Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/df39

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

Submissions:

Submission Selection

1 Submission [active] 4/24/2024 7:42:41 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

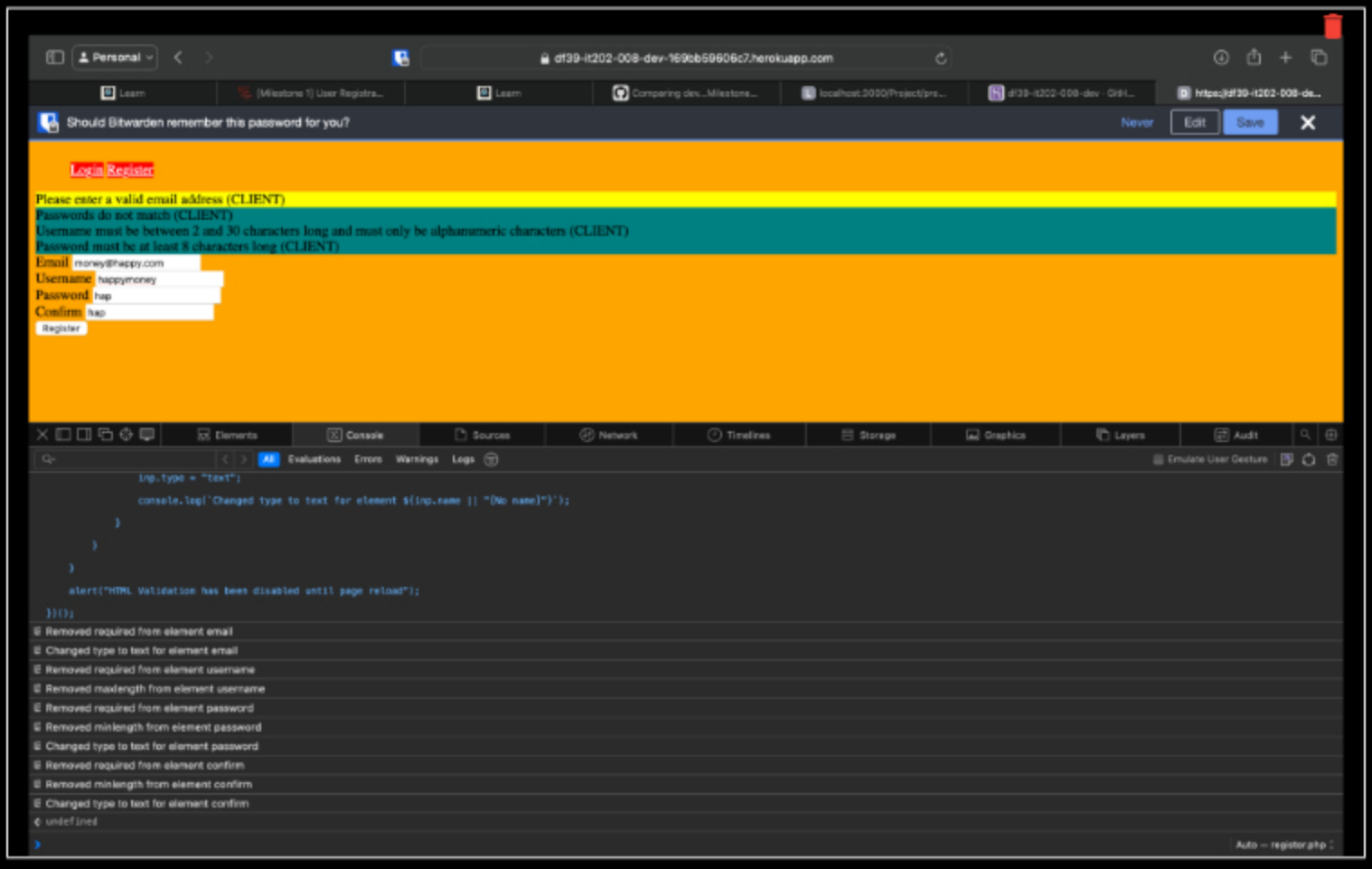
*The checkboxes are for your own tracking

#	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

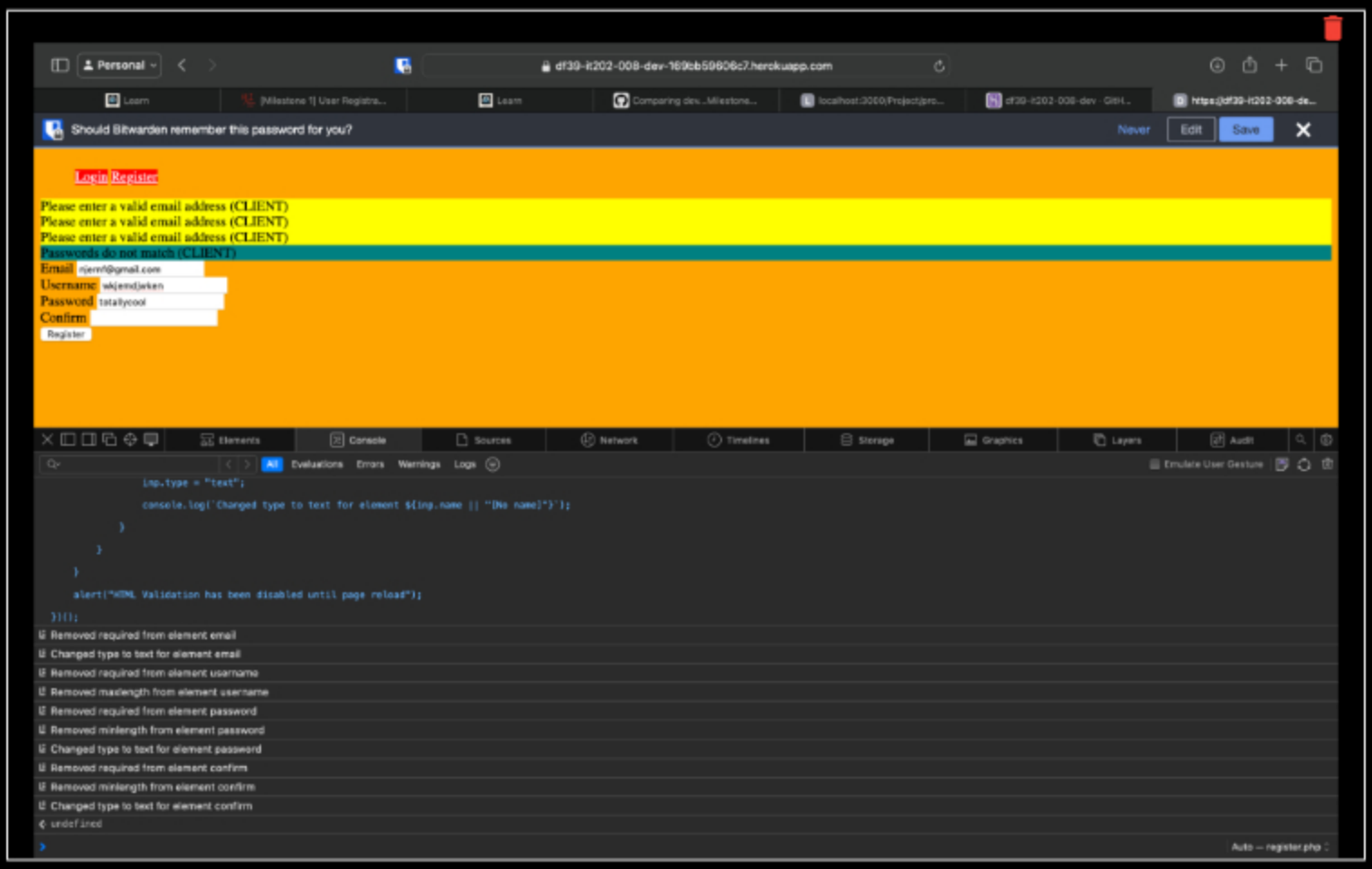
Small Medium Large



Javascript Validation of email format, username format, passwords matching and password being longer than 8 characters.

Checklist Items (1)

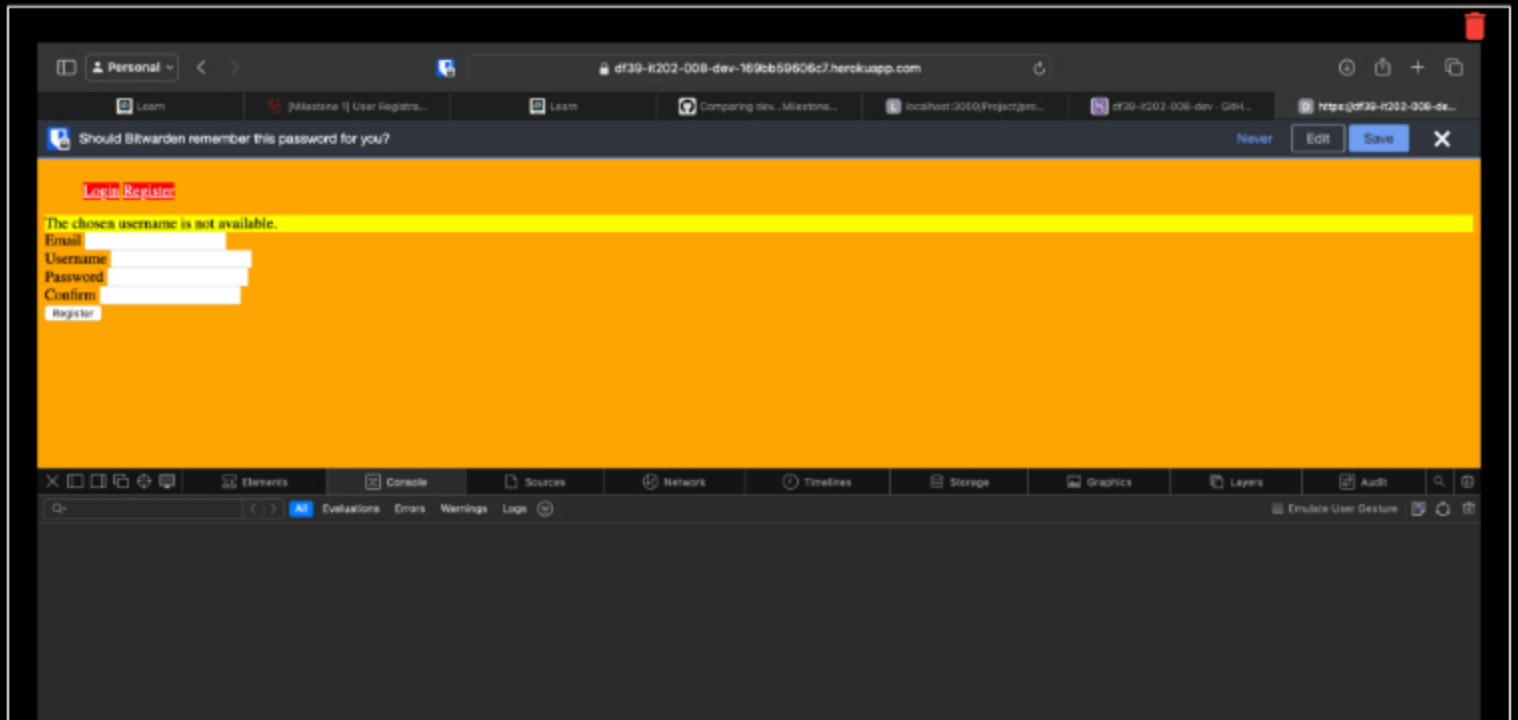
#3 Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required

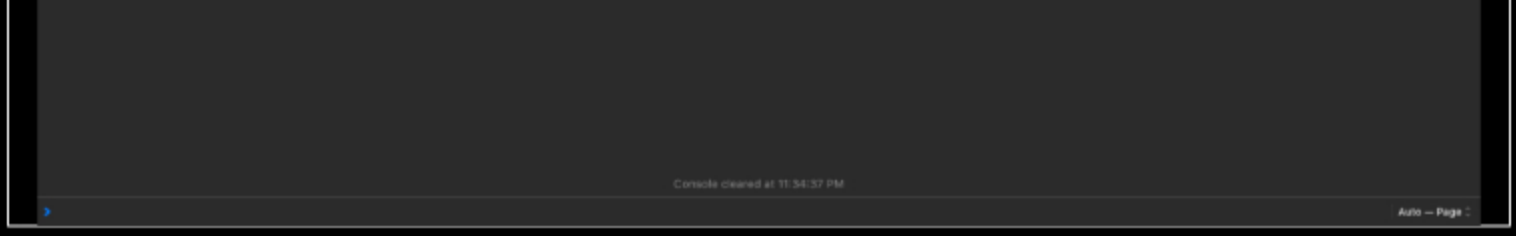


This shows that each field must be required.

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

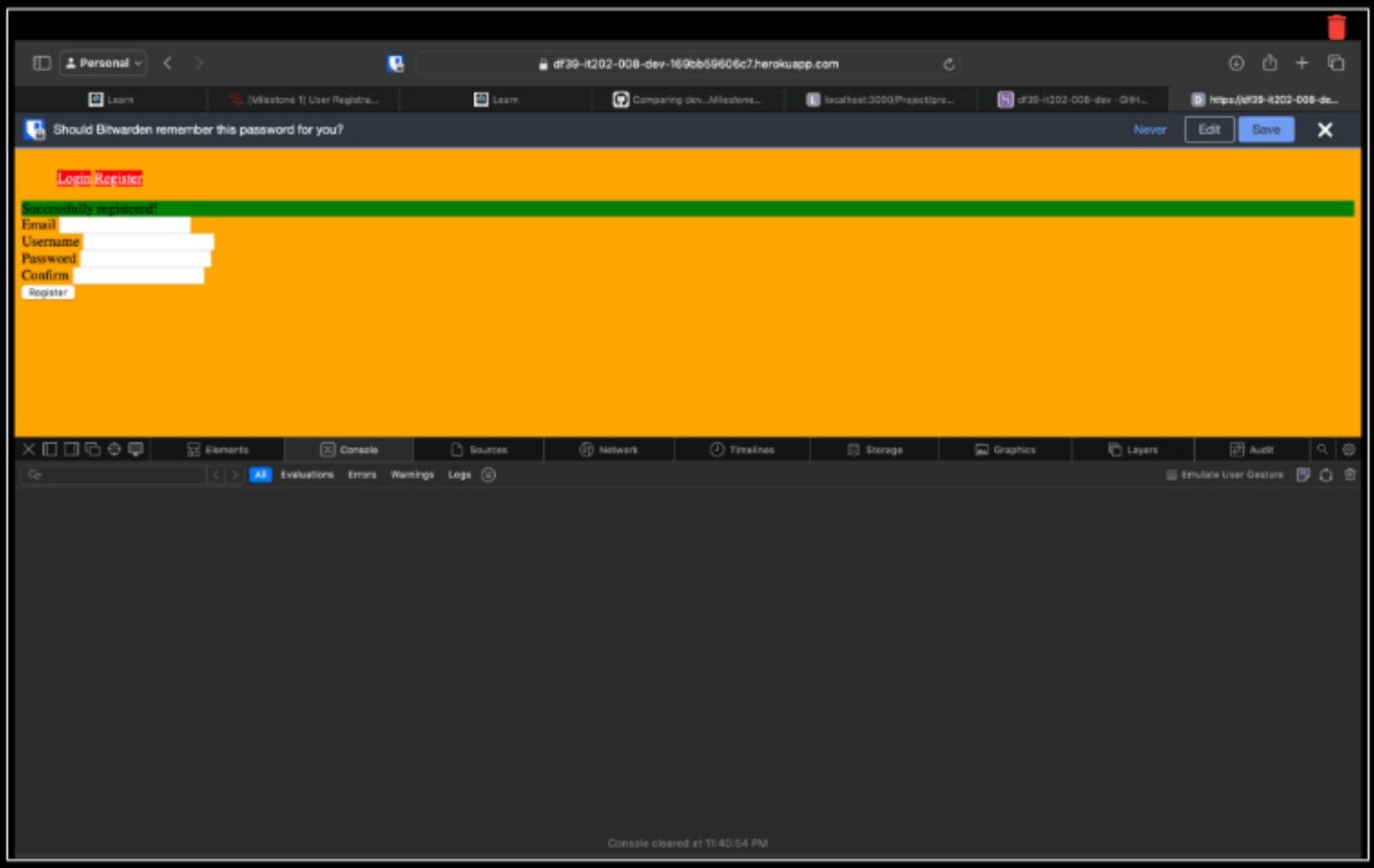




This demonstrates username already In use message.

Checklist Items (1)

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



This shows user-friendly message of new account being made, also shows CSS applied to site, and heroku dev deployment link in URL bar.

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

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

COLLAPSE

Task #2 - Points: 1

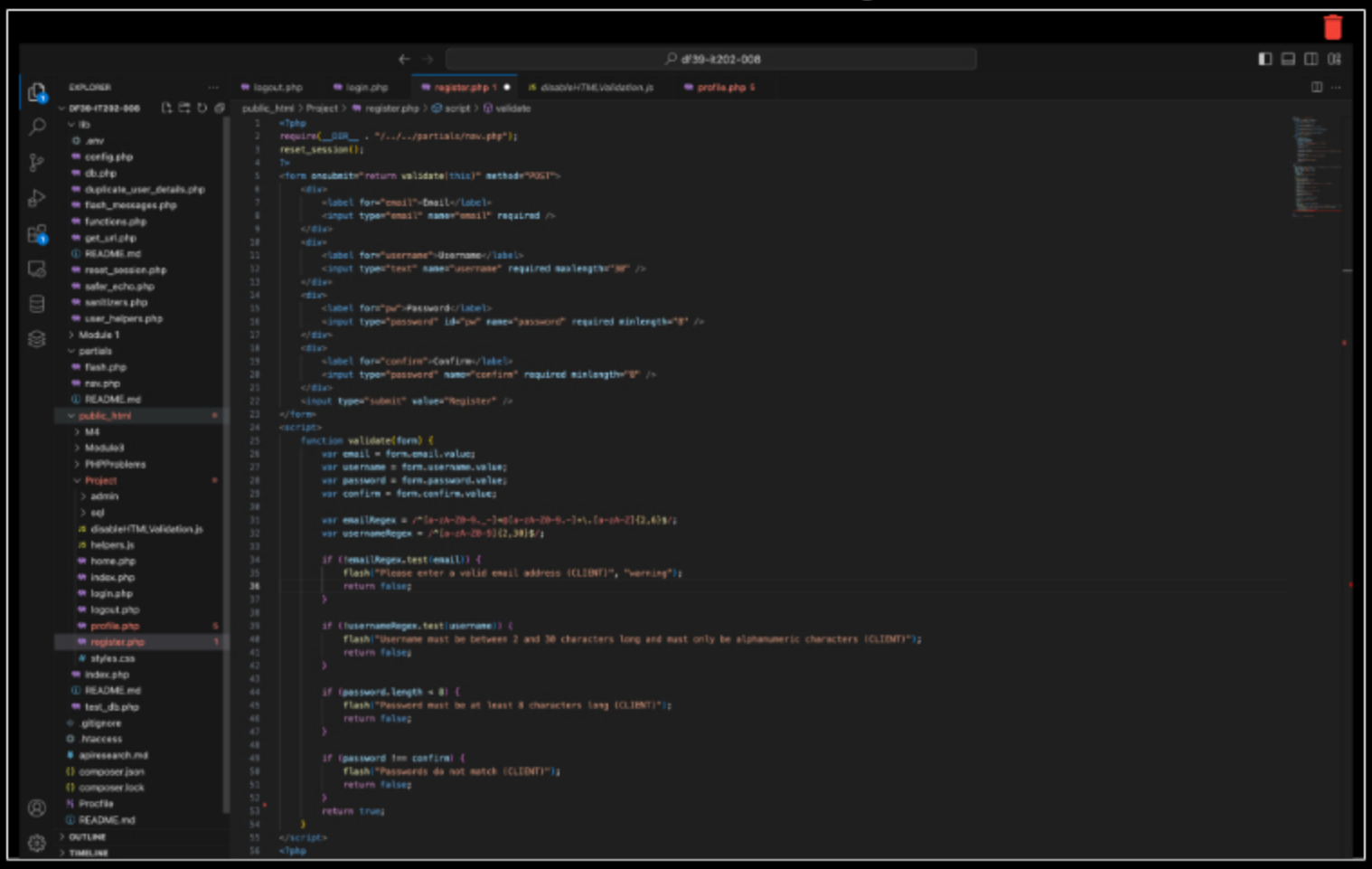
Text: Screenshot of the form code

Details:
Should have appropriate input types for the fiel

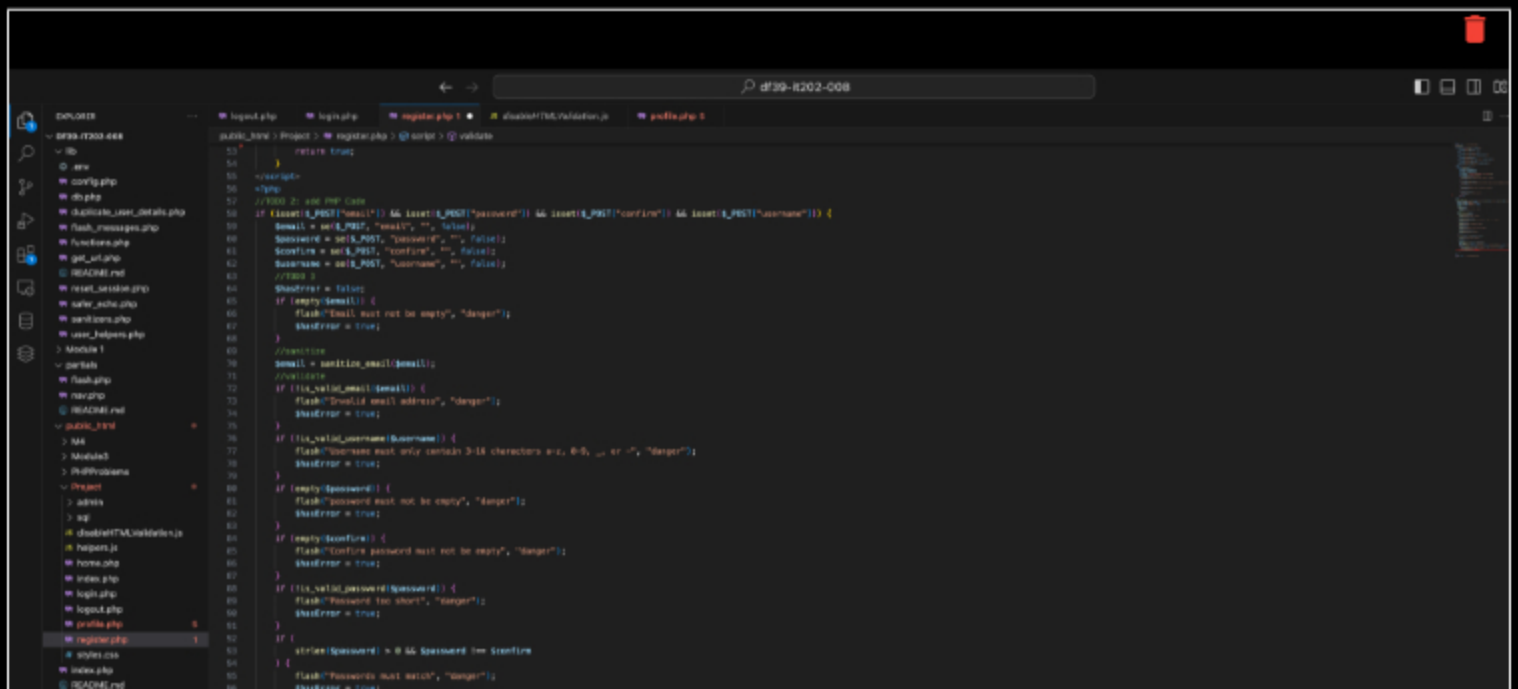
Task Screenshots:

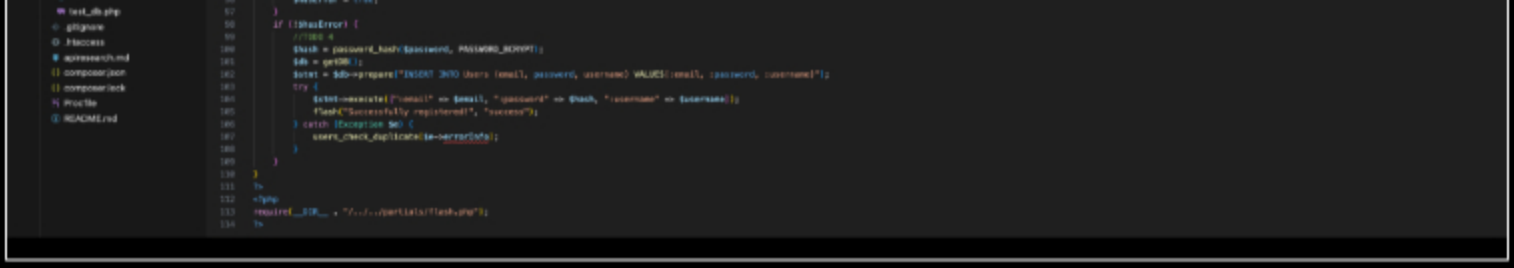
Gallery Style: Large View

Small Medium Large



Screenshot of JavaScript validation





Screenshots of the form code of register.php

^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

```
        <label for="confirm">Confirm</label>
        <input type="password" name="confirm" required minlength="8" />
    </div>
    <input type="submit" value="Register" />
</form>
<script>
    function validate(form) {
        // DF39 4/1/2024
        var email = form.email.value;
        var username = form.username.value;
        var password = form.password.value;
        var confirm = form.confirm.value;

        var emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
        var usernameRegex = /^[a-zA-Z0-9]{2,30}$/;

        if (!emailRegex.test(email)) {
            flash("Please enter a valid email address (CLIENT)", "warning");
            return false;
        }

        if (!usernameRegex.test(username)) {
            flash("Username must be between 2 and 30 characters long and must only be alphanumeric characters (CLIENT)");
            return false;
        }

        if (password.length < 8) {
            flash("Password must be at least 8 characters long (CLIENT)");
            return false;
        }

        if (password !== confirm) {
            flash("Passwords do not match (CLIENT)");
            return false;
        }
        return true;
    }
</script>
<?php
```

Javascript validation code with my UCID and date.

Checklist Items (2)

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

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

```php
</script>
<?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");
        $hasError = true;
    }
    if (empty($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) {
        //TODO 4
```

Validation code of the server side (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

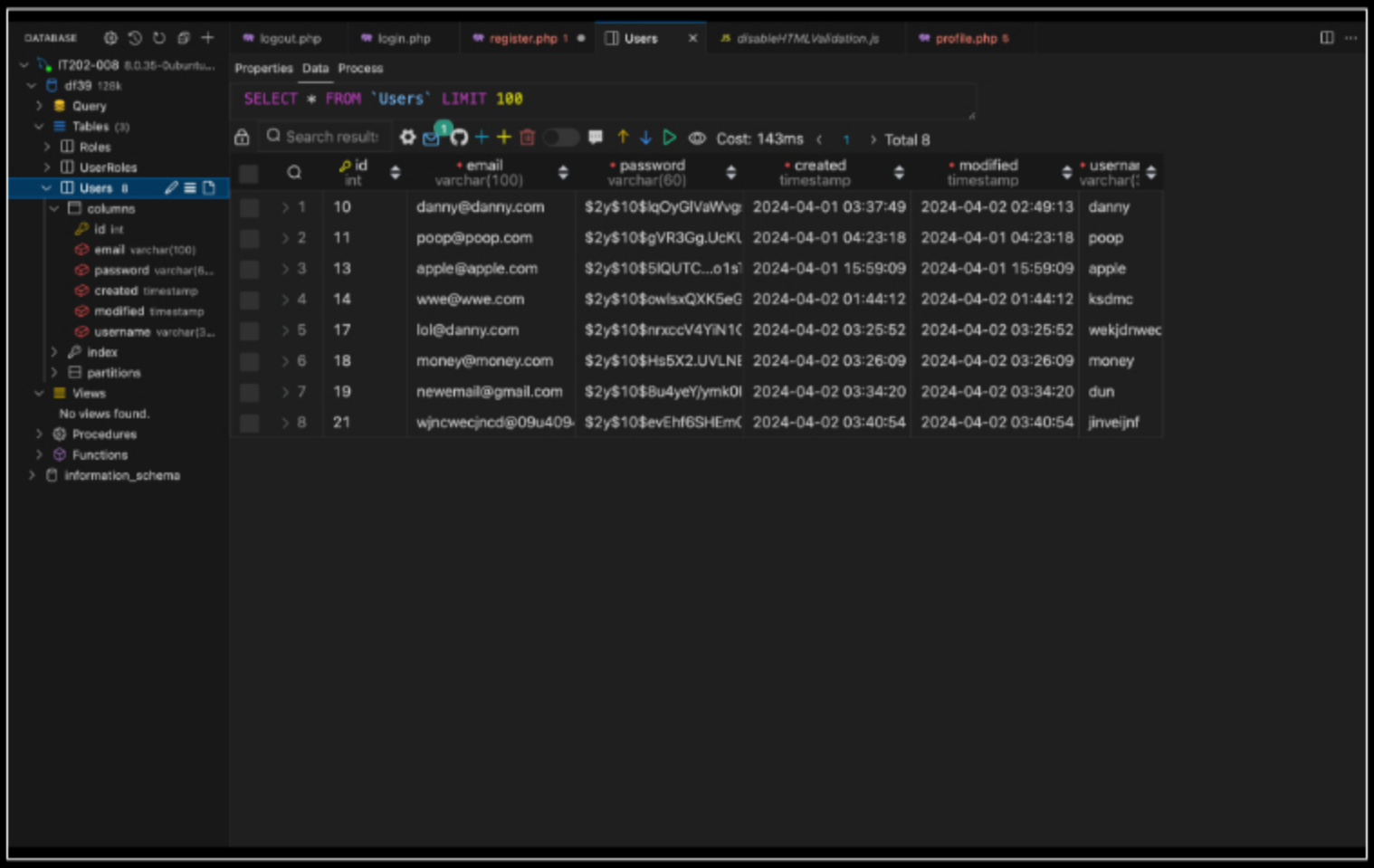
*The checkboxes are for your own tracking

#	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 shown with password hashed. Also contains, email, password, username and id fields. Left side of screenshot shows my ucid for the database

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:

When the page loads the java script validation loads with the page but is not yet executed, when the user puts information in the fields the java script (client side) validate function is called to ensure the format of the information is correct. When the user inputs the email, user name, password and confirm password the form is then submitted. If it is incorrect a flash message will pop up telling them what is incorrect. Once the form is submitted the data is sent to the server and the PHP code will check if the email, username, password and confirmation password are set. If they are set the php code will then add these user inputs into the $_POST array and sanitize the input data using the se function. This is for an extra layer of security. The data is then validated to ensure the format of username, password, email and password confirm. If the format is correct it then hashes the password and then pulls the data base using getDB and inserts the data into its appropriate table (Users Table) and creates a new user.

COLLAPSE

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/dunnfall/df39-it202-008/pull/19

User Login (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of form on website page

Checklist

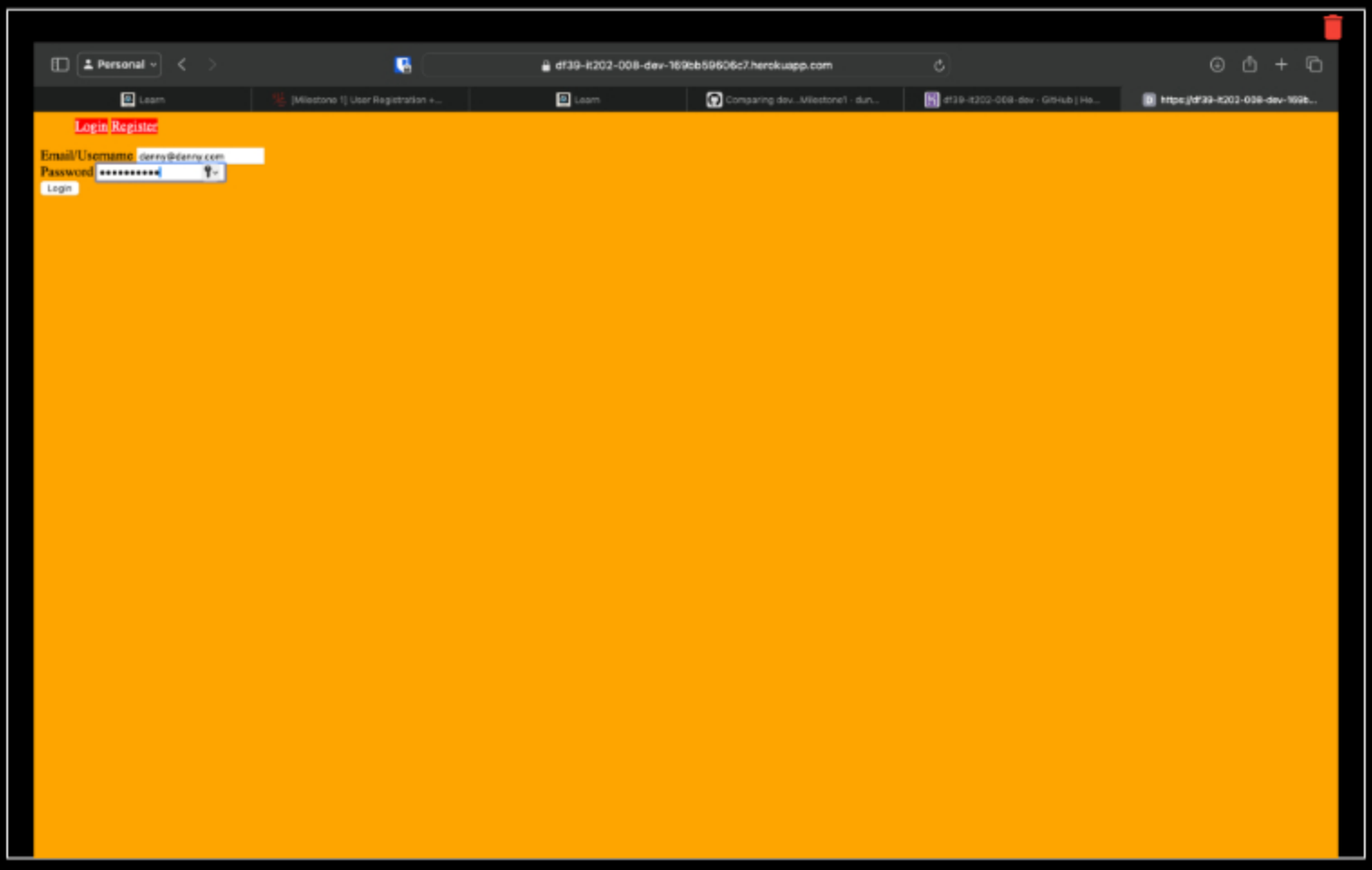
*The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



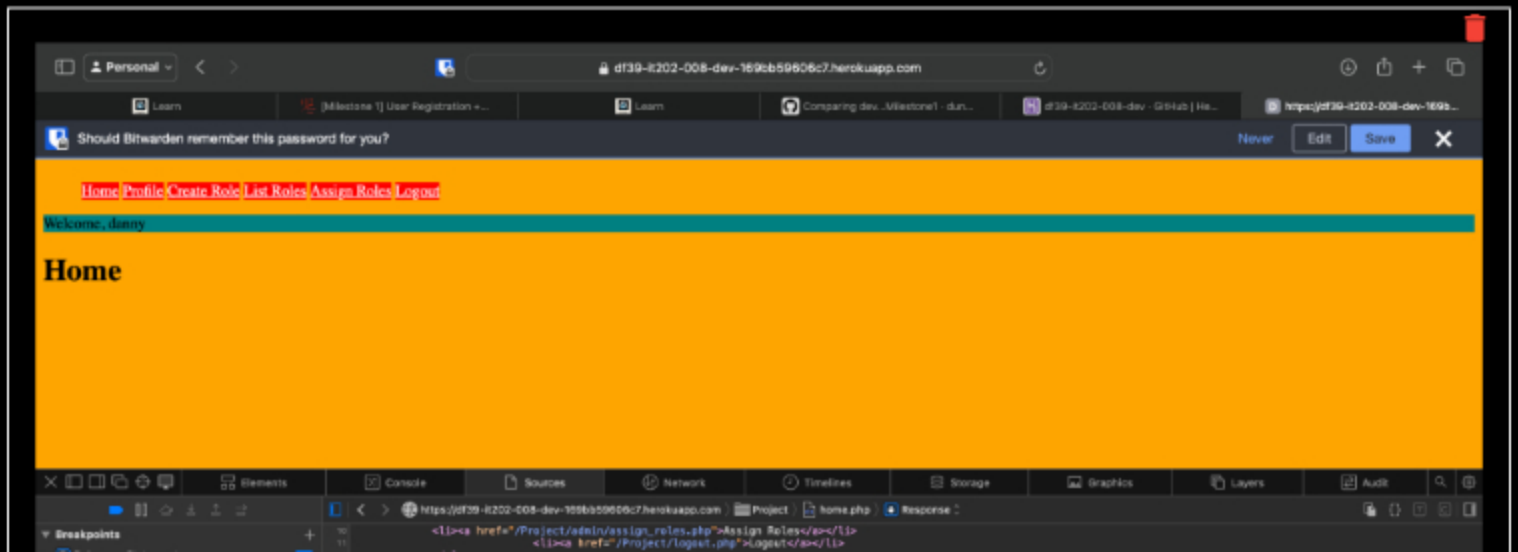
Shows heroku dev url, CSS applied on login site, correct login data is shown.

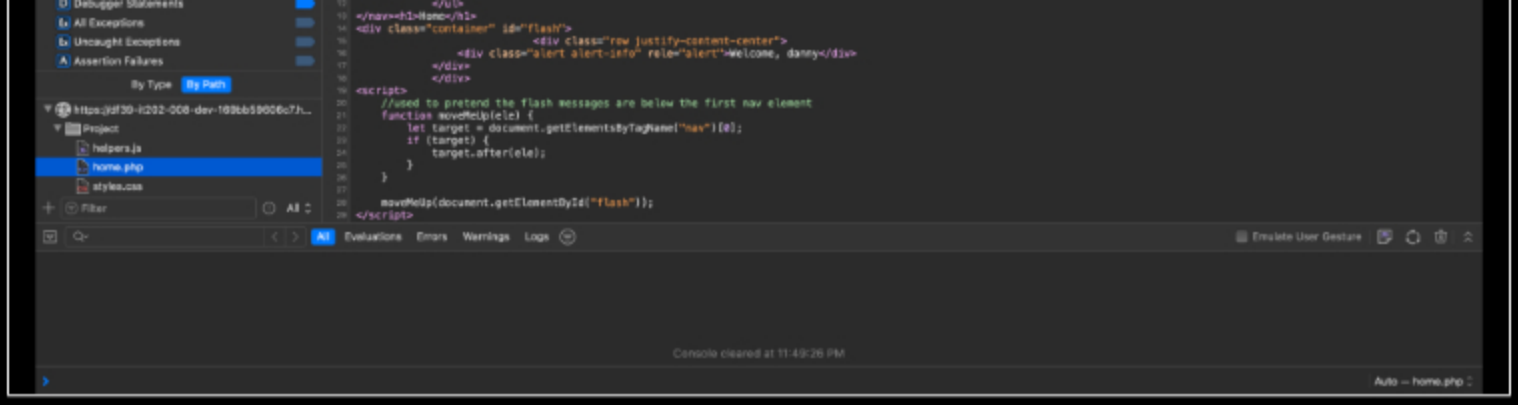
Checklist Items (3)

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

#2 Should have thoughtful CSS applied

#3 Include correct data where username is used to login

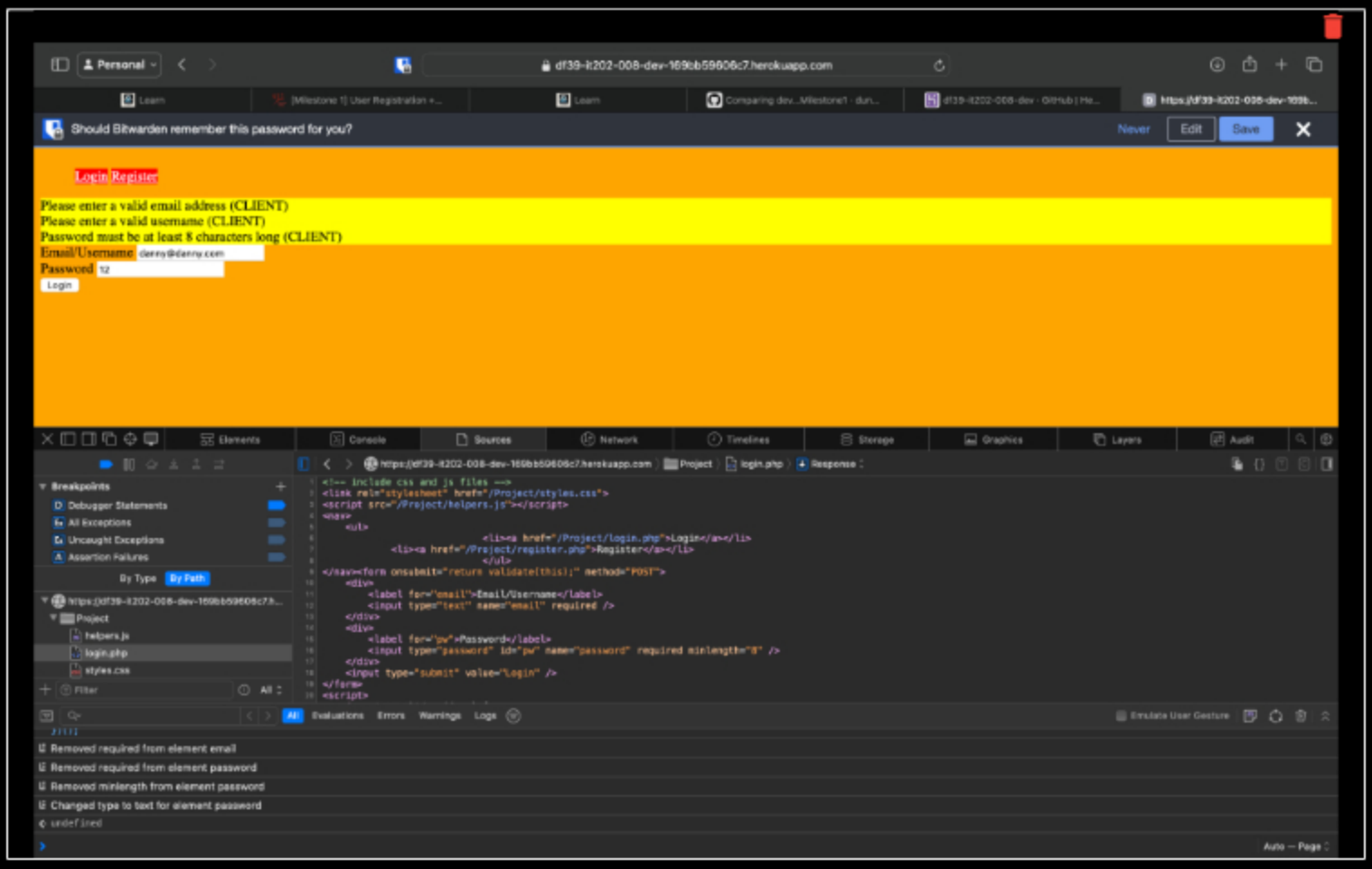




Login success

Checklist Items (1)

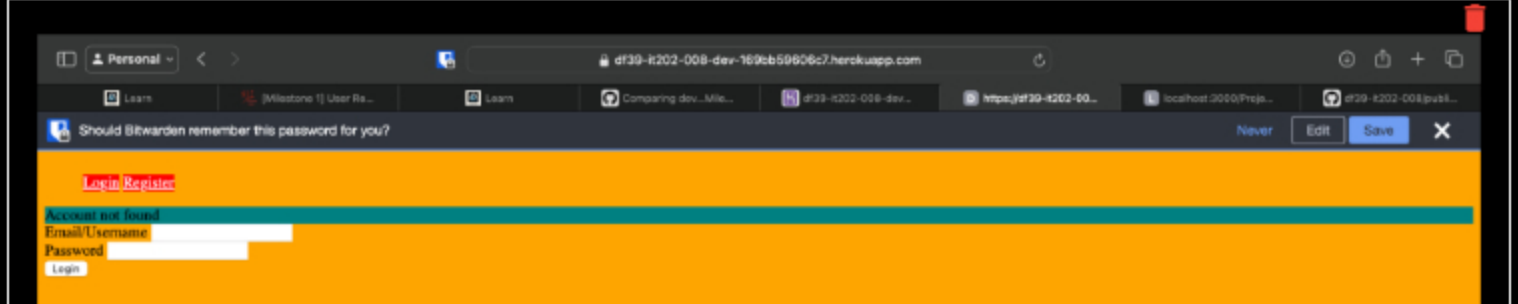
#5 Show success login message



Javascript validation for email/username formate and password format.

Checklist Items (1)

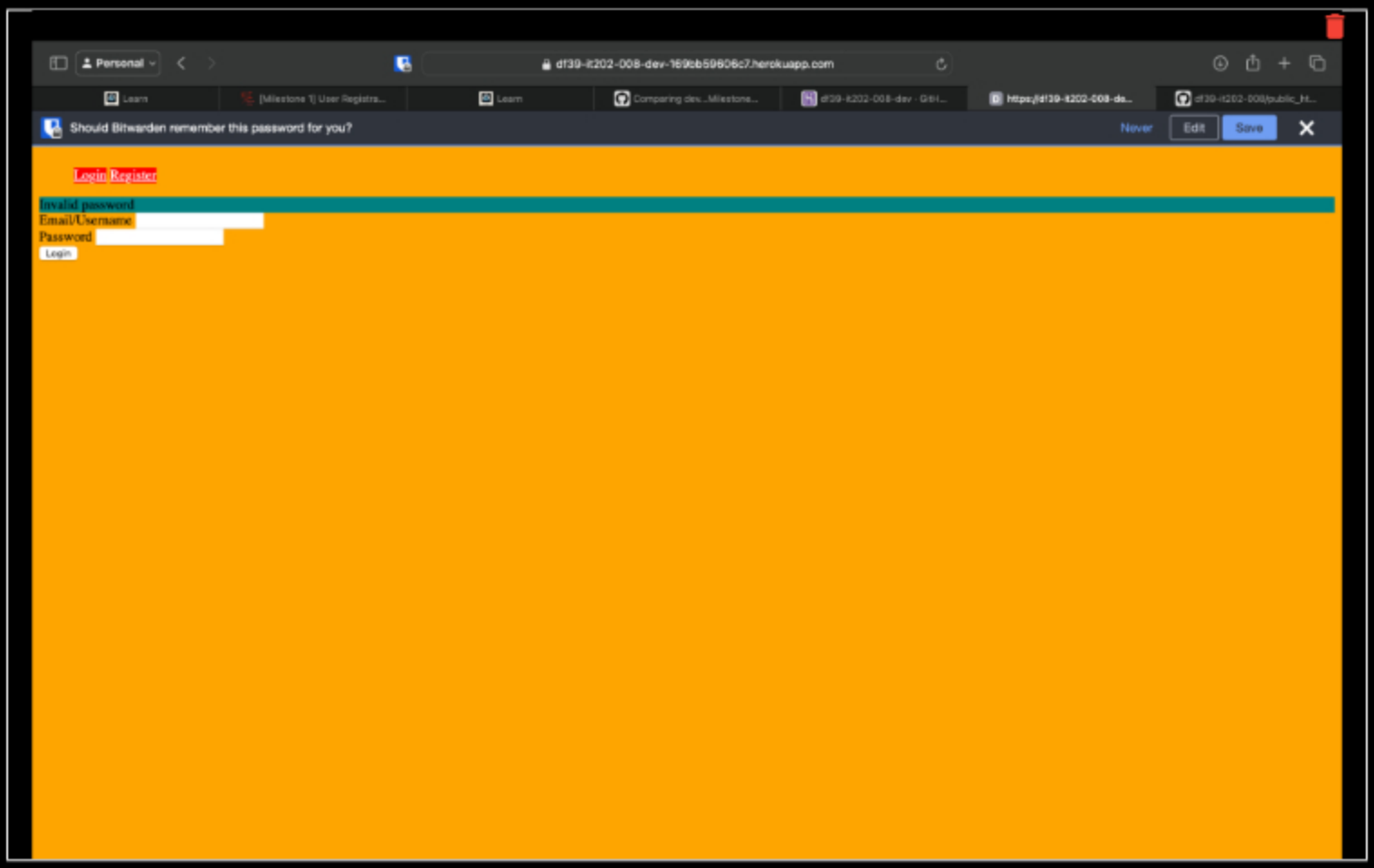
#6 Demonstrate JavaScript validation for each field [can be combined] (username format, email format, password format, and each field being required)



Account Not Found Message

Checklist Items (1)

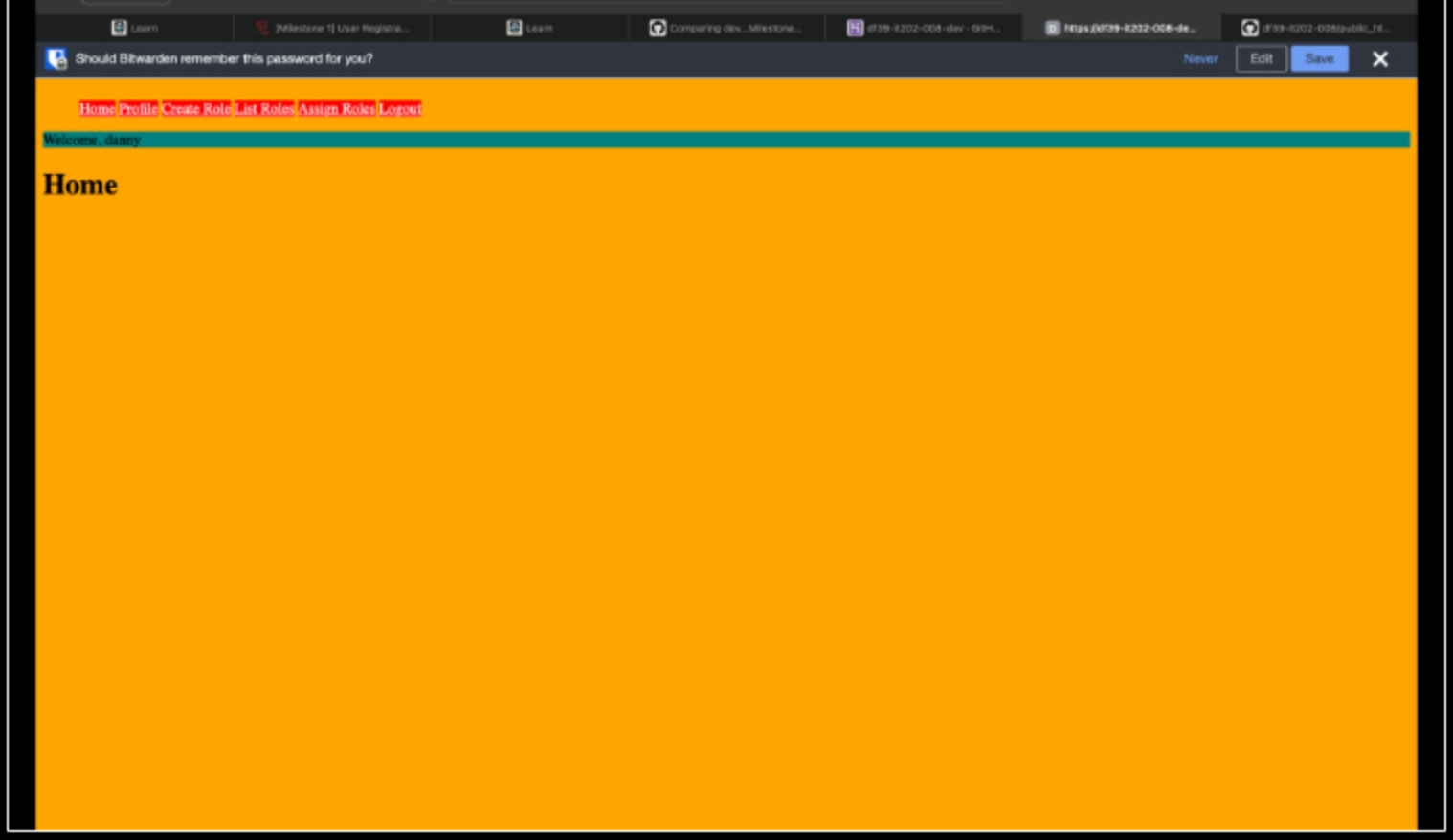
#7 Demonstrate user-friendly message of when an account doesn't exist



Invalid Password message

Checklist Items (1)

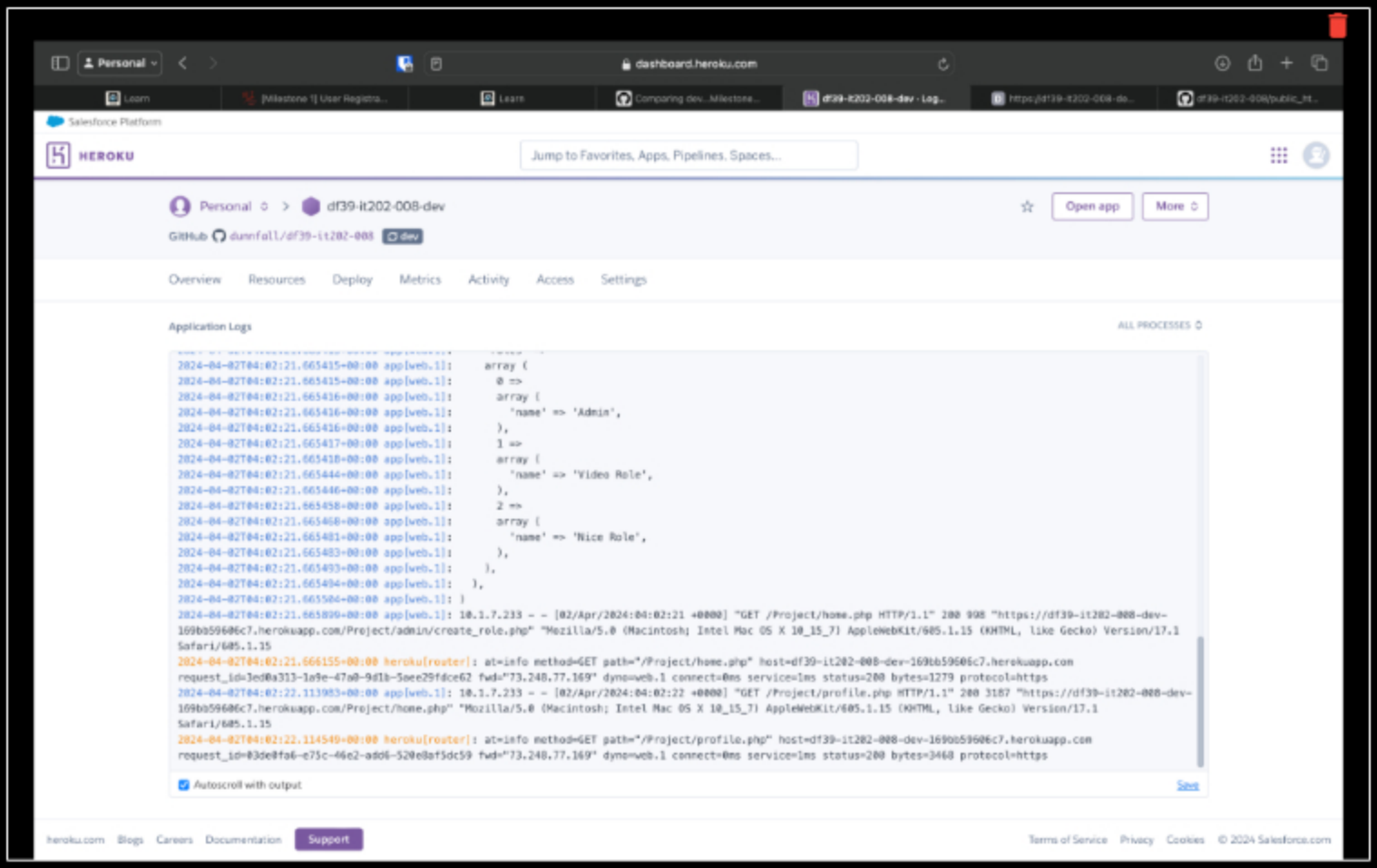
#8 Demonstrate user-friendly message of when password doesn't match what's in the DB



Successful login, and the destination page

Checklist Items (1)

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



Session data being set from 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

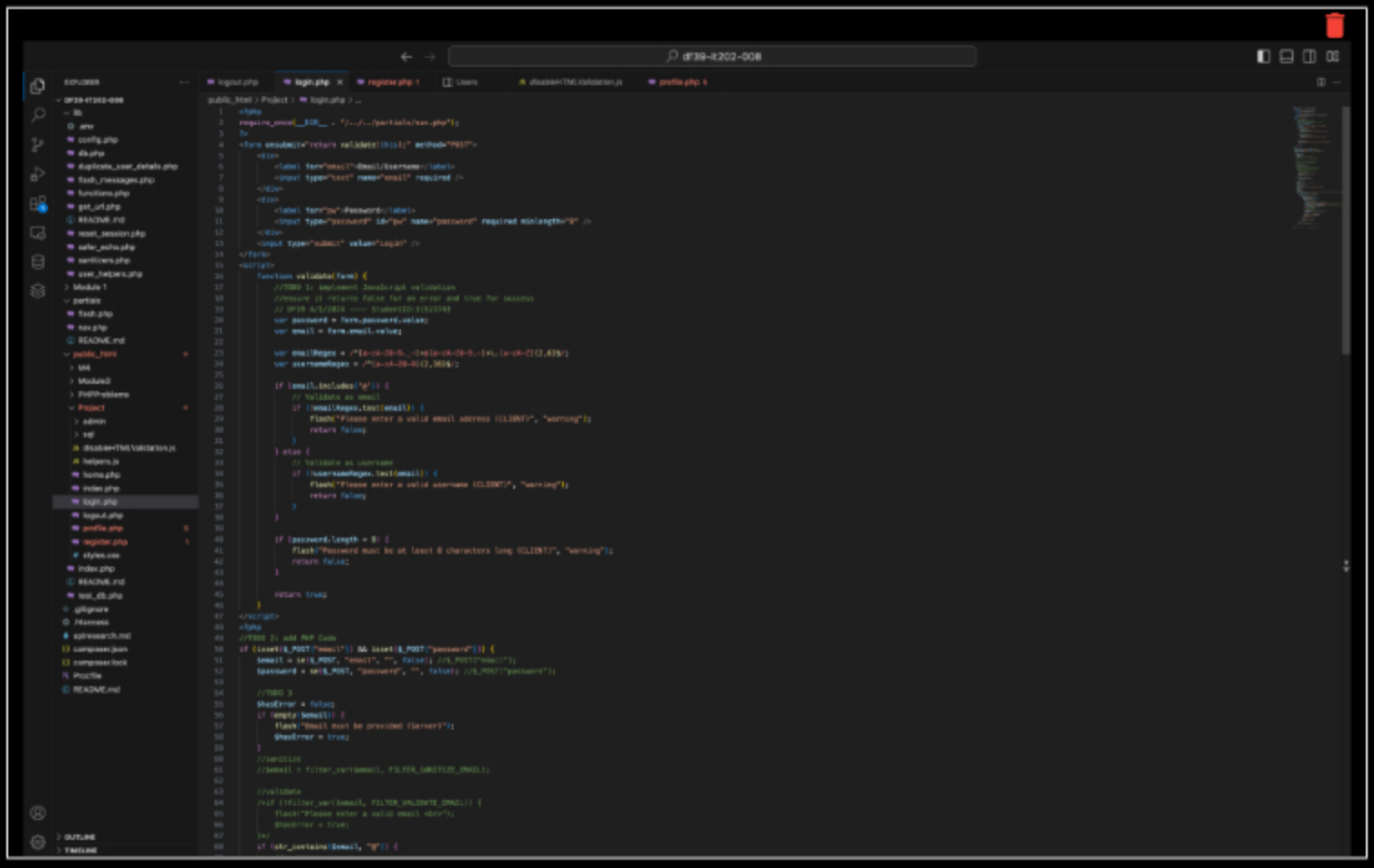
*The checkboxes are for your own tracking

#	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

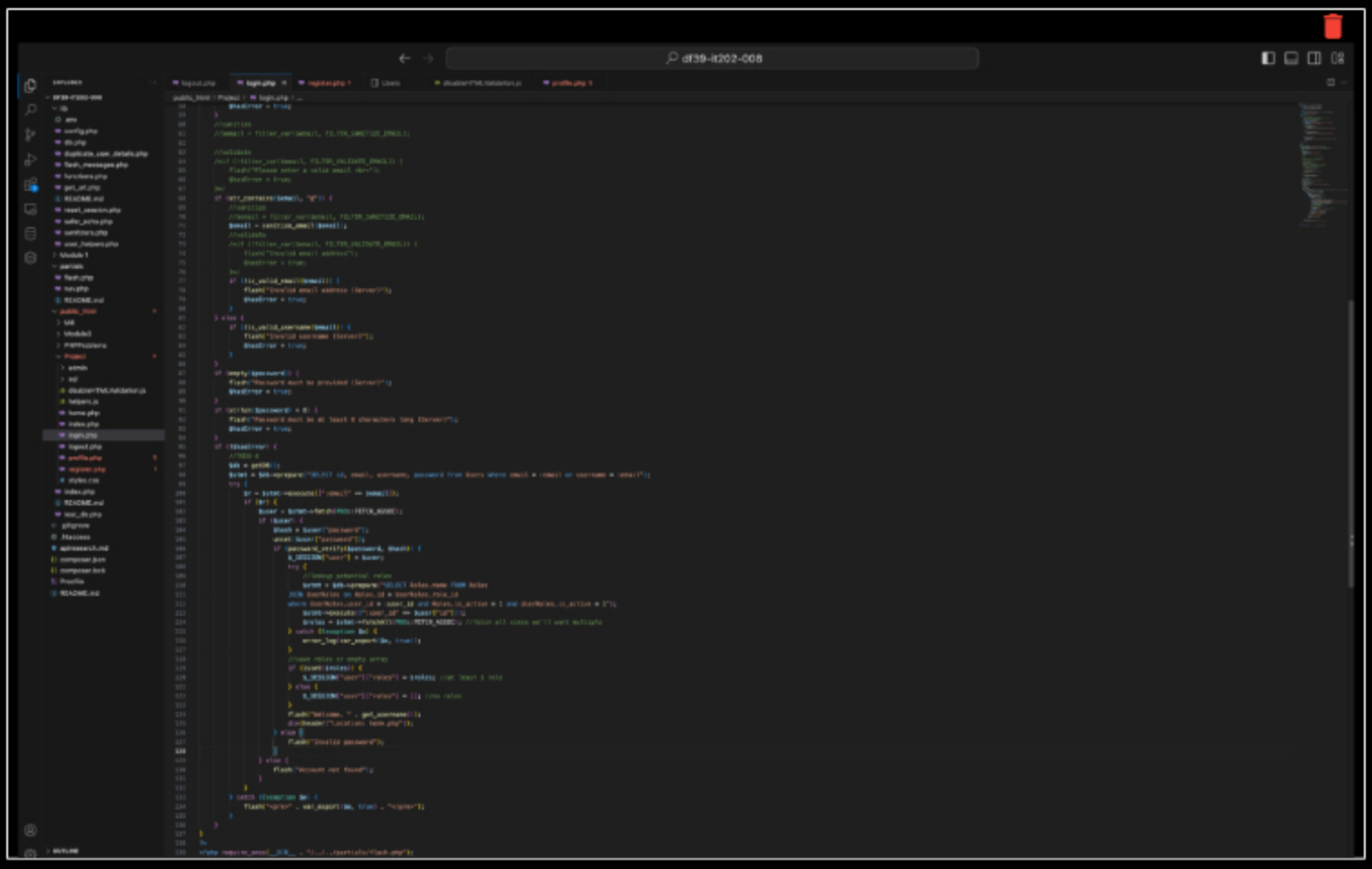
Small Medium Large



Input fields shown at the top, username and email are in a combined field.

Checklist Items (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)



Continuation of first screenshot

Checklist Items (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)

```
<script>
    function validate(form) {
        //TODO 1: implement JavaScript validation
        //ensure it returns false for an error and true for success
        // DF39 4/1/2024 ----- StudentID:31523743
        var password = form.password.value;
        var email = form.email.value;

        var emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
        var usernameRegex = /^[a-zA-Z0-9]{2,16}$/;

        if (email.includes('@')) {
            // Validate as email
            if (!emailRegex.test(email)) {
                flash("Please enter a valid email address (CLIENT)", "warning");
                return false;
            }
        } else {
            // Validate as username
            if (!usernameRegex.test(email)) {
                flash("Please enter a valid username (CLIENT)", "warning");
                return false;
            }
        }

        if (password.length < 8) {
            flash("Password must be at least 8 characters long (CLIENT)", "warning");
            return false;
        }
```

```
        return true;
    }
</script>
```

JavaScript validations with UCID and date

Checklist Items (1)

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

```php
<?php
//TODO 2: add PHP Code
//DF39 4/1/2024
if (isset($_POST["email"]) && isset($_POST["password"])) {
    $email = se($_POST, "email", "", false); //$_POST["email"];
    $password = se($_POST, "password", "", false); //$_POST["password"];

    //TODO 3
    $hasError = false;
    if (empty($email)) {
        flash("Email must be provided (Server)");
        $hasError = true;
    }
    //sanitize
    //$email = filter_var($email, FILTER_SANITIZE_EMAIL);

    //validate
    /*if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        flash("Please enter a valid email <br>");
        $hasError = true;
    }*/
    if (str_contains($email, "@")) {
        //sanitize
        //$email = filter_var($email, FILTER_SANITIZE_EMAIL);
        $email = sanitize_email($email);
        //validate
        /*if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
            flash("Invalid email address");
            $hasError = true;
        }*/
        if (!is_valid_email($email)) {
            flash("Invalid email address (Server)");
            $hasError = true;
        }
    } else {
        if (!is_valid_username($email)) {
            flash("Invalid username (Server)");
            $hasError = true;
        }
    }
    if (empty($password)) {
        flash("Password must be provided (Server)");
        $hasError = true;
    }
    if (strlen($password) < 8) {
        flash("Password must be at least 8 characters long (Server)");
        $hasError = true;
    }
    if (!$hasError) {
        //TODO 4
```

PHP validations

Checklist Items (2)

#3 Show PHP validations (include any lib content)

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

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

#2	1	Explain how the session works and why/how it's used

Response:

When the login page is loaded again the java-script validation is loaded with it but not called until the user attempts to submit the form. Once submitted the java script validate function is called and checks the format of the username/email and password using regex. If incorrect format is found a flash message will appear telling the user what formatting is incorrect. If the information is valid the form is submitted. The form data is added to the $_POST array. The php code will then sanitize the data using the se function. After sanitization the server side validation occurs. The PHP code will check if the formatting of the username/email, and password field is correct and not empty. If the format is wrong or fields are empty it will send a flash message to the user with the flash function and change hasError to true. If the format is good PHP will then query the database to fetch data associated with the user email or username. If none is found the user will see an a "Account Not Found Message". After getting the user data the hashed password is matched up to the user inputed password. If it is correct the user will be logged in. Once the user is logged in their user data information is stored in the session to keep them logged in. The user data including the roles they have are also fetched and stored. The user after being authenticated is then directed to the home page.

COLLAPSE

Task #4 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/dunnfall/df39-it202-008/pull/21

User Logout (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Capture the following screenshots

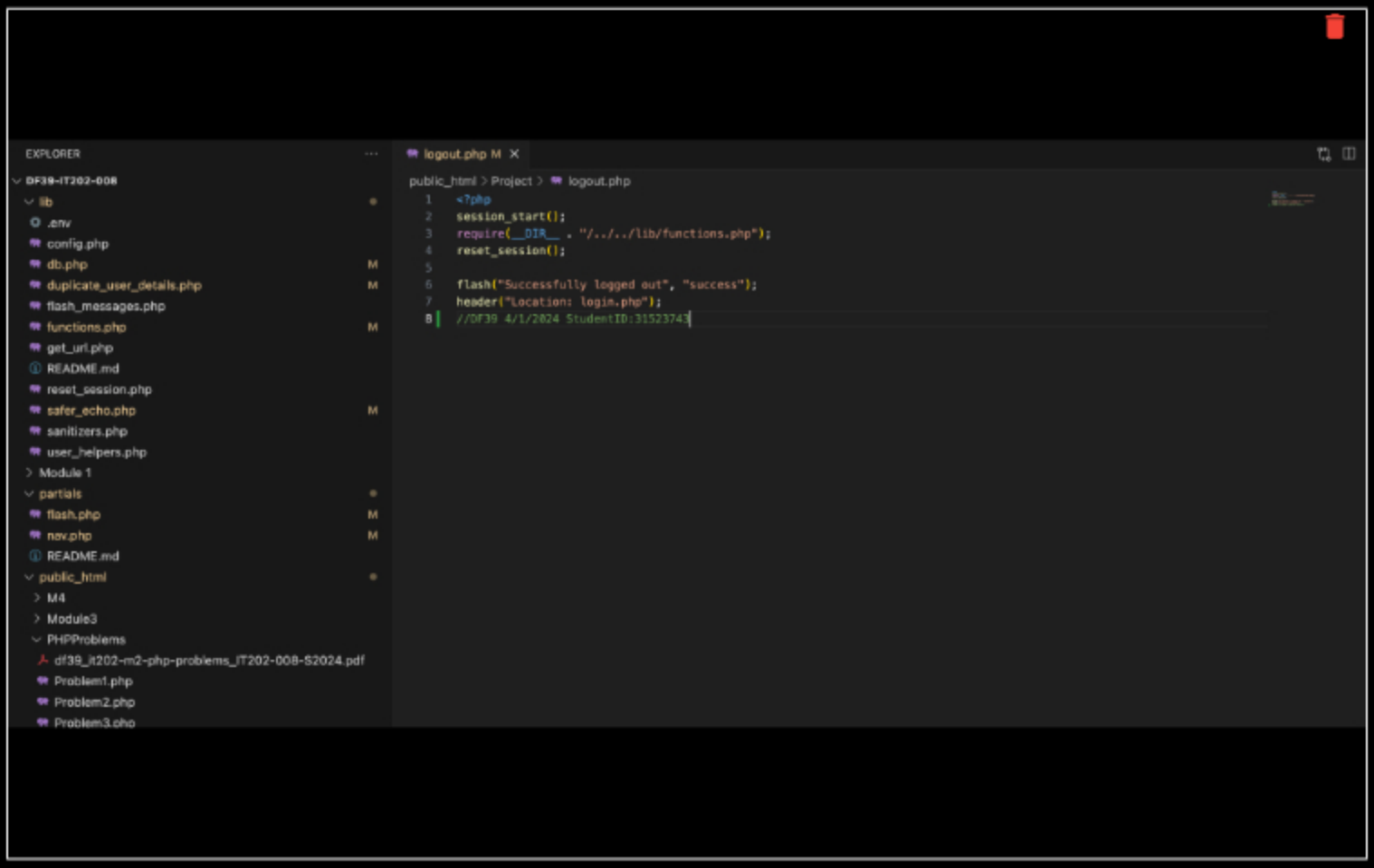
Checklist *The checkboxes are for your own tracking

#	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

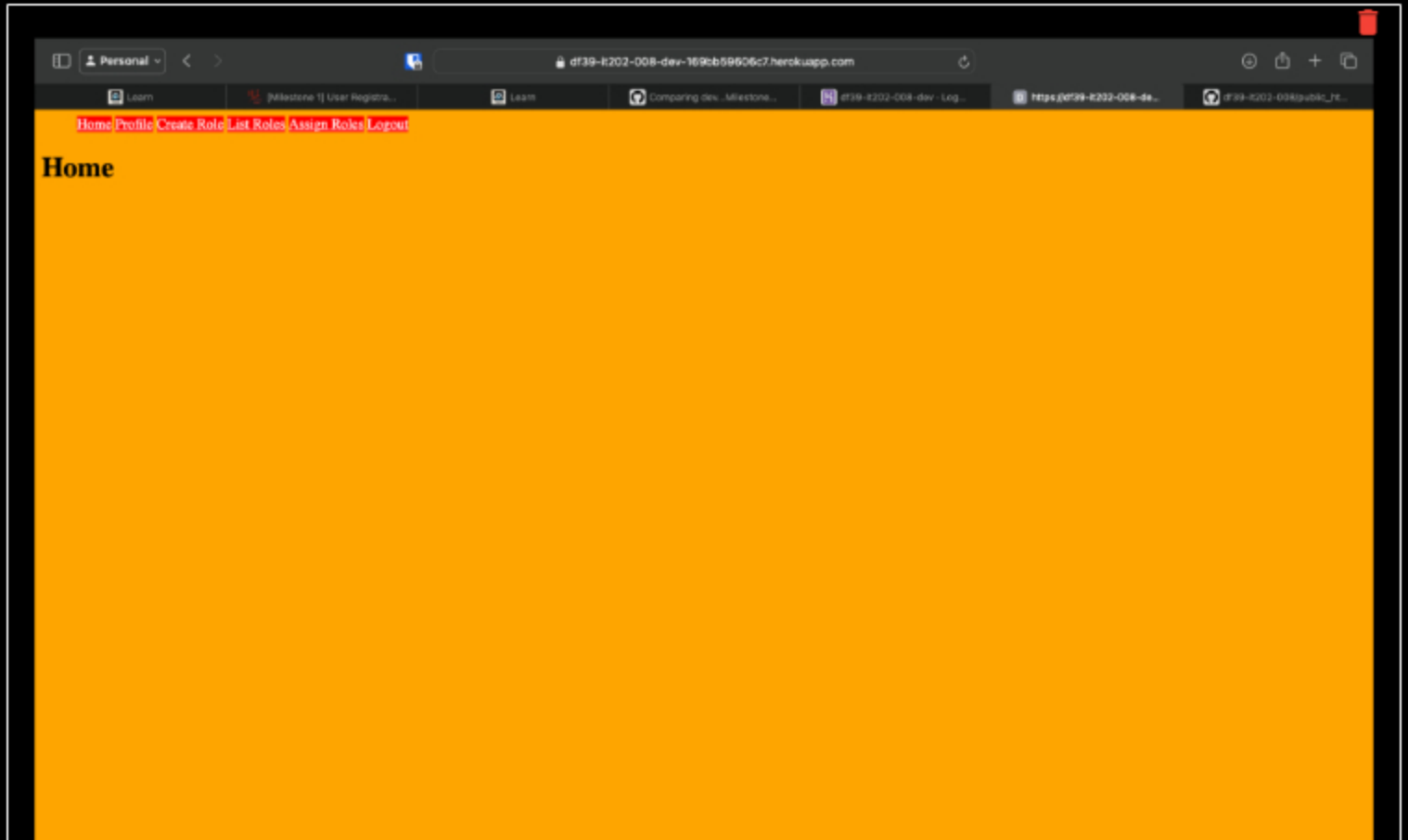
Small Medium Large



Logout related code, with session being destroyed and UCID and date.

Checklist Items (1)

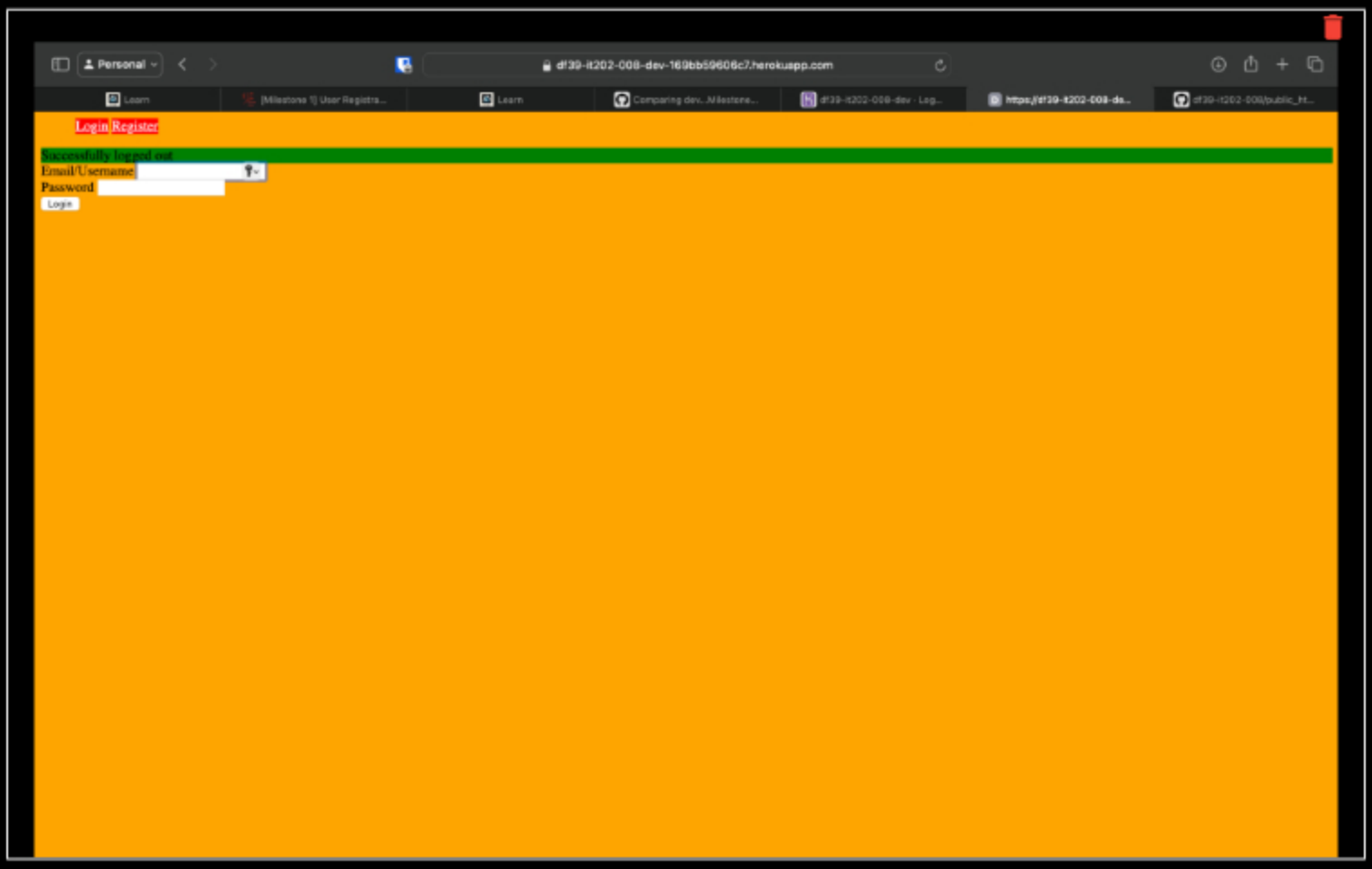
#3 Screenshot of the logout-related code showing the session is destroyed (code). Ensure ucid/date comment is present.



Logged in on navigation page

Checklist Items (1)

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



Redirect to login with logged out message

Checklist Items (1)

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

COLLAPSE

Task #2 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/dunnfall/df39-it202-008/pull/21

Basic Security Rules and Roles (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Authentication Screenshots

Checklist

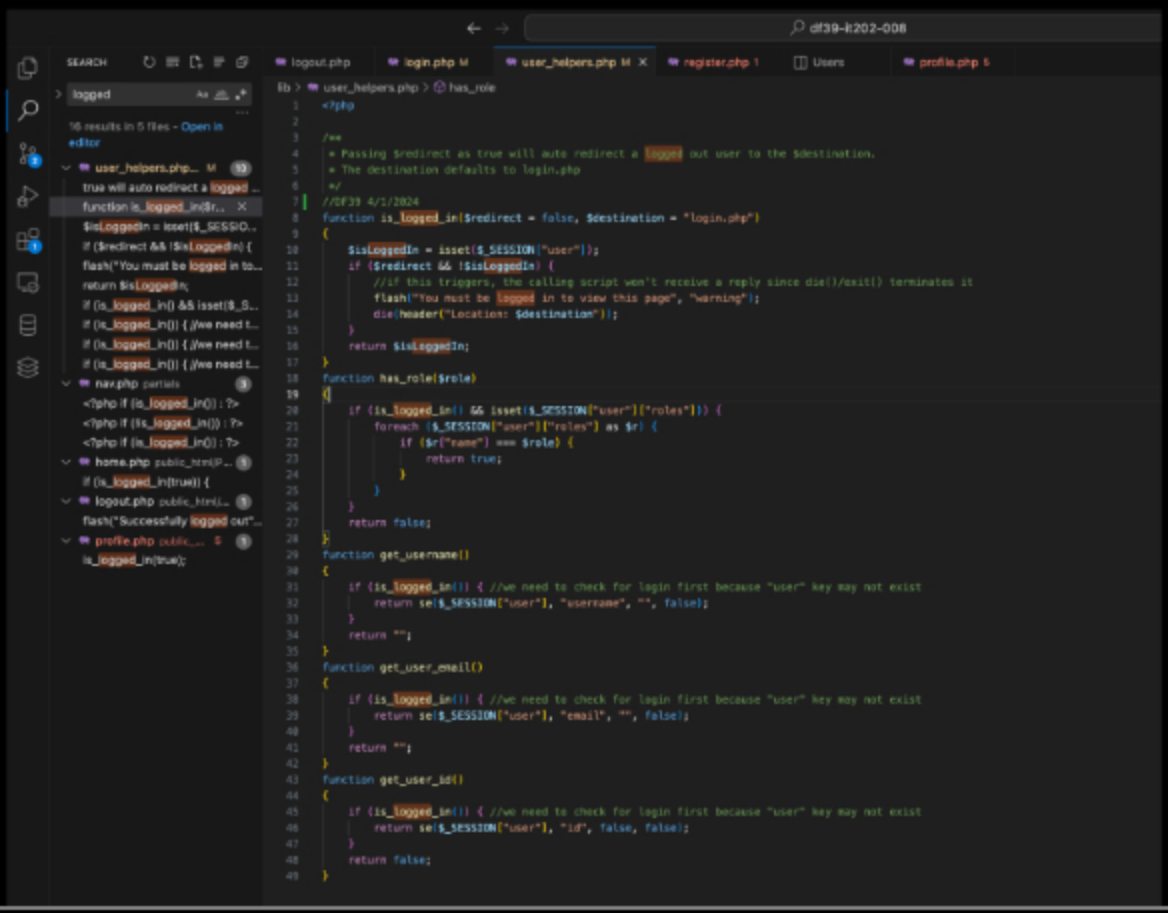
*The checkboxes are for your own tracking

#	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

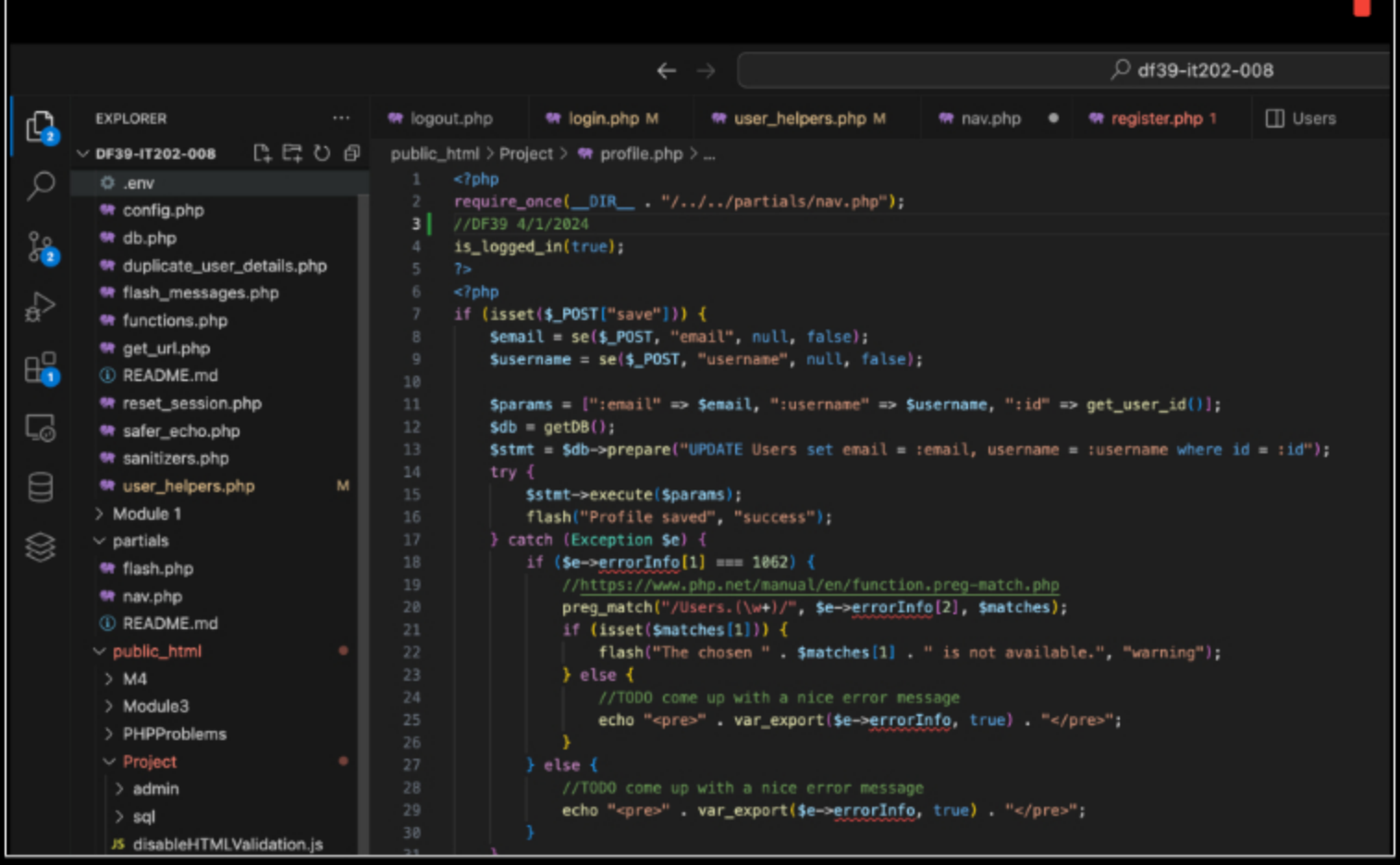
Small Medium Large



Login check function in user helpers

Checklist Items (1)

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

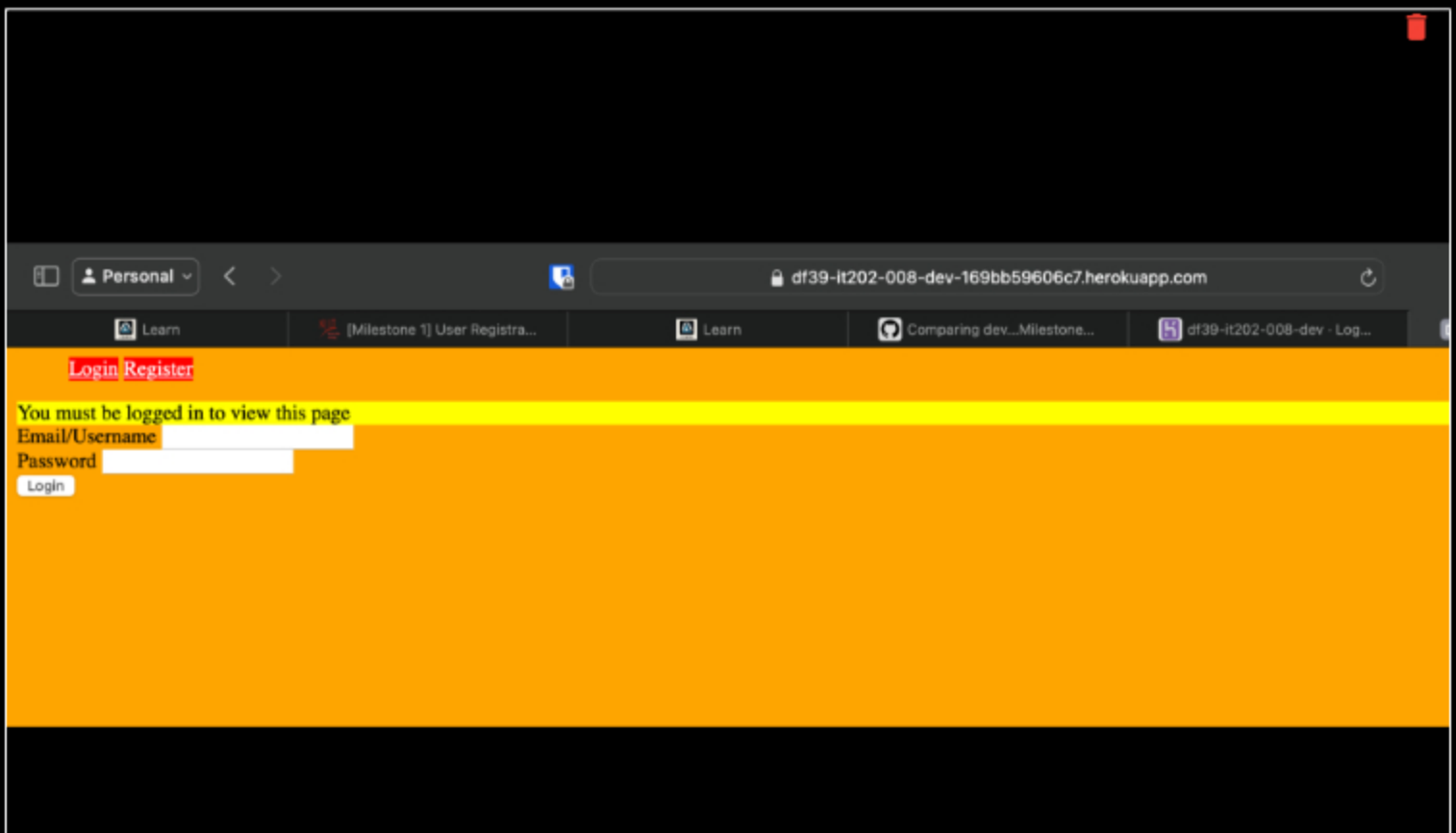


The logged in function is being used in nav.php home.php and profile.php

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)



User friendly message of accessing a page that requires to be logged in

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

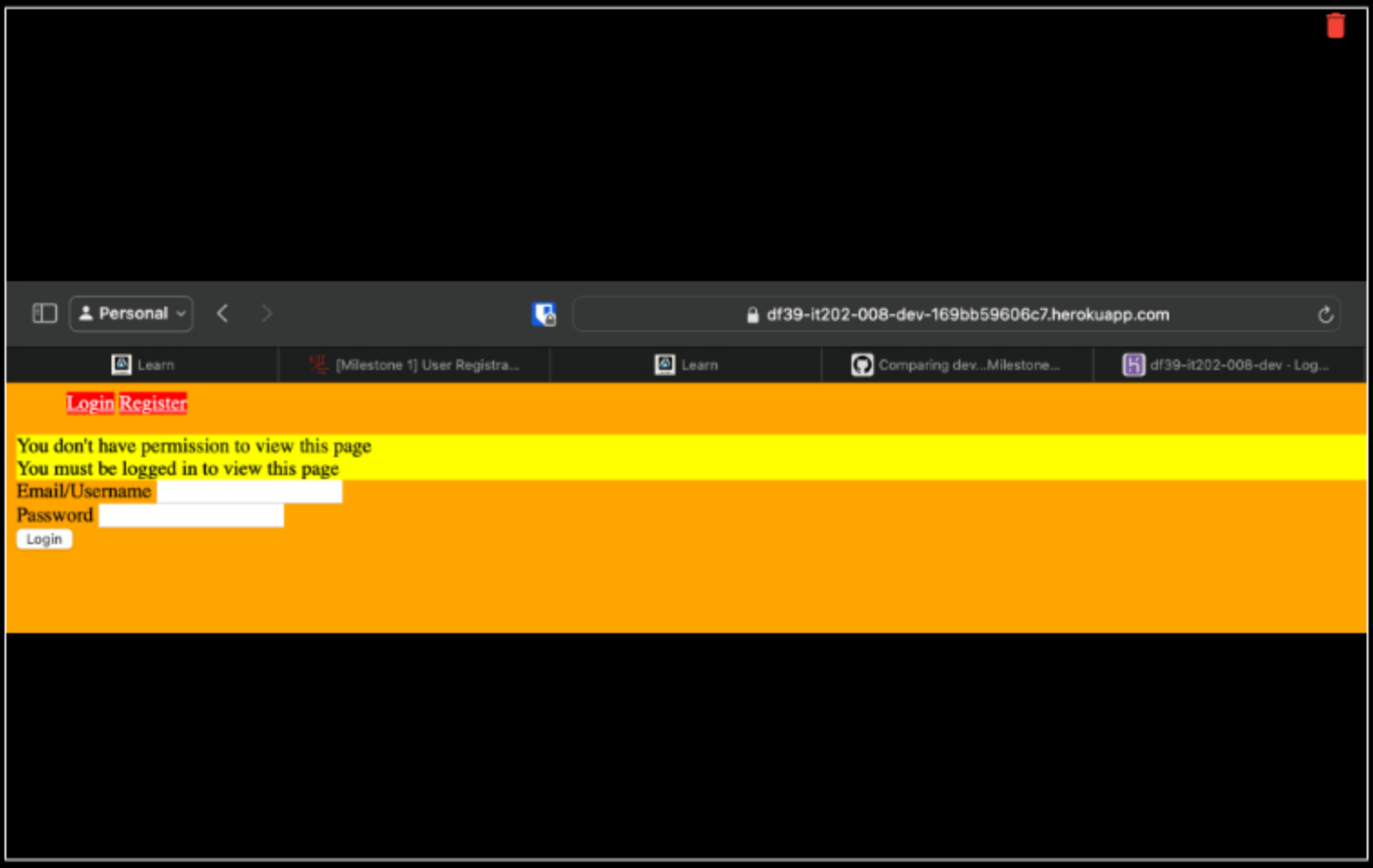
```php
//DF39 4/1/2024
function has_role($role)
{
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {
        foreach ($_SESSION["user"]["roles"] as $r) {
            if ($r["name"] === $role) {
                return true;
            }
        }
    }
    return false;
}
```

Function that checks for specific roles with UCID and date

Checklist Items (2)

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

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



Message when trying to access role-restricted page

Checklist Items (1)

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

```
<!-- include css and js files -->
<!-- DF39 4/1/2024 -->
<link rel="stylesheet" href="<?php echo get_url('styles.css'); ?>">
<script src="<?php echo get_url('helpers.js'); ?>"></script>
<nav>
    <ul>
        <?php if (is_logged_in()) : ?>
            <li><a href="<?php echo get_url('home.php'); ?>">Home</a></li>
            <li><a href="<?php echo get_url('profile.php'); ?>">Profile</a></li>
        <?php endif; ?>
        <?php if (!is_logged_in()) : ?>
            <li><a href="<?php echo get_url('login.php'); ?>">Login</a></li>
            <li><a href="<?php echo get_url('register.php'); ?>">Register</a></li>
        <?php endif; ?>
        <?php if (has_role("Admin")) : ?>
            <li><a href="<?php echo get_url('admin/create_role.php'); ?>">Create Role</a></li>
```

```php
            <li><a href="<?php echo get_url('admin/list_roles.php'); ?>">List Roles</a></li>
            <li><a href="<?php echo get_url('admin/assign_roles.php'); ?>">Assign Roles</a></li>
        <?php endif; ?>
        <?php if (is_logged_in()) : ?>
            <li><a href="<?php echo get_url('logout.php'); ?>">Logout</a></li>
        <?php endif; ?>
    </ul>
</nav>
```

This function of has_role is used in nav.php, assign_roles.php, create_role.php and list_roles.php

Checklist Items (1)

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

COLLAPSE

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

Checklist

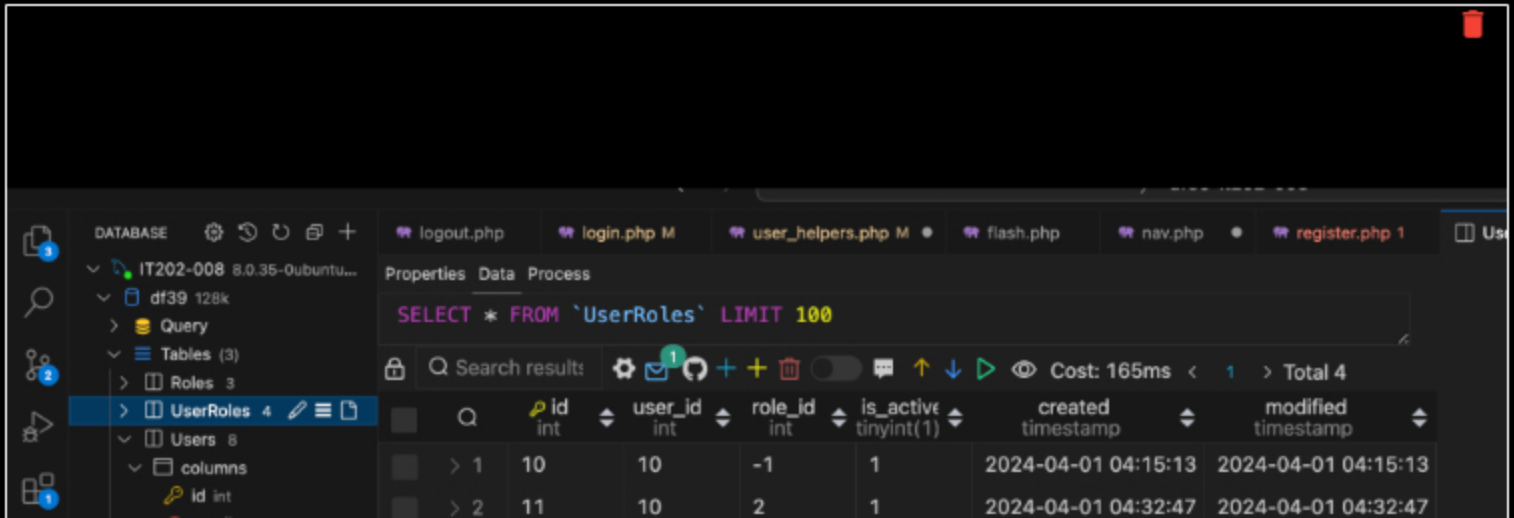
*The checkboxes are for your own tracking

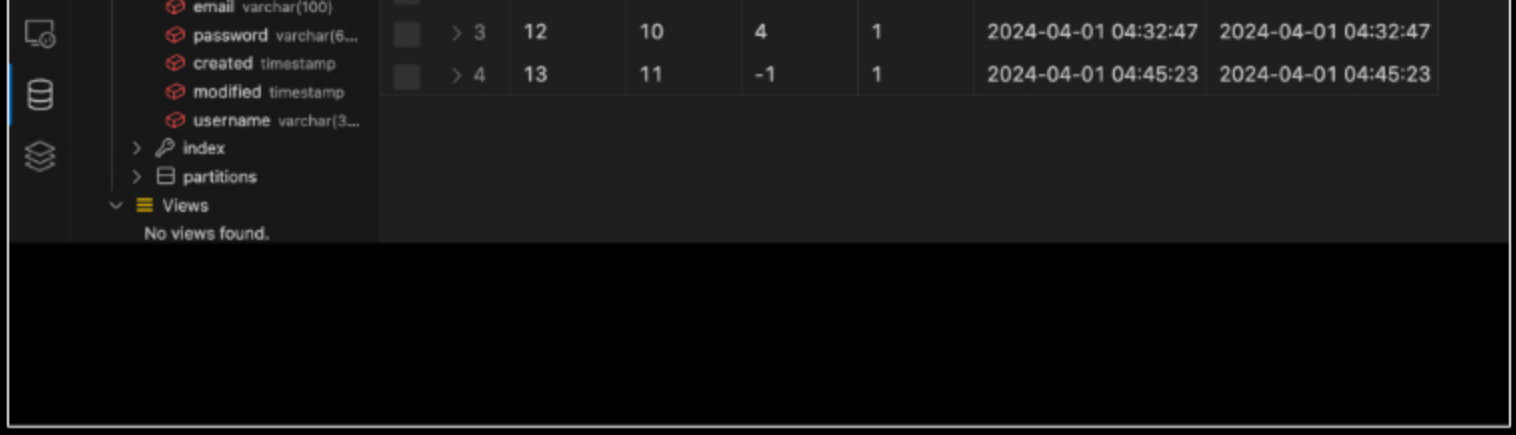
#	Points	Details
#1	1	At least one valid and enabled User->Role reference (UserRoles table)
#2	1	UserRoles Table should have id, user_id, role_id, is_active, created, and modified columns
#3	1	Roles Table should have id, name, description, is_active, modified, and created columns
#4	1	At least one valid 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



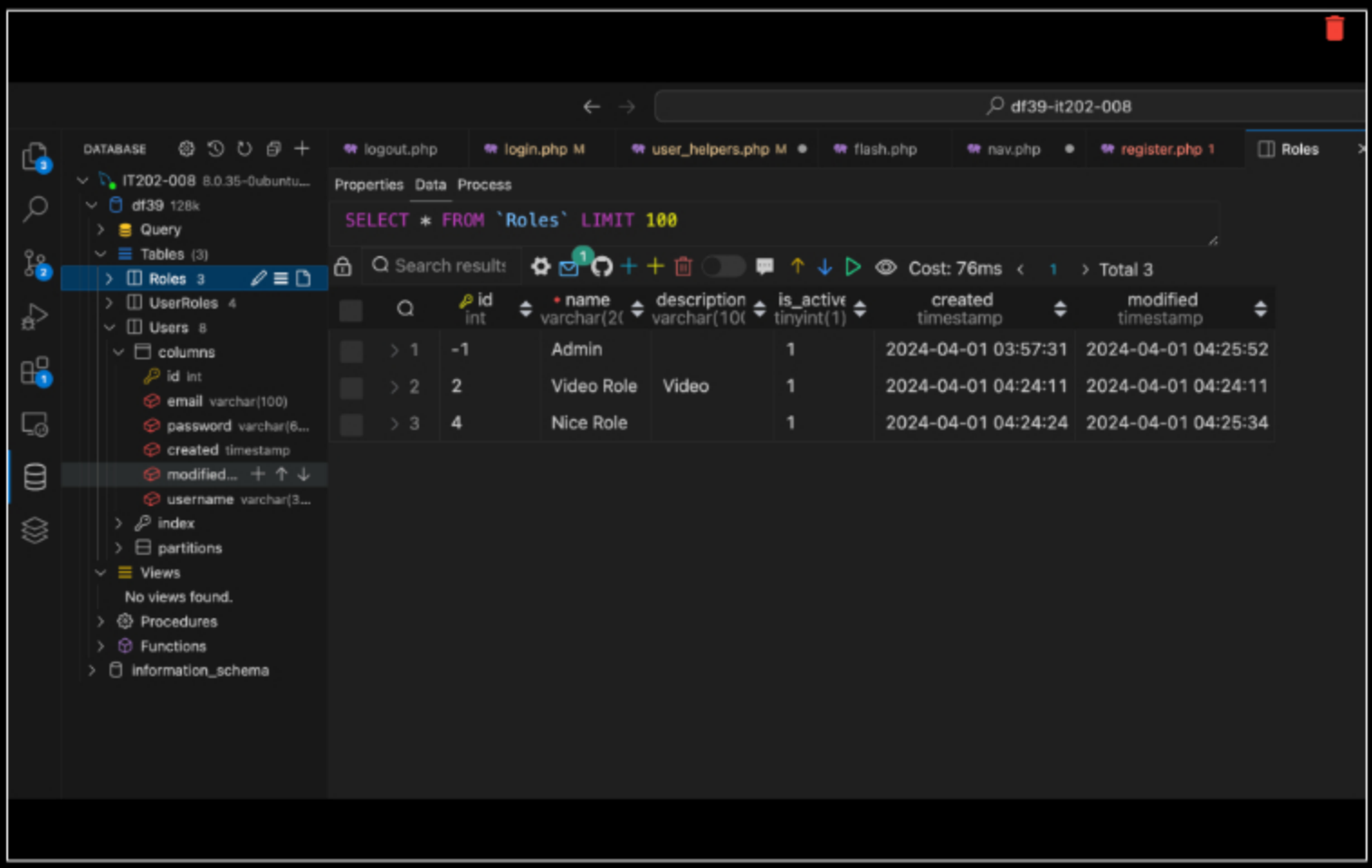


UserRoles has id, user_id, role_id, is_active, created, modified columns. Also has one valid and enabled user.

Checklist Items (2)

#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



Roles table with id, name, description, is_active, created, and modified columns. Left side panel has UCID with one valid and enable role

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)

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 purpose of the user roles table is to be a middle man table for other tables. It manages the relationship between the Users table and Roles table as well as assign roles to users and manage their permissions. Roles.is_active indicates if the role in the table is currently active and can be turned on and off, where as UserRoles.is_active is a soft switch where the user can activate to deactivate their role.

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/dunnfall/df39-it202-008/pull/34

COLLAPSE

User Profile (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: View Profile Website Page

Checklist

*The checkboxes are for your own tracking

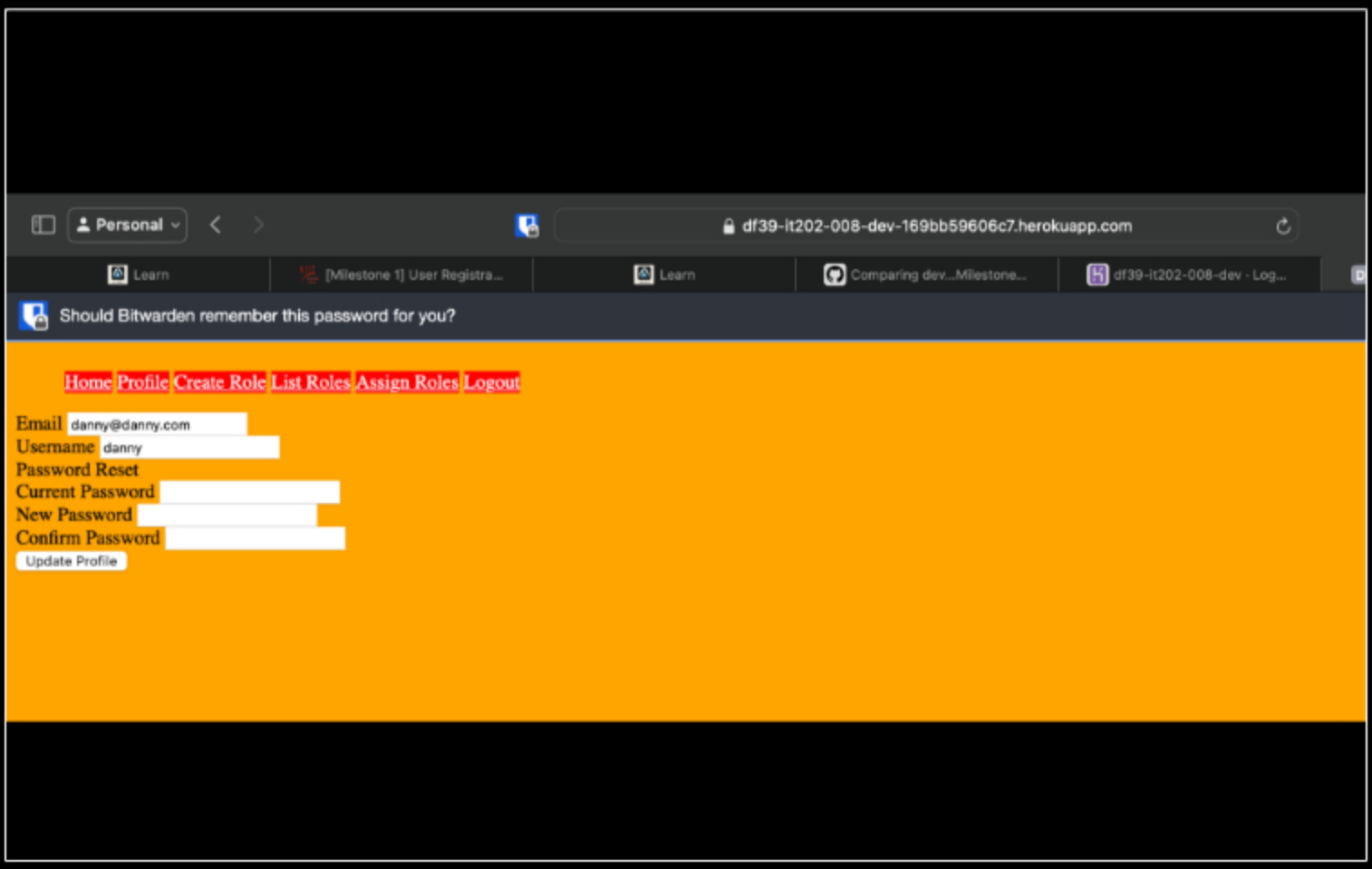
#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Show the profile form correctly populated on page load (username, email)
#4	1	Should have the following fields: username, email, current password, new password, confirm password (or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

password, confirm password (or similar)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Heroku dev in url bar CSS applied Profile page populated with username, email, password reset, new password, confirm password

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 the page is loaded the nav.php file is included to ensure the nav.php code is implemented at the top of the page for the user. As well as the is_logged_in(true) function is called to ensure the user is logged in. If the user is logged in it will call the nav.php file and display the nav and the rest of the profile.php file.

COLLAPSE

Task #3 - Points: 1

Text: Edit Profile Website Page

Checklist

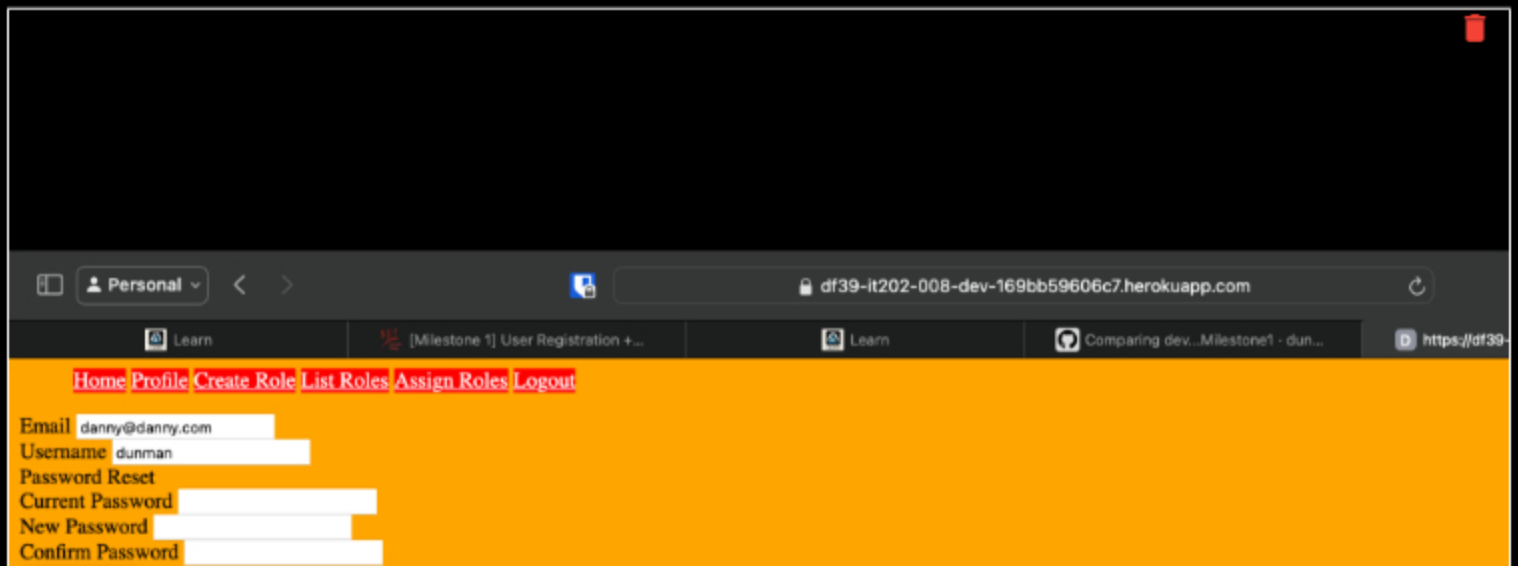
*The checkboxes are for your own tracking

#	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



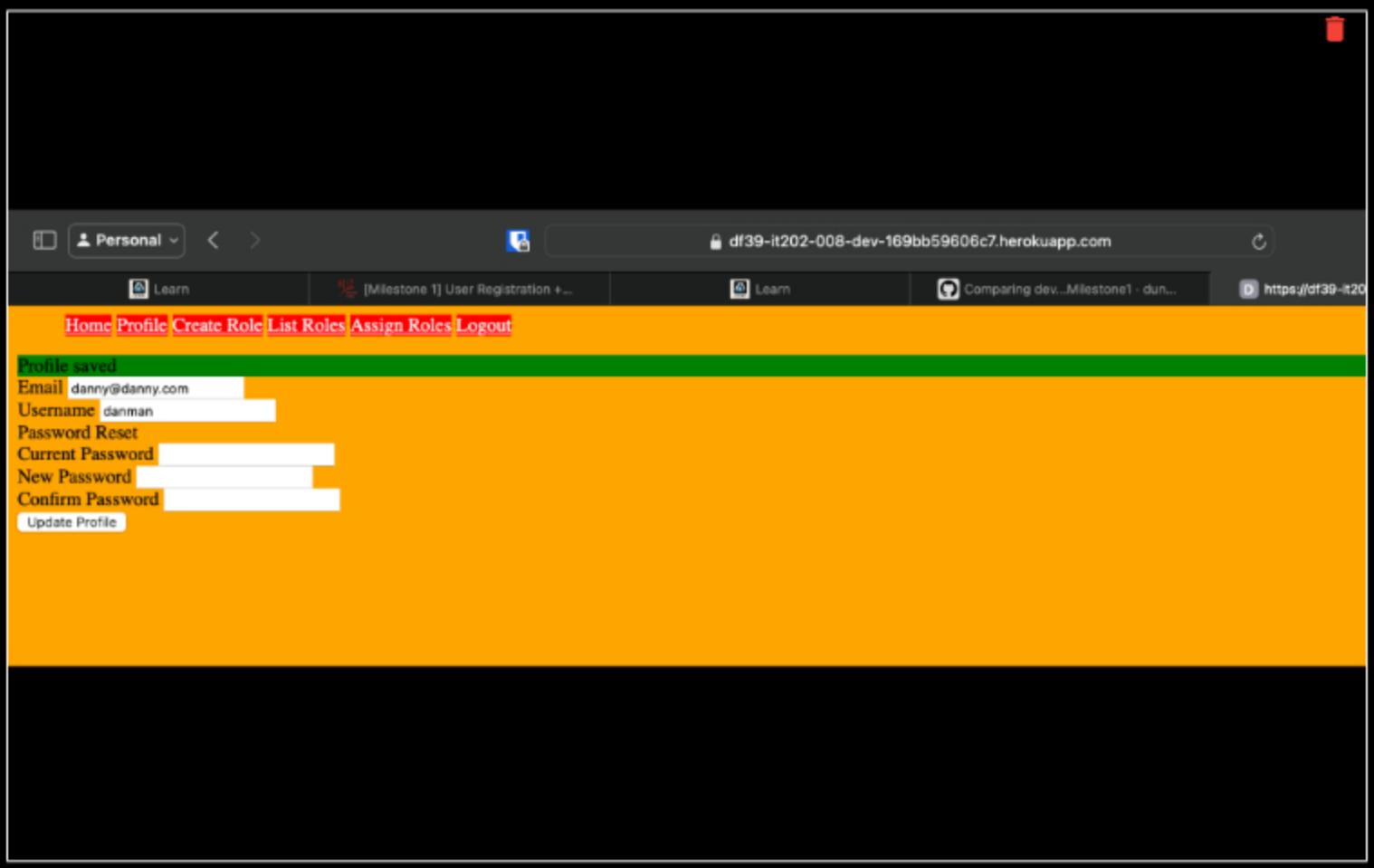
Heroku dev in URL CSS applied Before username change

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

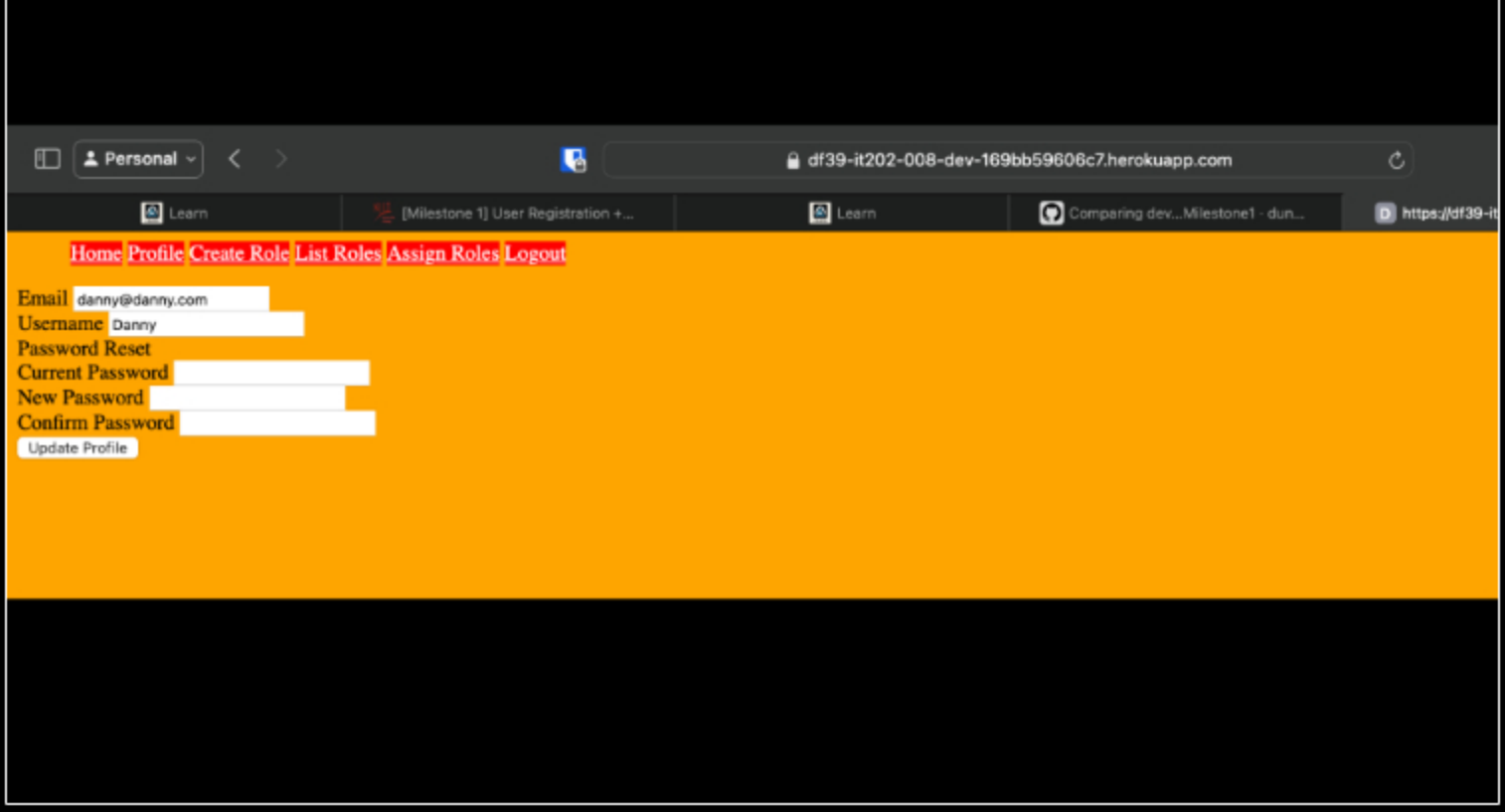
#3 Demonstrate with before and after of a username change (including success message)



After username change with success message

Checklist Items (1)

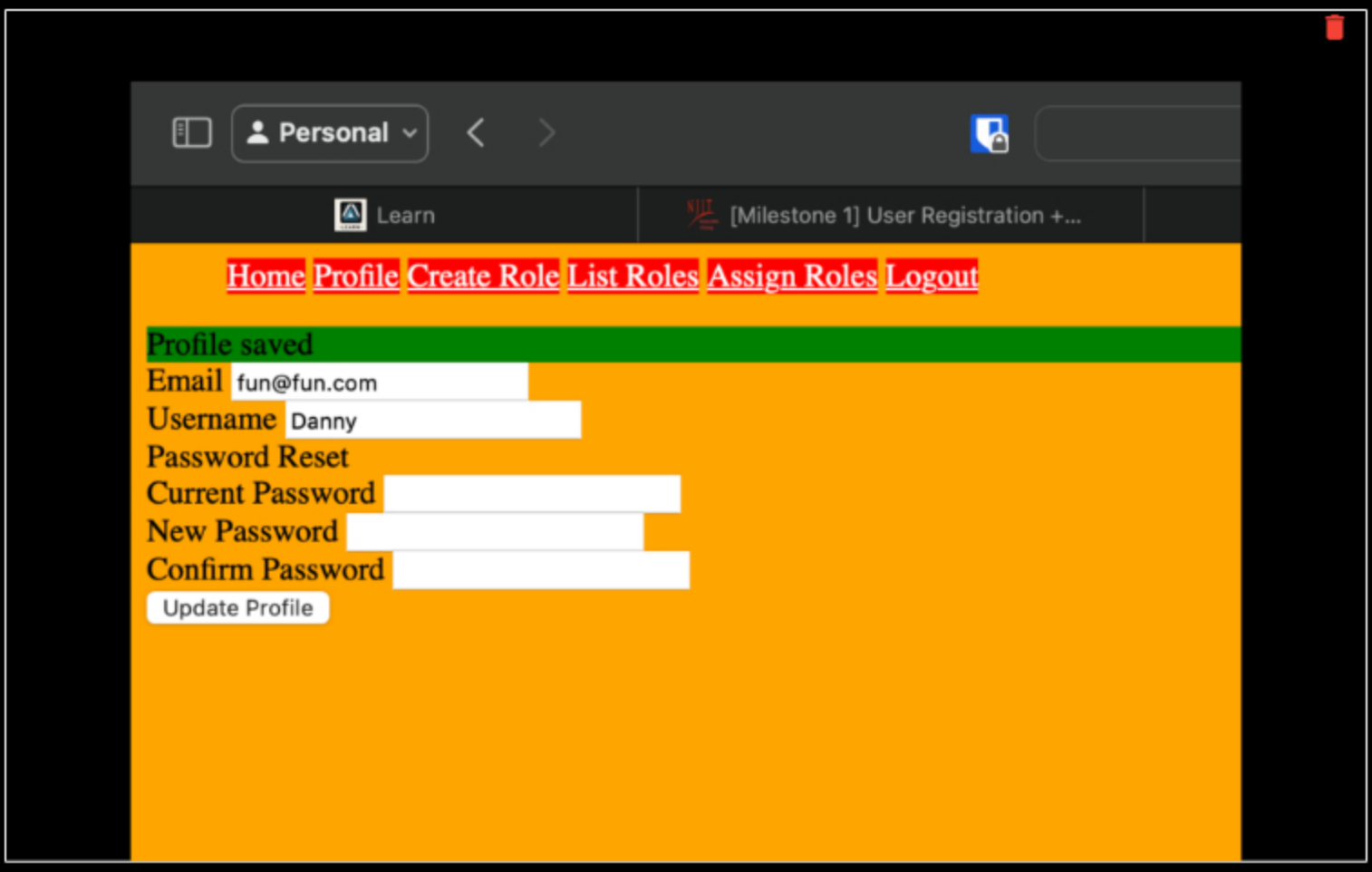
#3 Demonstrate with before and after of a username change (including success message)



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

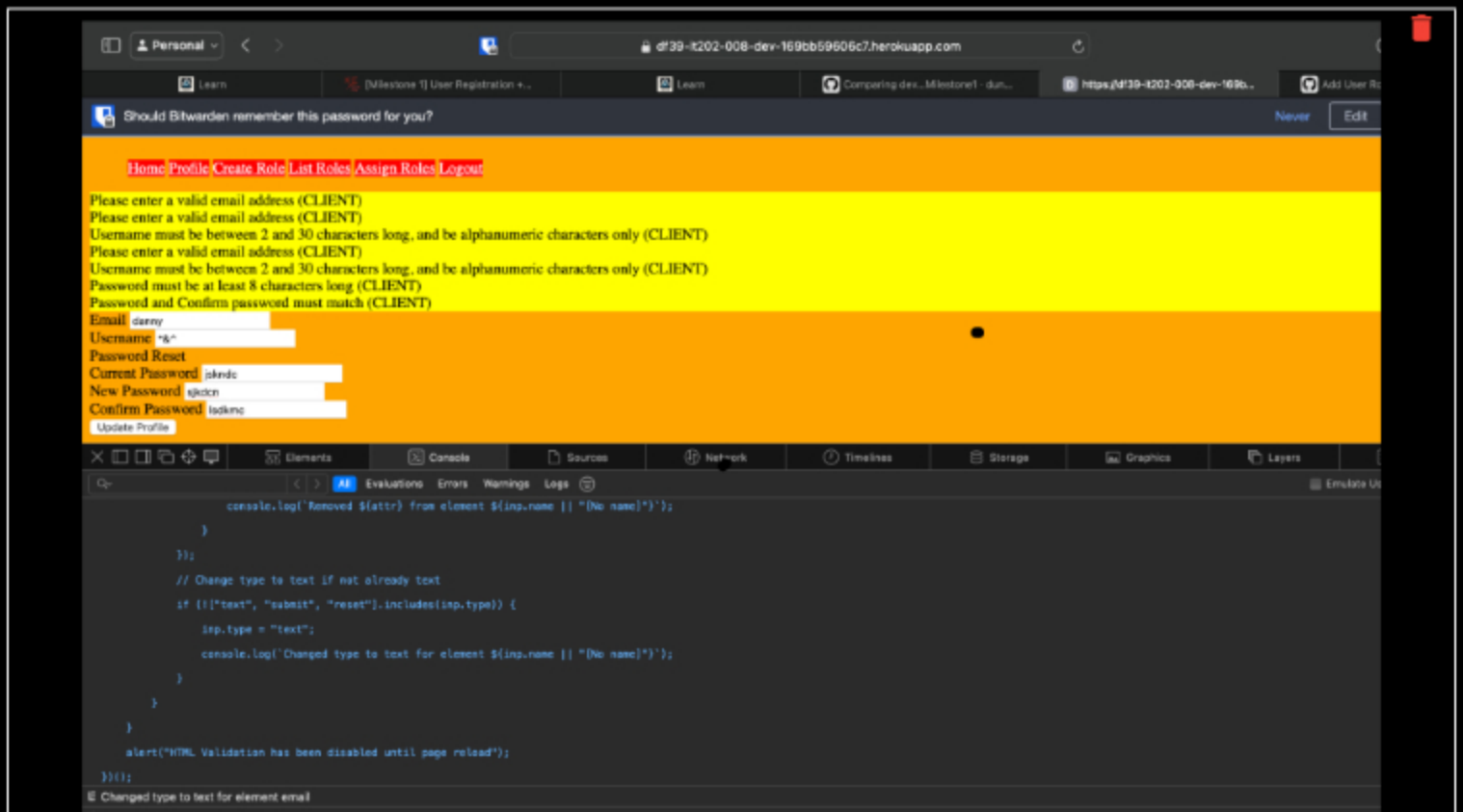
Checklist Items (1)

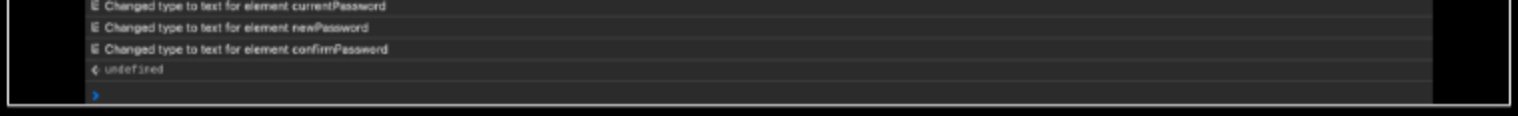
#4 Demonstrate with a before and after of an email change (including success message)

Updating password message

Checklist Items (1)

#5 Demonstrate the success message of updating password

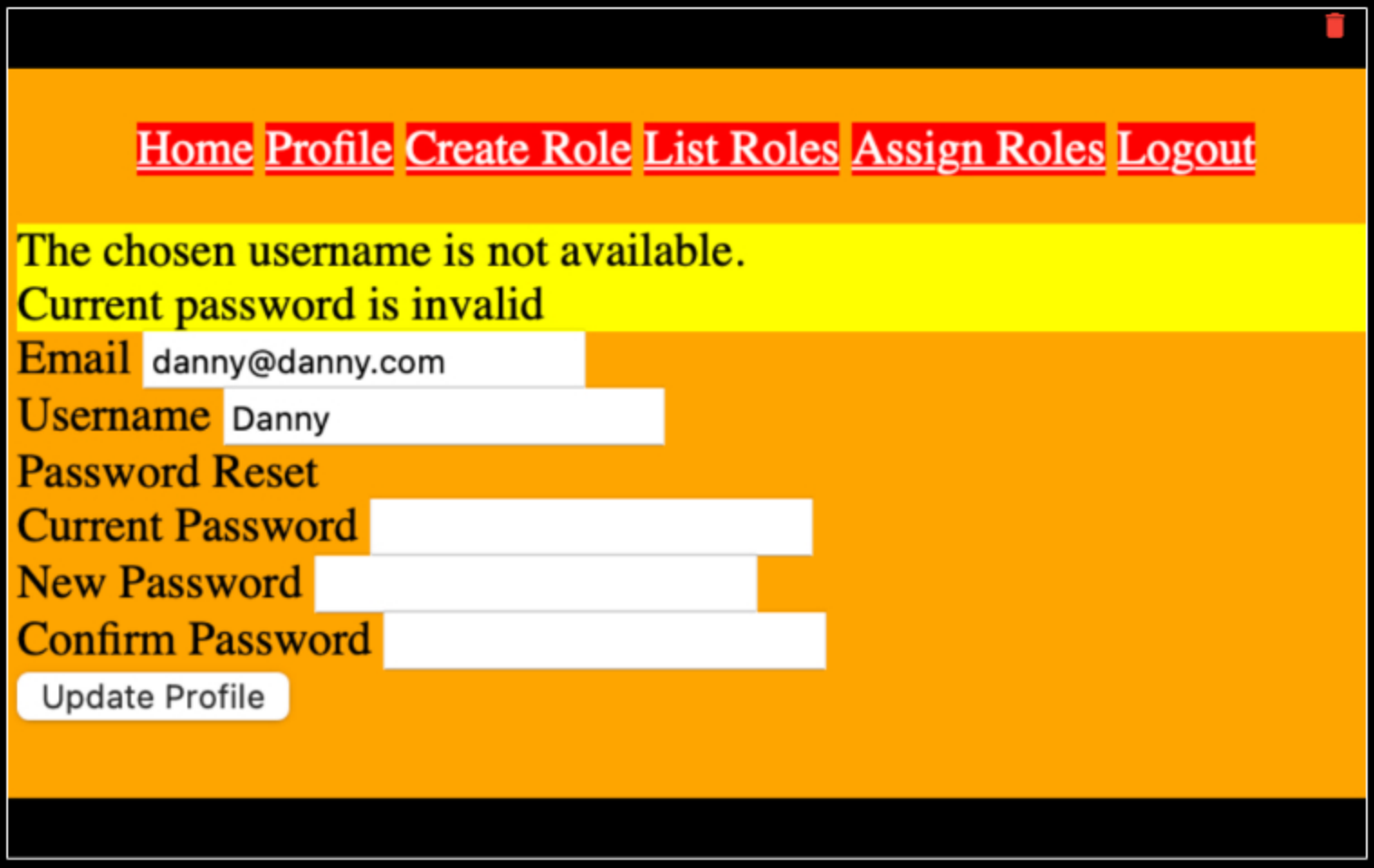




Username format message shown Email format message shown Password format message shown Confirm Password message shown

Checklist Items (1)

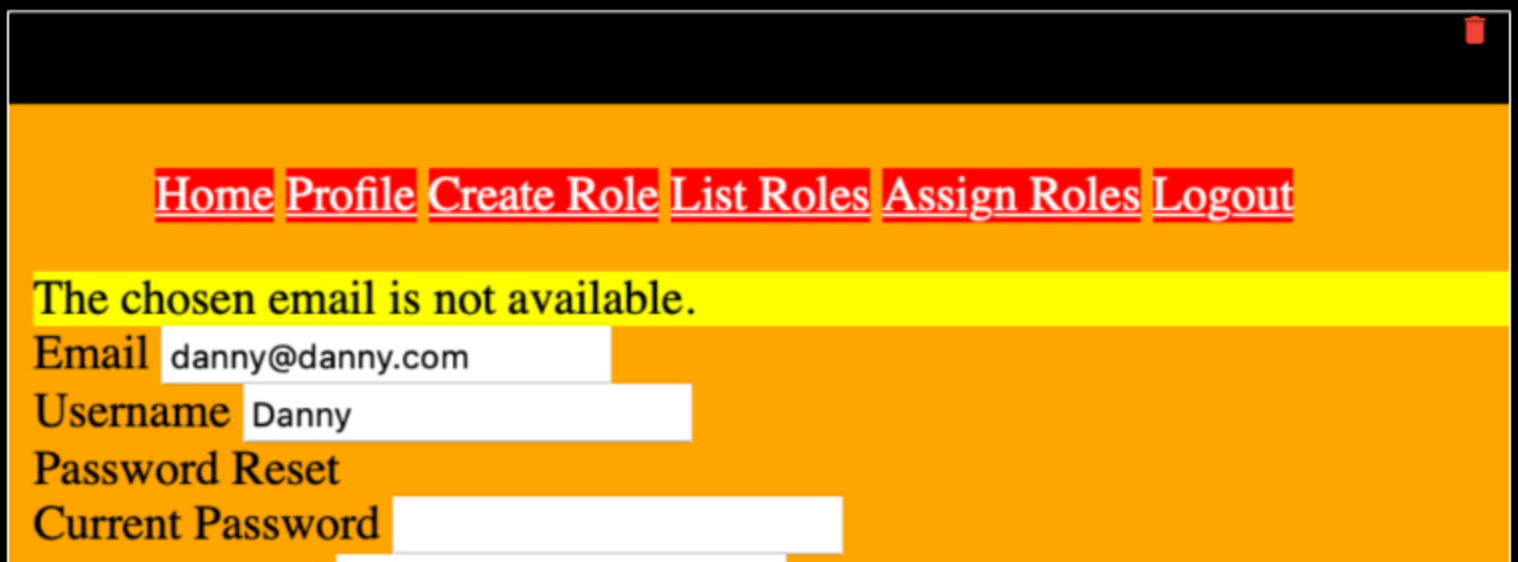
#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)



Username not available message shown Current password invalid message shown

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)



New Password

Confirm Password

Update Profile

Email not available message

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:

Username:
The PHP code will check if the form was submitted by checking if the save key is in the $_POST array, if it isn't the check fails. If the form gets submitted the username and email get sanitized by the PHP code. Before the form gets submitted client side validation will be checked to see if the format for the username/email is correct and if it isn't it will not submit the form and send a friendly flash message to the user. After the form is submitted the PHP will execute an SQL statement that updates the username and email. If an error occurs it will check if the error is 1062 which means a duplicate entry has been made and will display a friendly message to the user saying the entry is already in use. If there is no duplicate entry the username and email will be updated. PHP will then fetch these changes and store the user data in the session.

Password:
Before the form is submitted the client side validation (java script) will be called after the user attempts to submit the form and ensure formatting for the password is correct and that the password and confirm password fields are matching. If it isn't the form will not be submitted. Once the format is correct the form will be submitted. Server side will also check if the passwords match. If they match, a SQL statement will fetch the users current password from the data base, if the current password entry matches the user entered current password SQL statement is ran to update the users password in the data base. If they don't match a message will appear telling the user the current password is incorrect.

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/dunnfall/df39-it202-008/pull/31

Misc (1 pt.)

COLLAPSE

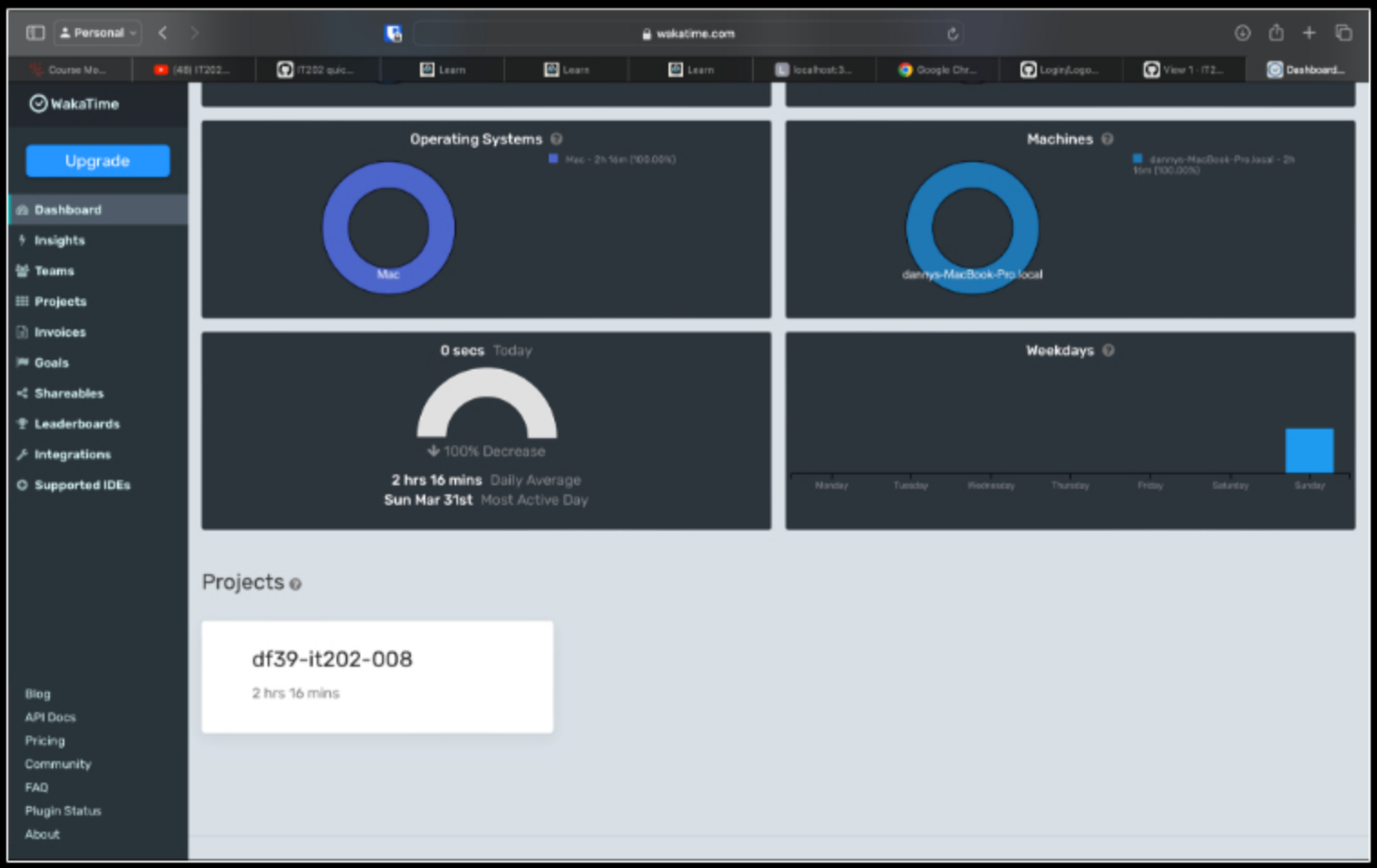
Task #1 - Points: 1

Text: Screenshot of wakatime

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Screenshot of Wakatime

COLLAPSE

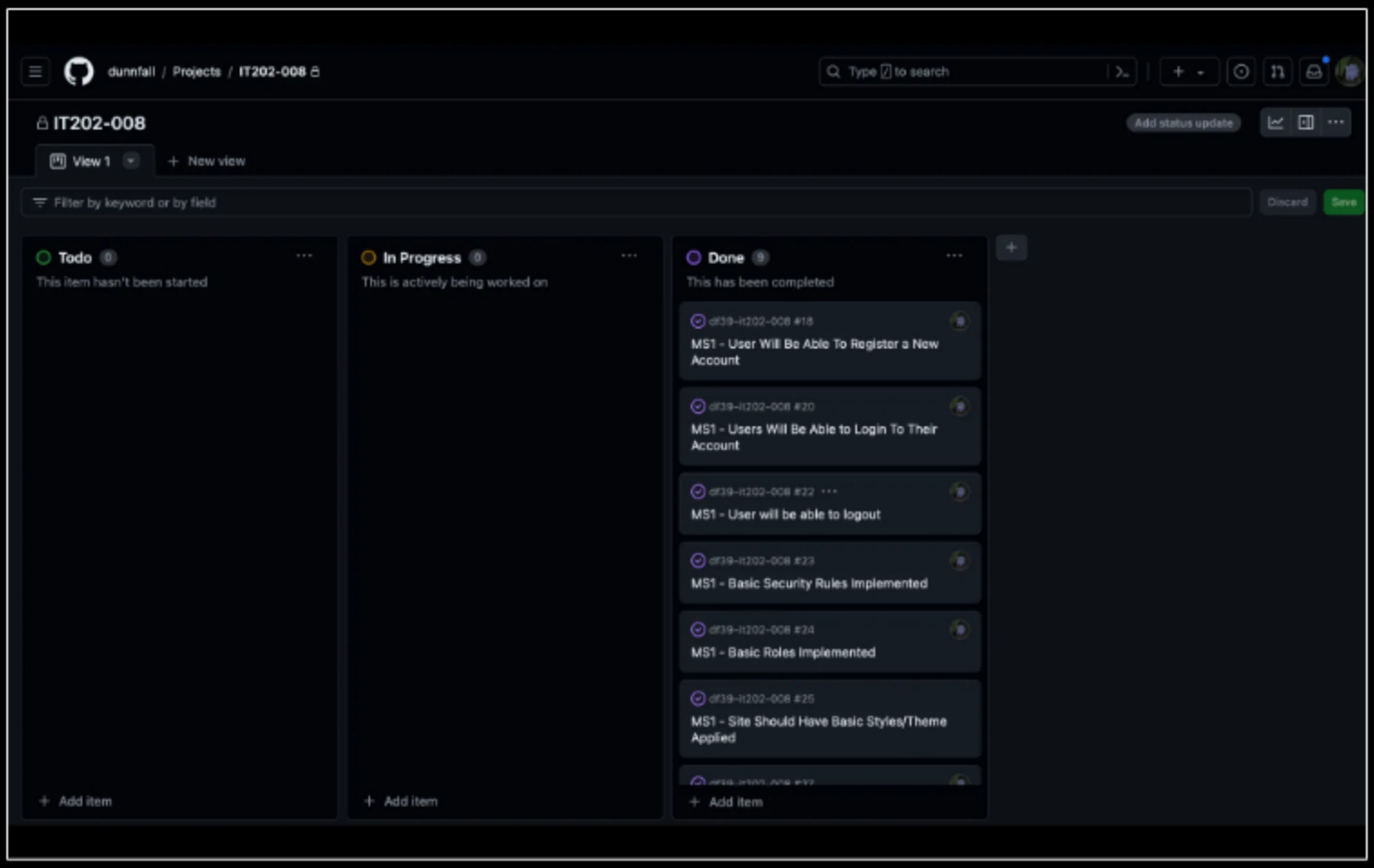
Task #2 - Points: 1

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 for Milestone 1

COLLAPSE

Task #3 - Points: 1

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

URL #1

https://github.com/users/dunnfall/projects/1

COLLAPSE

Task #4 - Points: 1

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

URL #1

https://df39-it202-008-prod-8b39da6847aa.herokuapp.com/Project/login.php

End of Assignment