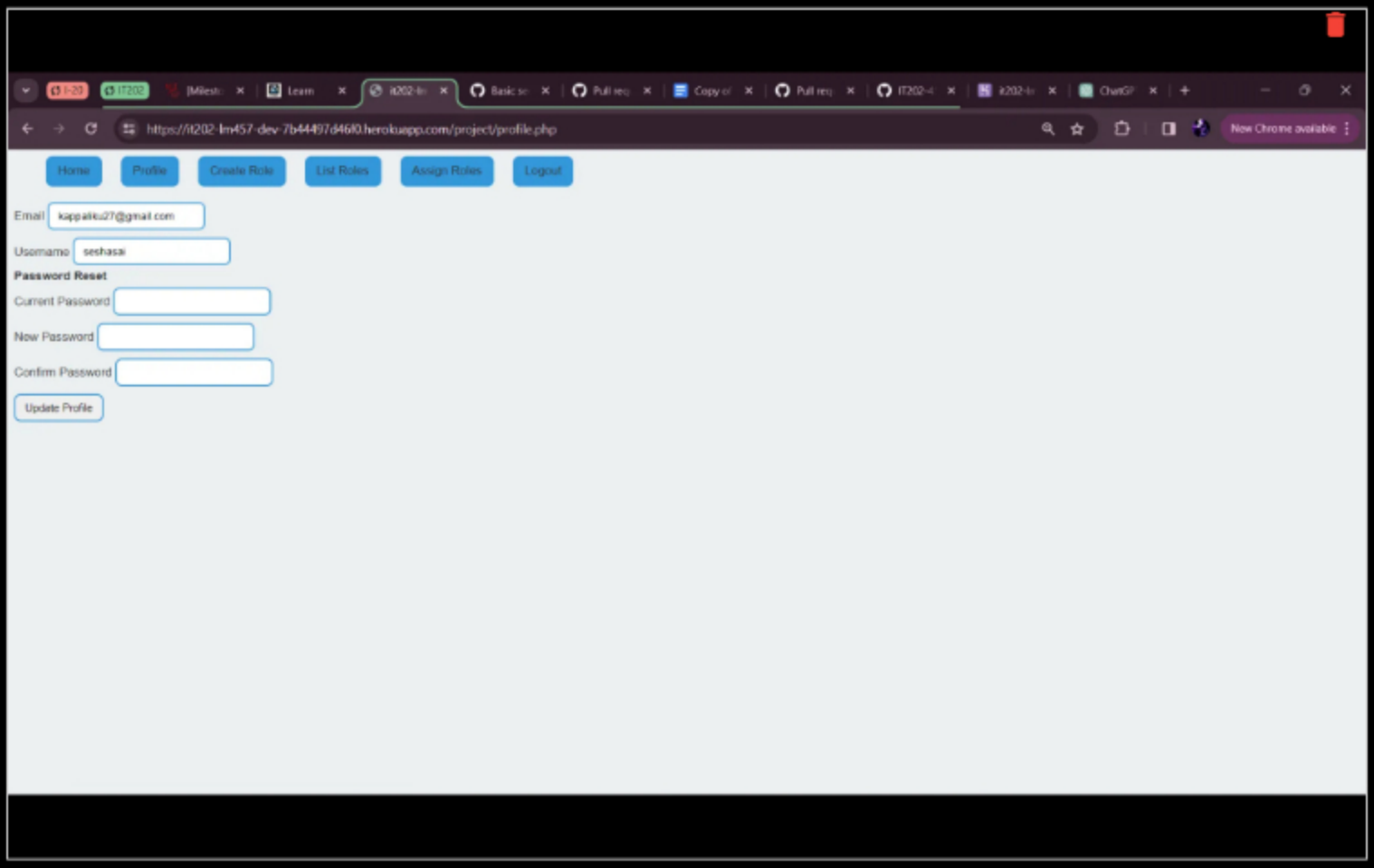
#3 Demonstrate with before and after of a username change (including success message)



After username change with success message.

Checklist Items (1)

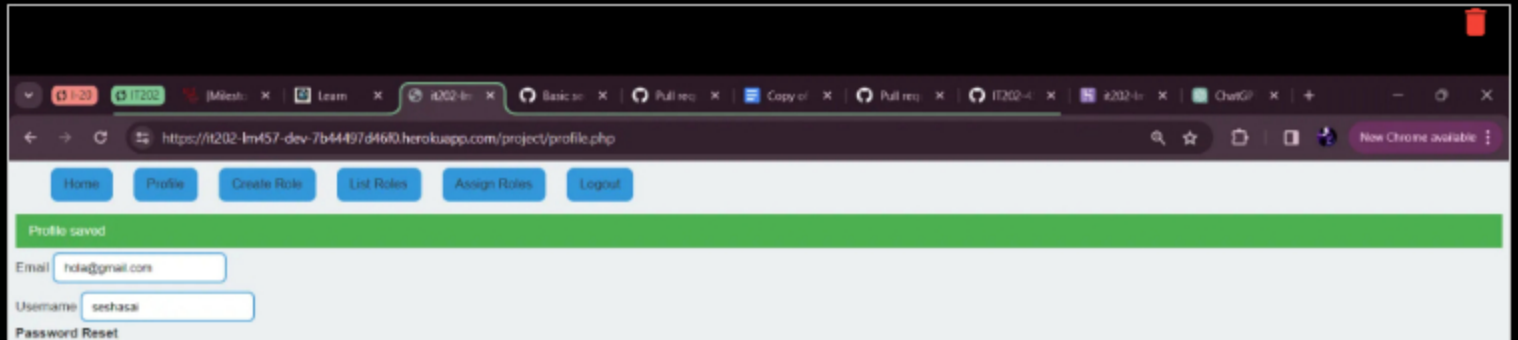
#3 Demonstrate with before and after of a username change (including success message)

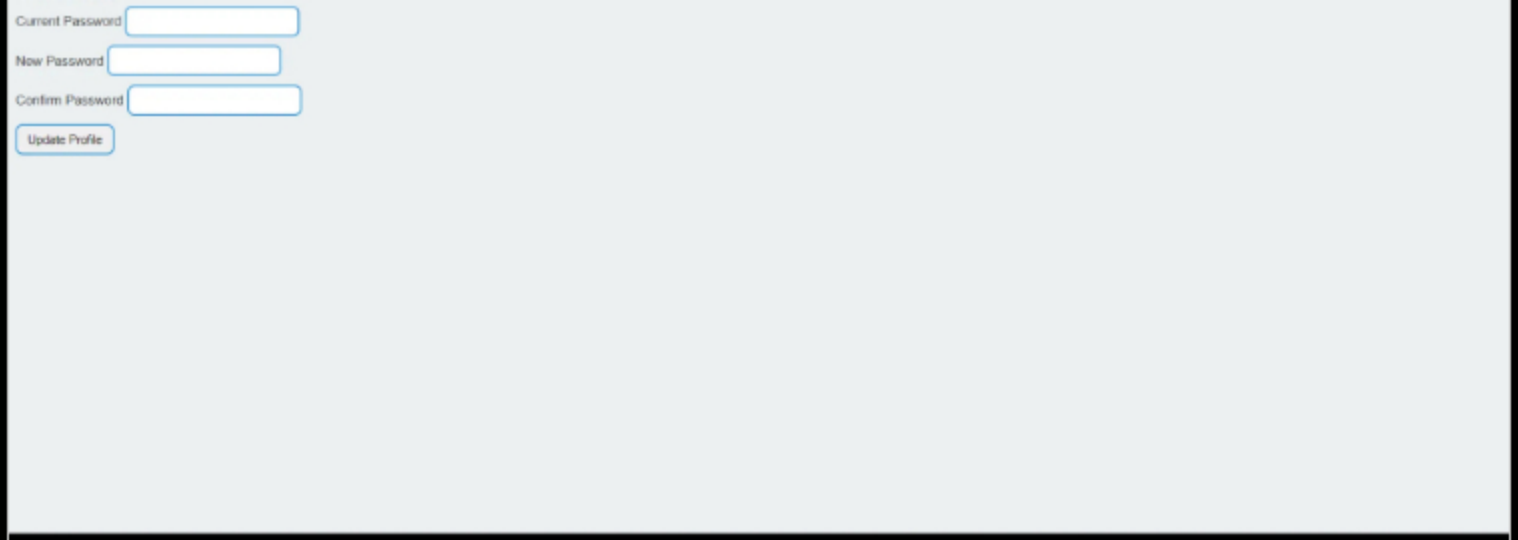


Before email change.

Checklist Items (1)

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

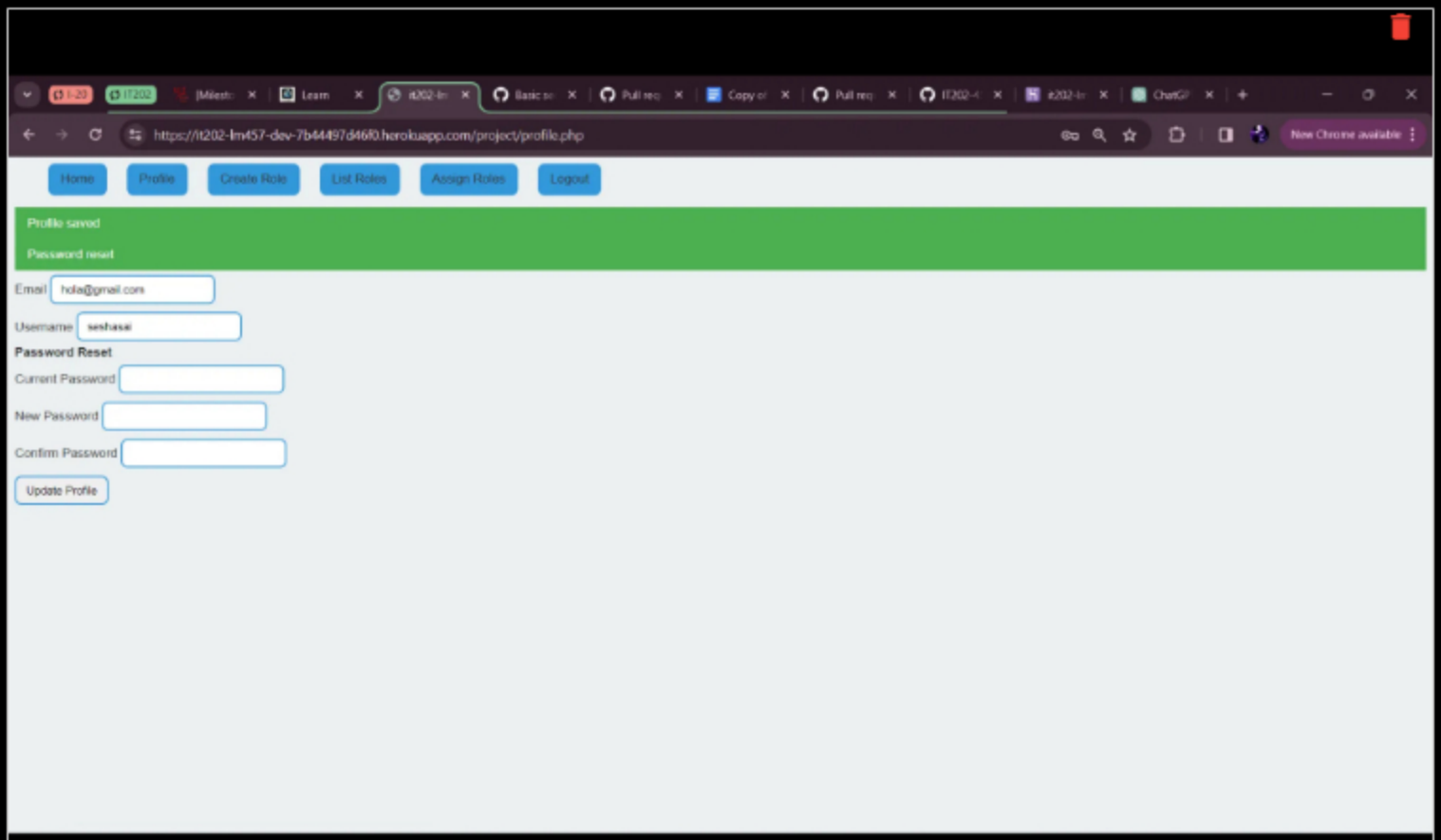




After email change with success message.

Checklist Items (1)

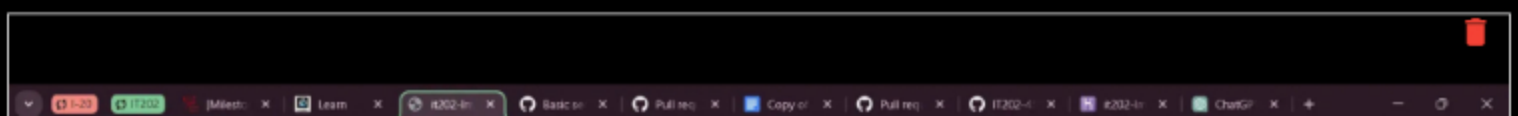
#4 Demonstrate with a before and after of an email change (including success message)

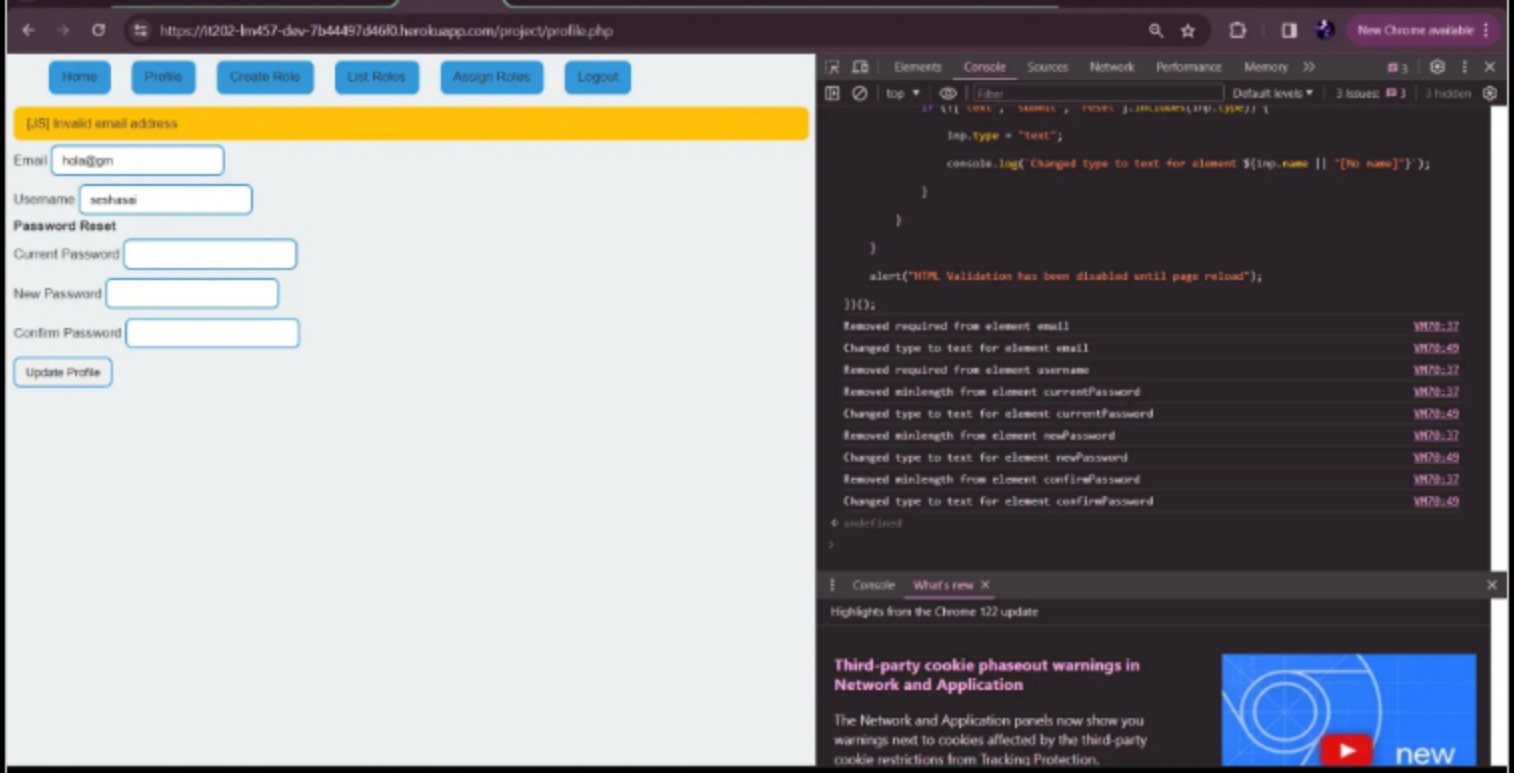


Success messages after updating the password.

Checklist Items (1)

#5 Demonstrate the success message of updating password

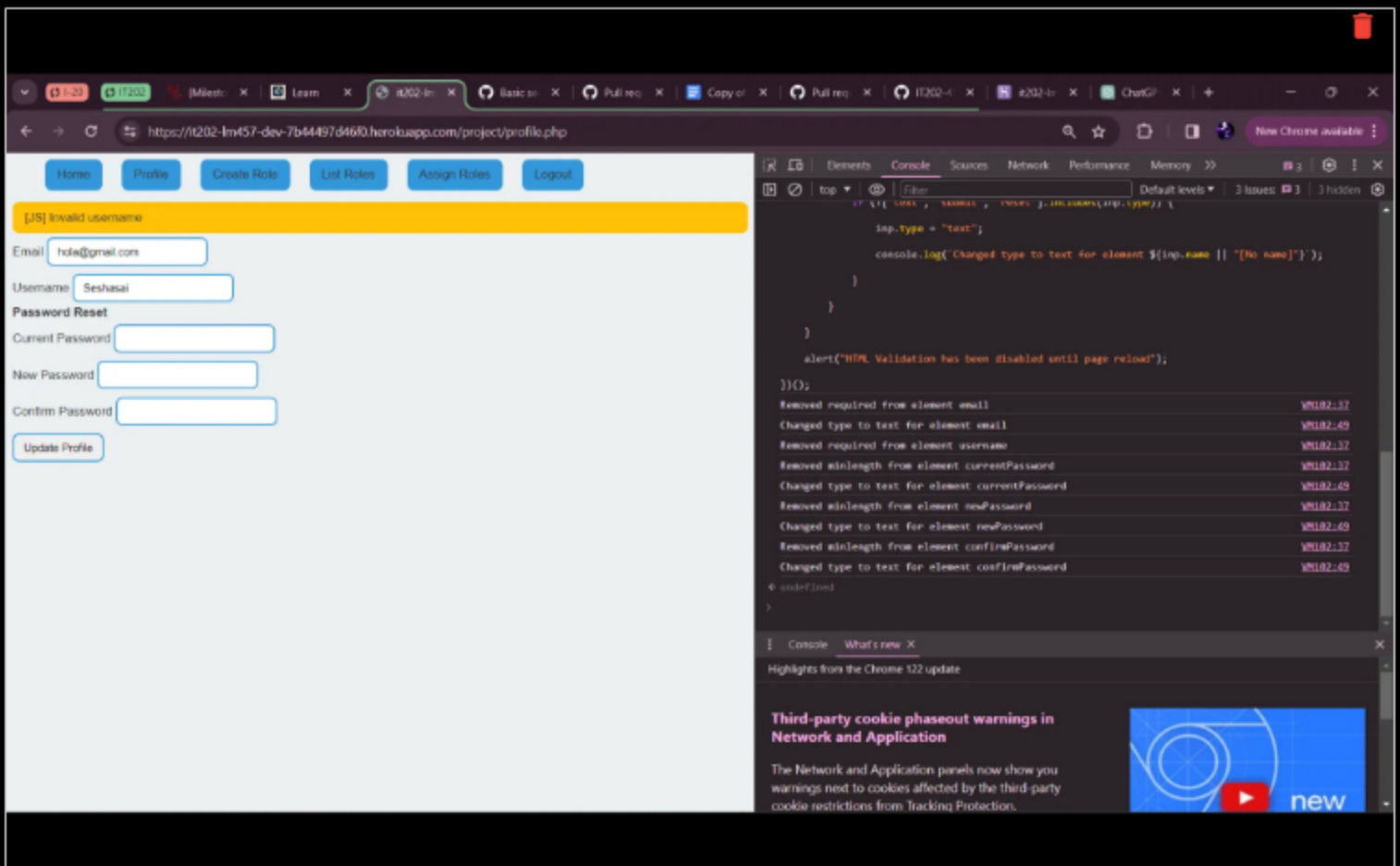




Invalid email JS validation.

Checklist Items (1)

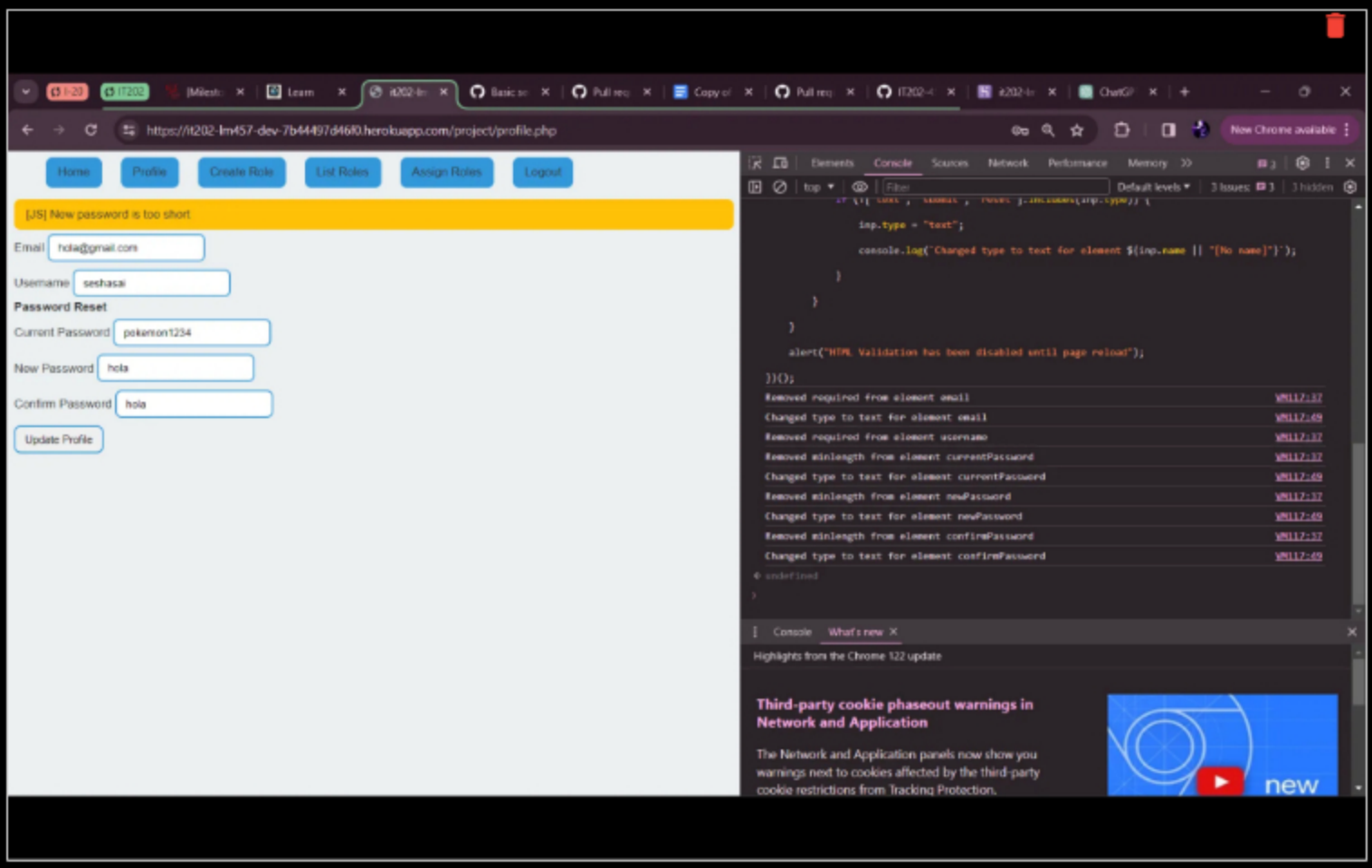
#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)



Invalid username JS validation.

Checklist Items (1)

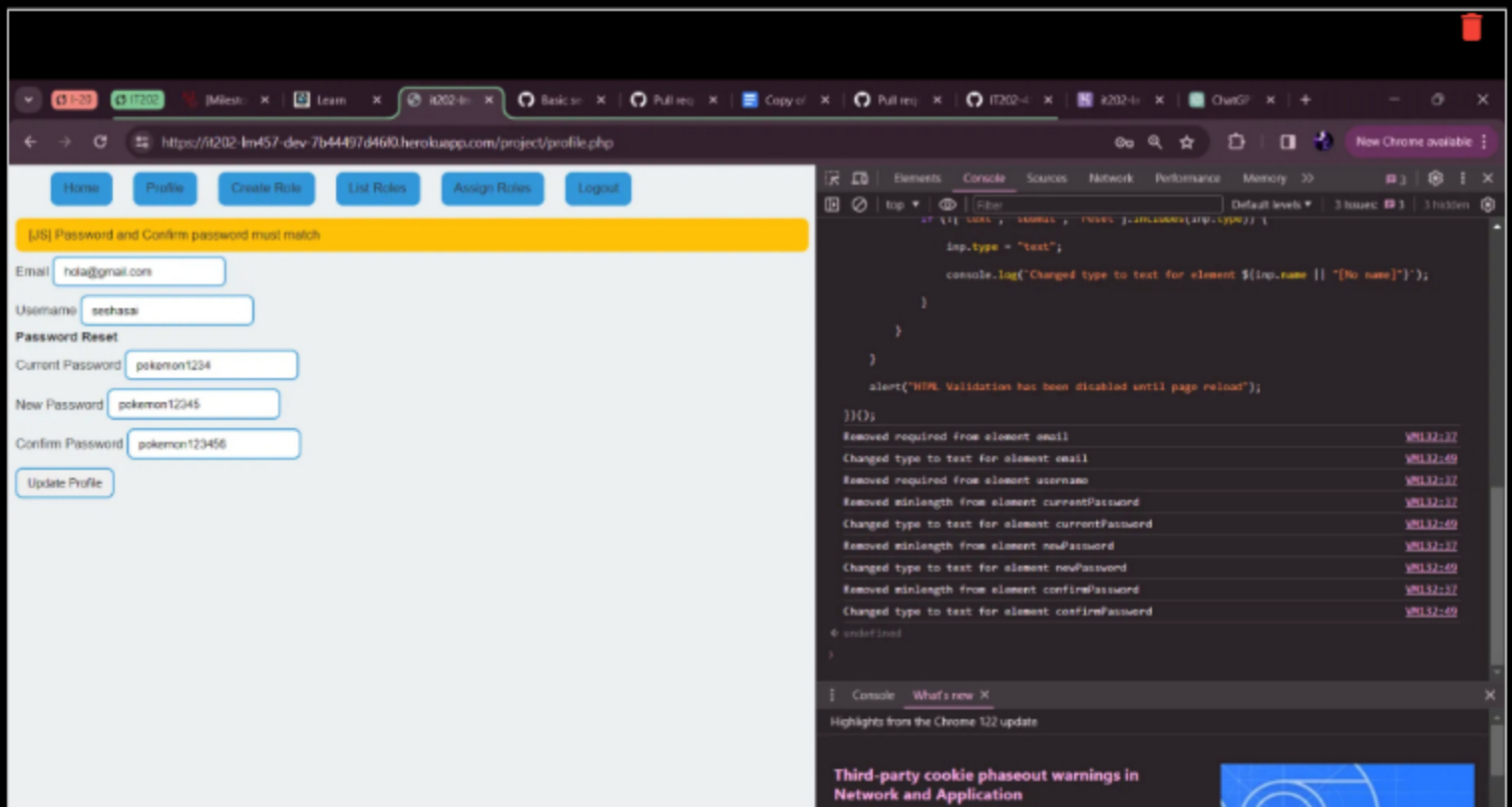
#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)



Invalid (short) password JS validation.

Checklist Items (1)

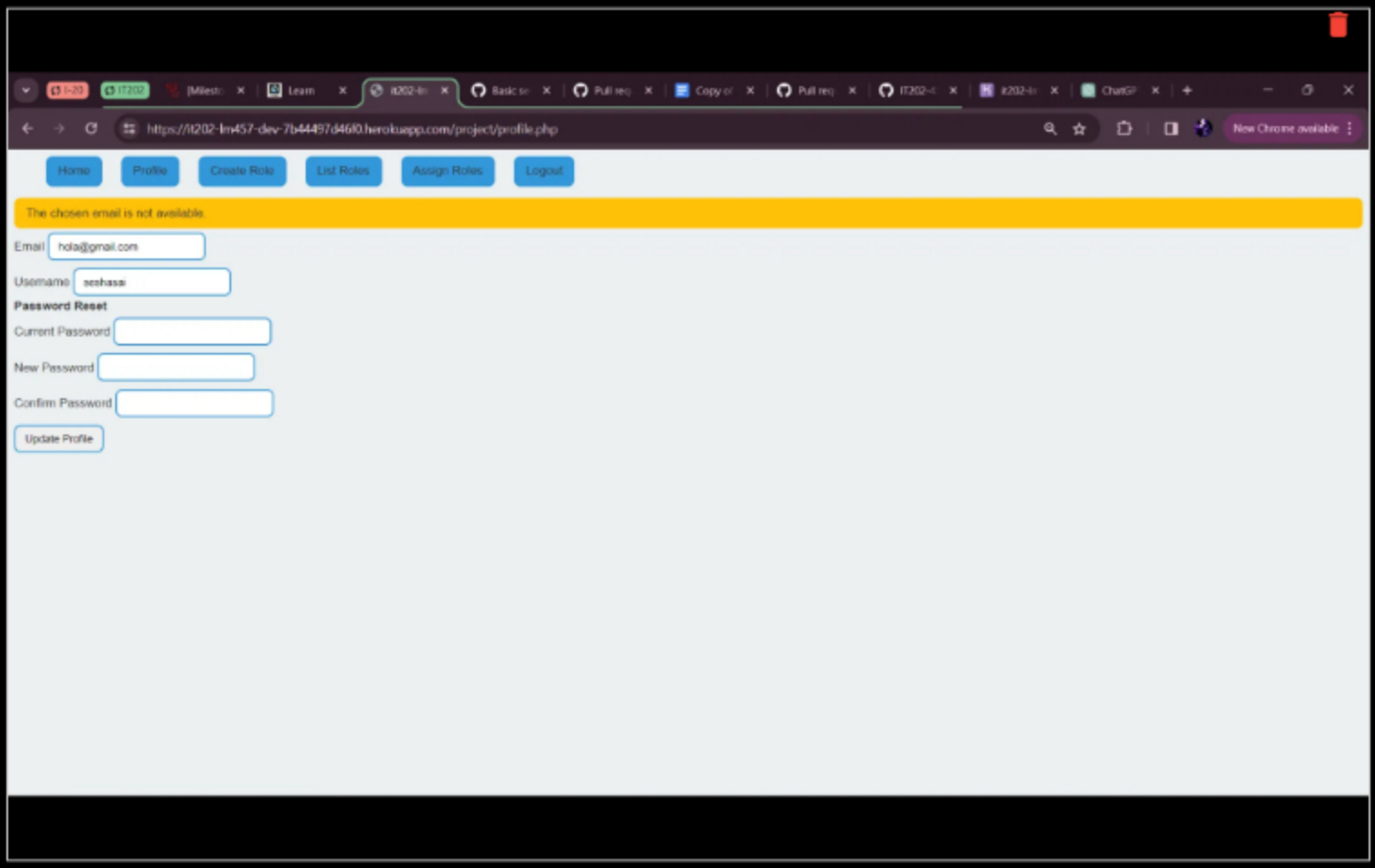
#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)



Password mismatch JS validation

Checklist Items (1)

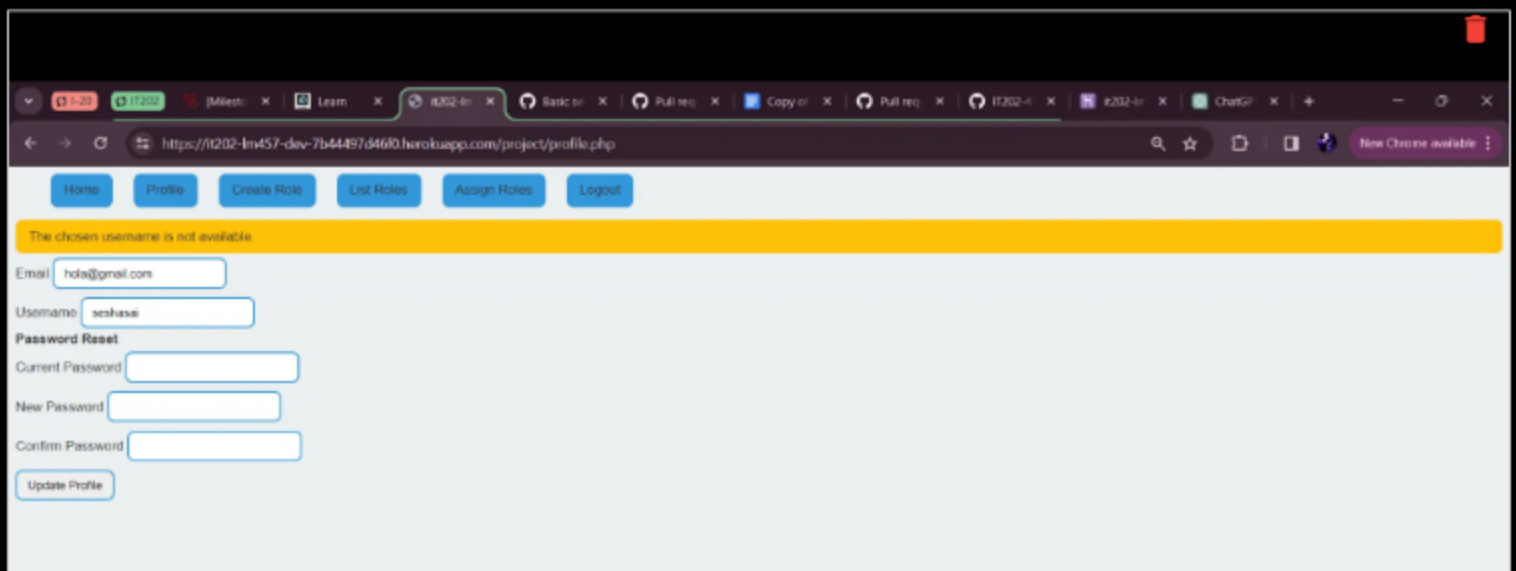
#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)



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

Checklist Items (1)

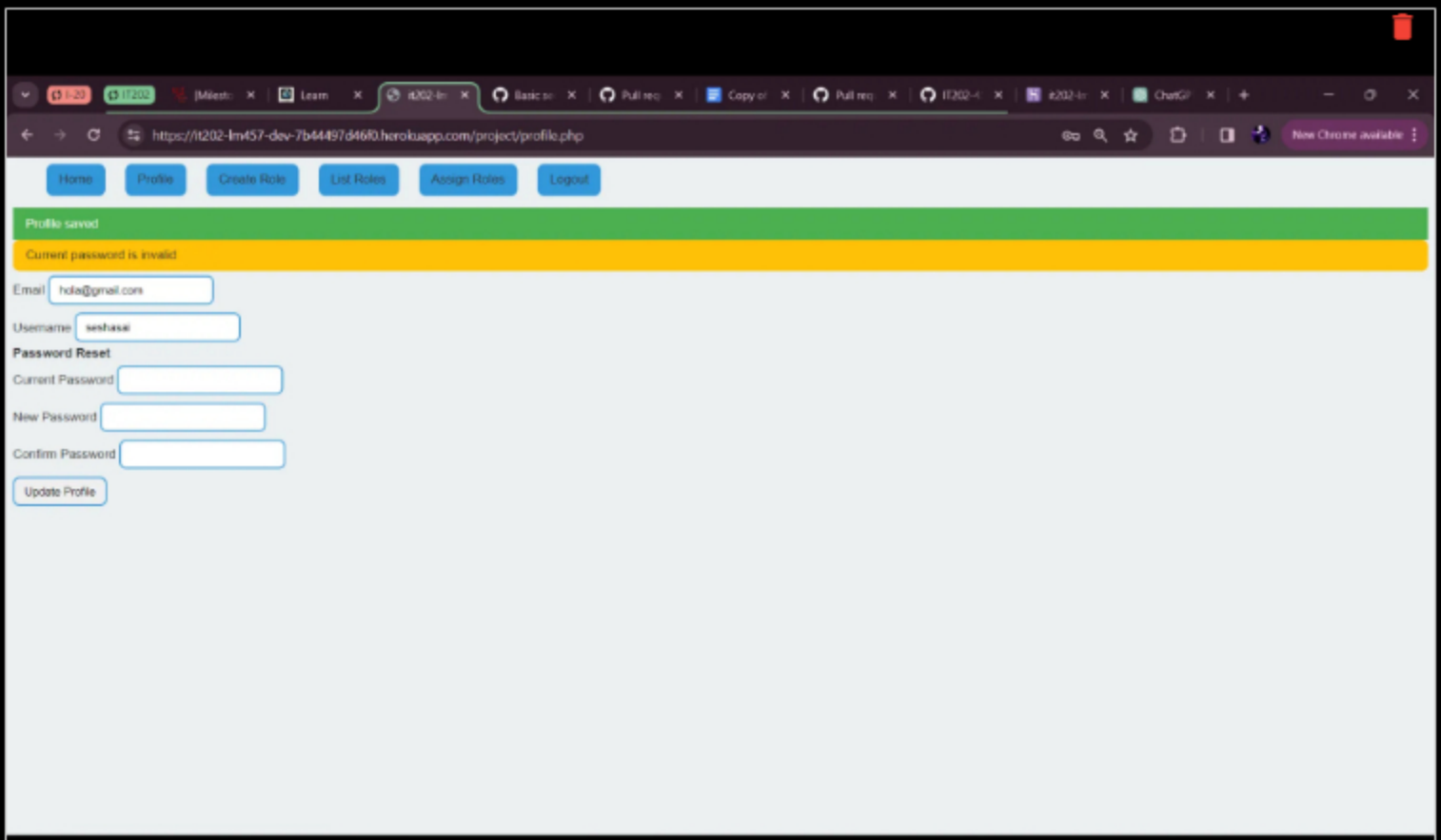
#7 Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)



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)



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)

COLLAPSE

Task #4 - Points: 1

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

Checklist

*The checkboxes are for your own tracking

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Updating Username/Email
#2	1	Updating password
#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.

COLLAPSE

Task #5 - 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/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

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of wakatime

Task Screenshots:

Gallery Style: Large View

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 2024-03-25 until 2024-03-31

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

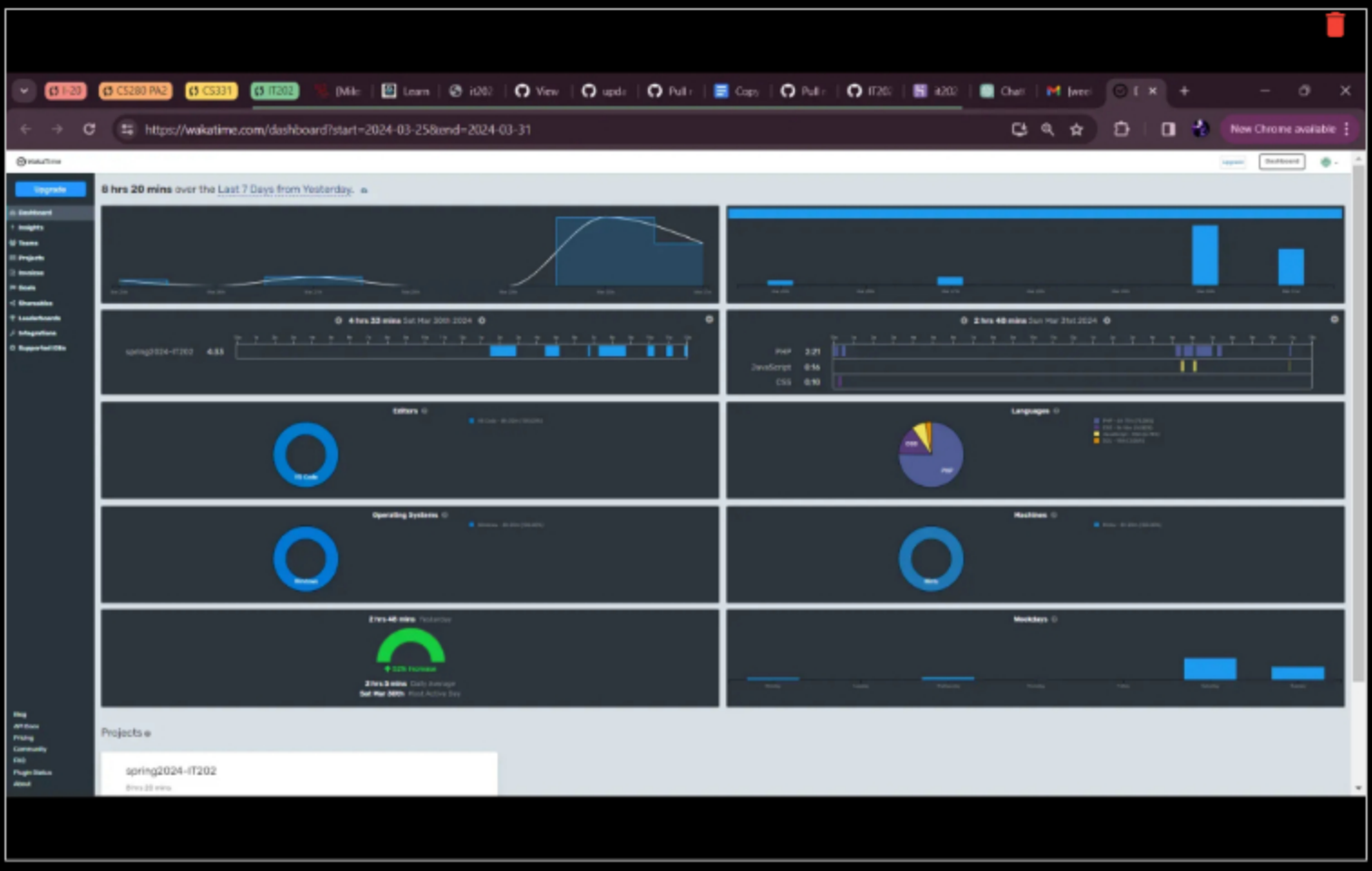
Operating Systems:

Windows 8 hrs 20 mins

Machines:

Mintu 8 hrs 20 mins

Updating files for Milestone1 plus JS validation in register, login, and profile pages.



WakaTime Dashboard.

COLLAPSE

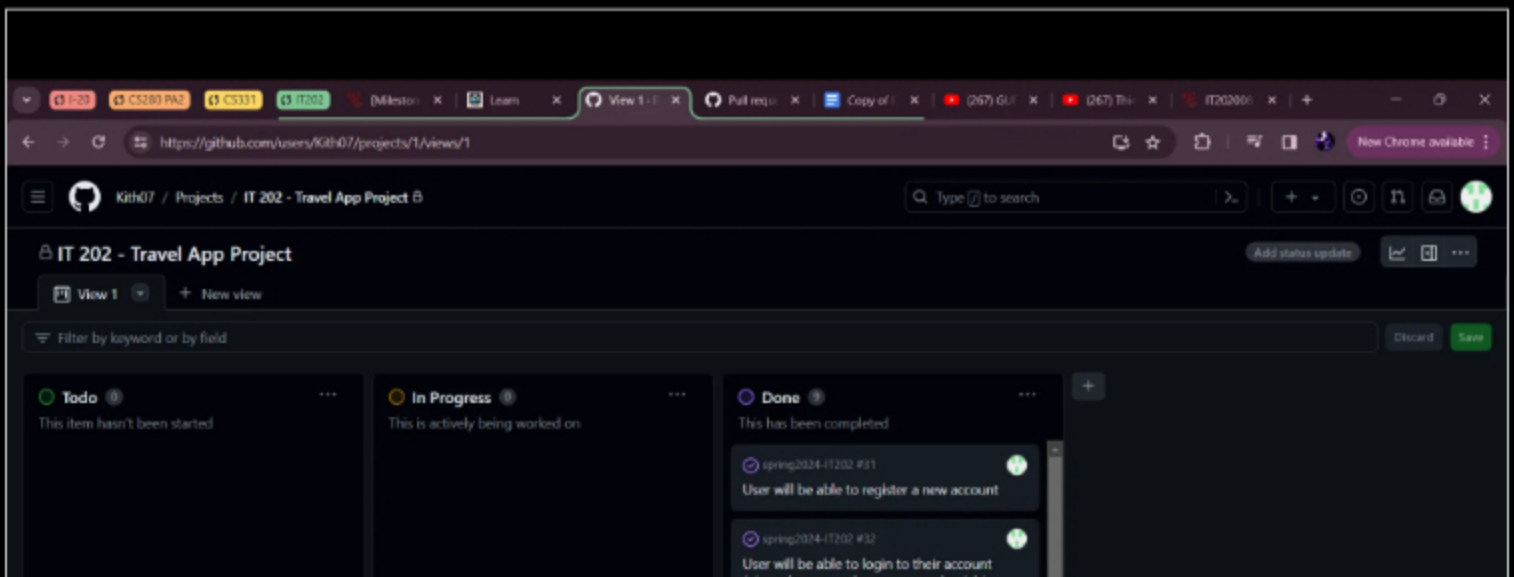
Task #2 - Points: 1

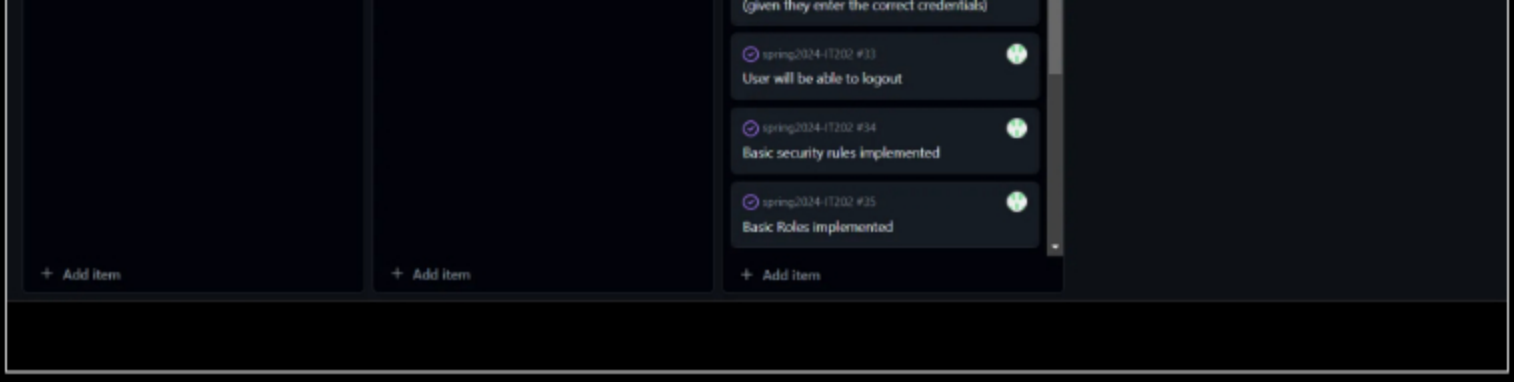
Text: Screenshot of your project board from GitHub (tasks should be in the proper column)

Task Screenshots:

Gallery Style: Large View

Small Medium Large





Github repo Project board screenshot for Milestone1.

^COLLAPSE^

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

^COLLAPSE^

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