Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/njz5

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

Submissions:

Submission Selection

1 Submission [active] 4/1/2024 7:38:55 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

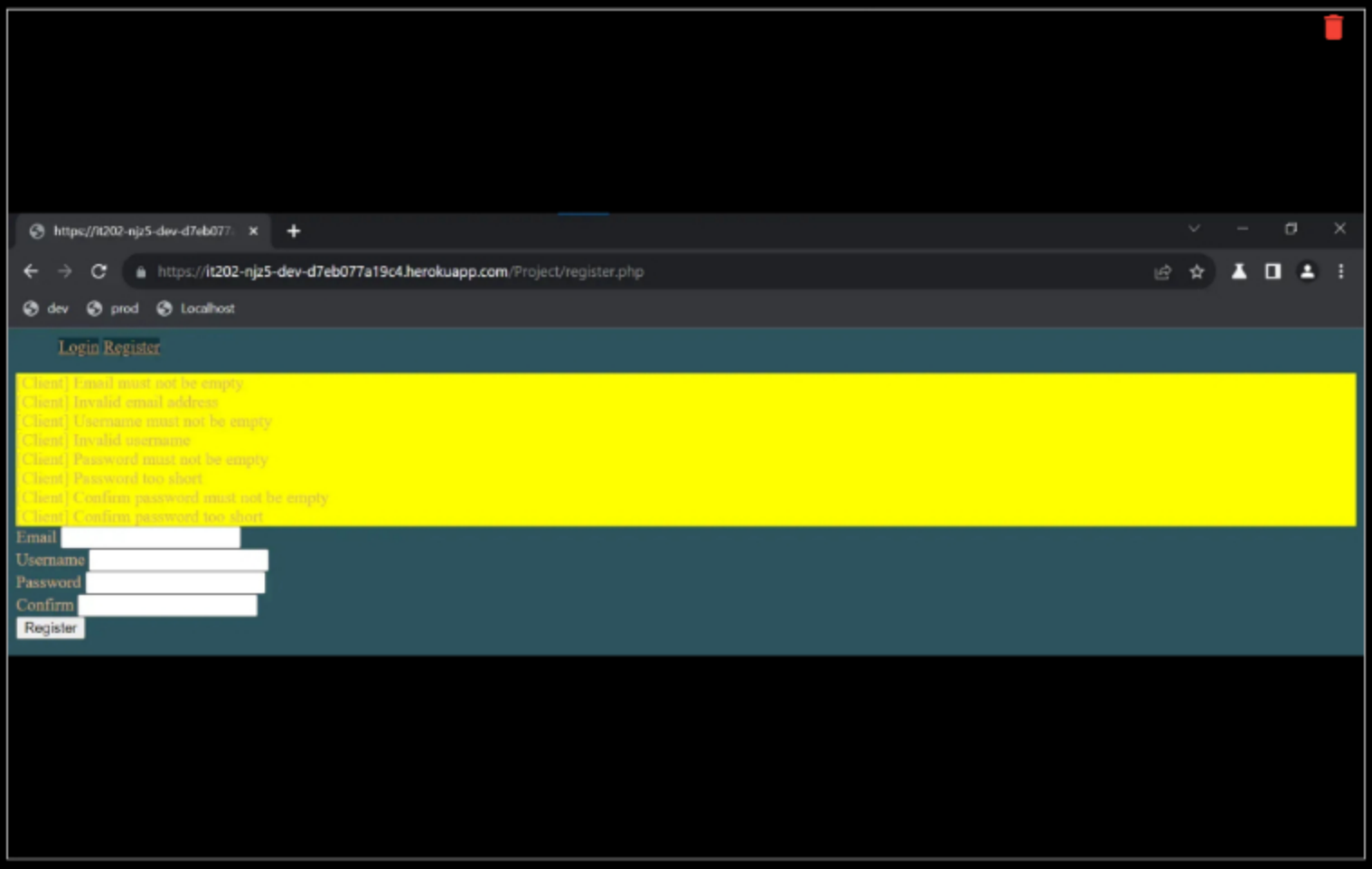
*The checkboxes are for your own tracking

#	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



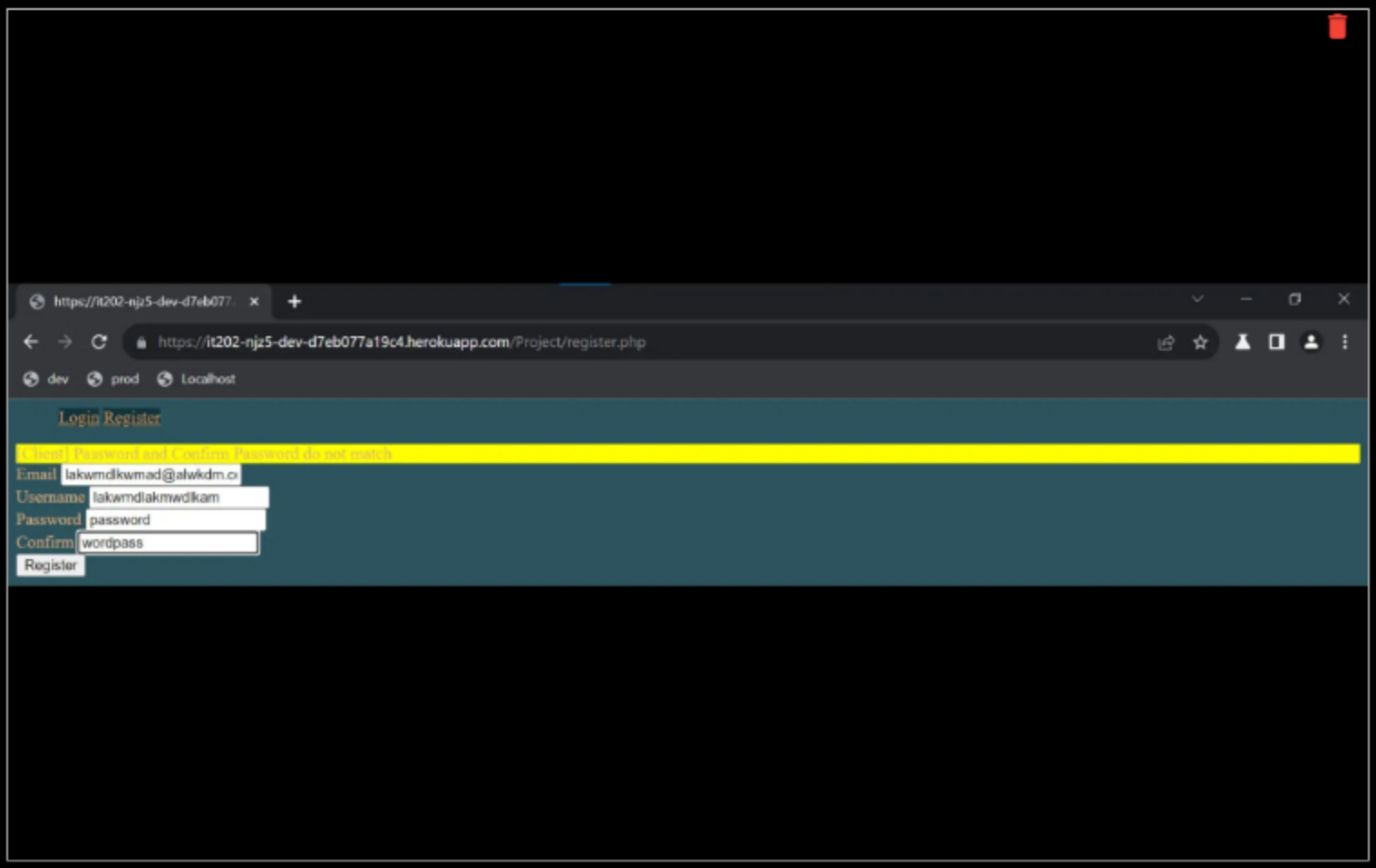
Empty registration.

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

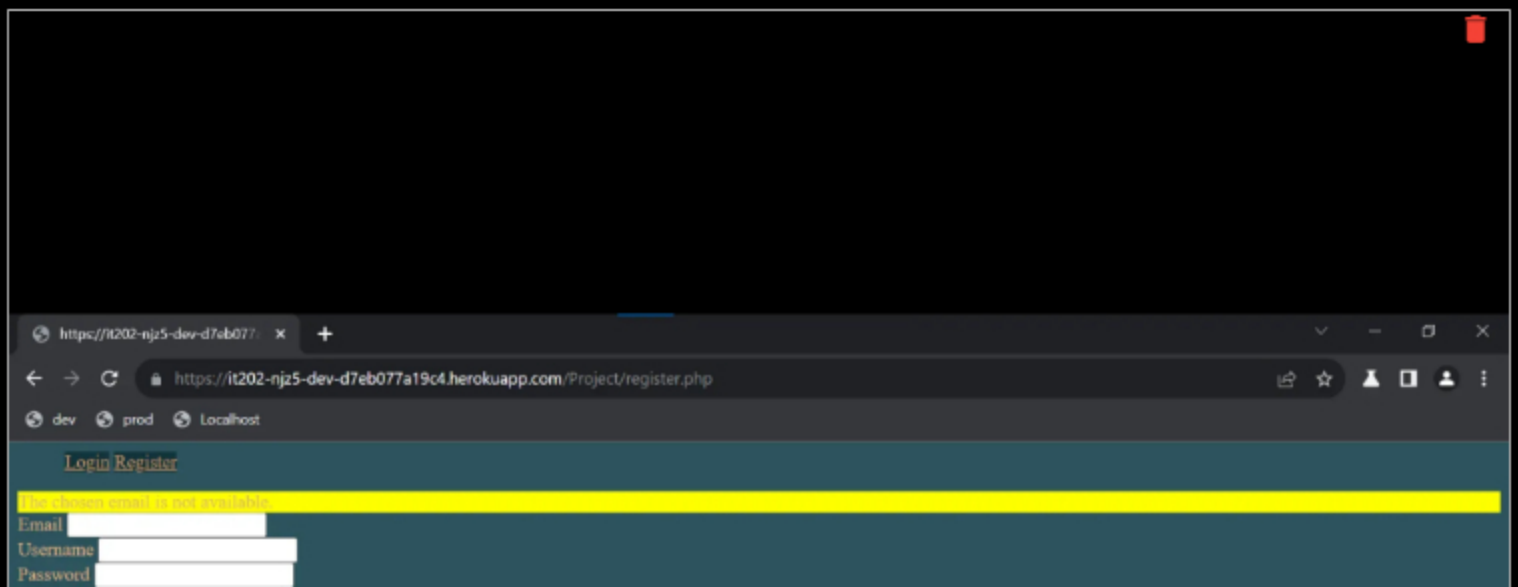
#3 Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required

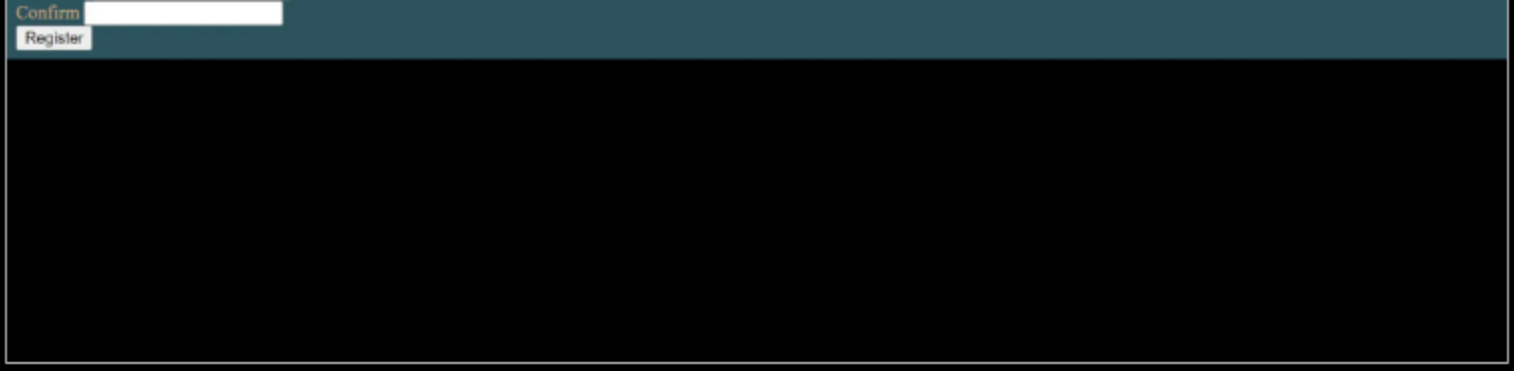


Passwords do not match.

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

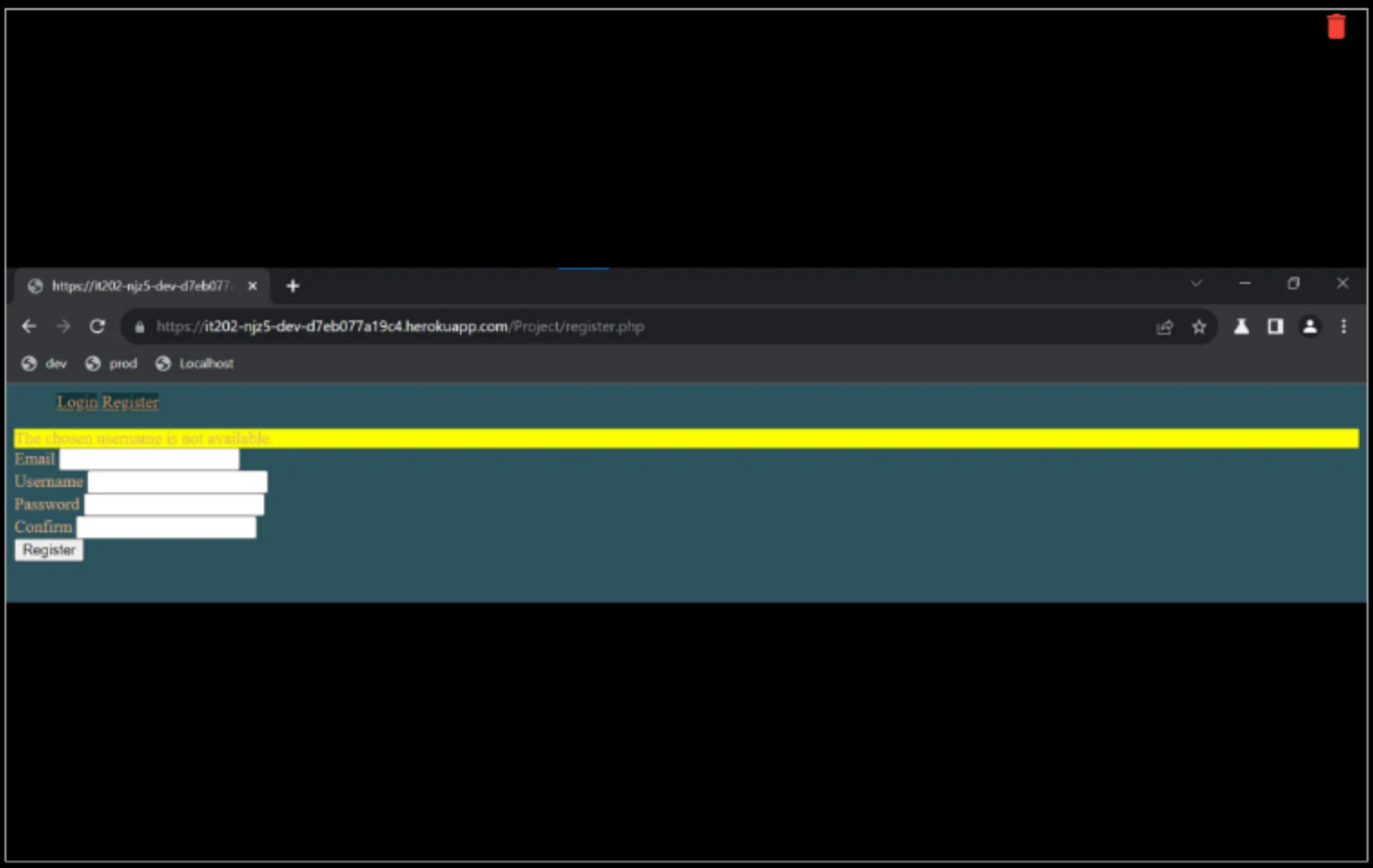




Email in use.

Checklist Items (1)

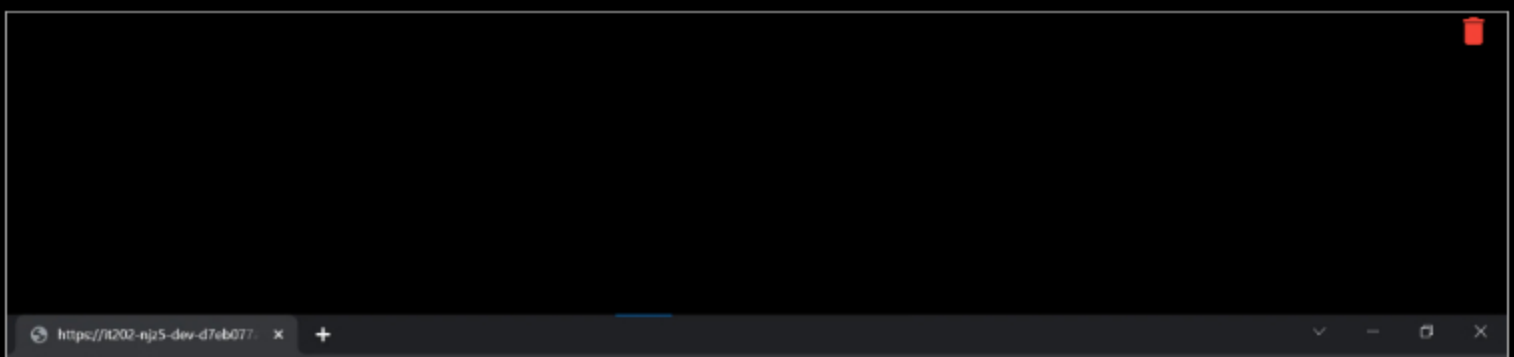
#4 Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)

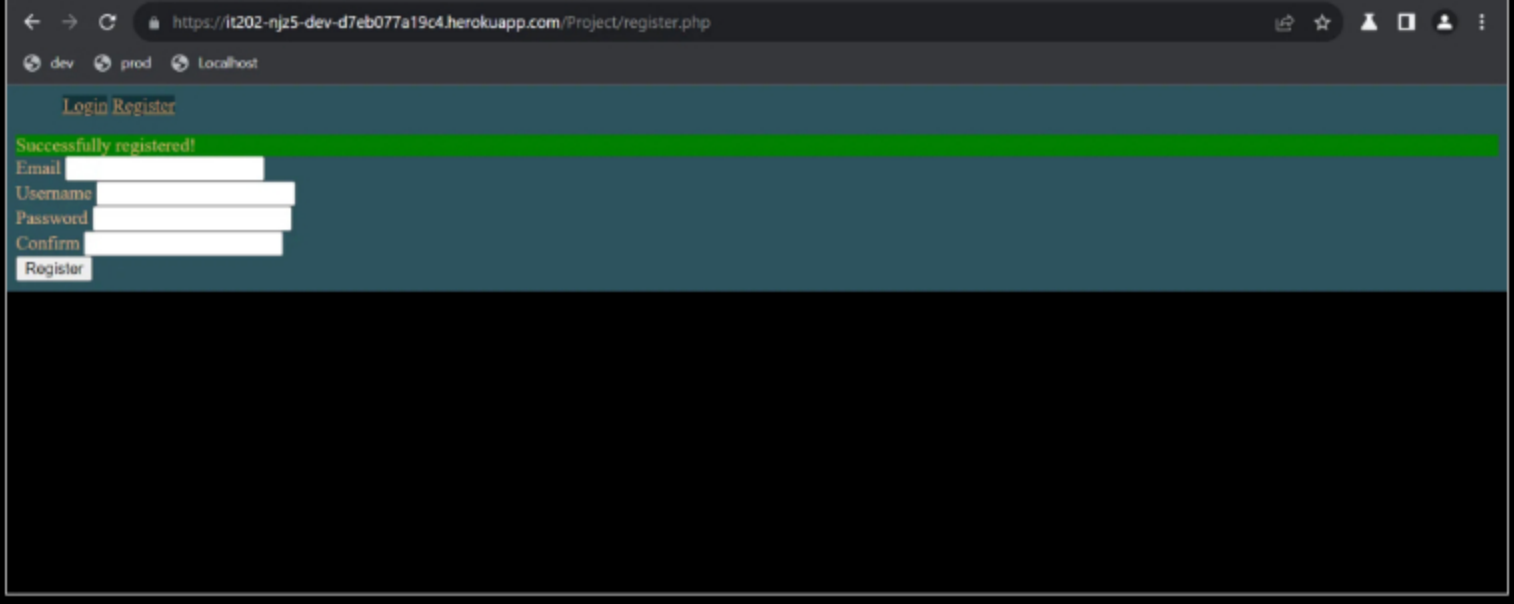


Username in use.

Checklist Items (1)

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





Successfully registered.

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

```
4   <form onsubmit="return validate(this)" method="POST">
5       <div>
6           <label for="email">Email</label>
7           <input id="email" type="email" name="email" required/>
8       </div>
9       <div>
10          <label for="username">Username</label>
11          <input id="username" type="text" name="username" required maxlength="30" />
12      </div>
13      <div>
14          <label for="pw">Password</label>
15          <input type="password" id="pw" name="password" required minlength="8" />
16      </div>
17      <div>
```

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

Registration form code.

COLLAPSE

Task #3 - Points: 1

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

Checklist

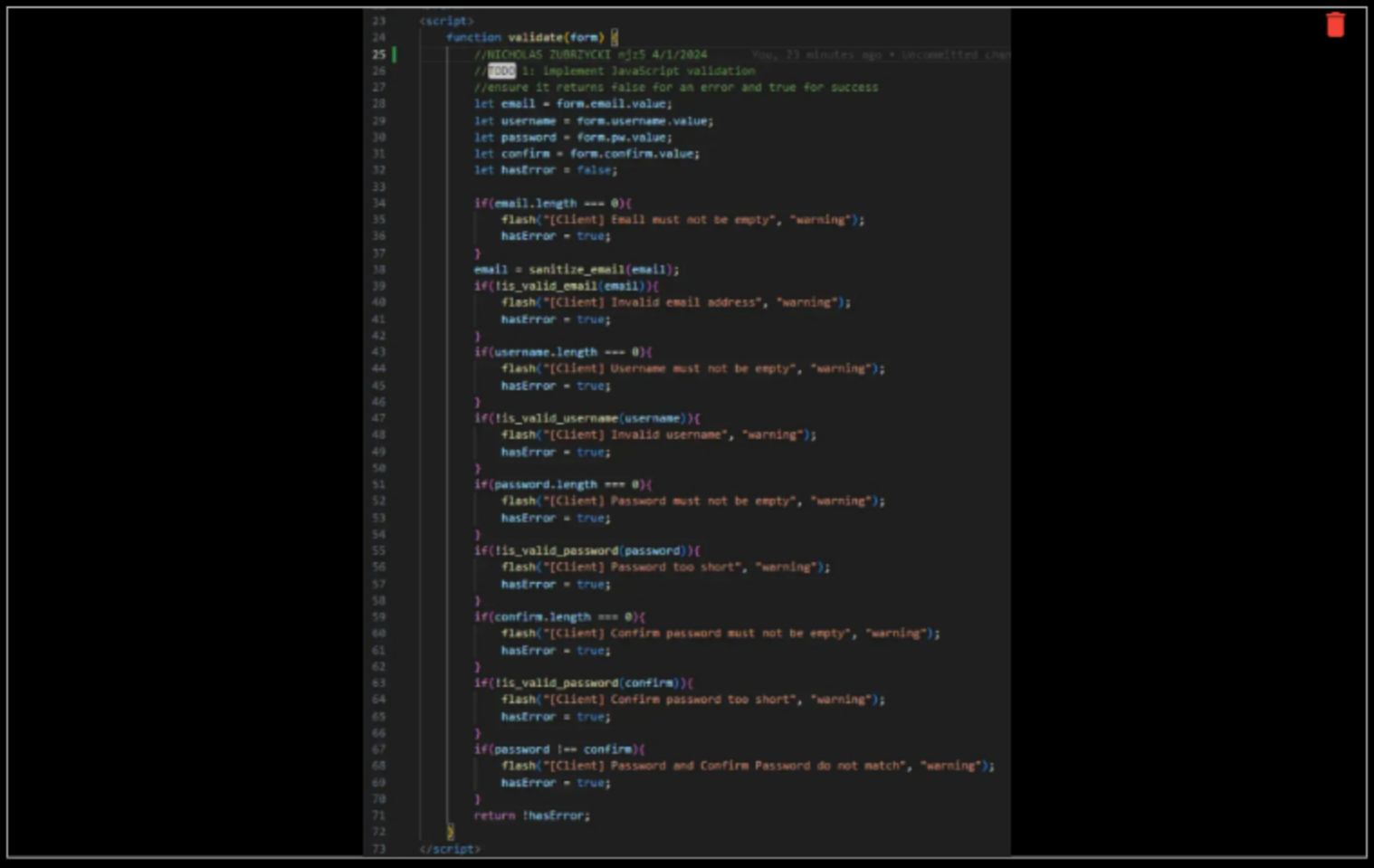
*The checkboxes are for your own tracking

#	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



```
<script>
    function validate(form) {
        //NICHOLAS ZUBRZYCKI mjz5 4/1/2024      You, 23 minutes ago • Uncommitted chan
        //TODO 1: implement JavaScript validation
        //ensure it returns false for an error and true for success
        let email = form.email.value;
        let username = form.username.value;
        let password = form.pw.value;
        let confirm = form.confirm.value;
        let hasError = false;

        if(email.length === 0){
            flash("[Client] Email must not be empty", "warning");
            hasError = true;
        }
        email = sanitize_email(email);
        if(!is_valid_email(email)){
            flash("[Client] Invalid email address", "warning");
            hasError = true;
        }
        if(username.length === 0){
            flash("[Client] Username must not be empty", "warning");
            hasError = true;
        }
        if(!is_valid_username(username)){
            flash("[Client] Invalid username", "warning");
            hasError = true;
        }
        if(password.length === 0){
            flash("[Client] Password must not be empty", "warning");
            hasError = true;
        }
        if(!is_valid_password(password)){
            flash("[Client] Password too short", "warning");
            hasError = true;
        }
        if(confirm.length === 0){
            flash("[Client] Confirm password must not be empty", "warning");
            hasError = true;
        }
        if(!is_valid_password(confirm)){
            flash("[Client] Confirm password too short", "warning");
            hasError = true;
        }
        if(password !== confirm){
            flash("[Client] Password and Confirm Password do not match", "warning");
            hasError = true;
        }
        return !hasError;
    }
</script>
```

JavaScript validation in register

JavaScript validation in register.

Checklist Items (1)

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

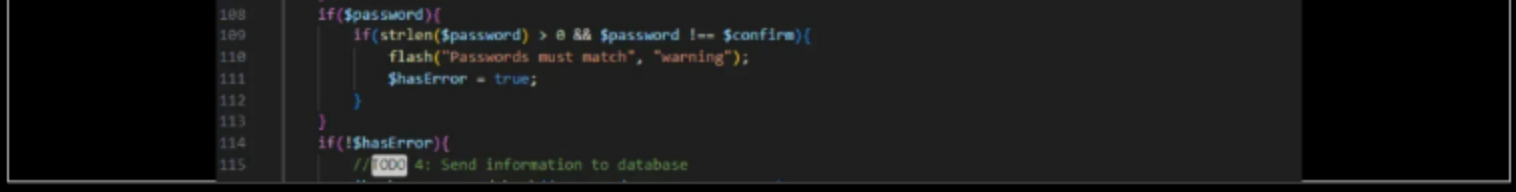
```javascript
//NICHOLAS ZUBRZYCKI njz5 4/1/2024
function sanitize_email(email = "")
{
    //Remove all characters except specified, according to PHP documentation.
    return email.replace(/[^a-zA-Z0-9!#$%&'*+-=?^_`{|}~@.[]]/g,'');
}

function is_valid_email(email = "")
{
    //One or more chars, @, One or more chars, ., One or more chars
    let emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    if(emailRegex.test(email)){
        return true;
    }
    return false;
}

function is_valid_username(username)
{
    let usernameRegex = /^[a-z0-9_-]{3,16}$/
    if(usernameRegex.test(username)){
        return true;
    }
    return false;
}

function is_valid_password(password)
{
    return password.length >= 8;
}
```

JavaScript validation functions in helpers.js.

Checklist Items (1)

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

```php
<?php
//TODO 2: add PHP Code
//NICHOLAS ZUBRZYCKI njz5 4/1/2024      You, 2 minutes ago • Uncommitted changes
if(isset($_POST["email"]) && isset($_POST["username"]) && isset($_POST["password"]) && isset($_POST["confirm"])){
    $email = se($_POST, "email", "", false);
    $username = se($_POST, "username", "", false);
    $password = se($_POST, "password", "", false);
    $confirm = se($_POST, "confirm", "", false);
    //TODO 3: validate/use
    $hasError = false;
    if(empty($email)){
        flash("Email must not be empty", "warning");
        $hasError = true;
    }
    $email = sanitize_email($email);
    if(!is_valid_email($email)){
        flash("Invalid email address", "warning");
        $hasError = true;
    }
    if(!is_valid_username($username)) {
        flash("Username must only be alphanumeric and can only contain - or _", "warning");
        $hasError = true;
    }
    if(empty($password)){
        flash("Password must not be empty", "warning");
        $hasError = true;
    }
    if(empty($confirm)){
        flash("Confirm password must not be empty", "warning");
        $hasError = true;
    }
    if(!is_valid_password($password)){
        flash("Password is too short", "warning");
        $hasError = true;
    }
```



```php
if($password){
    if(strlen($password) > 0 && $password !== $confirm){
        flash("Passwords must match", "warning");
        $hasError = true;
    }
}
if(!$hasError){
    //TODO 4: Send information to database
```

PHP validation in register.

Checklist Items (1)

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

```php
<?php
//NICHOLAS ZUBRZYCKI njz5 4/1/2024
function sanitize_email($email = "")
{
    return filter_var(trim($email), FILTER_SANITIZE_EMAIL);
}
function is_valid_email($email = "")
{
    return filter_var(trim($email), FILTER_VALIDATE_EMAIL);
}
function is_valid_username($username)
{
    return preg_match('/^[a-z0-9_-]{3,16}$/', $username);
}
function is_valid_password($password)
{
    if($password){
        return strlen($password) >= 8;
    }
}
```

PHP validation functions in sanitizers.php.

Checklist Items (1)

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

COLLAPSE

Task #4 - Points: 1

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

Checklist

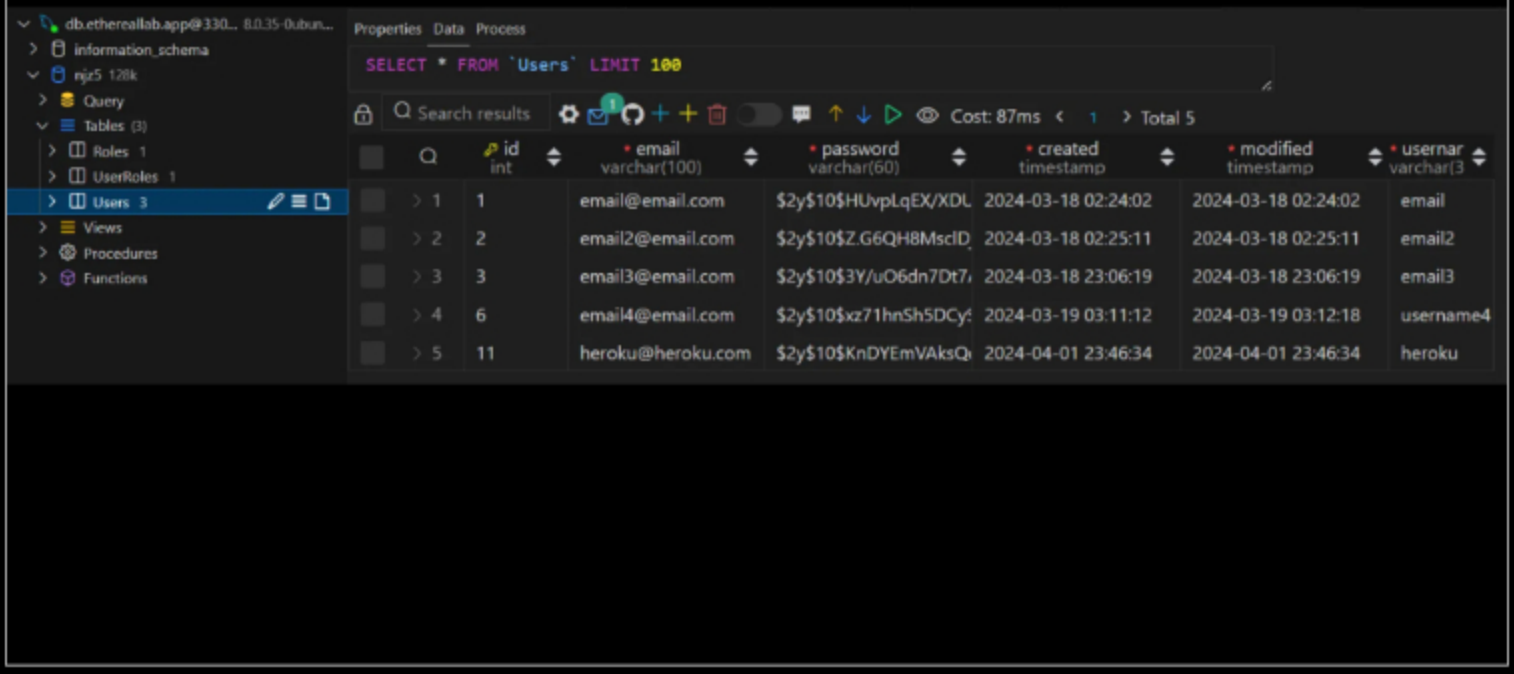
*The checkboxes are for your own tracking

#	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.

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 user loads an empty page on register.php. After inputting their information into each field (email, username, password, and confirm password), the user submits their information. First the HTML validation validates the data using form requirements. If that is successful, then the JavaScript validation validates the data by inspecting the values of input. If that is successful, then the PHP validation validates the data using the POST data. If all layers of validation confirm the data is good, then email, username, and password are inserted into the users table. The data for id, created, and modified is automatic, so the database handles that itself.

COLLAPSE

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/NickZub/njz5-it202-008/pull/12

URL #2
https://github.com/NickZub/njz5-it202-008/pull/22

URL #3
https://github.com/NickZub/njz5-it202-008/pull/24

User Login (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of form on website page

Checklist

*The checkboxes are for your own tracking

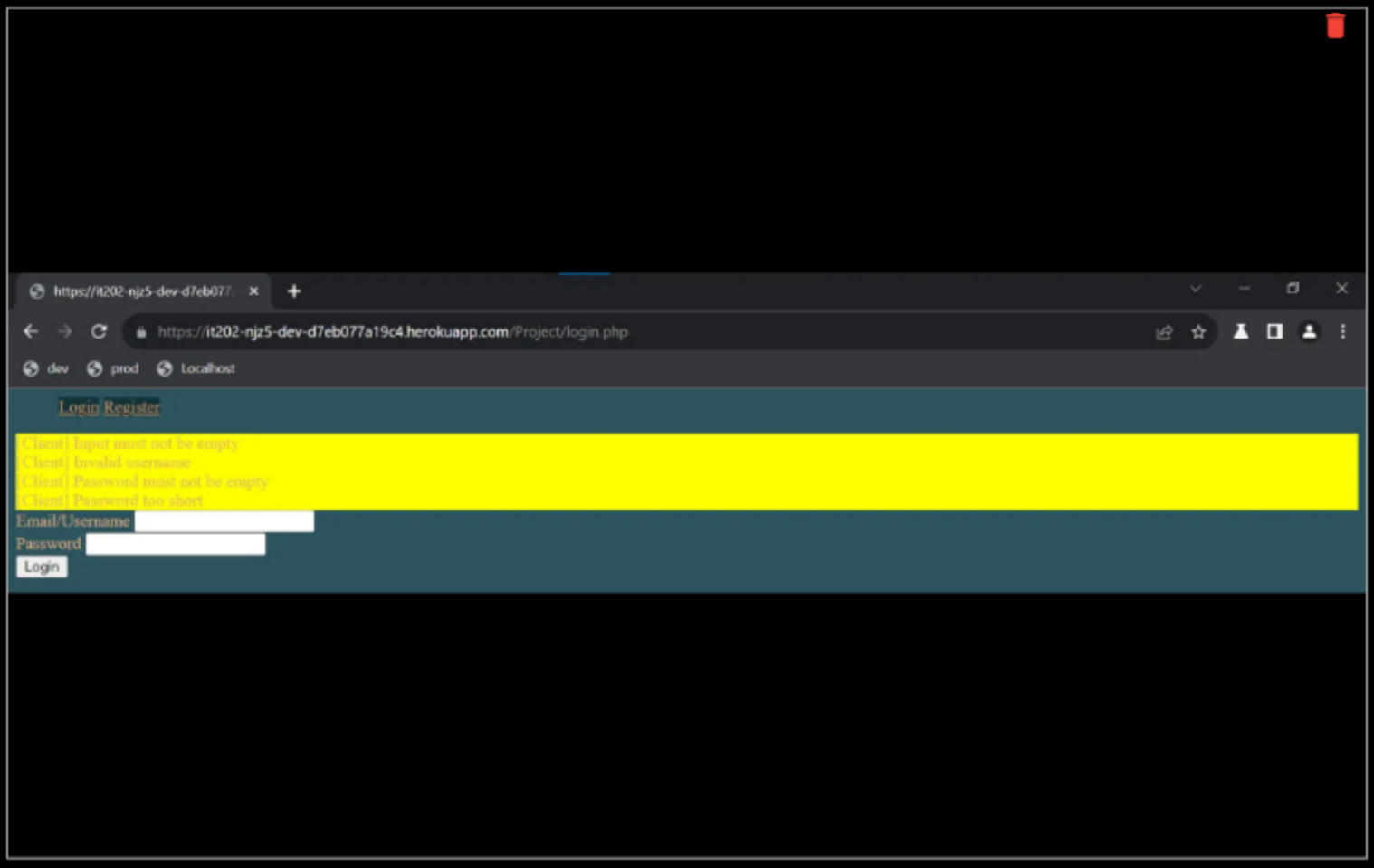
#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Include correct data where username is used to login
#4	1	Include correct data where email is used to login
#5	1	Show success login message
#6	1	Demonstrate JavaScript validation for 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



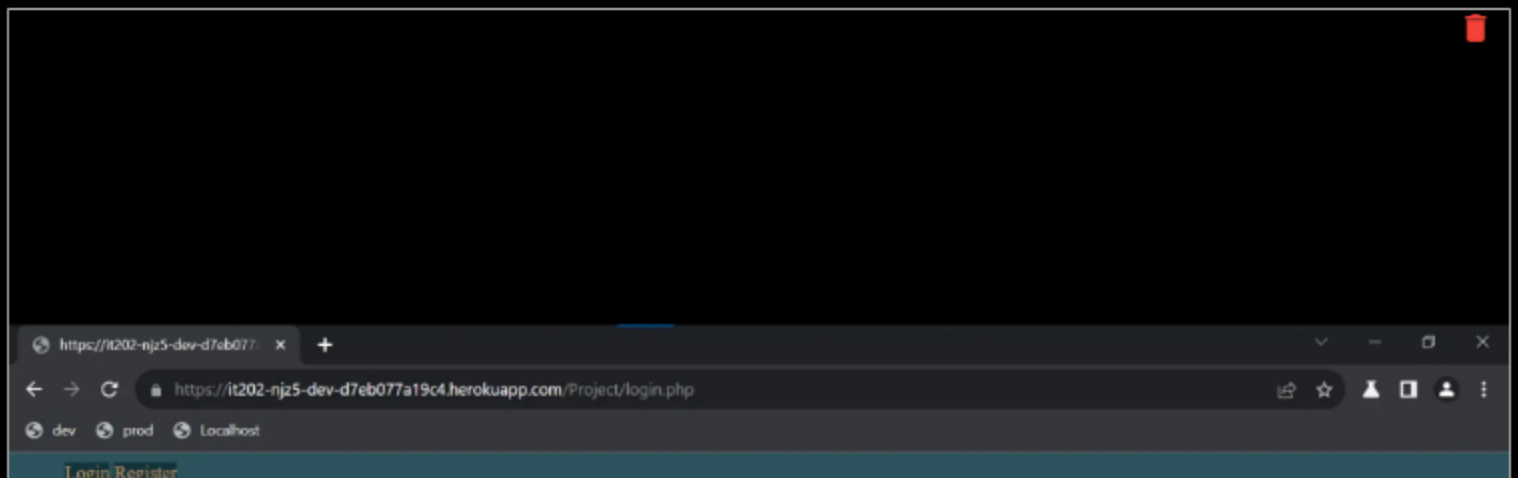
Empty login.

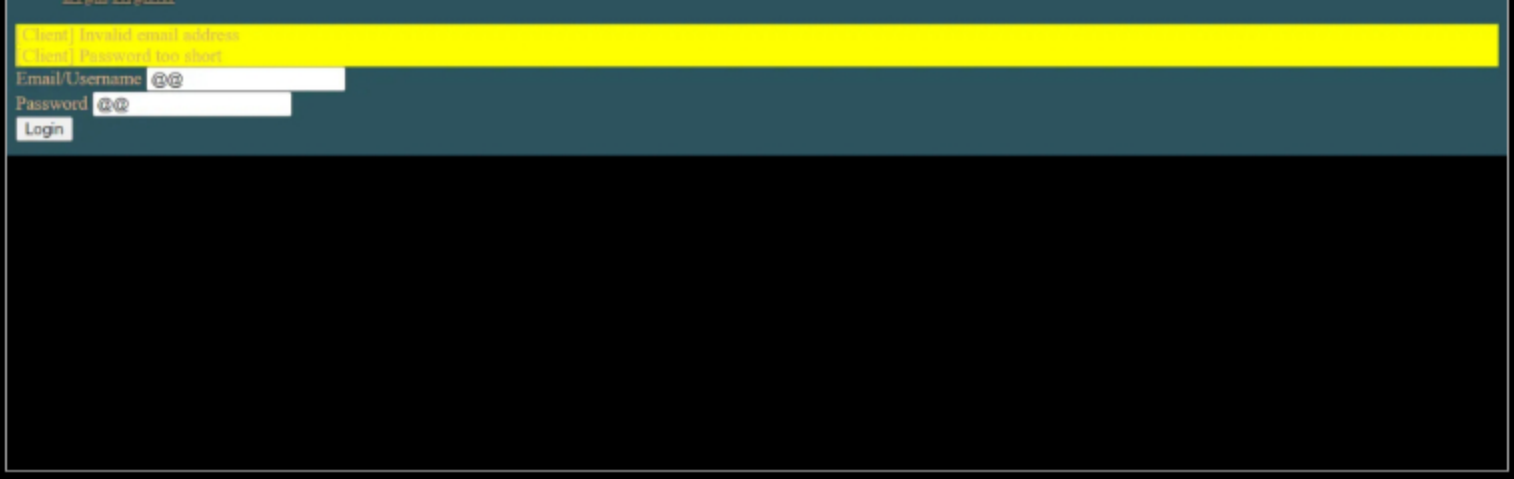
Checklist Items (3)

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

#2 Should have thoughtful CSS applied

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





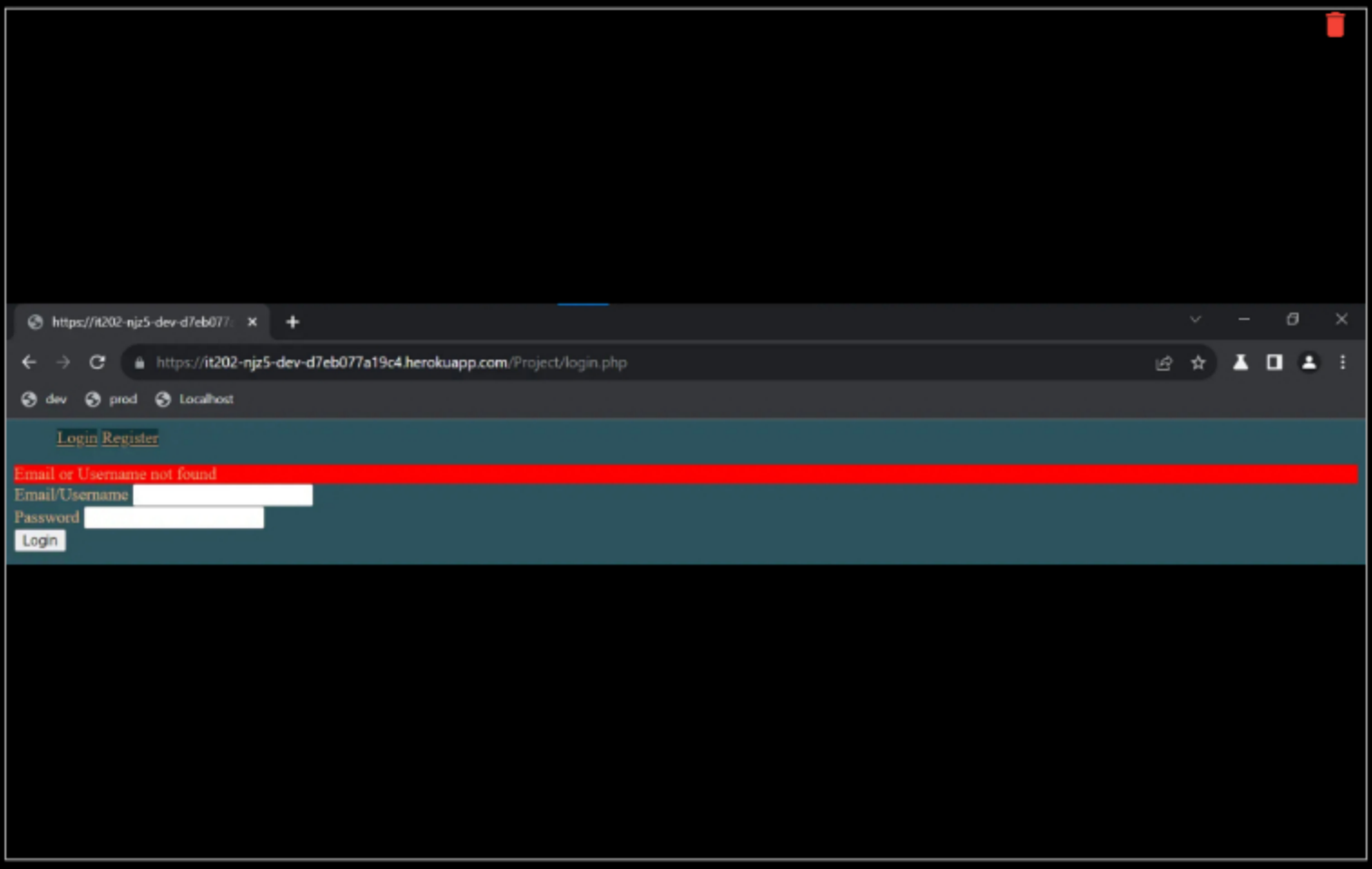
Invalid email and password.

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

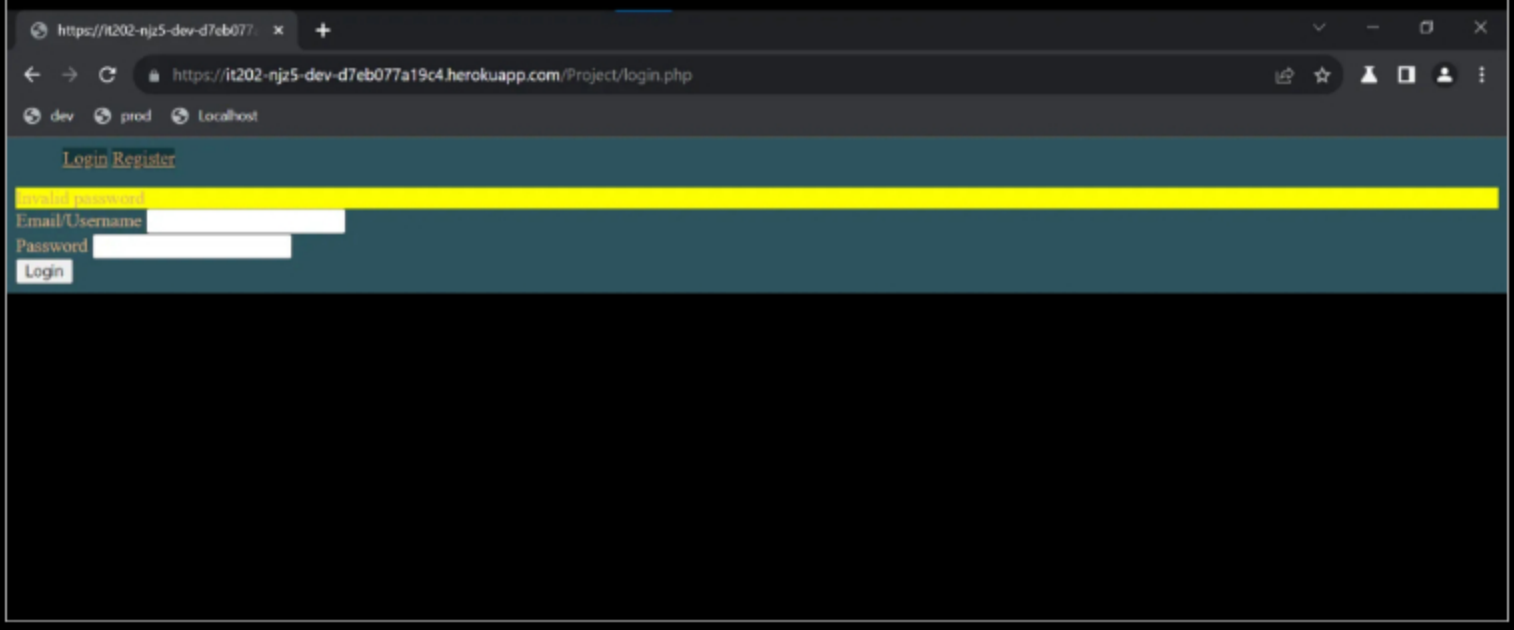
#6 Demonstrate JavaScript validation for each field [can be combined] (username format, email format, password format, and each field being required)



Account doesn't exist.

Checklist Items (1)

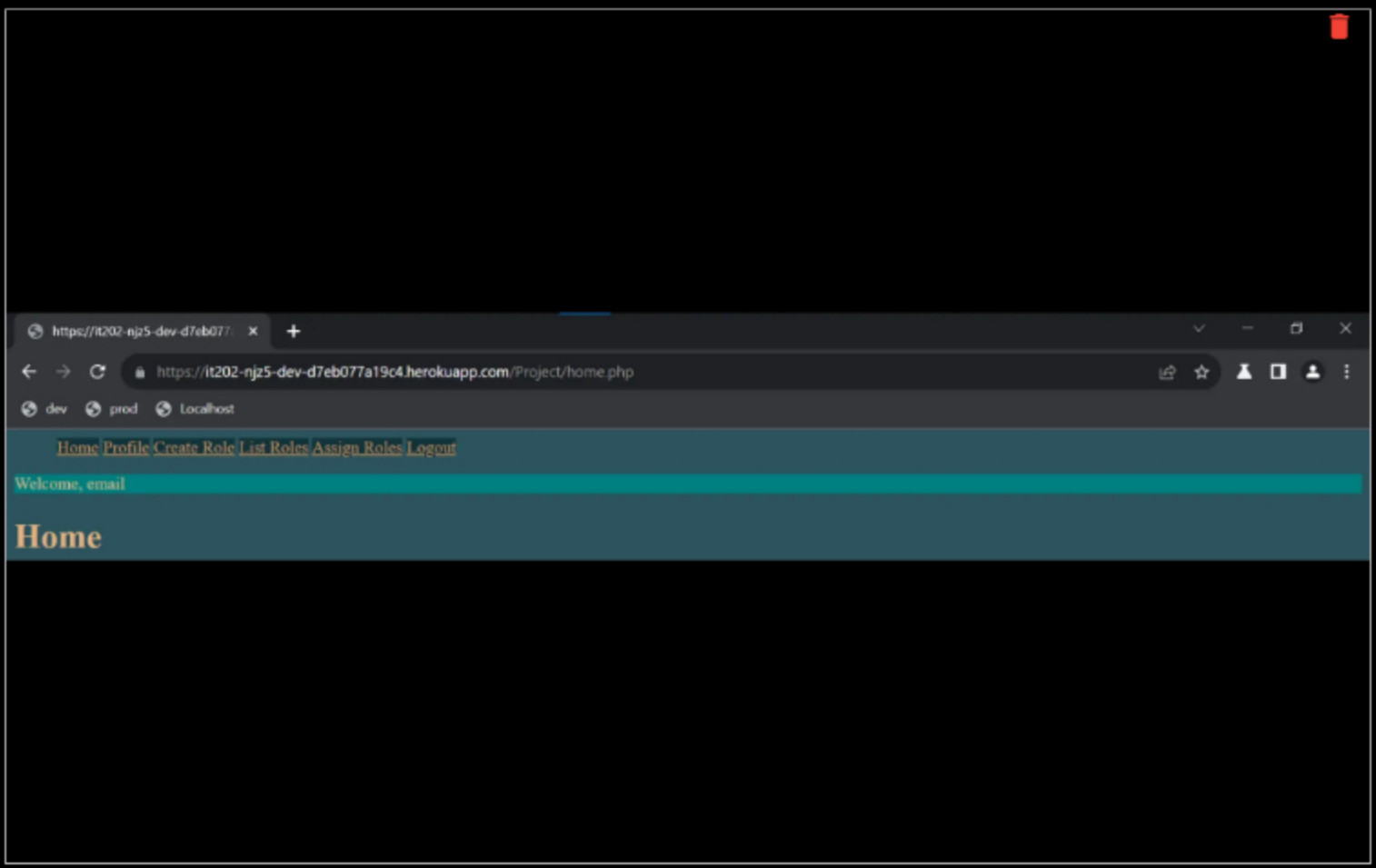
#7 Demonstrate user-friendly message of when an account doesn't exist



Password doesn't match database entry.

Checklist Items (1)

#8 Demonstrate user-friendly message of when password doesn't match what's in the DB

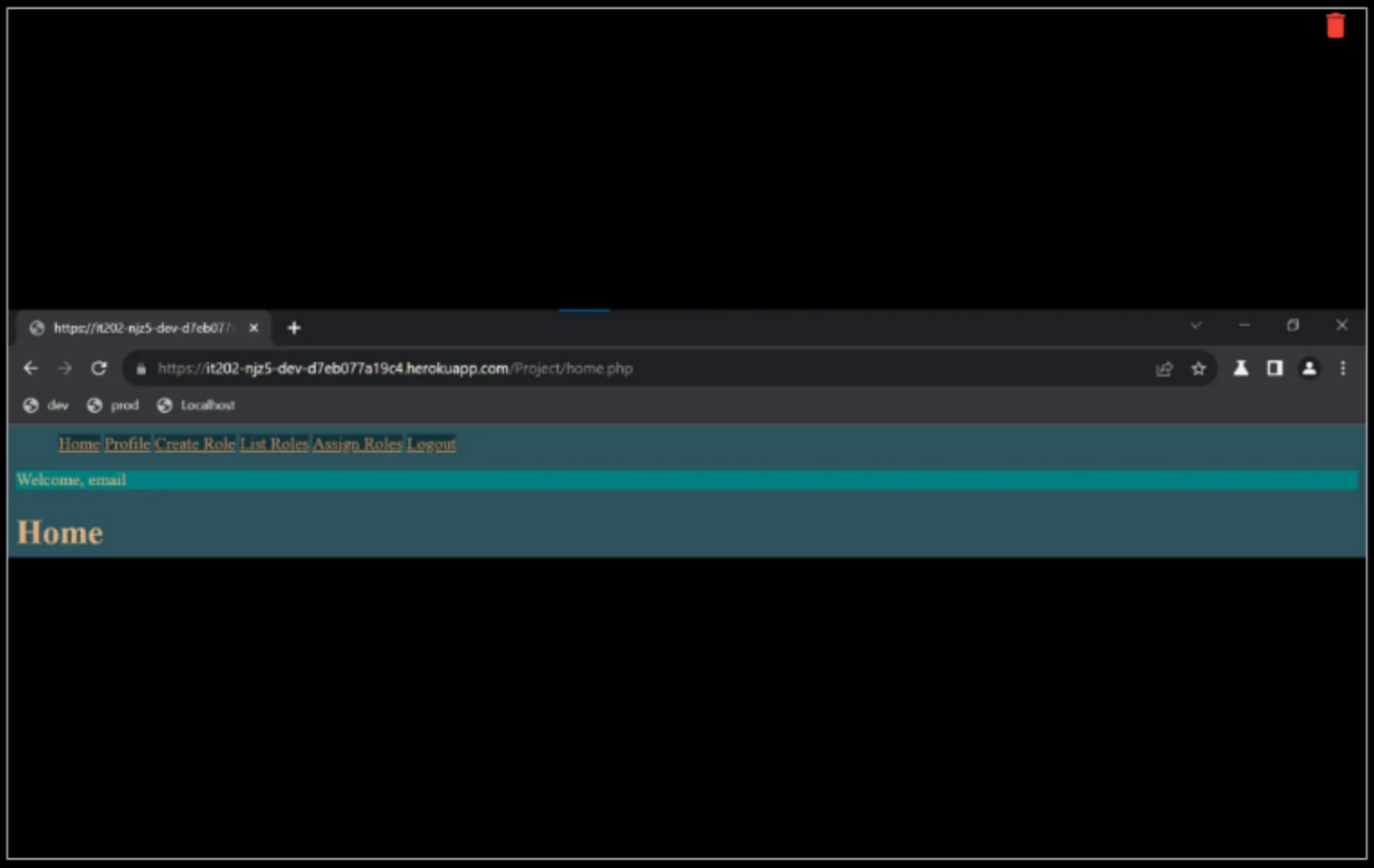


Successful username login.

Checklist Items (2)

#3 Include correct data where username is used to login

#9 Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)

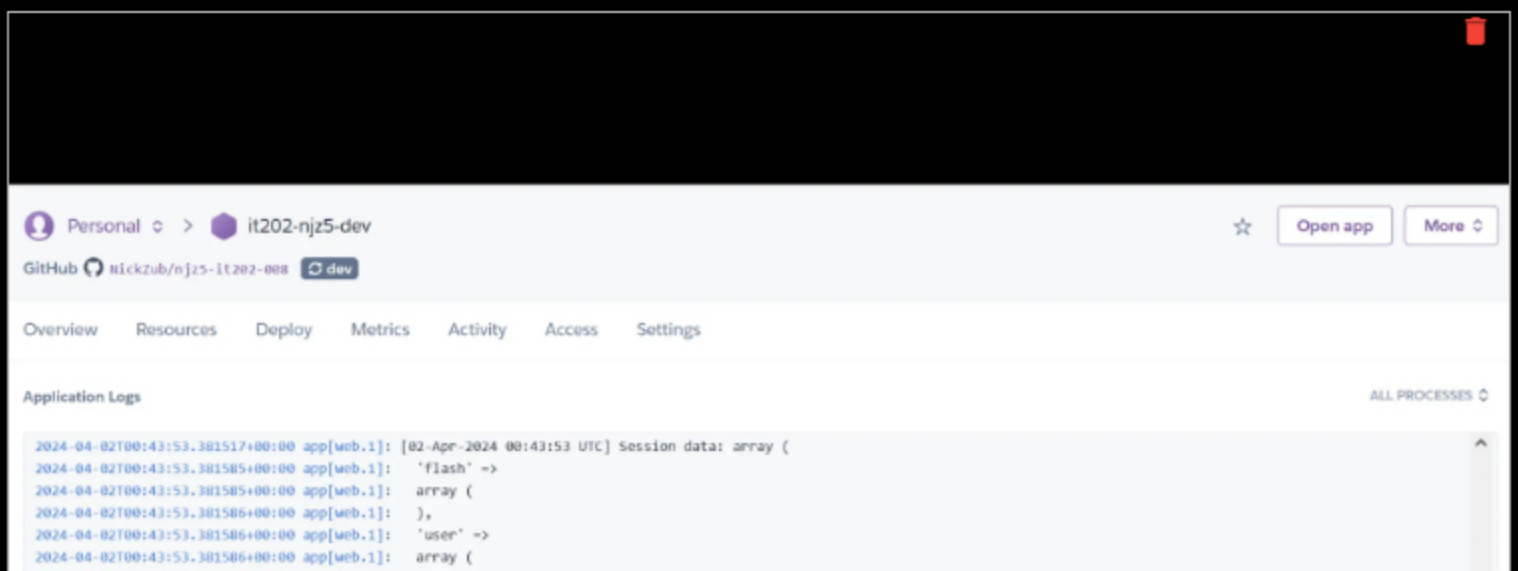


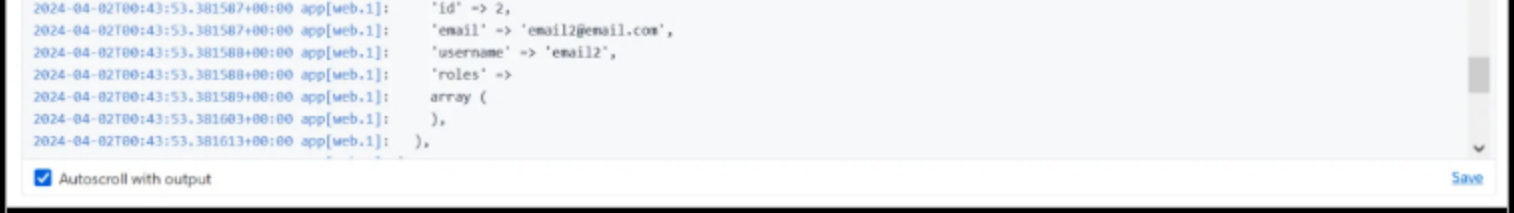
Successful email login (looks the same).

Checklist Items (2)

#4 Include correct data where email is used to login

#9 Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)





Session data in server logs.

Checklist Items (1)

#10 Demonstrate session data being set (captured from server logs)

COLLAPSE

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
3  //NICHOLAS ZUBRZYCKI njz5 4/1/2024
4  ?>
5  <form onsubmit="return validate(this)" method="POST">
6      <div>
7          <label for="email">Email/Username</label>
8          <input type="text" id="email" name="email" required />
9      </div>
10     <div>
11         <label for="pw">Password</label>
12         <input type="password" id="pw" name="password" required minlength="8" />
13     </div>
```

```
    <input type="submit" value="Login" />
</form>
```

Login form code with combined field.

Checklist Items (2)

#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)

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

```
<script>
    function validate(form) {
        //TODO 1: implement JavaScript validation
        //ensure it returns false for an error and true for success
        //NICHOLAS ZUBRZYCKI njz5 4/1/2024
        let input = form.email.value;
        let password = form.pw.value;
        let hasError = false;
        if(input.length === 0){
            flash("[Client] Input must not be empty", "warning");
            hasError = true;
        }
        if(input.includes('@')){
            input = sanitize_email(input);
            if(!is_valid_email(input)){
                flash("[Client] Invalid email address", "warning");
                hasError = true;
            }
        }
        else{
            if(!is_valid_username(input)){
                flash("[Client] Invalid username", "warning");
                hasError = true;
            }
        }
        if(password.length === 0){
            flash("[Client] Password must not be empty", "warning");
            hasError = true;
        }
        if(!is_valid_password(password)){
            flash("[Client] Password too short", "warning");
            hasError = true;
        }
        return !hasError;
    }
</script>
```

JavaScript validation for login.

Checklist Items (1)

#2 Show JavaScript validations (include any extra files related)

```
//NICHOLAS ZUBRZYCKI njz5 4/1/2024
function sanitize_email(email = "")
{
    //Remove all characters except specified, according to PHP documentation.
    return email.replace(/[^a-zA-Z0-9!#$%&'*+-=?^_`{|}~@.[]]/g,'');
}

function is_valid_email(email = "")
{
    //One or more chars, @, One or more chars, ., One or more chars
    let emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
```

```
    if(emailRegex.test(email)){
        return true;
    }
    return false;
}

function is_valid_username(username)
{
    let usernameRegex = /^[a-z0-9_-]{3,16}$/
    if(usernameRegex.test(username)){
        return true;
    }
    return false;
}

function is_valid_password(password)
{
    return password.length >= 8;
}
```

JavaScript validation functions in helpers.js.

Checklist Items (1)

#2 Show JavaScript validations (include any extra files related)

```
<?php
//TODO 2: add PHP Code
//NICHOLAS ZUBRZYCKI njz5 4/1/2024
if (isset($_POST["email"]) && isset($_POST["password"])) {
    $email = se($_POST, "email", "", false);
    $password = se($_POST, "password", "", false);

    //TODO 3
    $hasError = false;
    if (empty($email)) {
        flash("Email must not be empty", "warning");
        $hasError = true;
    }
    if (str_contains($email, "@")) {
        //sanitize
        $email = sanitize_email($email);
        //validate
        if (!is_valid_email($email)) {
            flash("Invalid email address", "warning");
            $hasError = true;
        }
    } else {
        if (!is_valid_username($email)) {
            flash("Invalid username", "warning");
            $hasError = true;
        }
    }
    if (empty($password)) {
        flash("Password must not be empty", "warning");
        $hasError = true;
    }
    if (!is_valid_password($password)) {
        flash("Password too short", "warning");
        $hasError = true;
    }
    if (!$hasError) {
        //flash("Welcome, $email");
        //TODO 4
        $db = getDB();
        $stmt = $db->prepare("SELECT id, email, username, password from Users
        where email = :email or username = :email");
```

PHP validation in login.

Checklist Items (1)

#3 Show PHP validations (include any lib content)

```
<?php
//NICHOLAS ZUBRZYCKI njz5 4/1/2024
```

```php
function sanitize_email($email = "")
{
    return filter_var(trim($email), FILTER_SANITIZE_EMAIL);
}
function is_valid_email($email = "")
{
    return filter_var(trim($email), FILTER_VALIDATE_EMAIL);
}
function is_valid_username($username)
{
    return preg_match('/^[a-z0-9_-]{3,16}$/', $username);
}
function is_valid_password($password)
{
    if($password){
        return strlen($password) >= 8;
    }
}
```

PHP validation functions in sanitizers.php.

Checklist Items (1)

#3 Show PHP validations (include any lib content)

^COLLAPSE ^

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 visiting login.php, an empty page is loaded. After inputting email/username and password, the input values are validated through HTML, JavaScript (client-side), and PHP (server-side) validation. If proper data is submitted, a database query is made to search for the email or username that is inputted. If the query is successful, the passwords are verified and compared, but if the query is unsuccessful, a not found error is returned. Sessions are similar to cookies, except they are stored on the server side rather than client side. Login requires nav and nav starts the session, so the login page is able to access the session data. If proper data is submitted, then the database query result information is stored in the session. This means if a user logs in once, they are able to access other pages using that same account even after closing the tab, without having to login again.

COLLAPSE

Task #4 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/NickZub/njz5-it202-008/pull/13

URL #2
https://github.com/NickZub/njz5-it202-008/pull/14

URL #3
https://github.com/NickZub/njz5-it202-008/pull/16

URL #4
https://github.com/NickZub/njz5-it202-008/pull/19

URL #5
https://github.com/NickZub/njz5-it202-008/pull/21

User Logout (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Capture the following screenshots

Checklist

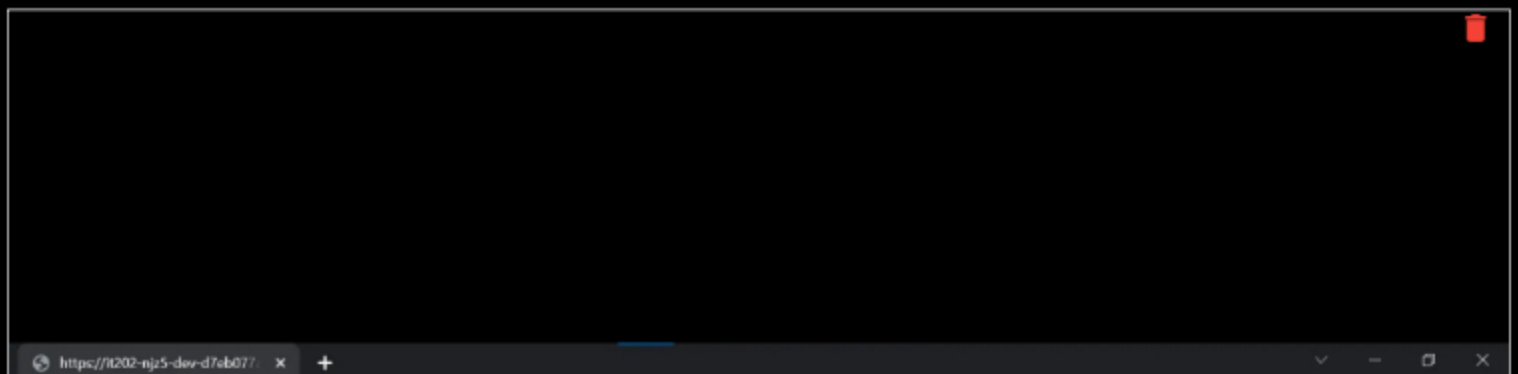
*The checkboxes are for your own tracking

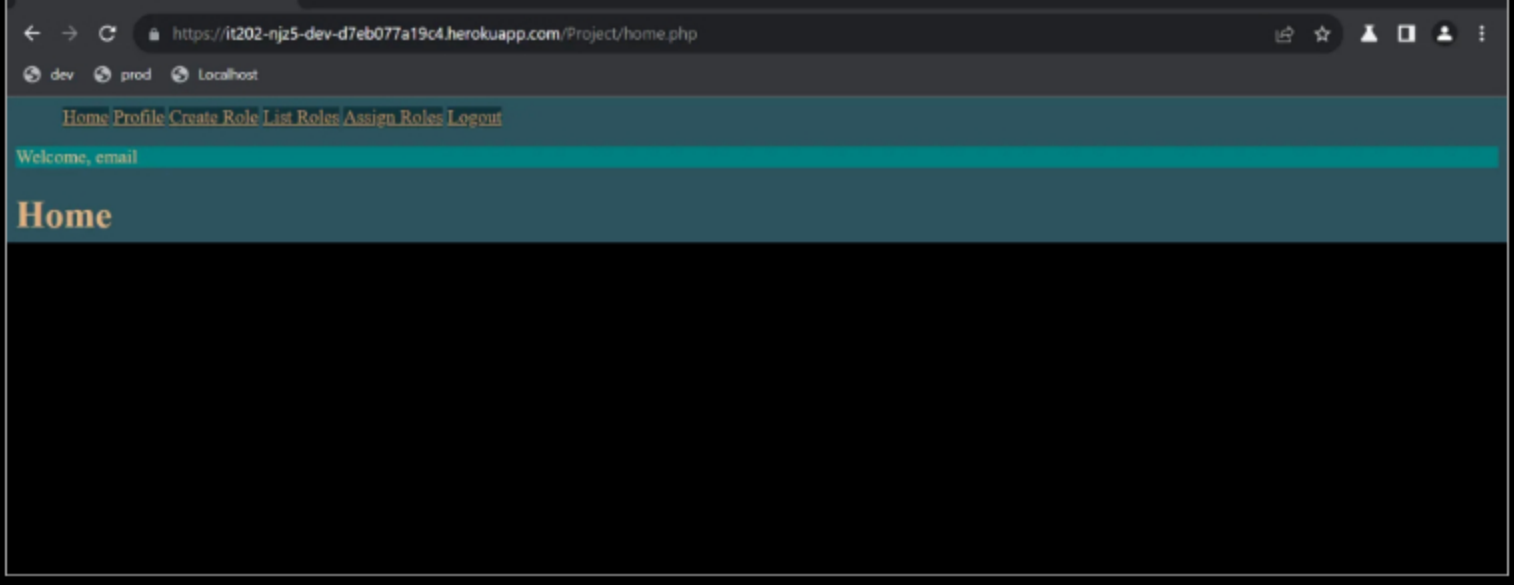
#	Points	Details
#1	1	Screenshot of the navigation when logged in (site)
#2	1	Screenshot of the redirect to login with the user-friendly logged-out message (site)
#3	1	Screenshot of the logout-related code showing the session is destroyed (code). Ensure ucid/date comment is present.

Task Screenshots:

Gallery Style: Large View

Small Medium Large

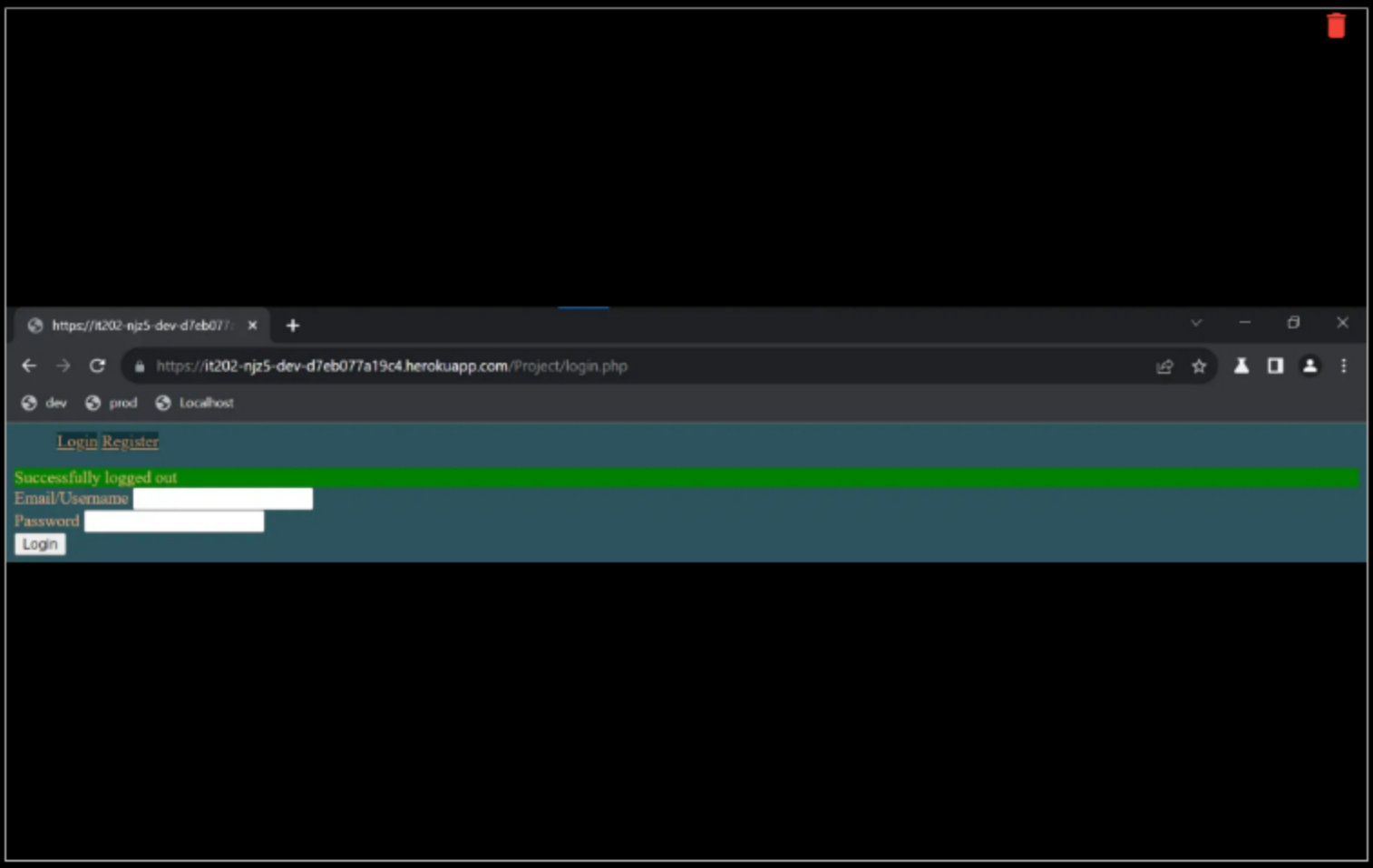




Logged in navigation.

Checklist Items (1)

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



Logout redirect with message.

Checklist Items (1)

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

```php
<?php
session_start();
require(__DIR__ . "/../../lib/functions.php");
reset_session();
//NICHOLAS ZUBRZYCKI njz5 4/1/2024
flash("Successfully logged out", "success");
header("Location: login.php");

```

Logout.php code.

Checklist Items (1)

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

```php
<?php
function reset_session()
{
    session_unset();
    session_destroy();
    session_start();
}
//NICHOLAS ZUBRZYCKI njz5 4/1/2024
```

Reset session function.

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/NickZub/njz5-it202-008/pull/13

URL #2
https://github.com/NickZub/njz5-it202-008/pull/19

COLLAPSE

Basic Security Rules and Roles (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Authentication Screenshots

Checklist

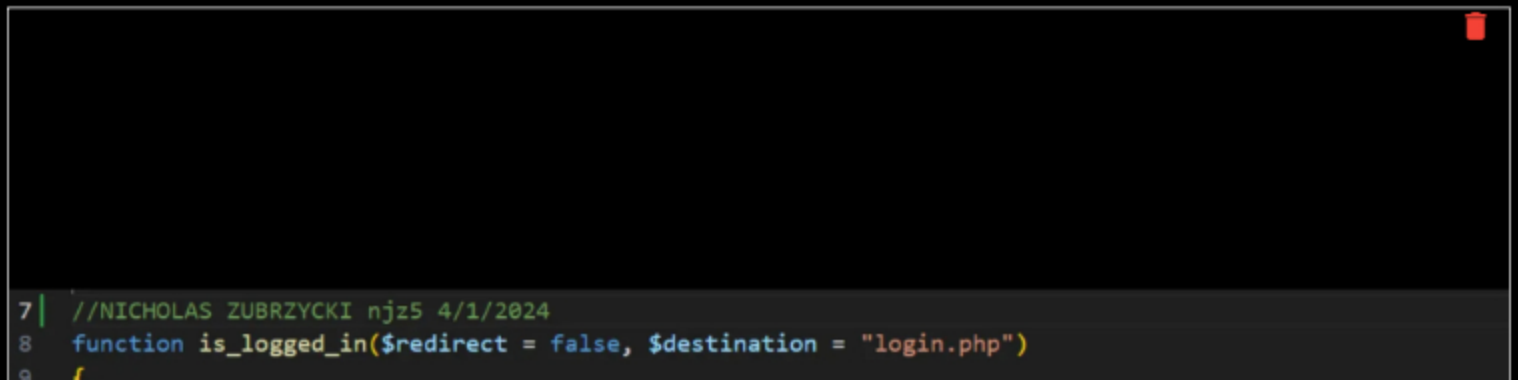
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Screenshot of the function that checks if a user is logged in
#2	1	Screenshot of the login check function being used (i.e., profile likely). Also caption what pages it's used on
#3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)
#4	1	Demonstrate the user-friendly message of trying to manually access a login-protected page while being logged out

Task Screenshots:

Gallery Style: Large View

Small Medium Large



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

Function to check if a user is logged in.

Checklist Items (1)

#1 Screenshot of the function that checks if a user is logged in

```php
<?php
require_once(__DIR__ . "/../../partials/nav.php");
is_logged_in(true);
//NICHOLAS ZUBRZYCKI njz5 4/1/2024
?>
```

Use login function in profile.php.

Checklist Items (2)

#2 Screenshot of the login check function being used (i.e., profile likely). Also caption what pages it's used on

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

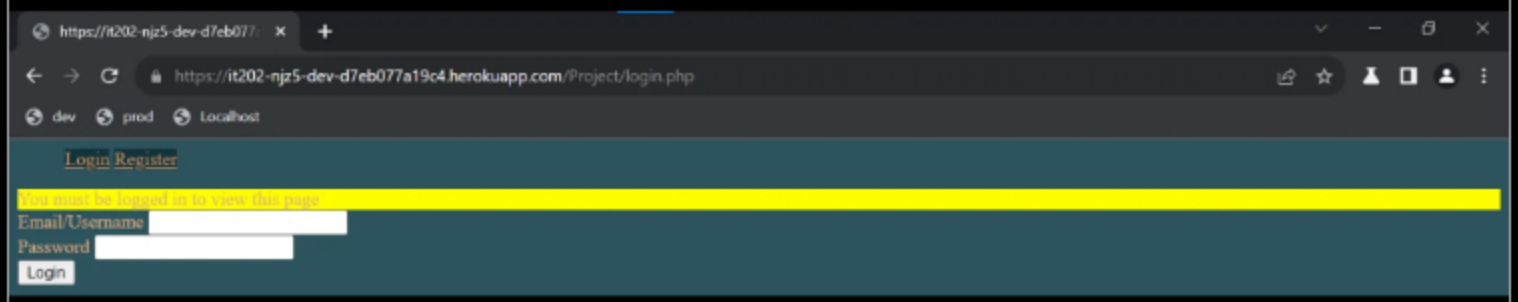
```php
<?php
require(__DIR__."/../../partials/nav.php");
?>
<h1>Home</h1>
<?php
if(is_logged_in(true)){
    error_log("Session data: " . var_export($_SESSION, true));
}
require(__DIR__ . "/../../partials/flash.php");
//NICHOLAS ZUBRZYCKI njz5 4/1/2024
?>
```

Use login function in home.php.

Checklist Items (2)

#2 Screenshot of the login check function being used (i.e., profile likely). Also caption what pages it's used on

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



Trying to access a login-protected page while logged out.

Checklist Items (1)

#4 Demonstrate the user-friendly message of trying to manually access a login-protected page while being logged out

COLLAPSE

Task #2 - Points: 1

Text: Authorization Screenshots

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Screenshot of the function that checks for a specific role
#2	1	Screenshot of the role check function being used. Also caption what pages it's used on
#3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)
#4	1	Demonstrate the user-friendly message of trying to manually access a role-protected page while being logged out

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```php
//NICHOLAS ZUBRZYCKI njz5 4/1/2024
function has_role($role)
{
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {
        foreach ($_SESSION["user"]["roles"] as $r) {
            if ($r["name"] === $role) {
                return true;
            }
        }
    }
    return false;
}
```

Function to check for a specific role.

Checklist Items (1)

#1 Screenshot of the function that checks for a specific role

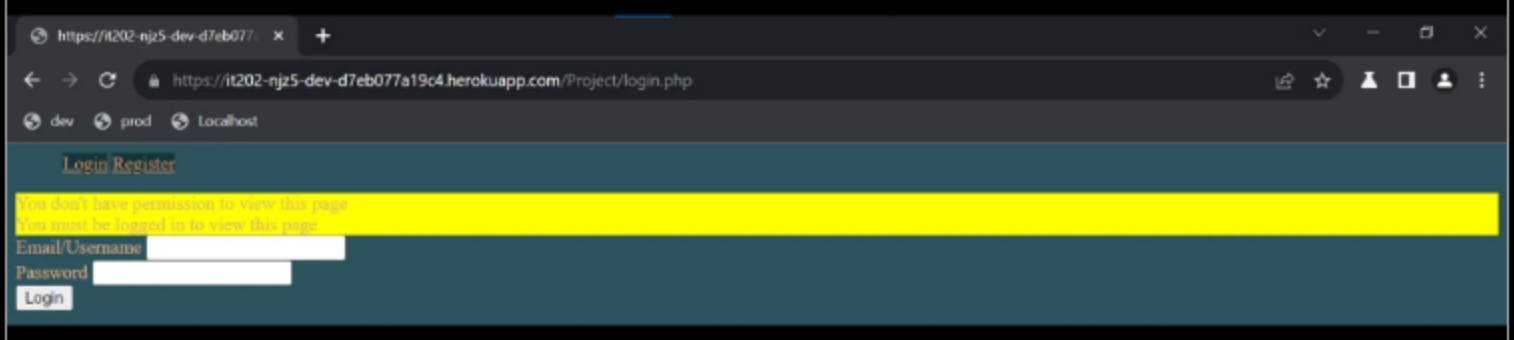
```php
require(__DIR__ . "/../../../partials/nav.php");
//NICHOLAS ZUBRZYCKI njz5 4/1/2024      You, 23 hours ago • Uncommit
if (!has_role("Admin")) {
    flash("You don't have permission to view this page", "warning");
    exit(header("Location: $BASE_PATH" . "/home.php"));
}
```

Role check function being used. It is used on all the admin pages that regular users do not have access to.

Checklist Items (2)

#2 Screenshot of the role check function being used. Also caption what pages it's used on

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



Trying to access a role-protected page while logged out.

Checklist Items (1)

#4 Demonstrate the user-friendly message of trying to manually access a role-protected page while being logged out

COLLAPSE

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

Checklist

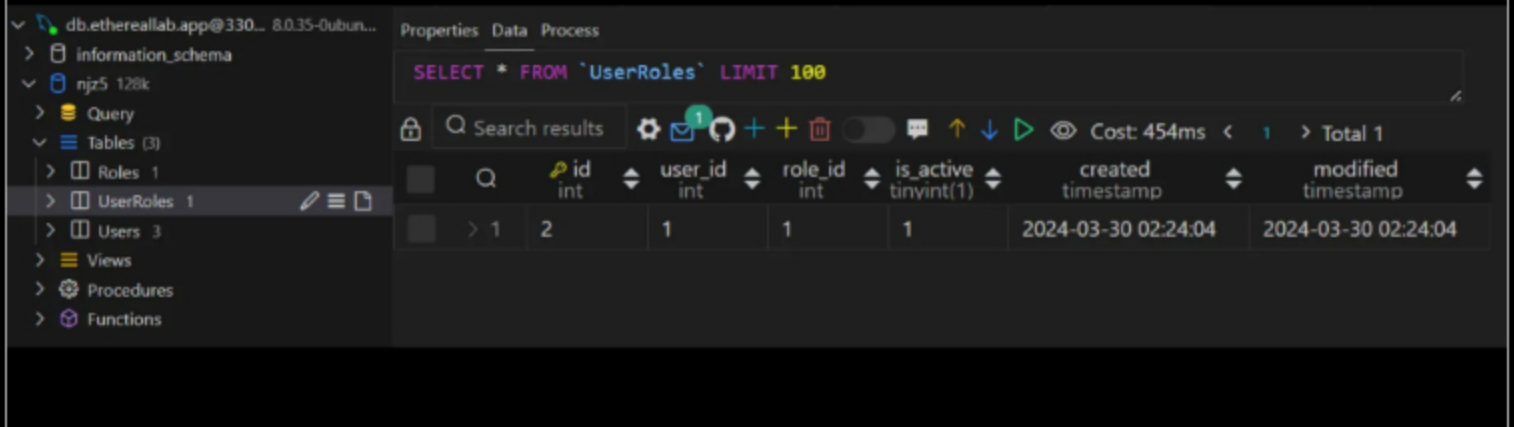
*The checkboxes are for your own tracking

#	Points	Details
#1	1	At least one valid and enabled User->Role reference (UserRoles table)
#2	1	UserRoles Table should have id, user_id, role_id, is_active, created, and modified columns
#3	1	Roles Table should have id, name, description, is_active, modified, and created columns
#4	1	At least one valid and enabled Role (Roles table)
#5	1	Ensure left panel or database name is present in each table screenshot (should contain your ucid)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



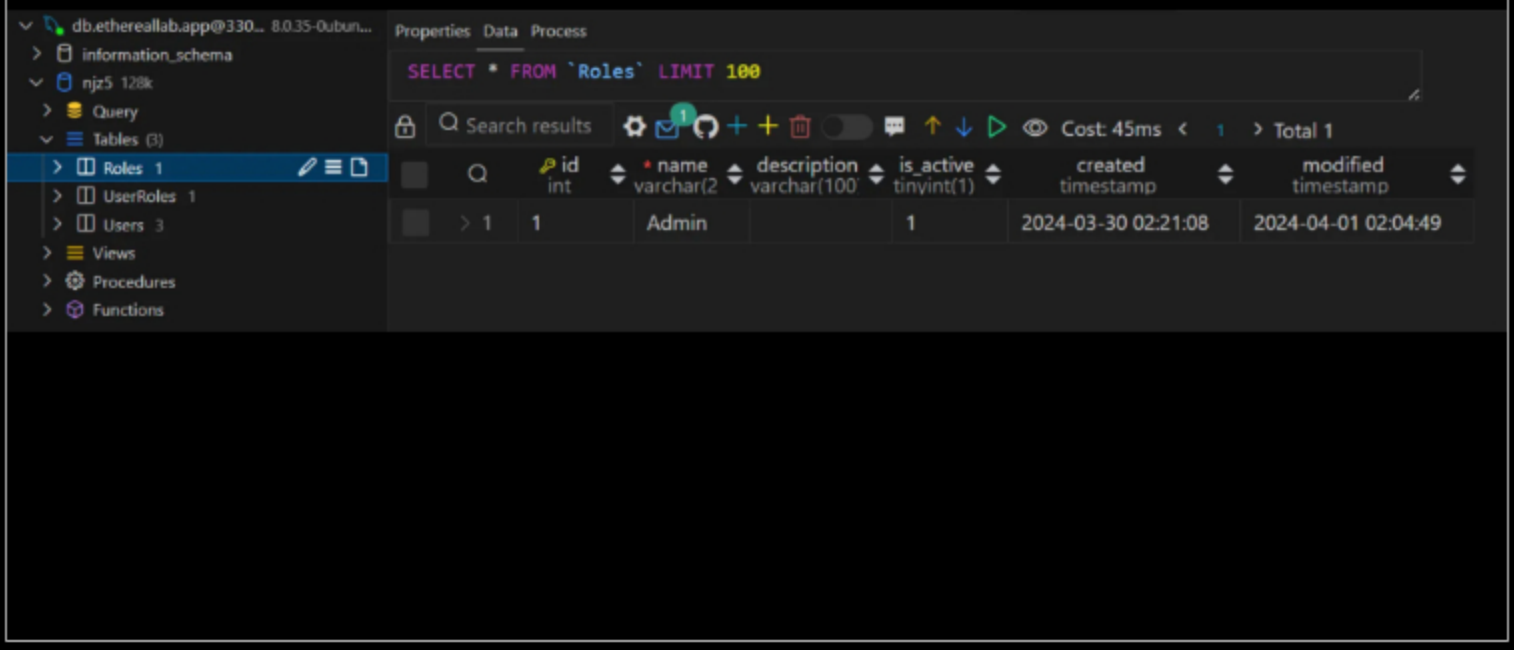
UserRoles table.

Checklist Items (3)

#1 At least one valid and enabled User->Role reference (UserRoles table)

#2 UserRoles Table should have id, user_id, role_id, is_active, created, and modified columns

#5 Ensure left panel or database name is present in each table screenshot (should contain your ucid)



Roles table.

Checklist Items (3)

#3 Roles Table should have id, name, description, is_active, modified, and created columns

#4 At least one valid and enabled Role (Roles table)

#5 Ensure left panel or database name is present in each table screenshot (should contain your ucid)

Task #4 - Points: 1

COLLAPSE

Task #4 - Points: 1

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

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	What's the purpose of the UserRoles table?
#2	1	How does Roles.is_active differ from UserRoles.is_active?

Response:

The UserRoles table is used to connect the users table and roles table together, resulting in the association between users and the respective roles they have. This allows you to easily identify and query what roles a specific user has. Roles.is_active is a global toggle for the role and affects every single user; UserRoles.is_active is a local toggle for the role and affects only that single user.

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/NickZub/njz5-it202-008/pull/20

User Profile (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: View Profile Website Page

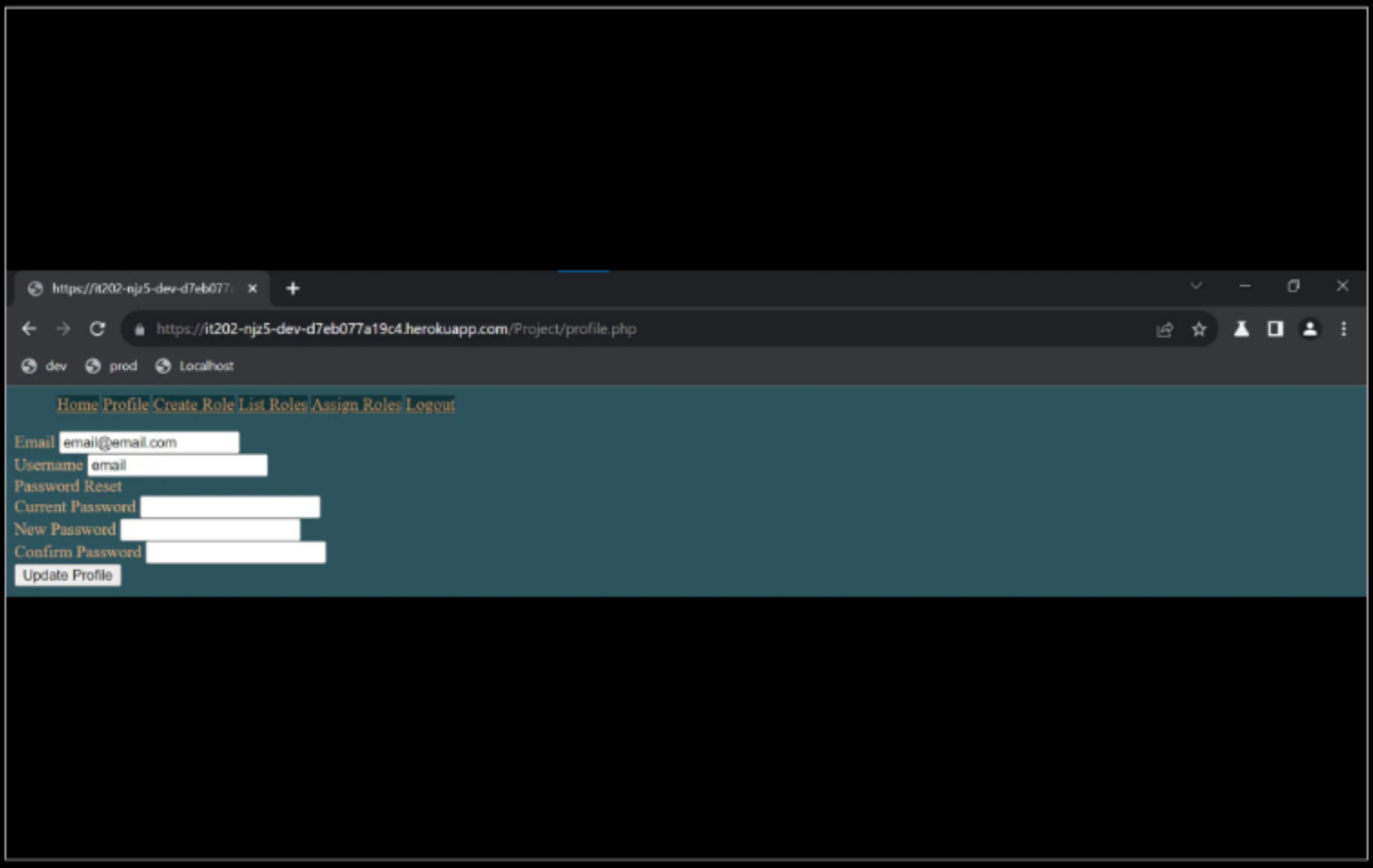
Checklist *The checkboxes are for your own tracking

#	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.

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:

The functions get_username and get_user_email in user_helpers.php are used to retrieve the values of email and

username from the session data. After retrieving these values in profile.php, the value attributes for the email and username fields in the form are populated using the safer echo of their values. This preloads their data so you can see it when the form loads.

COLLAPSE

Task #3 - Points: 1

Text: Edit Profile Website Page

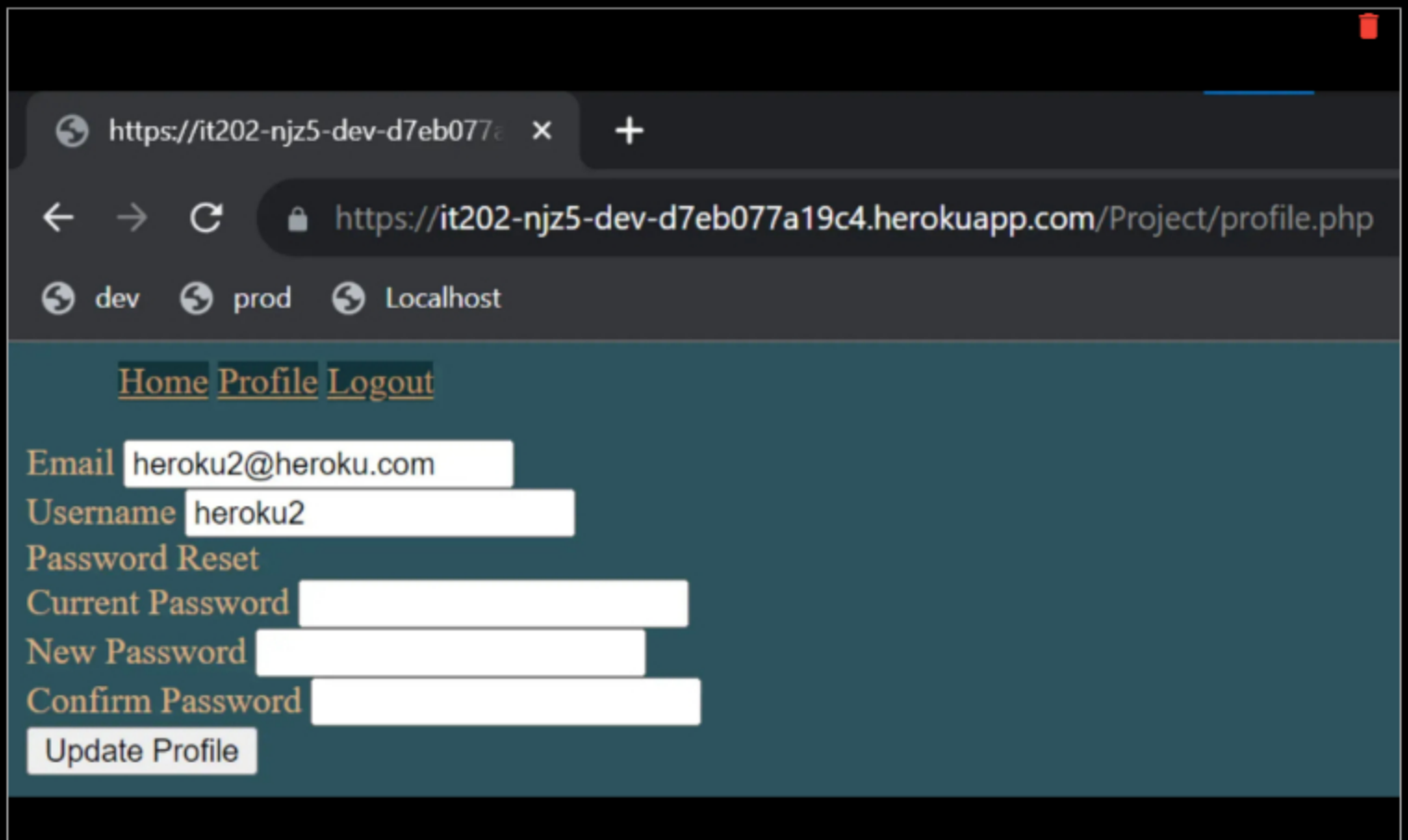
Checklist *The checkboxes are for your own tracking

#	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

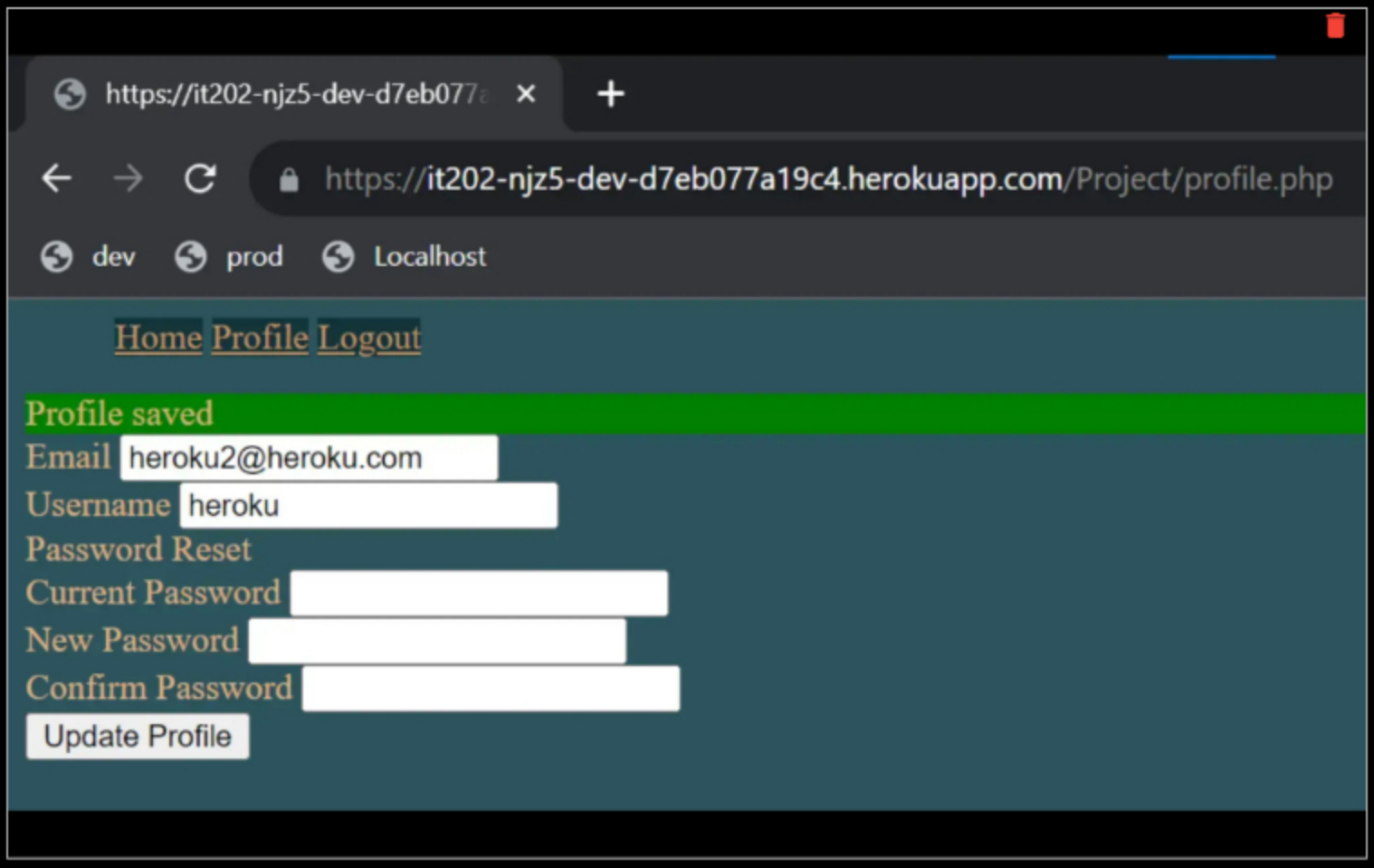
Small Medium Large



Before username change.

Checklist Items (1)

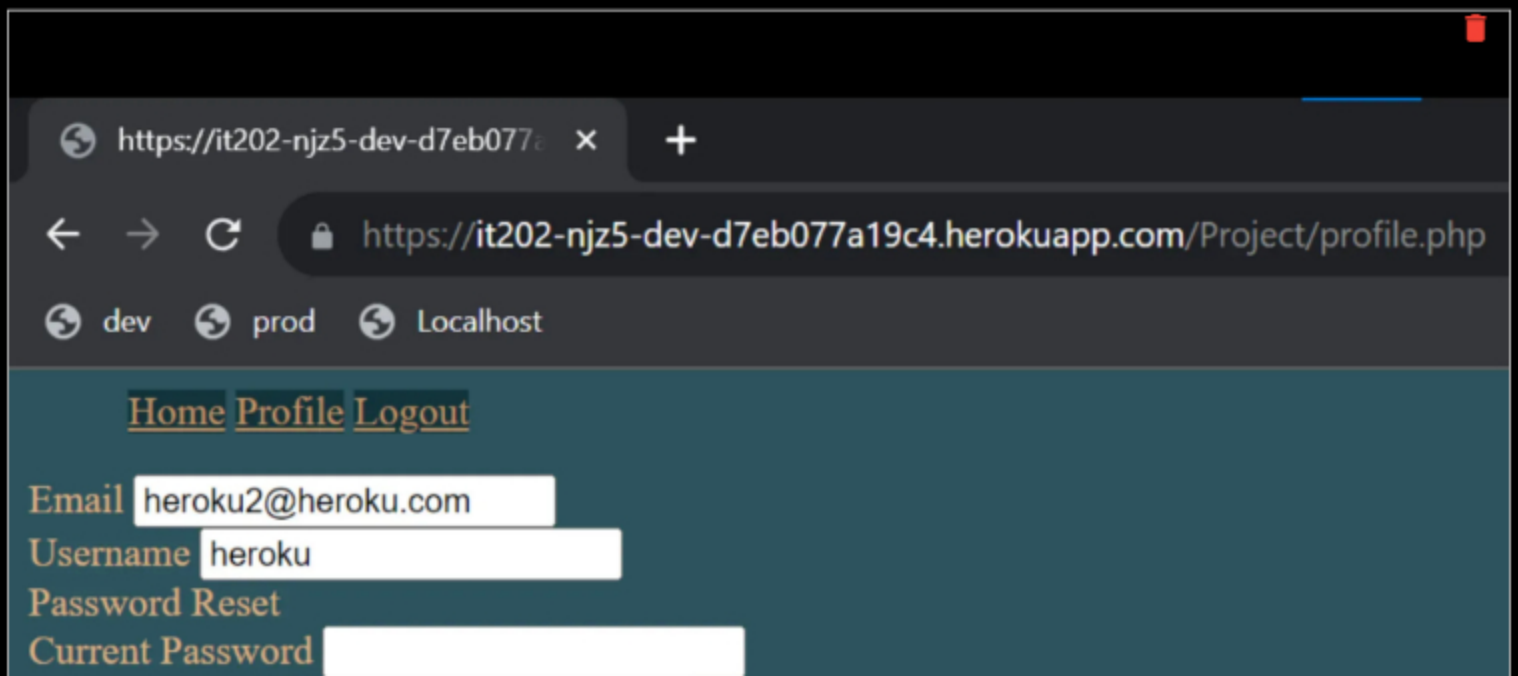
#3 Demonstrate with before and after of a username change (including success message)



After username change.

Checklist Items (1)

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



New Password

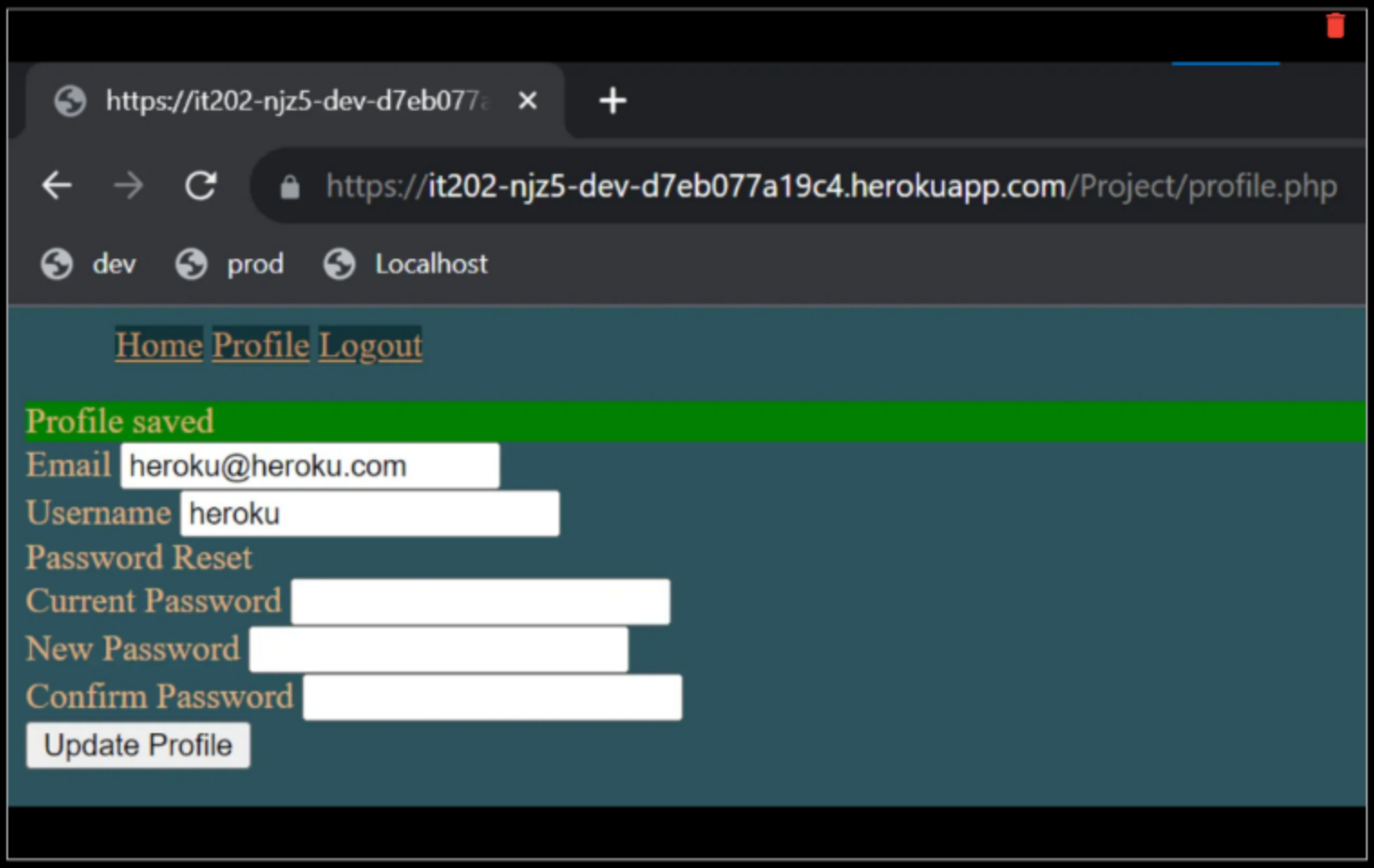
Confirm Password

Update Profile

Before email change.

Checklist Items (1)

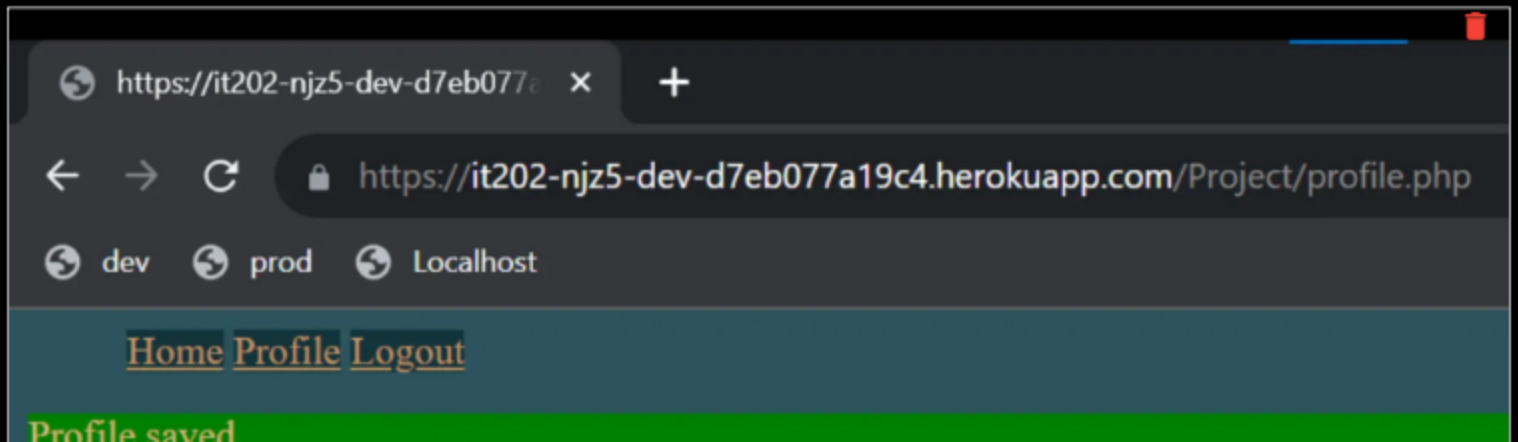
#4 Demonstrate with a before and after of an email change (including success message)



After email change.

Checklist Items (1)

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



Password reset

Email heroku@heroku.com

Username heroku

Password Reset

Current Password

New Password

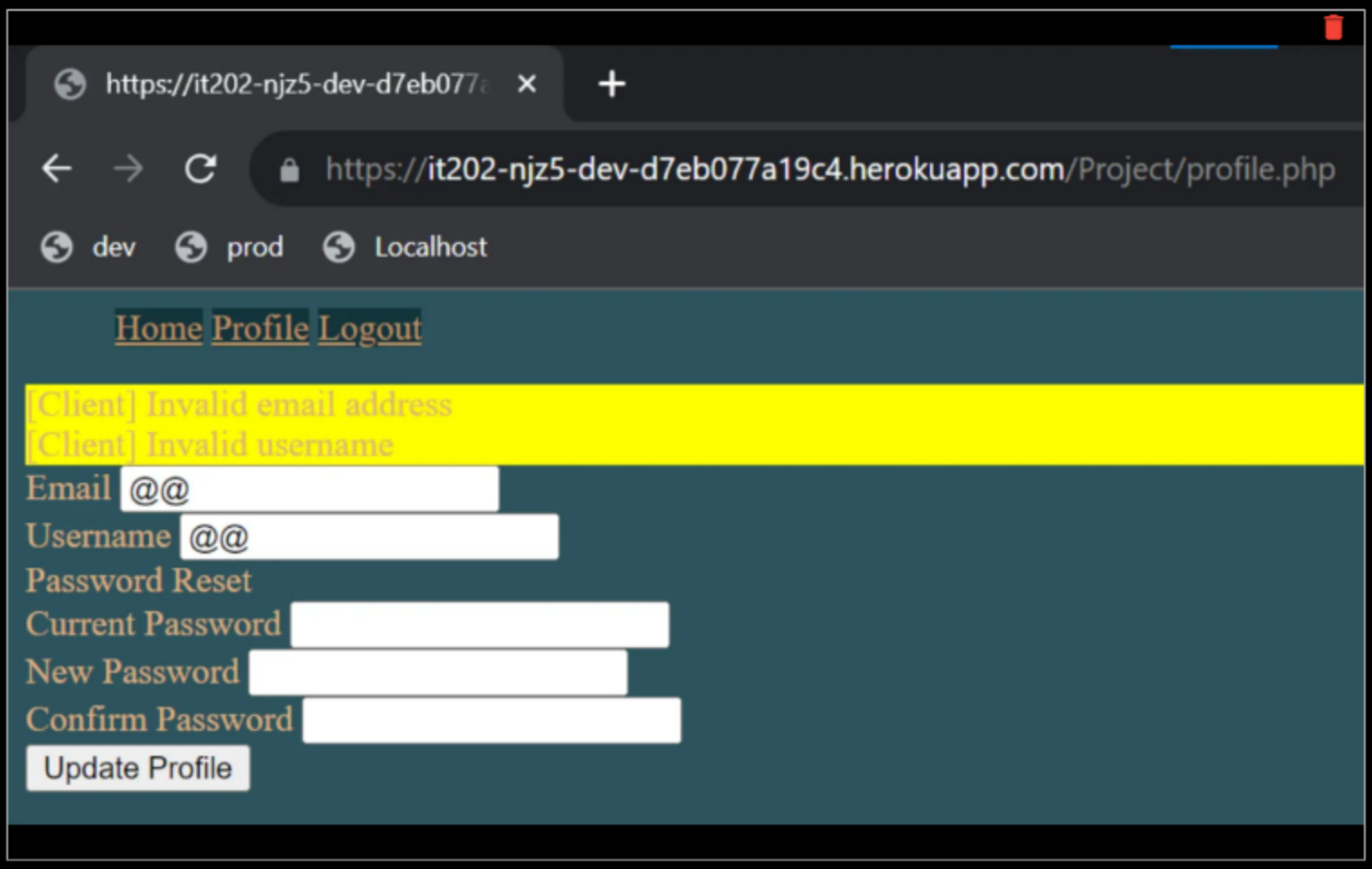
Confirm Password

Update Profile

Successfully updated password.

Checklist Items (1)

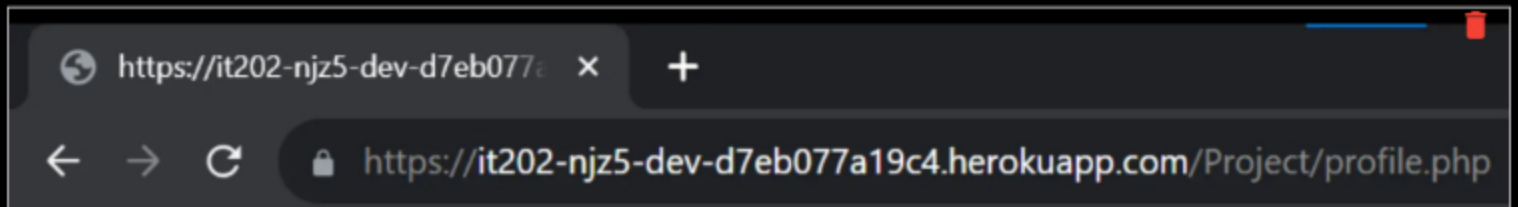
#5 Demonstrate the success message of updating password

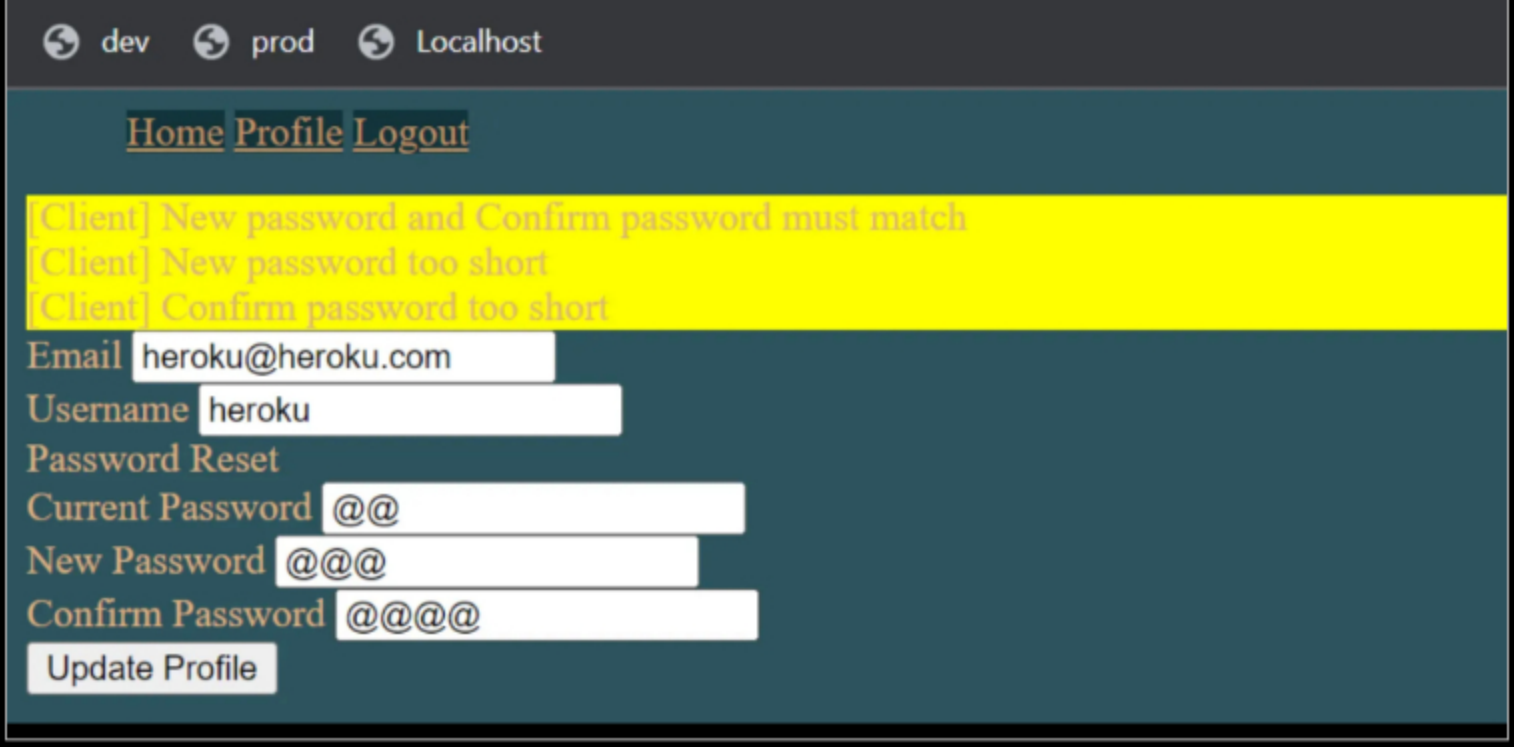


JavaScript email and username validation.

Checklist Items (1)

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

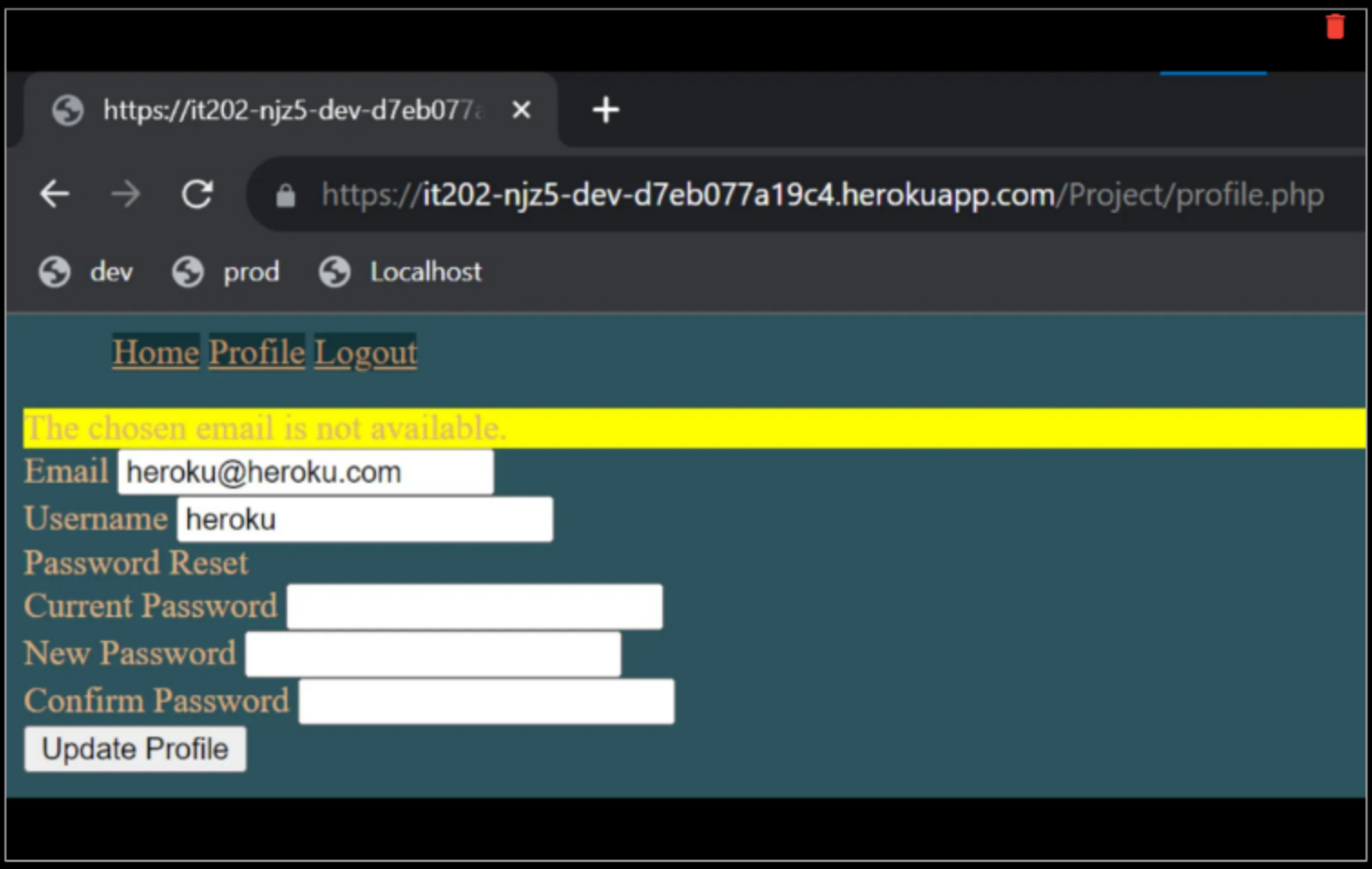




JavaScript password validation.

Checklist Items (1)

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

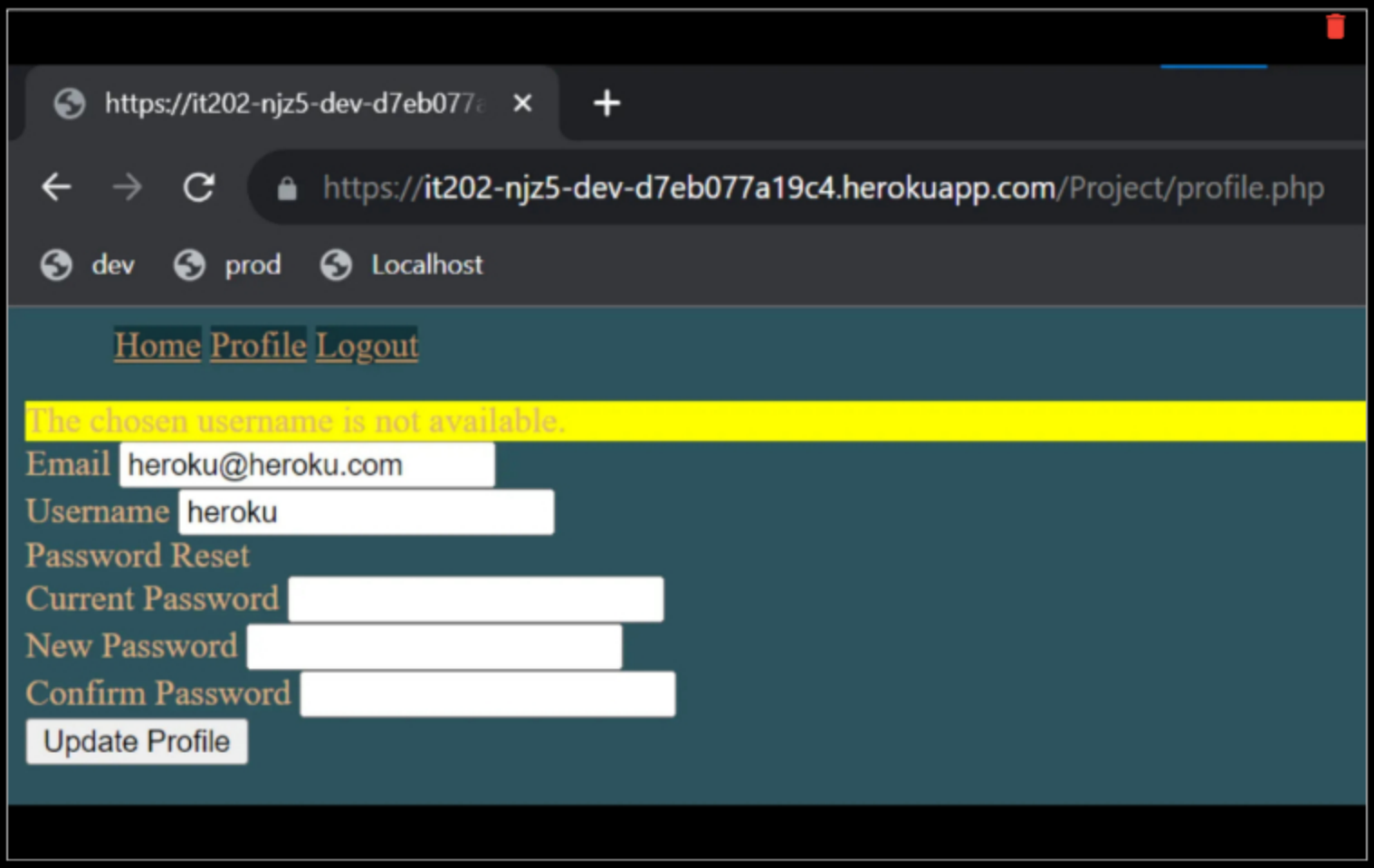


PHP email in use.

Checklist Items (1)

#7 Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already

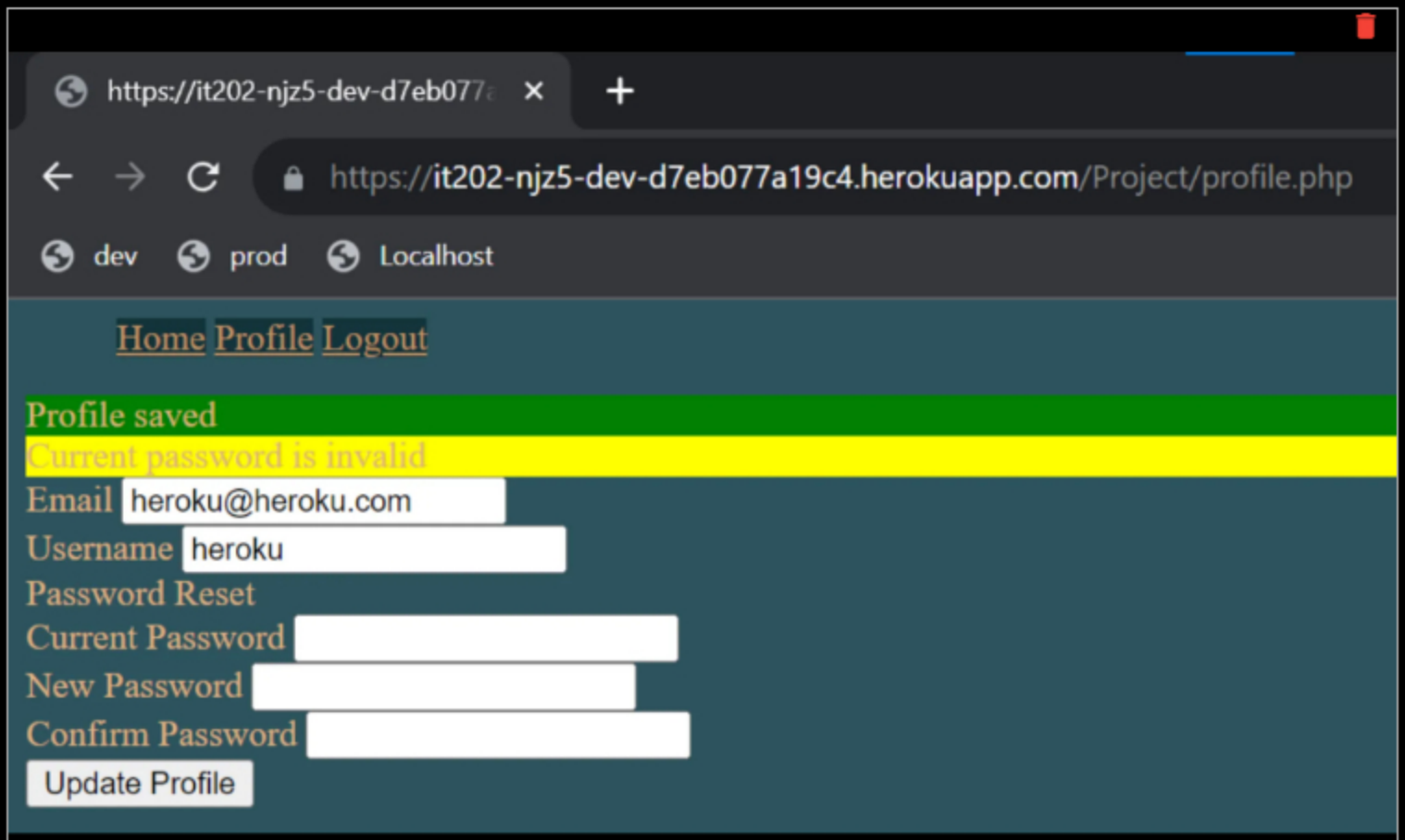
in use, current password doesn't match what's in the DB)



PHP username in use.

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)



PHP current password does not match database.

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

#	Points	Details
#1	1	Updating Username/Email
#2	1	Updating password
#3	1	Don't just show code, translate things to plain English

Response:

1. Username and email are checked for validity according to HTML, JavaScript, then PHP validations. If each passes, then an update statement with the new email and username parameters is queried against the database to change the current data. It uses the get_user_id() function to only change the data for that respective user.

2. If any value is entered for the passwords, the current password and new passwords are checked for validity using HTML, JavaScript, and PHP validations. If those pass, then the password verify function checks to see if the current password field value matches the password value in the database for that user id, which is done on the PHP server side. If it does, then the password is changed using an update statement.

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/NickZub/njz5-it202-008/pull/34

URL #2
https://github.com/NickZub/njz5-it202-008/pull/35

URL #3
https://github.com/NickZub/njz5-it202-008/pull/36

URL #4
https://github.com/NickZub/njz5-it202-008/pull/16

URL #5

https://github.com/NickZub/njz5-it202-008/pull/19

Misc (1 pt.)

COLLAPSE

COLLAPSE

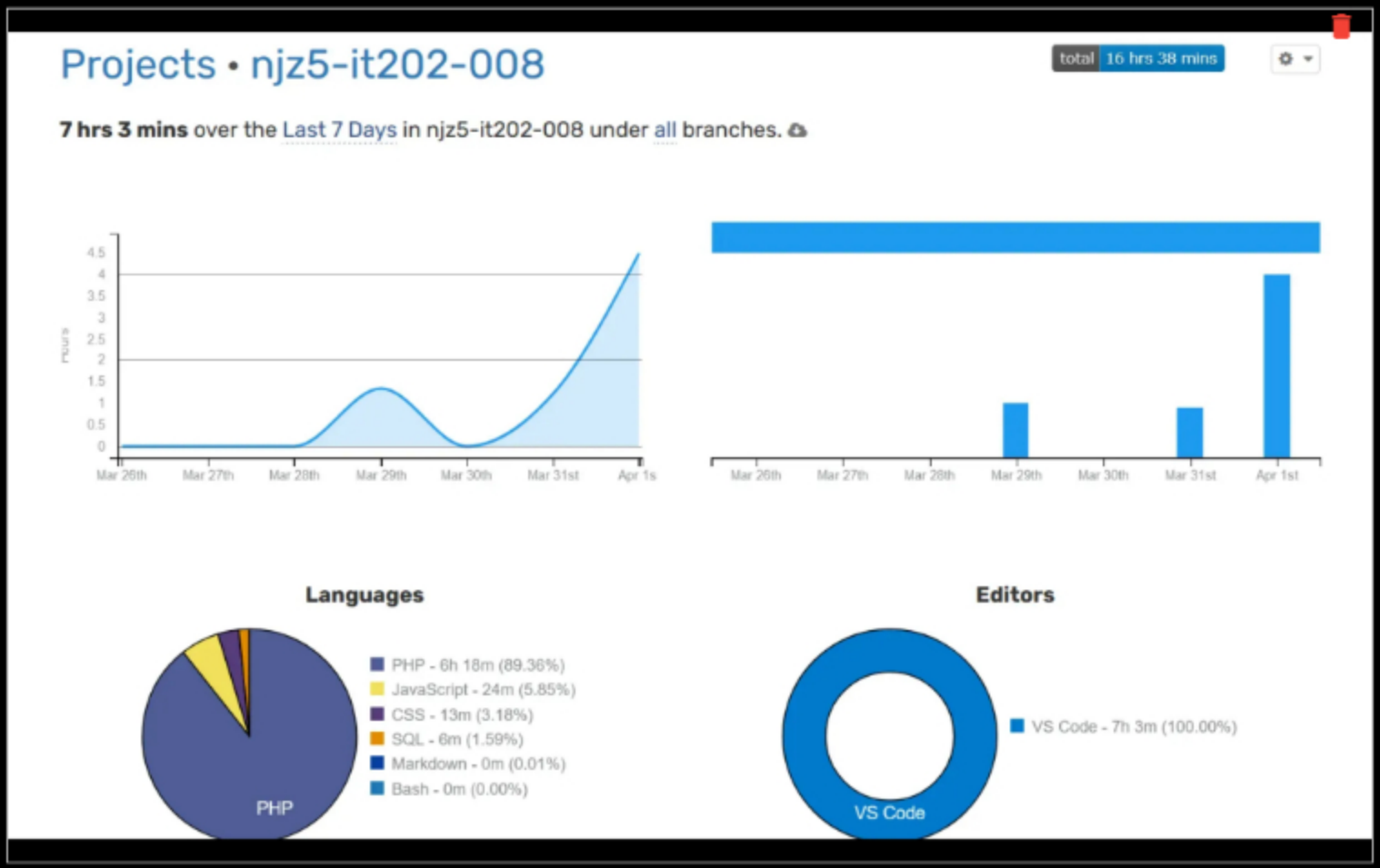
Task #1 - Points: 1

Text: Screenshot of wakatime

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Waka.

Waka 2.

Files

1 hr 51 mins	public_html/Project/login.php
1 hr 41 mins	...ic_html/Project/profile.php
33 mins	...c_html/Project/register.php
20 mins	lib/functions.php
19 mins	partials/nav.php
14 mins	lib/sanitizers.php
14 mins	lib/duplicate_user_details.php
14 mins	lib/sanitizers.js
13 mins	public_html/Project/home.php
13 mins	public_html/Project/styles.css
12 mins	lib/user_helpers.php
8 mins	public_html/Project/logout.php
8 mins	public_html/Project/helpers.js
6 mins	lib/reset_session.php
5 mins	public_html/index.php
5 mins	...roject/admin/list_roles.php
4 mins	.../005_insert_roles_admin.sql
2 mins	...oject/admin/create_role.php
2 mins	...ject/admin/assign_roles.php
2 mins	..._html/Project/sanitizers.js
1 min	partials/flash.php
1 min	lib/get_url.php
1 min	..._create_table_userroles.sql
32 secs	.../003_create_table_roles.sql
7 secs	lib/db.php
5 secs	lib/config.php

Branches

2 hrs 47 mins	Milestone1
2 hrs 25 mins	MS1-Feat-JavaScriptValidation
58 mins	MS1-InClass-Housekeeping
51 mins	MS1-Feat-UserRoles

Waka 3.

COLLAPSE

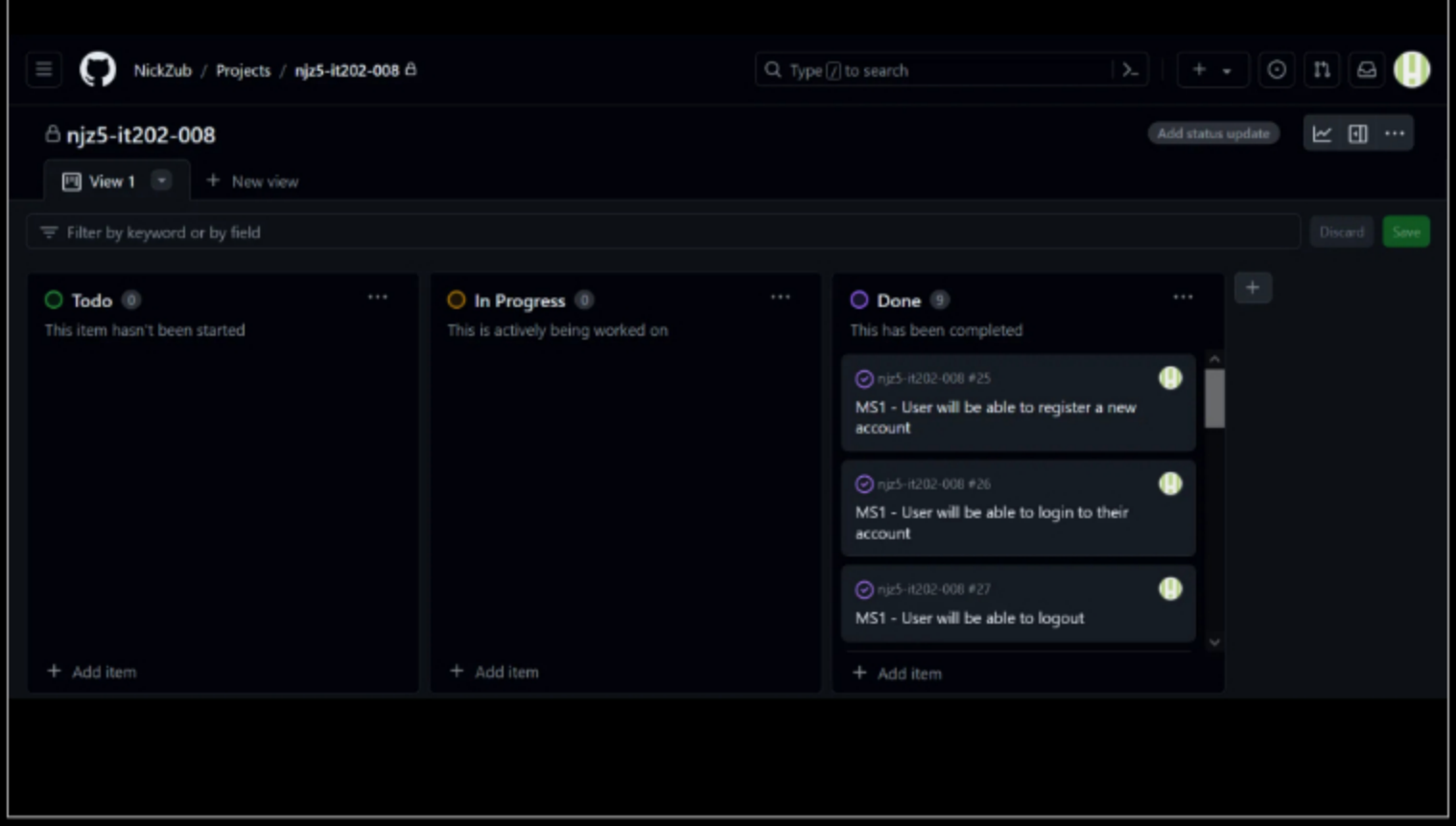
Task #2 - Points: 1

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Project board.

COLLAPSE

Task #3 - Points: 1

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

URL #1

https://github.com/users/NickZub/projects/2/views/1

COLLAPSE

Task #4 - Points: 1

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

URL #1

https://it202-njz5-prod-bedd6fae57b2.herokuapp.com/Project/login.php

End of Assignment