Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/ekh3

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

Submissions:

Submission Selection

1 Submission [active] 4/1/2024 1:52:32 AM

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

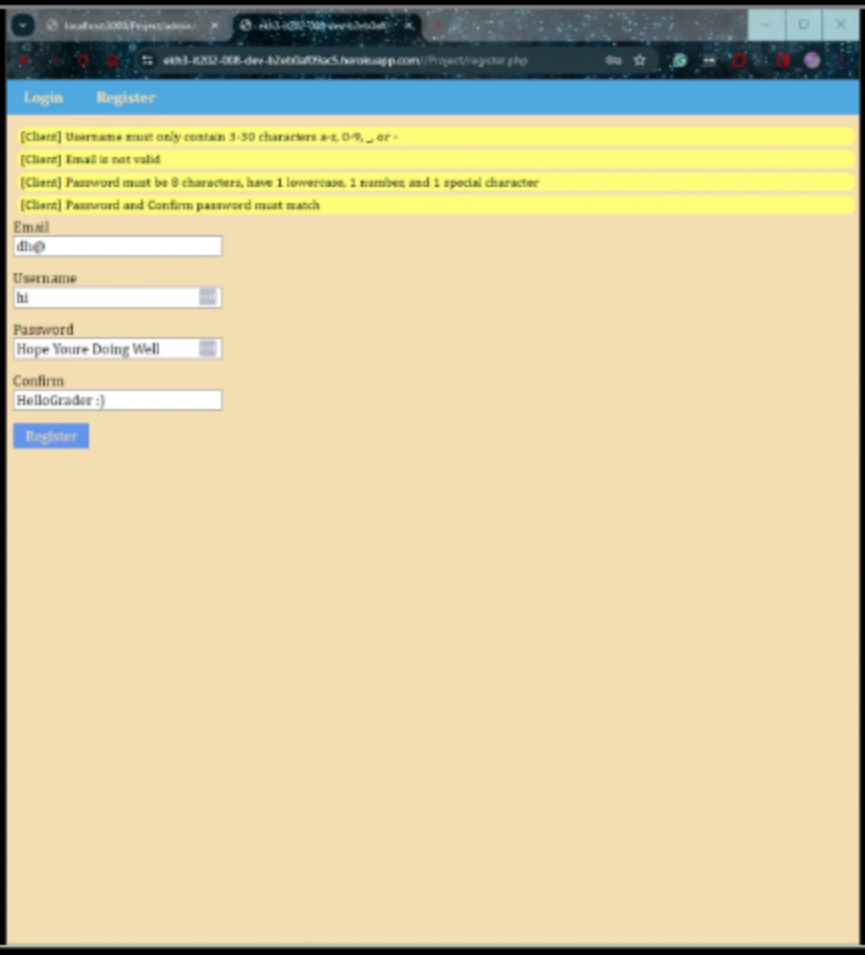
*The checkboxes are for your own tracking

#	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



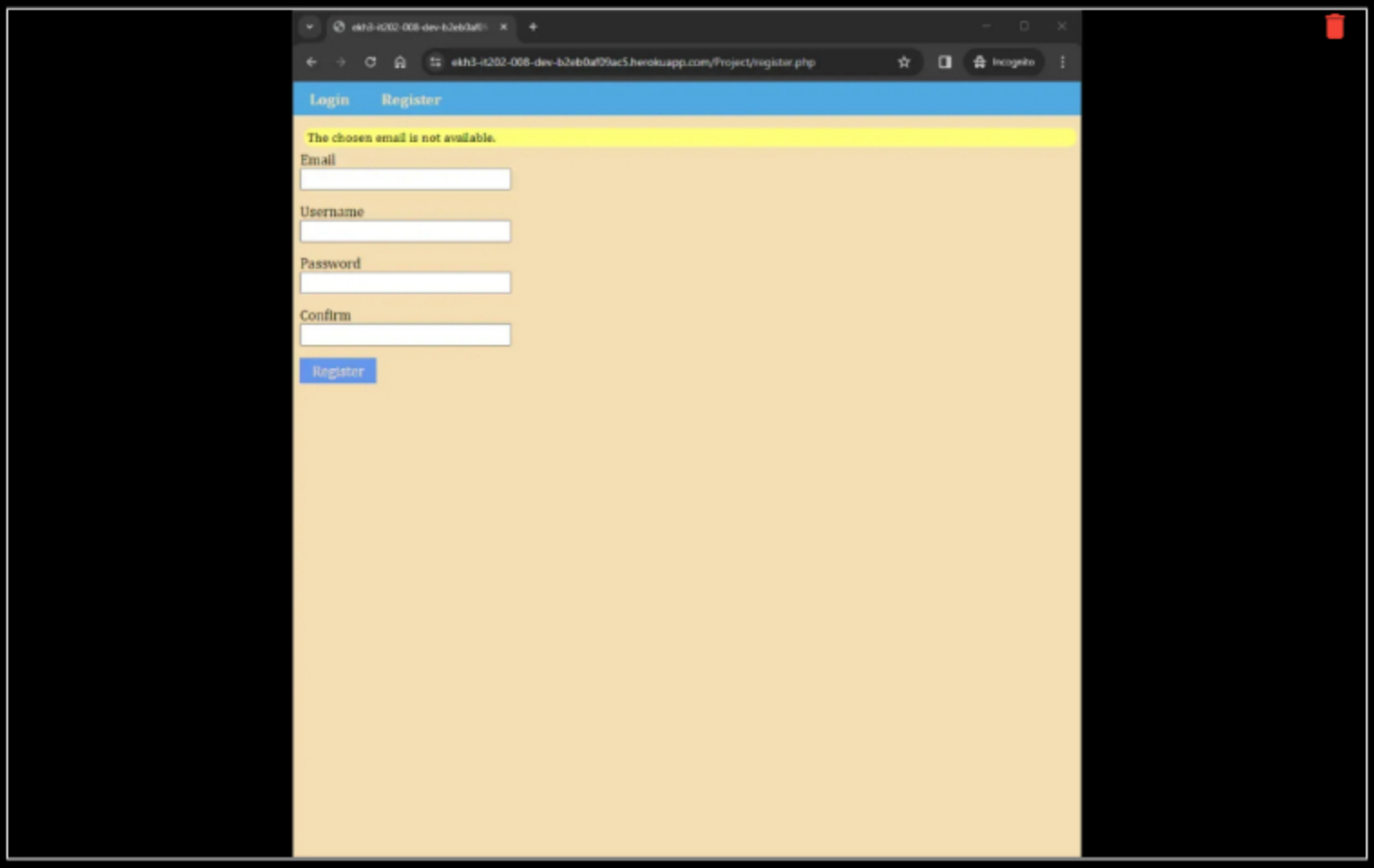
Demonstrates the JS validation for email, username, and password formats with the matching password requirements.

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

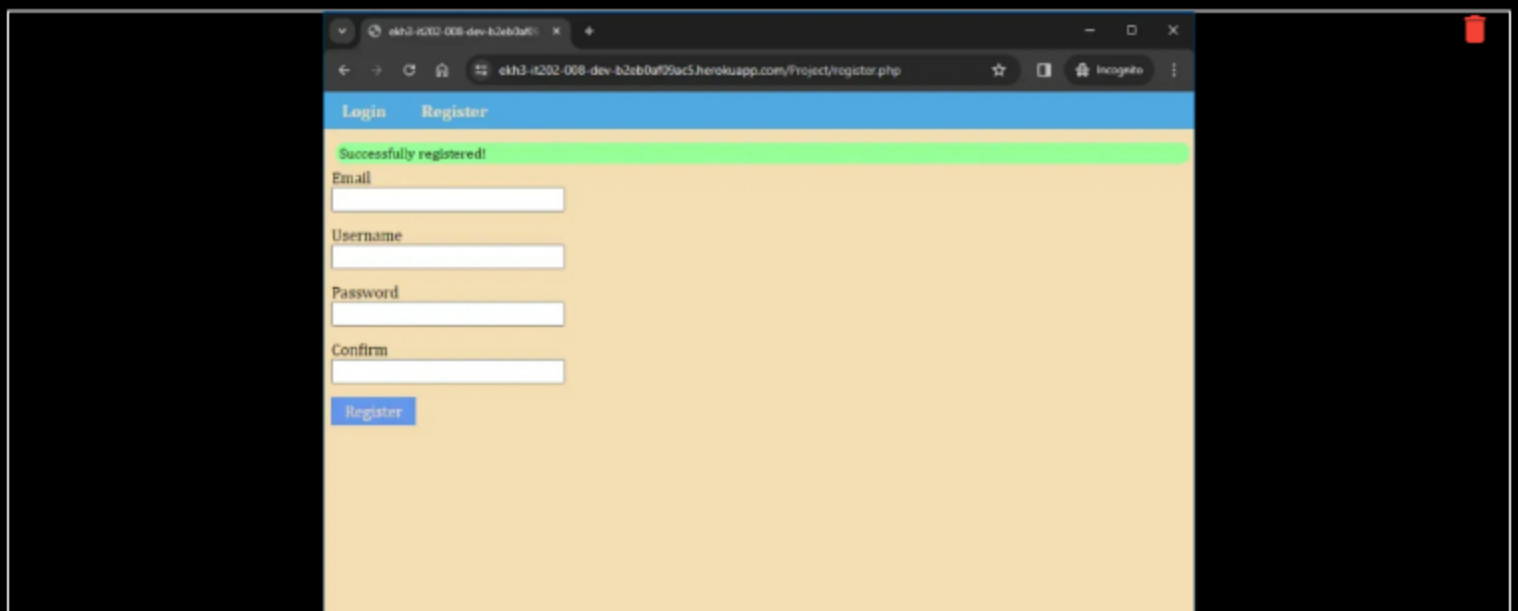
#3 Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required



Email in use

Checklist Items (1)

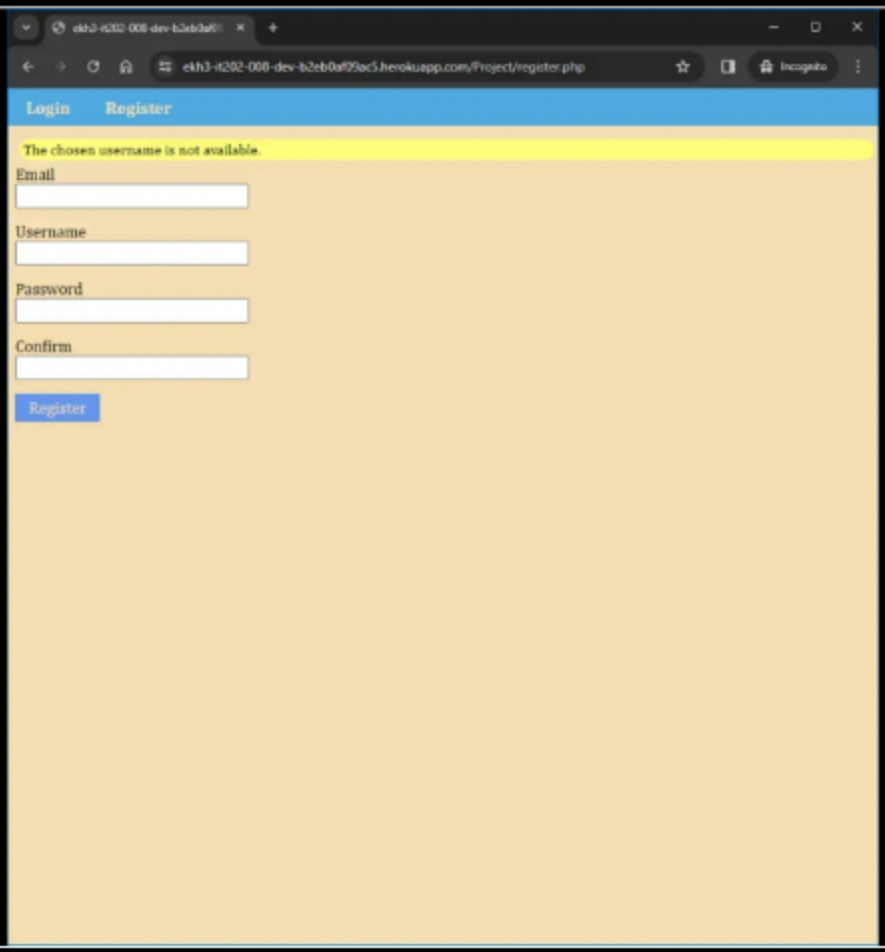
#4 Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)



Register success

Checklist Items (1)

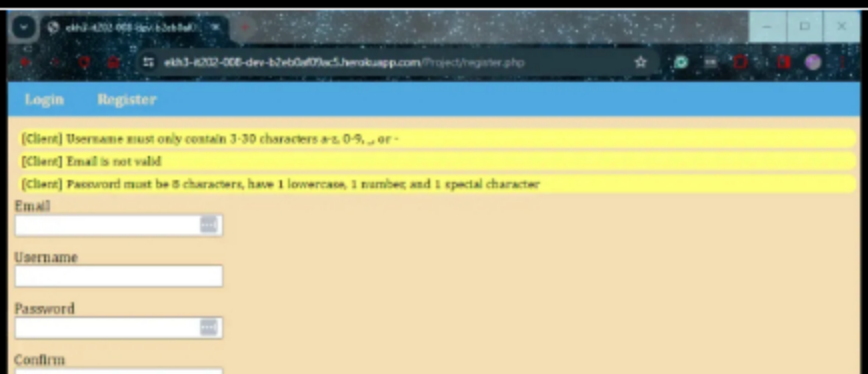
#6 Demonstrate user-friendly message of new account being created

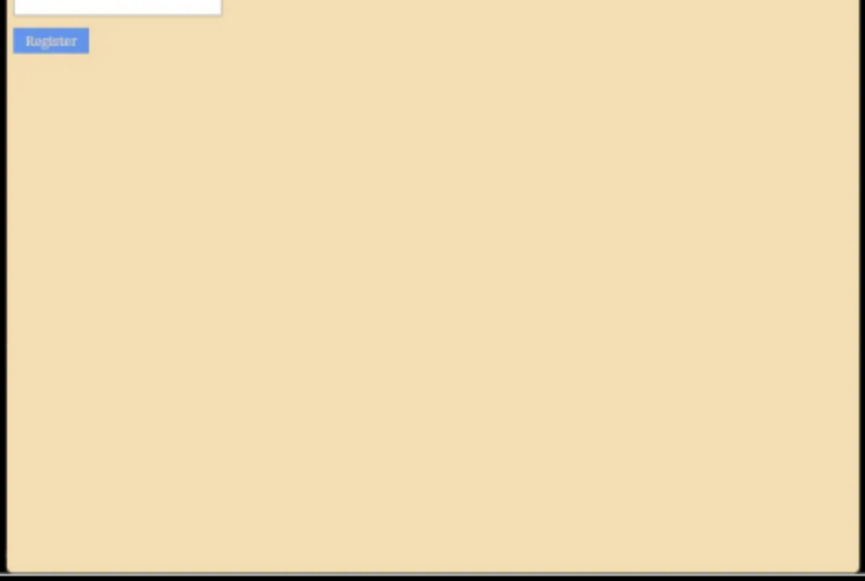


Username in use

Checklist Items (1)

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





Register required fields (if blank)

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

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

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

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

Form code for register

COLLAPSE

Task #3 - Points: 1

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

Checklist

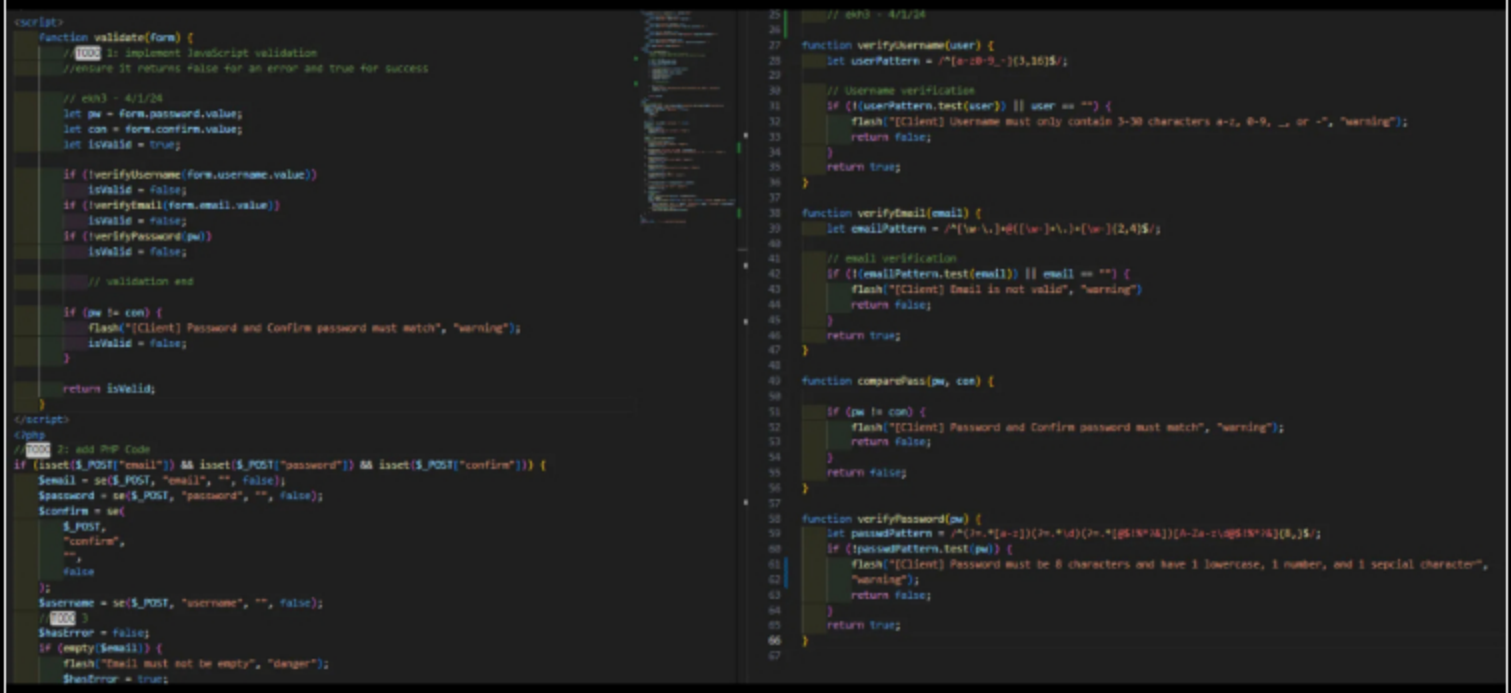
*The checkboxes are for your own tracking

#	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

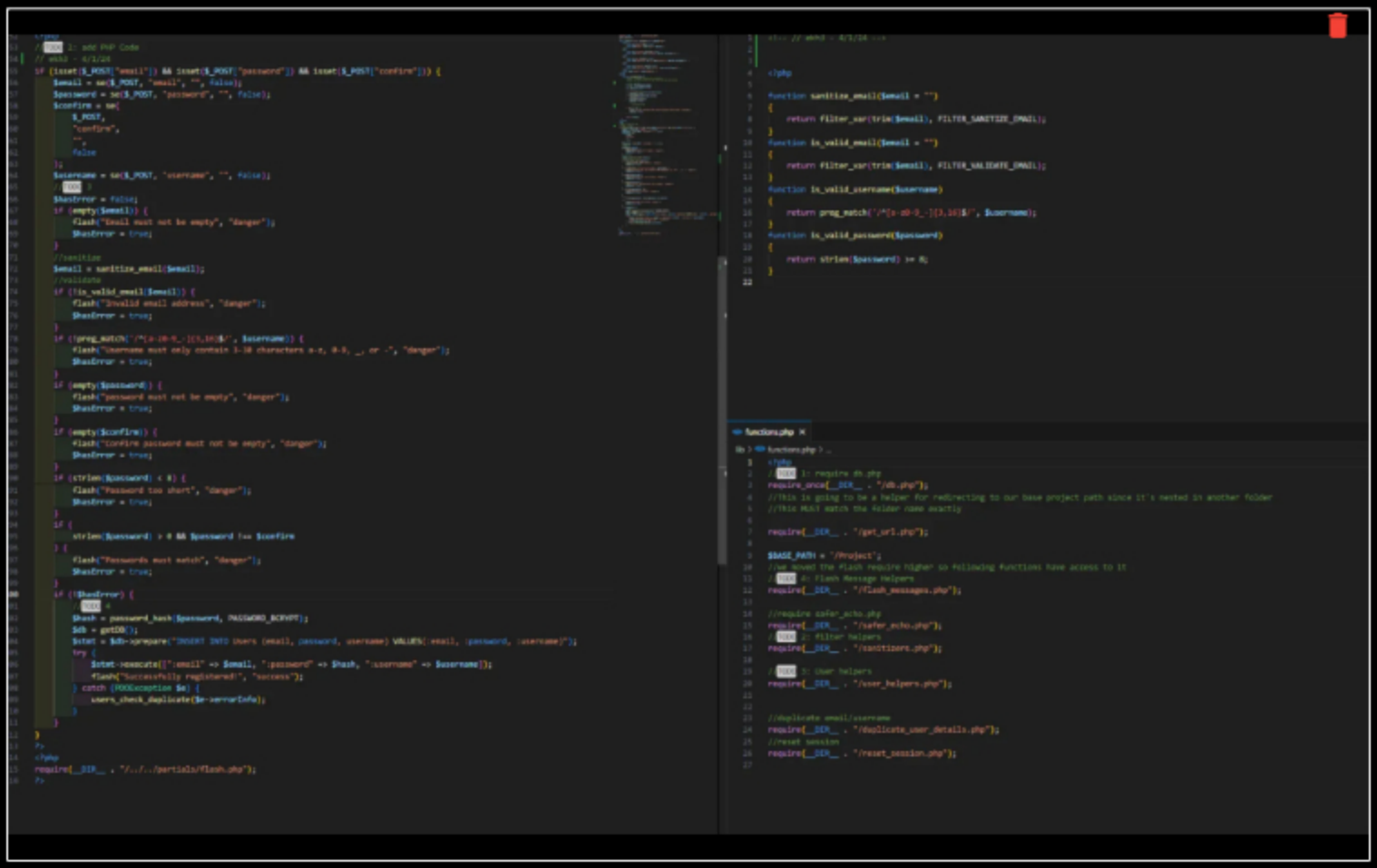
Small Medium Large



JS code

Checklist Items (1)

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

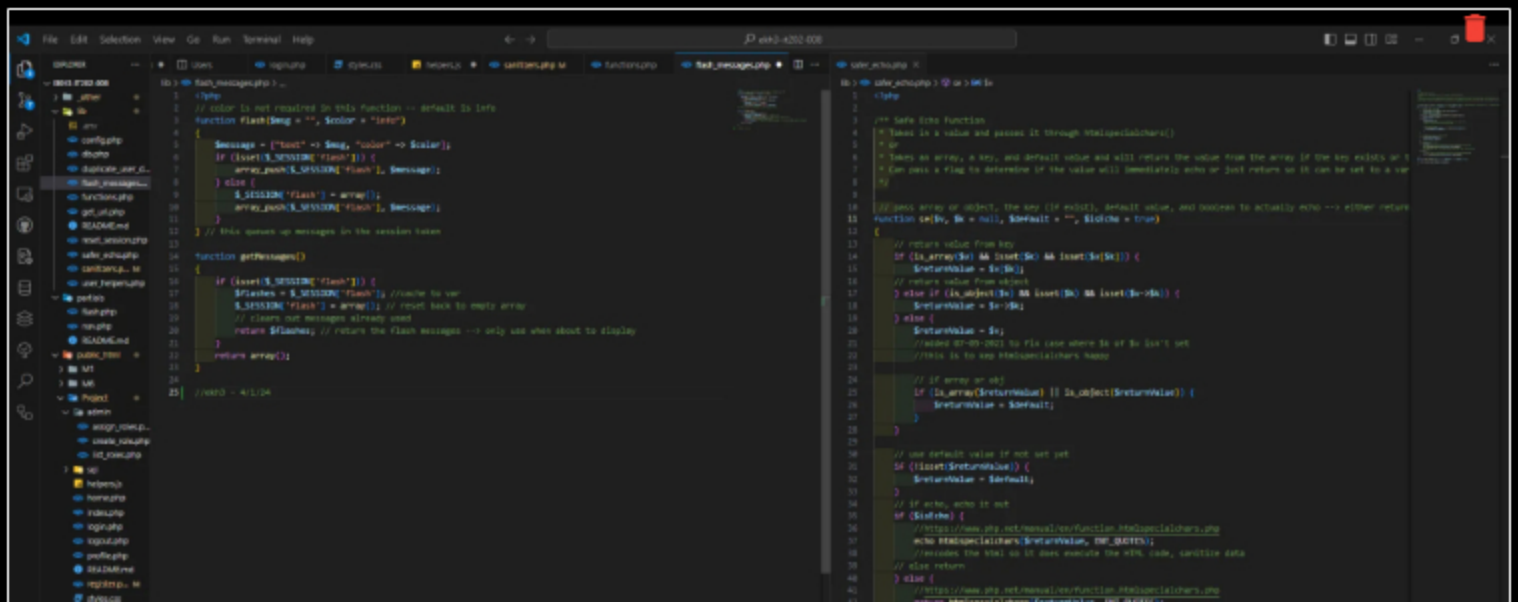


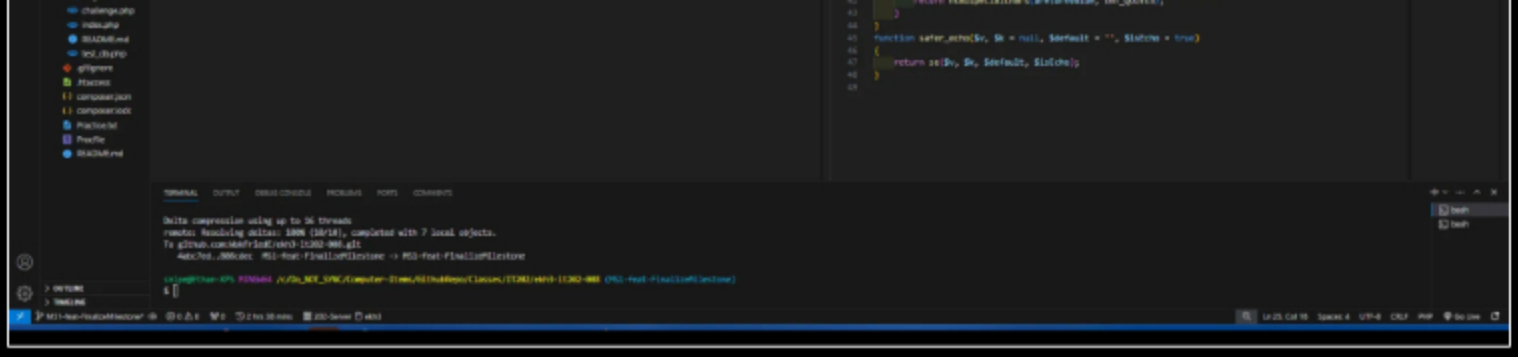
Register php code

Checklist Items (2)

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

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





Extra php functions

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

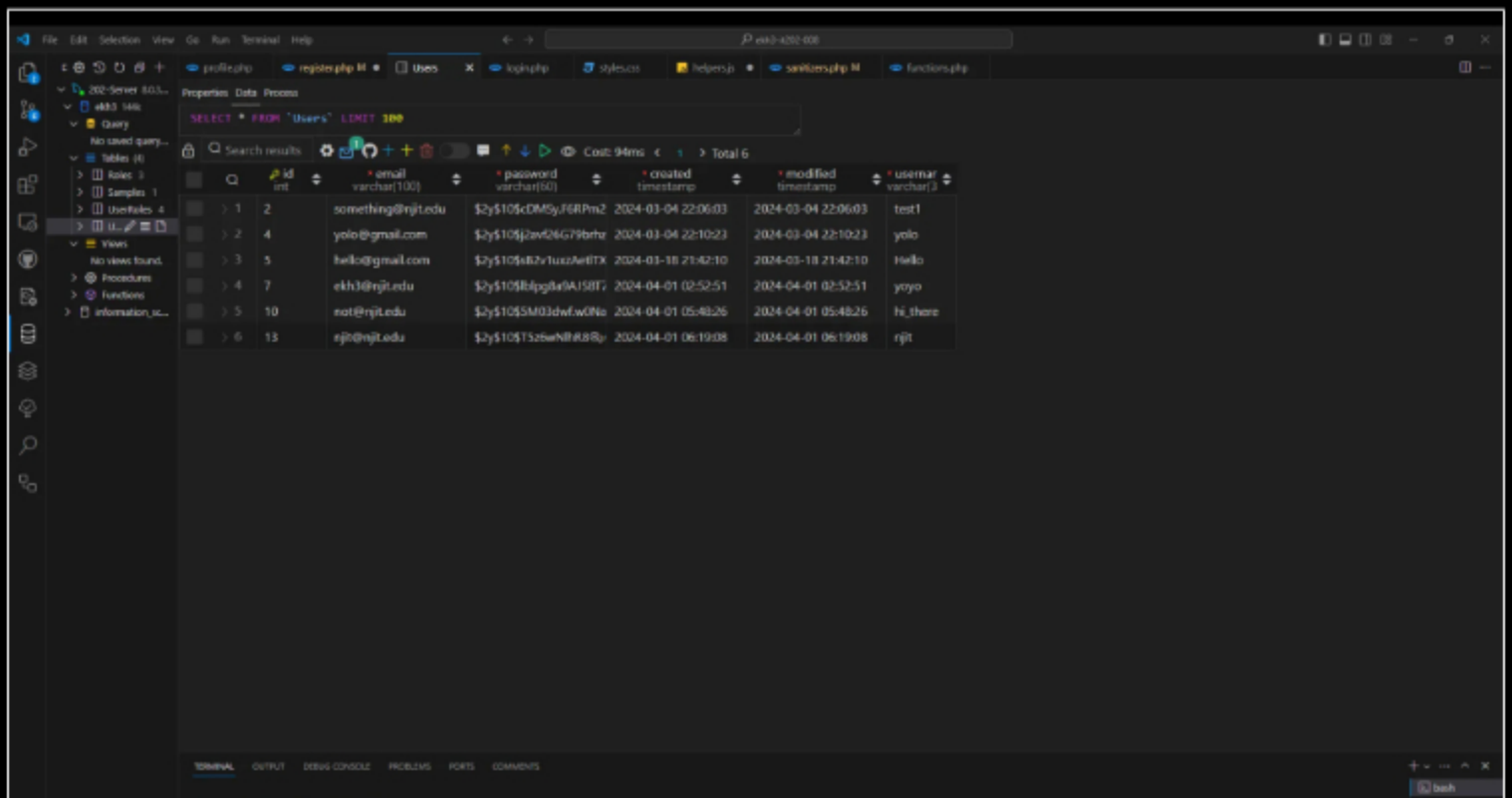
*The checkboxes are for your own tracking

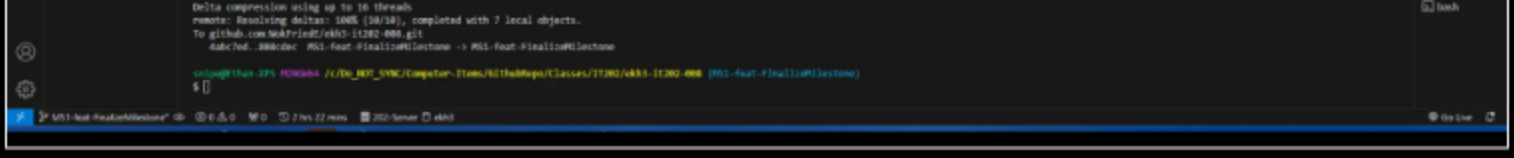
#	Points	Details
#1	1	Password should be hashed
#2	1	Should have email, password, username (unique), created, modified, and id fields
#3	1	Ensure left panel or database name is present (should contain your ucid)

Task Screenshots:

Gallery Style: Large View

Small Medium Large





User table with hashed password, unique values

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:

For registration, the user would be able to set their registration in the register page. Within the register page there would be 4 input box fields in a form which allows the user to put in their information. After entering their information and clicking register, the information from the form would initiate the validate() function in JS. It would try to validate all the patterns for each item, making sure the proper regex exists, using regex.test(str). If a field is not valid, it would trigger the a flash message and not submit as the return would be false.Otherwise it would go through and the JS returns true, the PHP validations would check again stopping the process and alerting them if anything is incorrect. Lastly, it would attempt to submit it to the database and if there is a duplicate value for a unique field, it would alert the user that the username or email is unavailable. Otherwise, the results would be sent to the databse and the password is hashed using bycrpyt method beforehand.

COLLAPSE

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/WokFriedE/ekh3-it202-008/pull/16

User Login (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of form on website page

Checklist

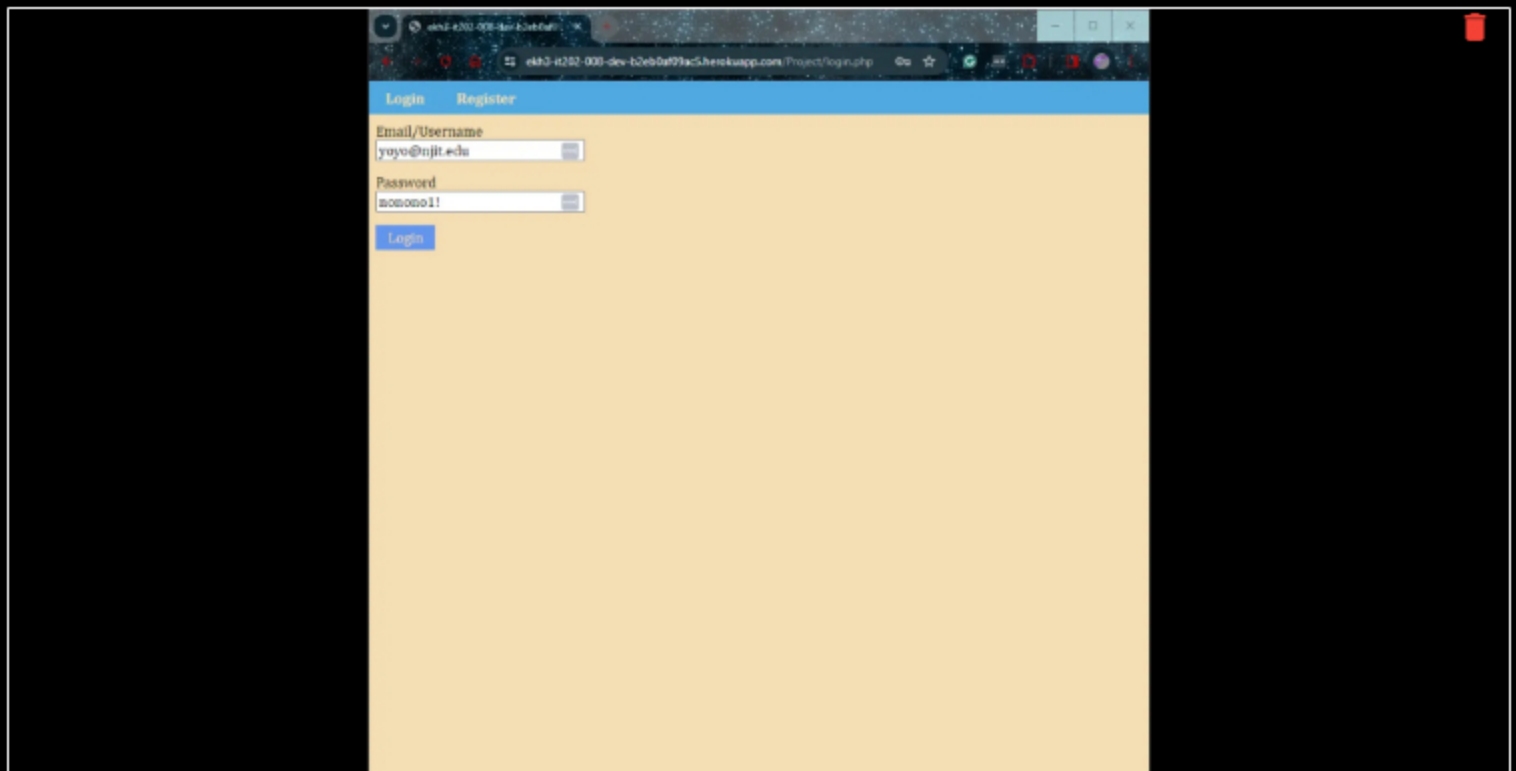
*The checkboxes are for your own tracking

#	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

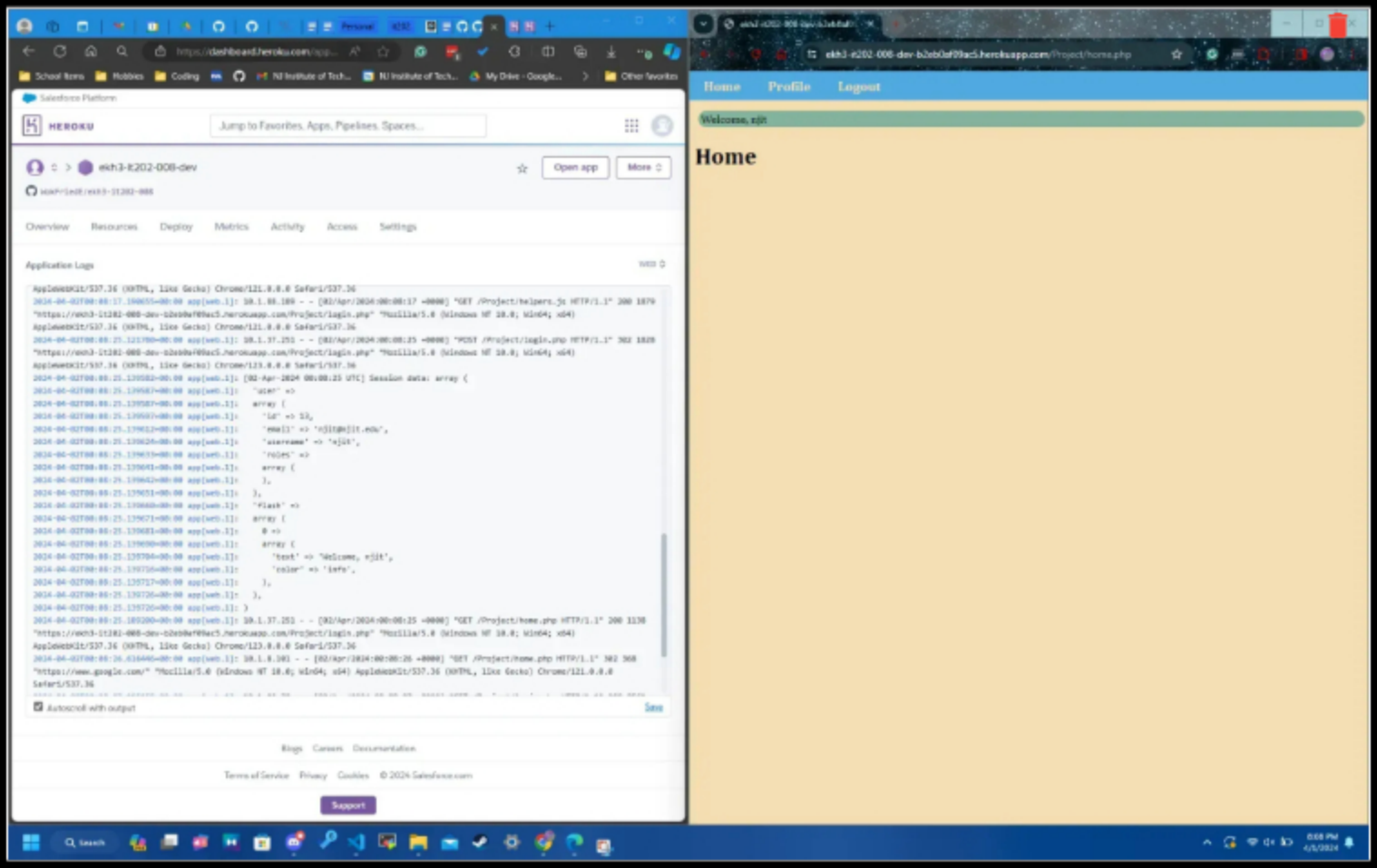


valid email (admin login)

Checklist Items (2)

#2 Should have thoughtful CSS applied

#4 Include correct data where email is used to login

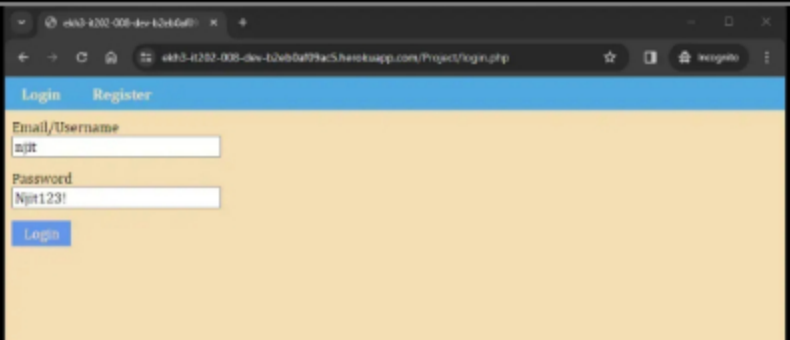


Successful login with the error log output of the session

Checklist Items (2)

#5 Show success login message

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



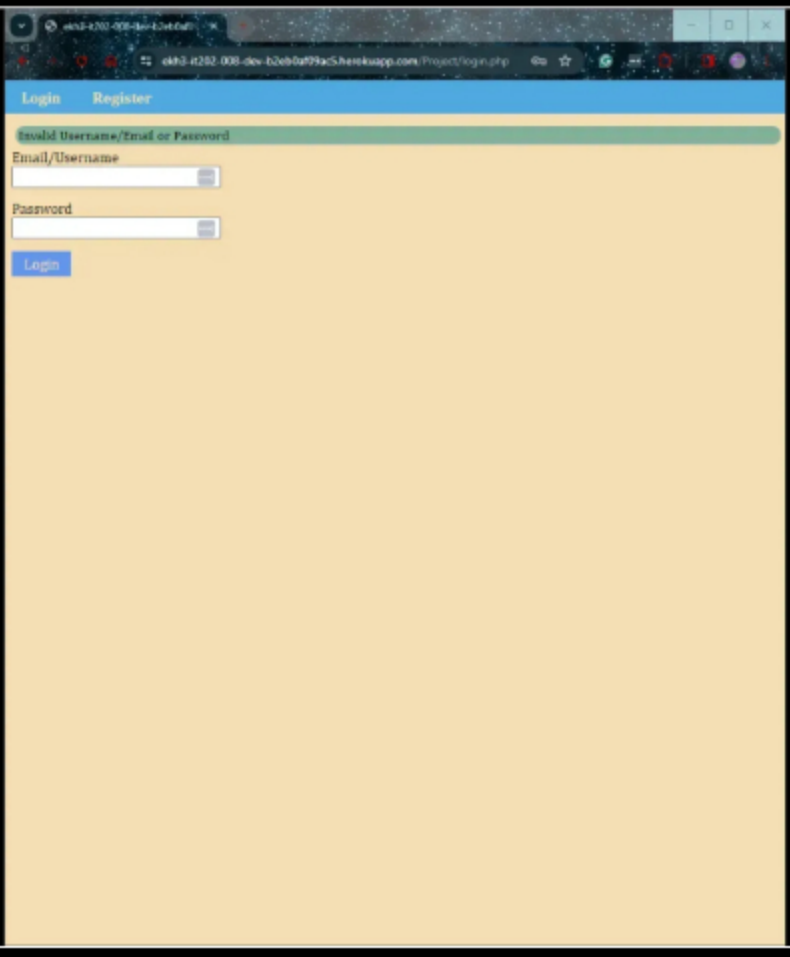
Proper username login (default user login)

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

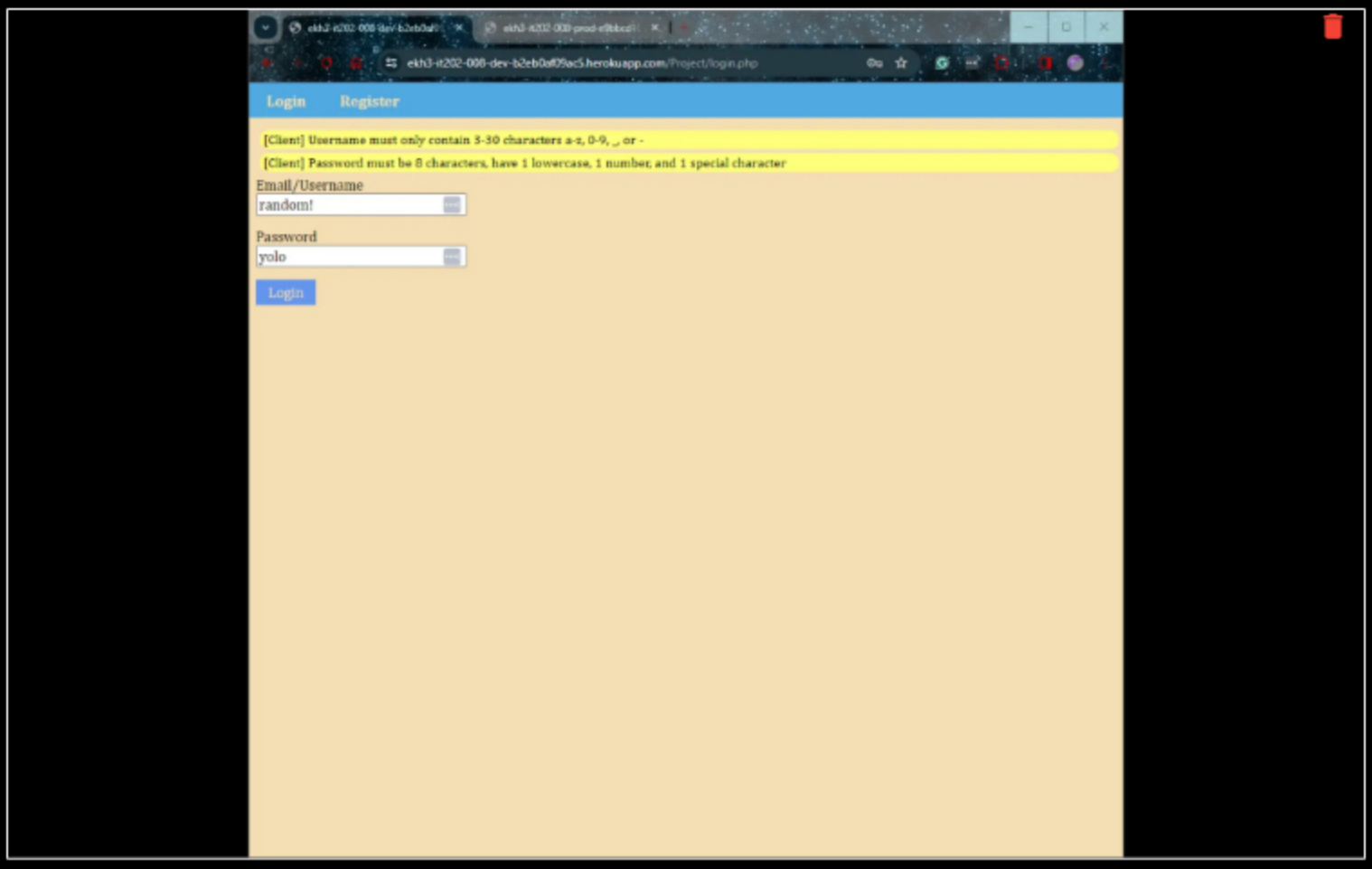


Message for invalid password and username/email does not exist.

Checklist Items (2)

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

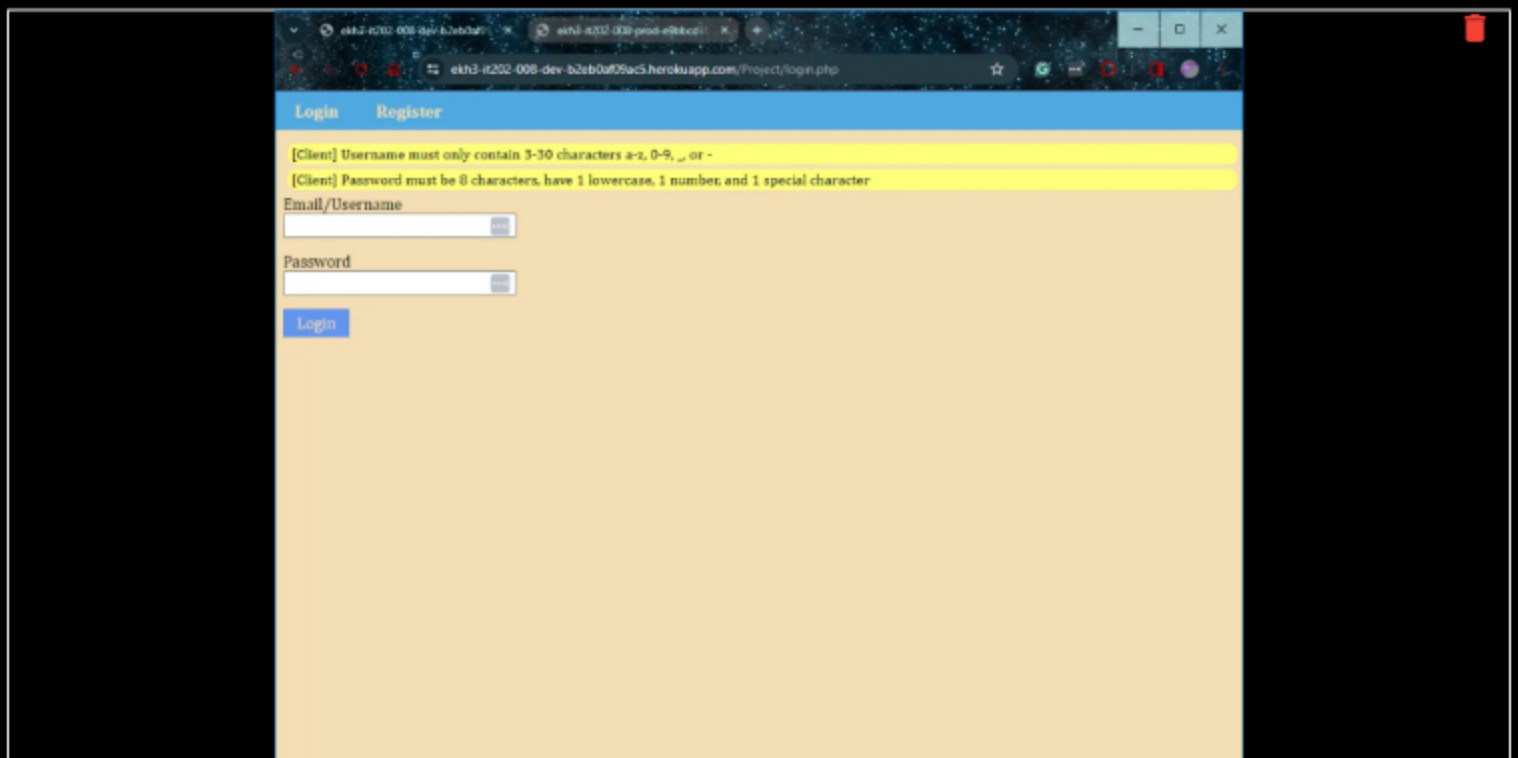
#8 Demonstrate user-friendly message of when password doesn't match what's in the DB



Validation for format for password and username

Checklist Items (1)

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



Validation with blank fields (enforces requirement)

Checklist Items (1)

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

^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

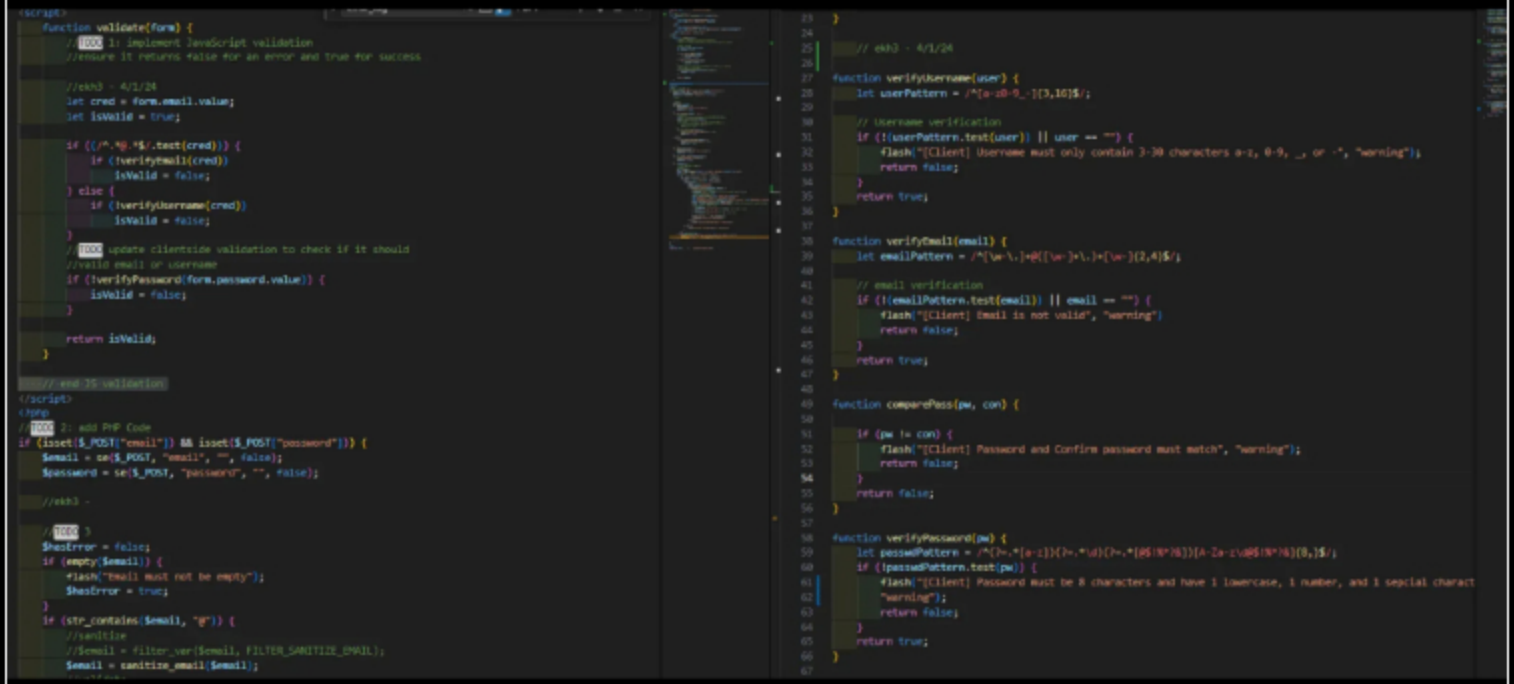
```
<!-- ekh3 - 4/1/24 -->
<form onsubmit="return validate(this)" method="POST">
    <div>
        <label for="email">Email/Username</label>
        <input type="text" name="email" required />
    </div>
    <div>
        <label for="pw">Password</label>
        <input type="password" id="pw" name="password" required minlength="8" />
    </div>
    <input type="submit" value="Login" />
</form>
```

Login form code (used one text input for email and username)

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)

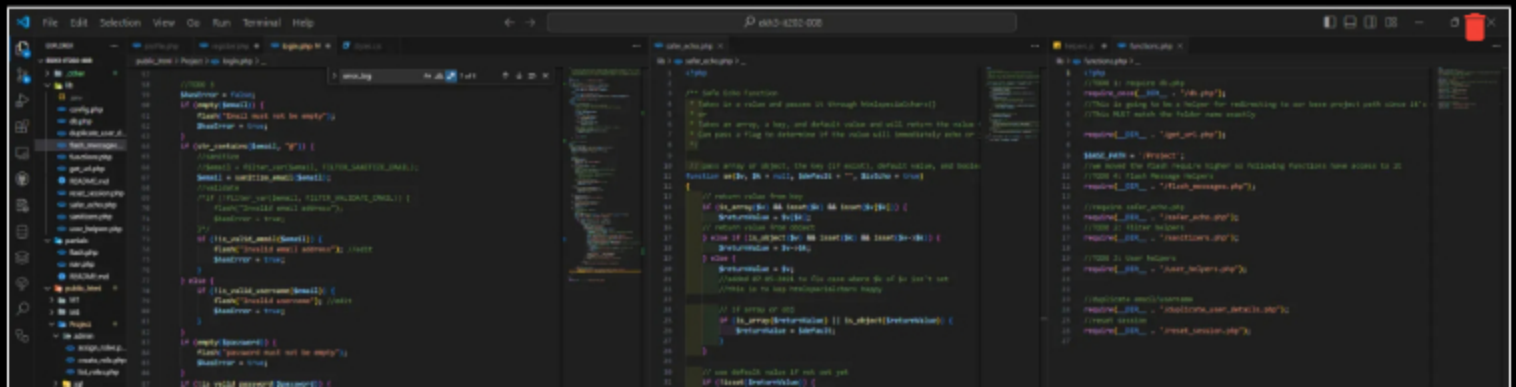


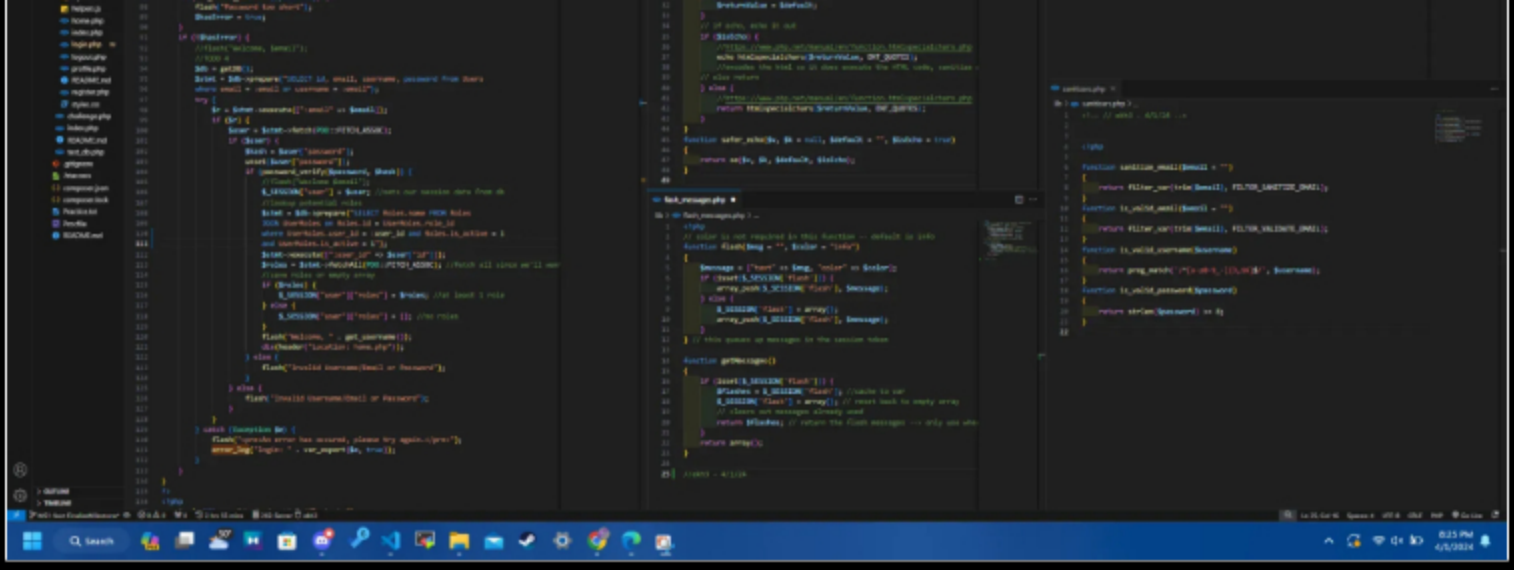
Log in code - JS validation

Checklist Items (2)

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

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





PHP verification code - safe_echo.php is not used

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 the user first enters the web app, they will be directed to the login.php screen where they will be allowed to enter their username/email and password. After entering the information and clicking the login button, it would validate the values. For the username, it would check if the credential has an @ symbol, if so, check the email verification in JS. If it does not have an @, check for the username validation in JS. Afterward, it checks for the password with JS, just like the credentials. If all is right, then the form will be submitted, otherwise, it will prompt flash messages and deny the form submission. If the username or email does not exist it will prompt the user saying "Invalid Username/Email or Password." The same would be done if the password does not match the credential. If the submission is successful, then PHP will do validations using regex, similar to JS. If anything messes up, a flash message will be prompted. If everything validates, a session will be created for the nav.php to call in other pages.

The cookie and session will stay in the local cache, working as an associative array where the user information is stored. It is being used so that other pages, like the profile, can use the information from the login page. In addition, using the roles tables set, it would assign the role to the user to change their view accordingly. The way it works is that when the page opens, the navbar would pull the session cookie. The PHP on each page will use that session key to get information and change accordingly.

COLLAPSE

Task #4 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/WokFriedE/ekh3-it202-008/pull/24

User Logout (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Capture the following screenshots

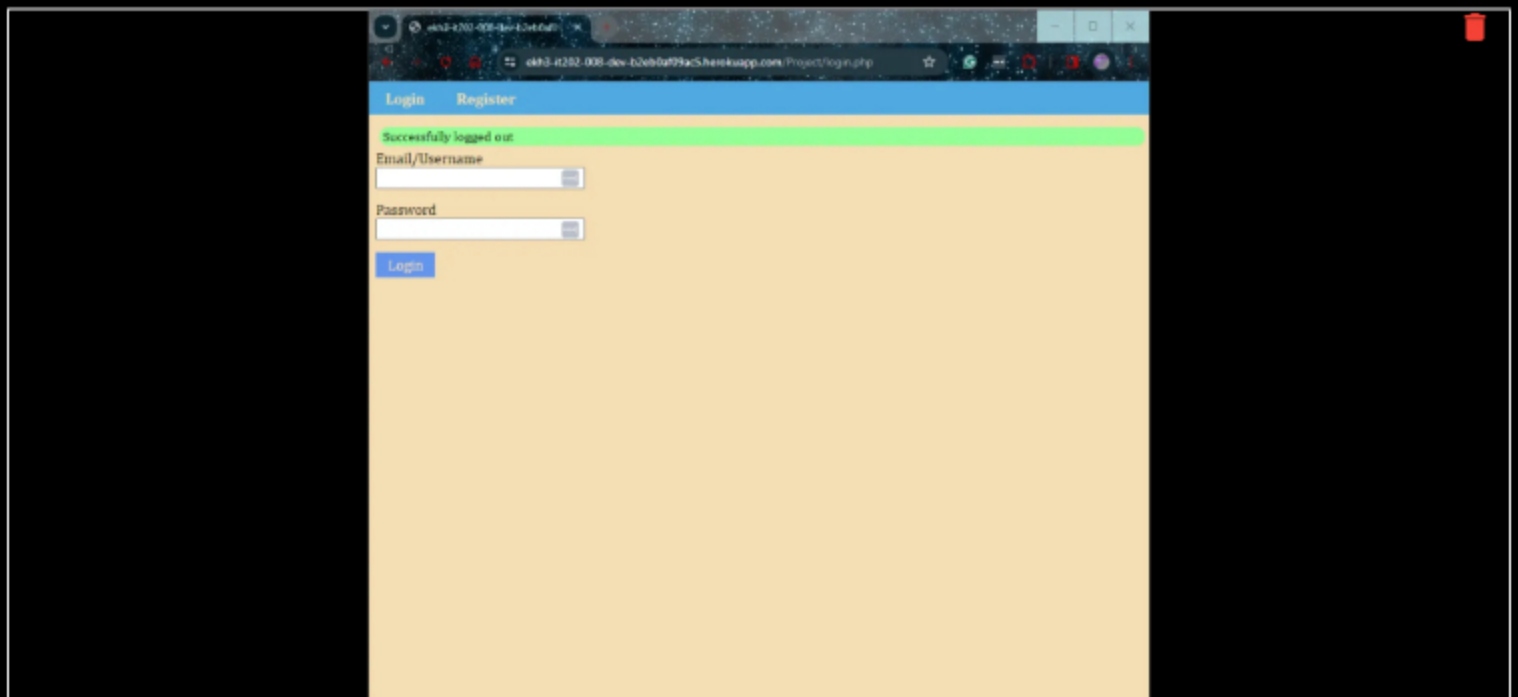
Checklist *The checkboxes are for your own tracking

#	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

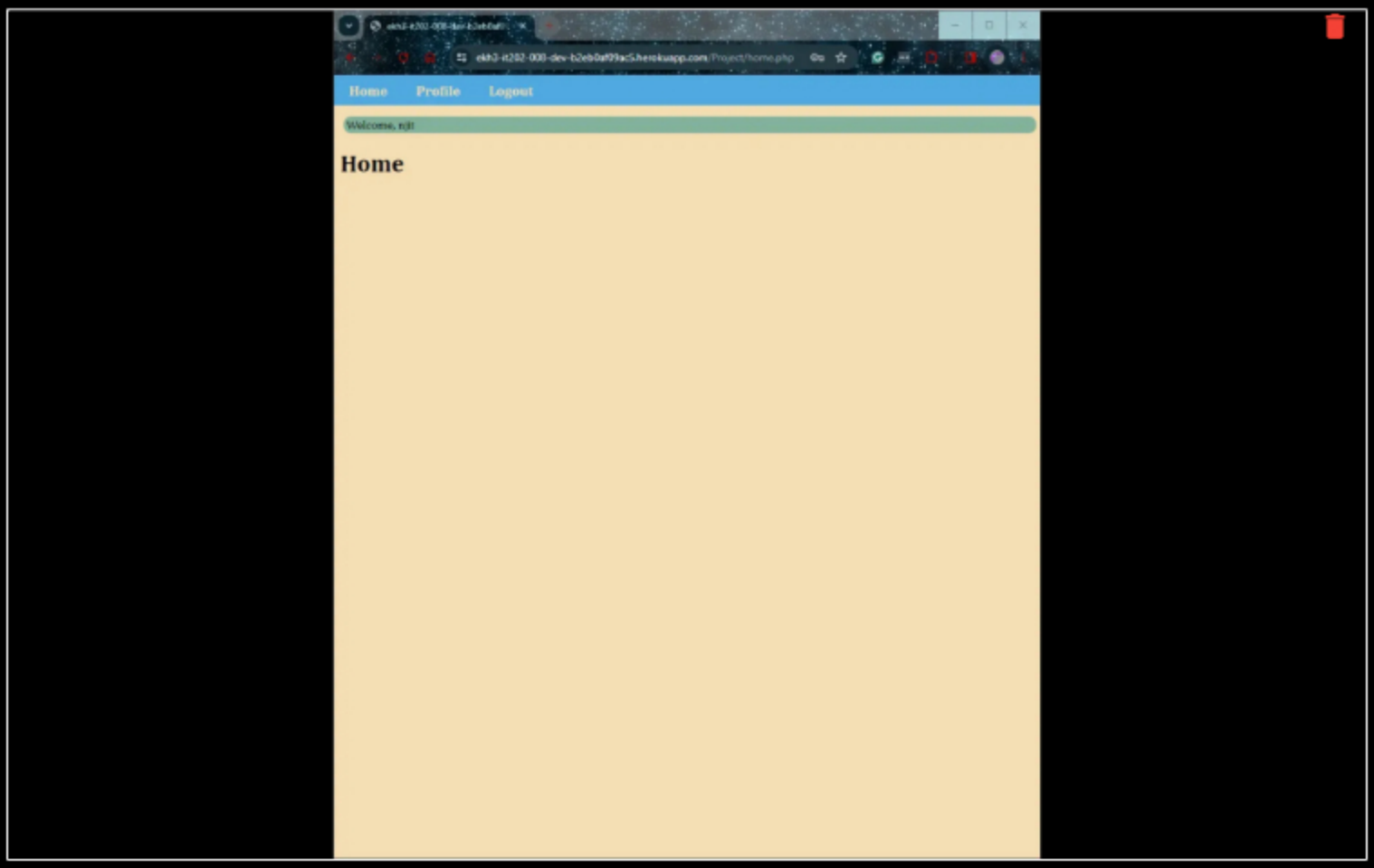
Small Medium Large



Successful log out and redirect

Checklist Items (1)

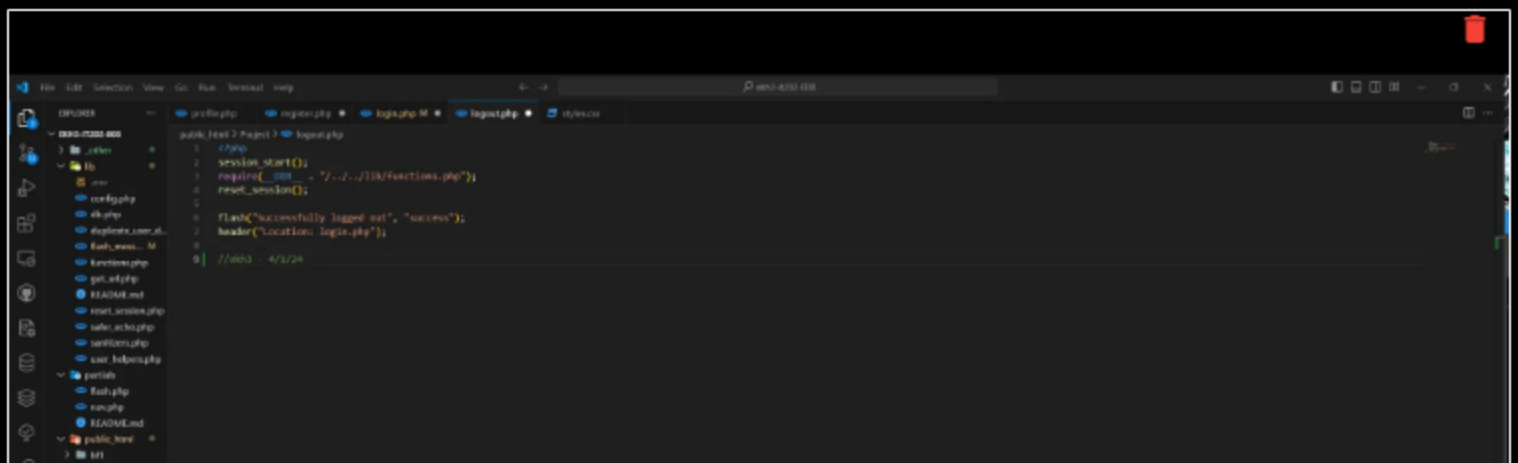
#2 Screenshot of the redirect to login with the user-friendly logged-out message (site)

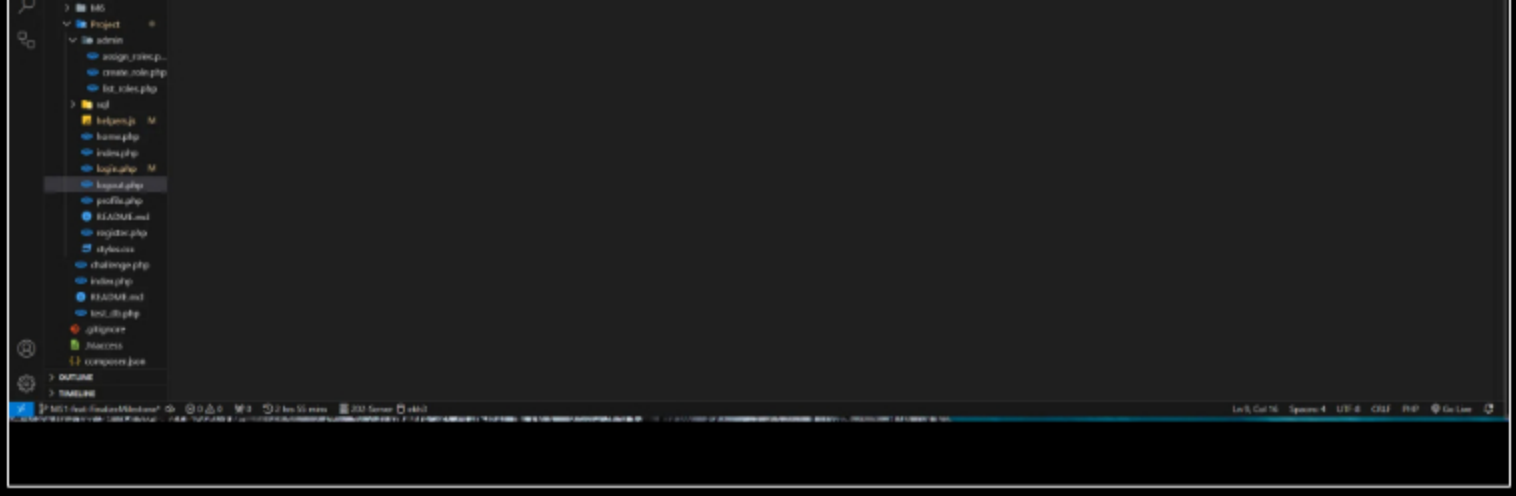


Logged in user nav - default user

Checklist Items (1)

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





log out code

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/WokFriedE/ekh3-it202-008/pull/14

Basic Security Rules and Roles (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Authentication Screenshots

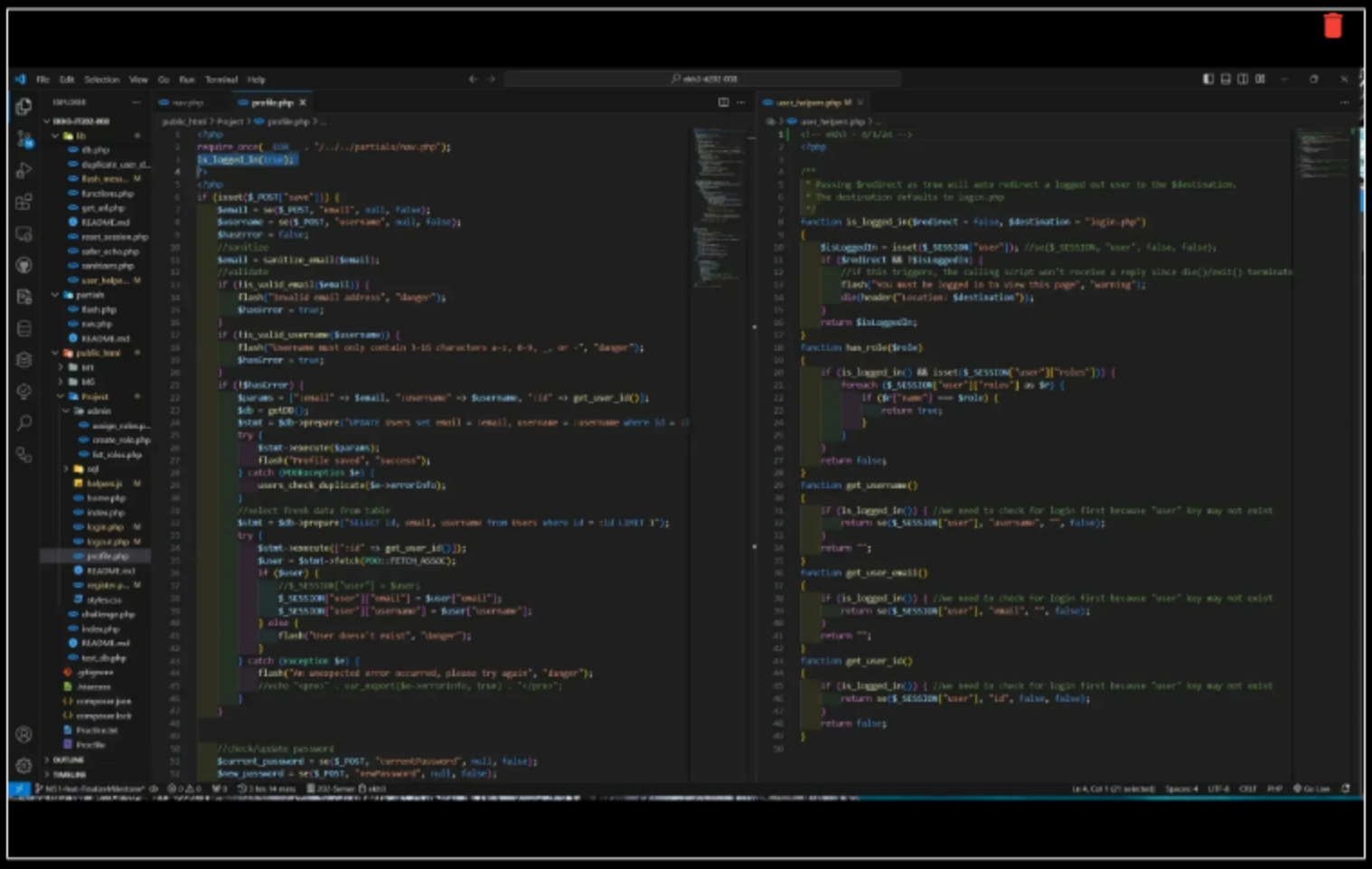
Checklist *The checkboxes are for your own tracking

#	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



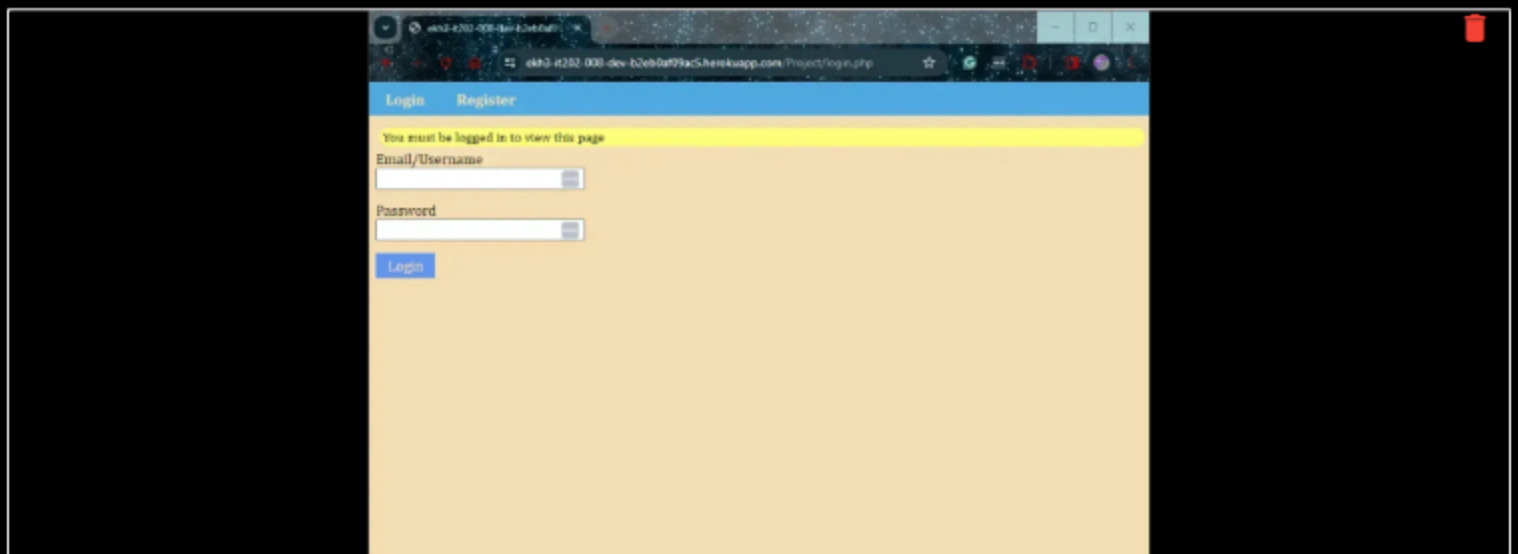
Code for is_logged_in() function (right) & use in profile (left) Used in profile.php and home.php

Checklist Items (3)

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

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

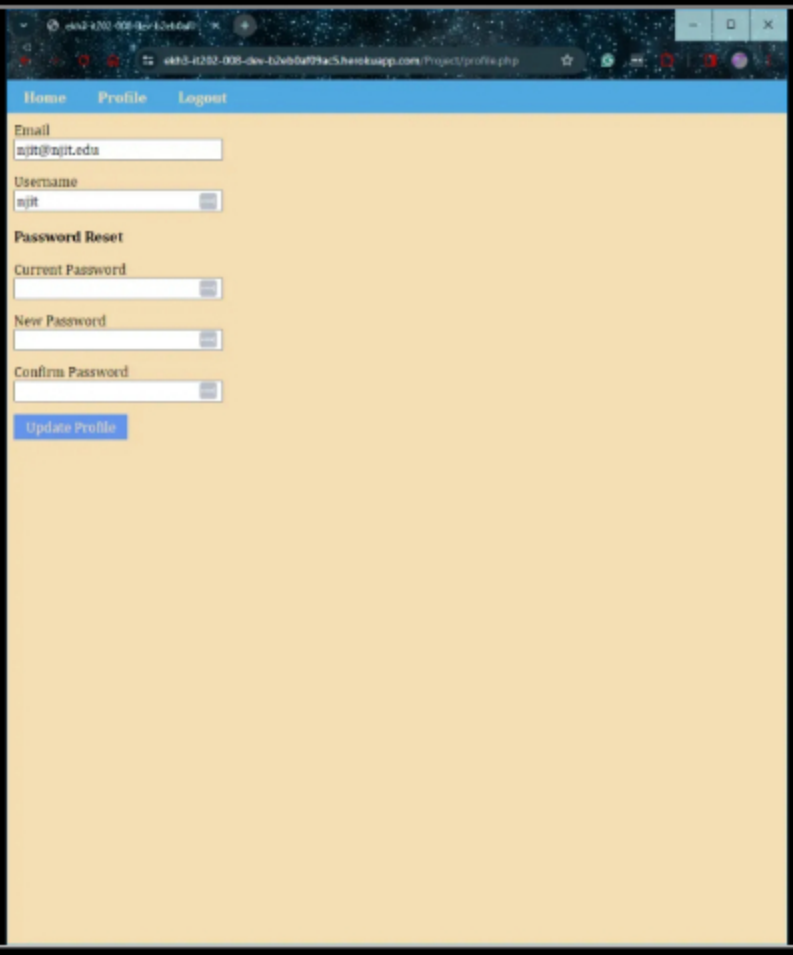
#3 Include ucid/date code comments in all screenshots (one per screenshot is sufficient)



trying ./profile.php without being logged in

Checklist Items (1)

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



Screenshot of profile - Login default user

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Authorization Screenshots

Checklist

*The checkboxes are for your own tracking

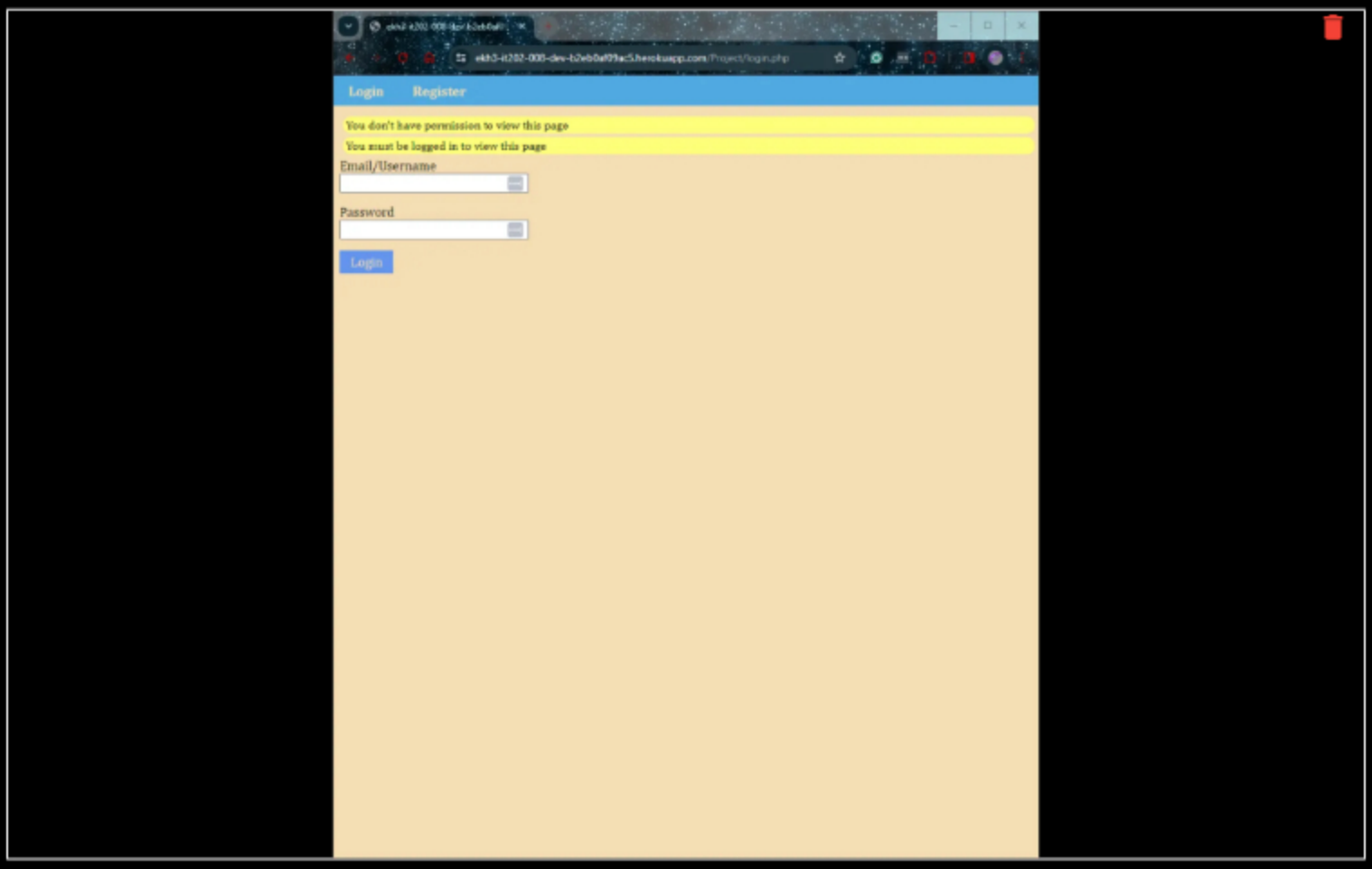
#	Points	Details
#1	1	Screenshot of the function that checks for a specific role

☐ #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

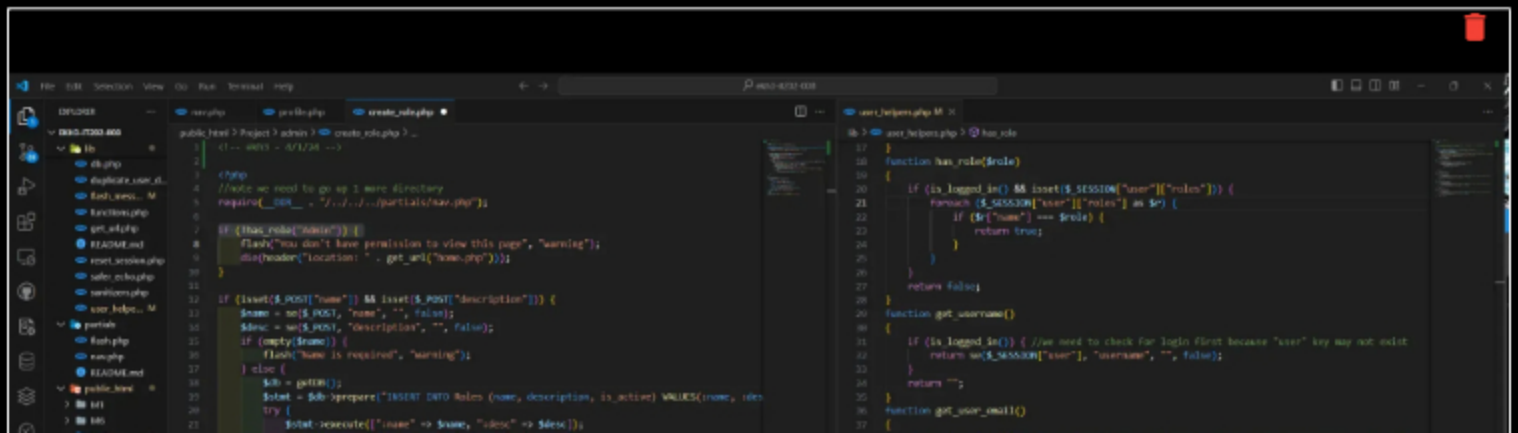
Small Medium Large

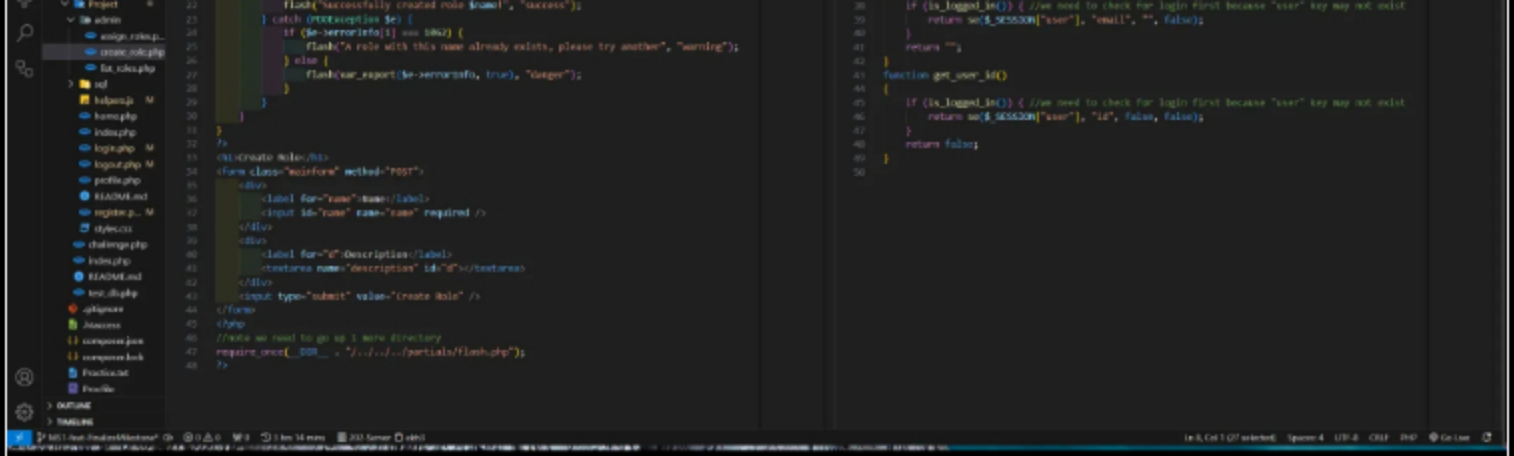


trying to go to ./admin/list_roles.php without logged in

Checklist Items (1)

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





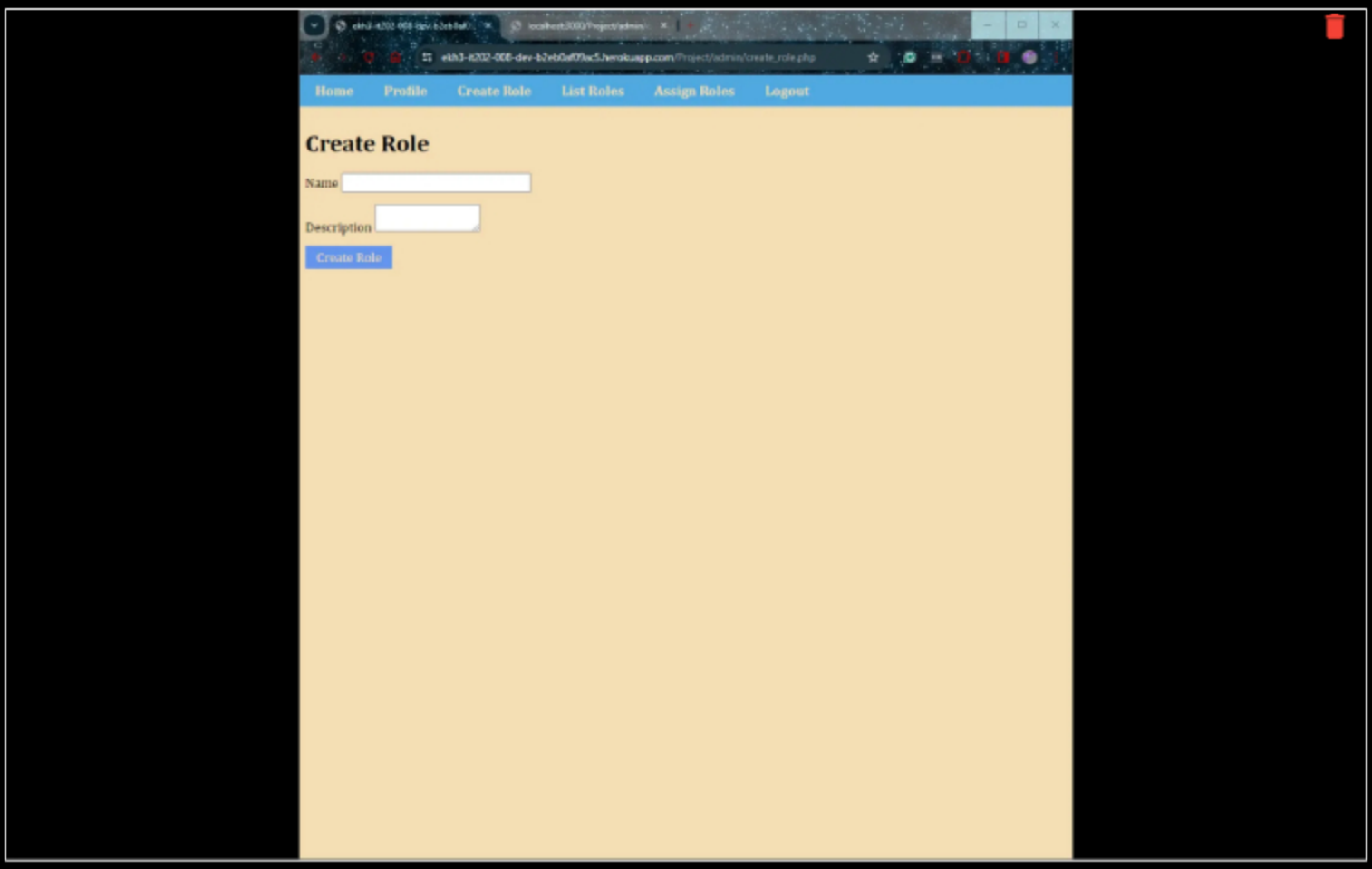
Function for has_role() on right; on left, create_role.php, assign_roles.php, and create_role.php use has_role("Admin")

Checklist Items (3)

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

#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)



Screenshot showing has_role() in action on site

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

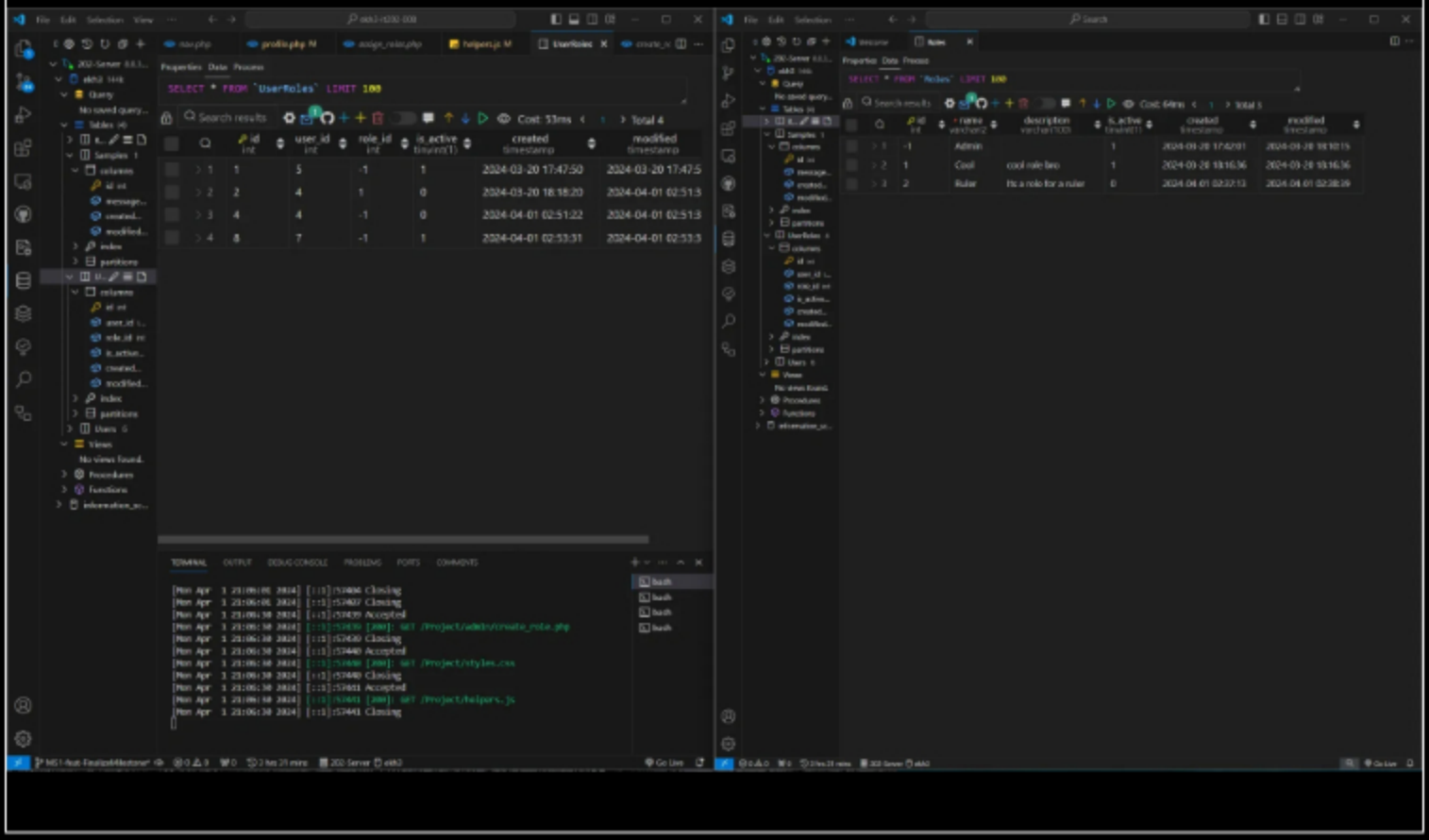
*The checkboxes are for your own tracking

#	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 table (left), Roles table (right) - Screenshot

Checklist Items (5)

#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

#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:

UserRoles table is used to build the relation of the Users from its respective table to the Roles table. As multiple users can have multiple roles that can be toggled, having it as its relational table would be easier to manage where the ID of the user is related to the ID of the role. In addition, using IDs instead of names allows for the roles to be edited without needing to alter the entire system.

As people can toggle their roles, it's easier to soft delete roles by disabling the relation of the user and the role, making it inactive; this would act as a UserRoles.is_active. In contrast the Roles.is_active would disable the role entirely, so regardless of the active status for a user, the role cannot be used; thus Roles.is_active is a macro switch for a role while UserRoles.is_active is per user and 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/WokFriedE/ekh3-it202-008/pull/24

COLLAPSE

User Profile (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: View Profile Website Page

Checklist

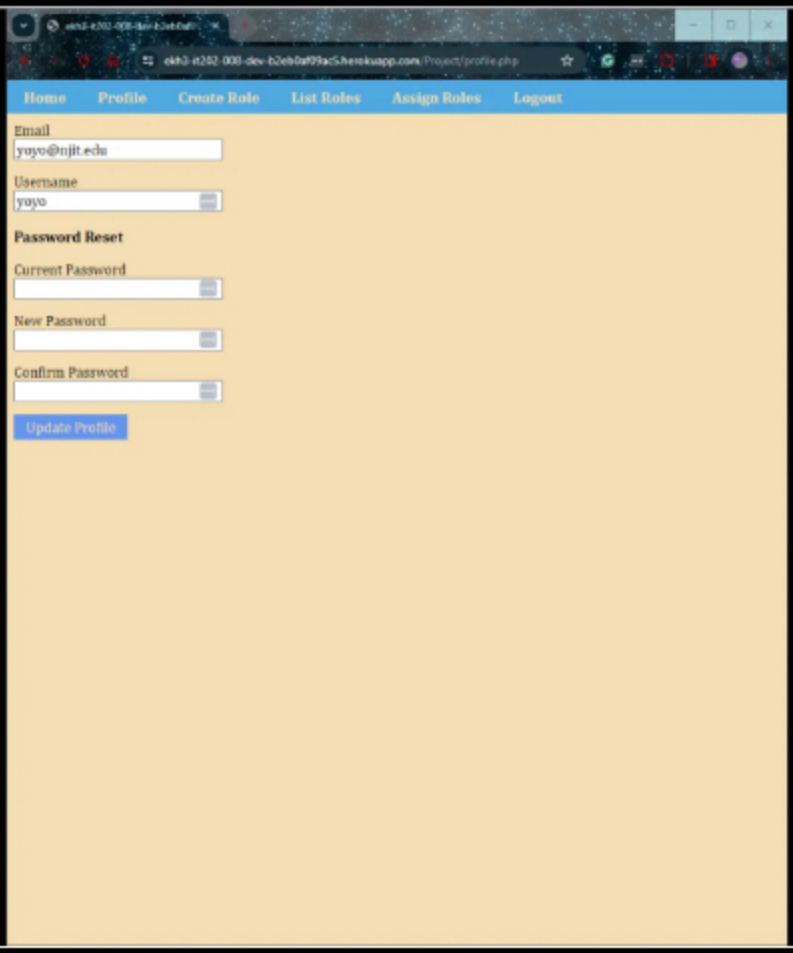
*The checkboxes are for your own tracking

#	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)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Profile screenshot with fields and auto populated items

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 user clicks on the profile page, it would check if the user is logged in using the is_logged_in() function from a PHP helper function. It would check if "user" is set to see if someone is logged in. On load, the form would generate with the values from $email and $username in their respective fields. The result is that they stay "sticky."

COLLAPSE

Task #3 - Points: 1

Text: Edit Profile Website Page

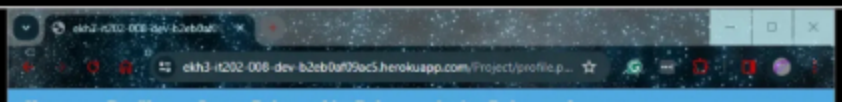
Checklist *The checkboxes are for your own tracking

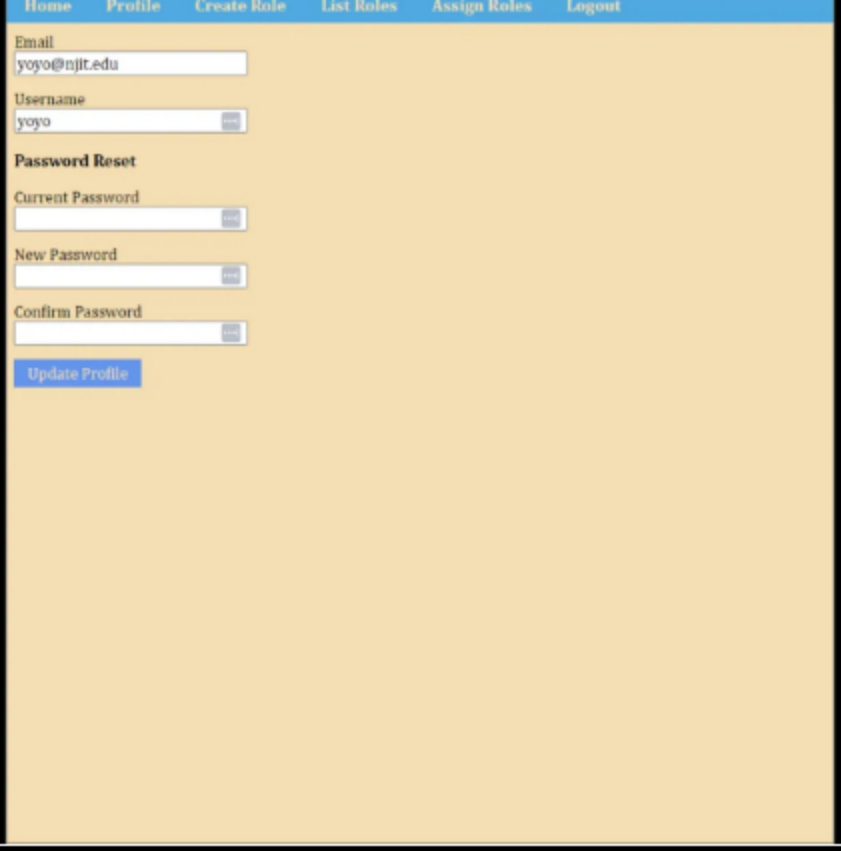
#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Demonstrate with before and after of a username change (including success message)
#4	1	Demonstrate with a before and after of an email change (including success message)
#5	1	Demonstrate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



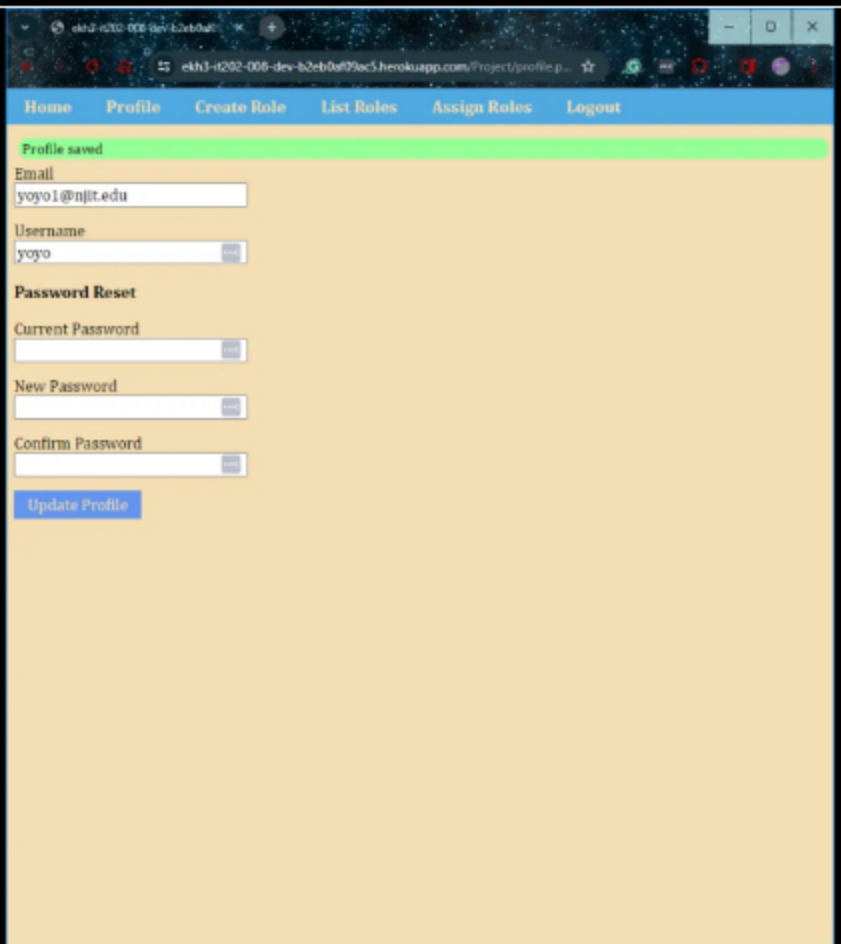


Profile before updates (both were reset to this before further updates made)

Checklist Items (2)

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

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



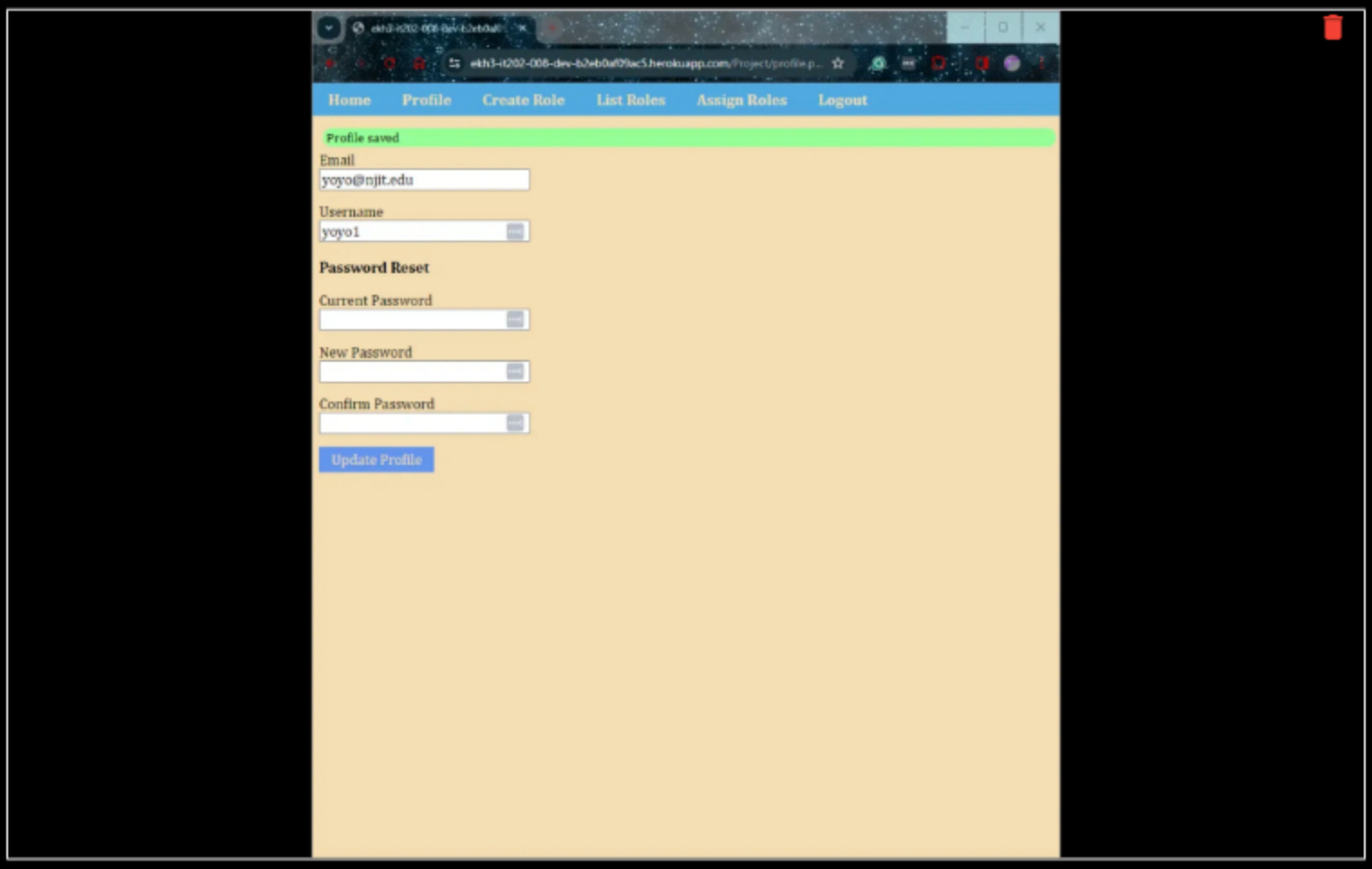
Profile after email change (yoyo@njit.edu -> yoyo1@njit.edu)

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

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



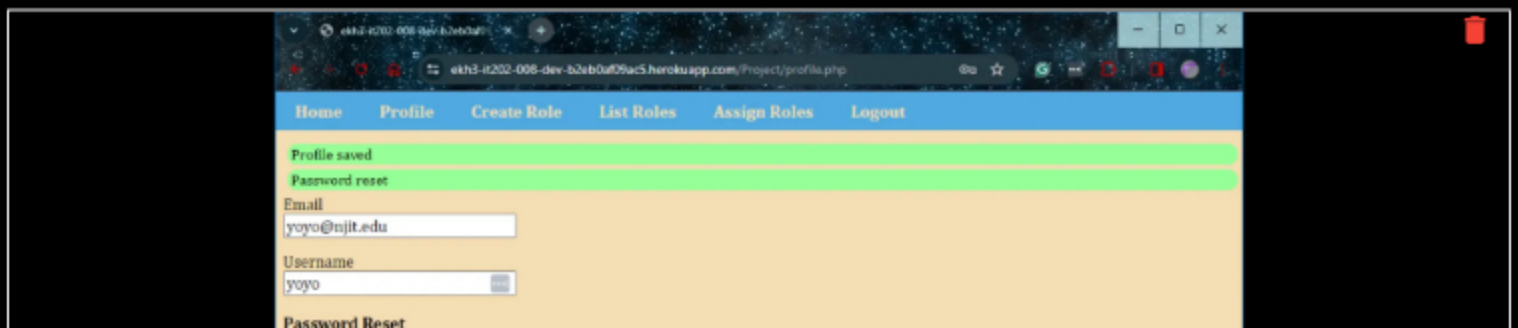
Profile after username change (yoyo –> yoyo1)

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)



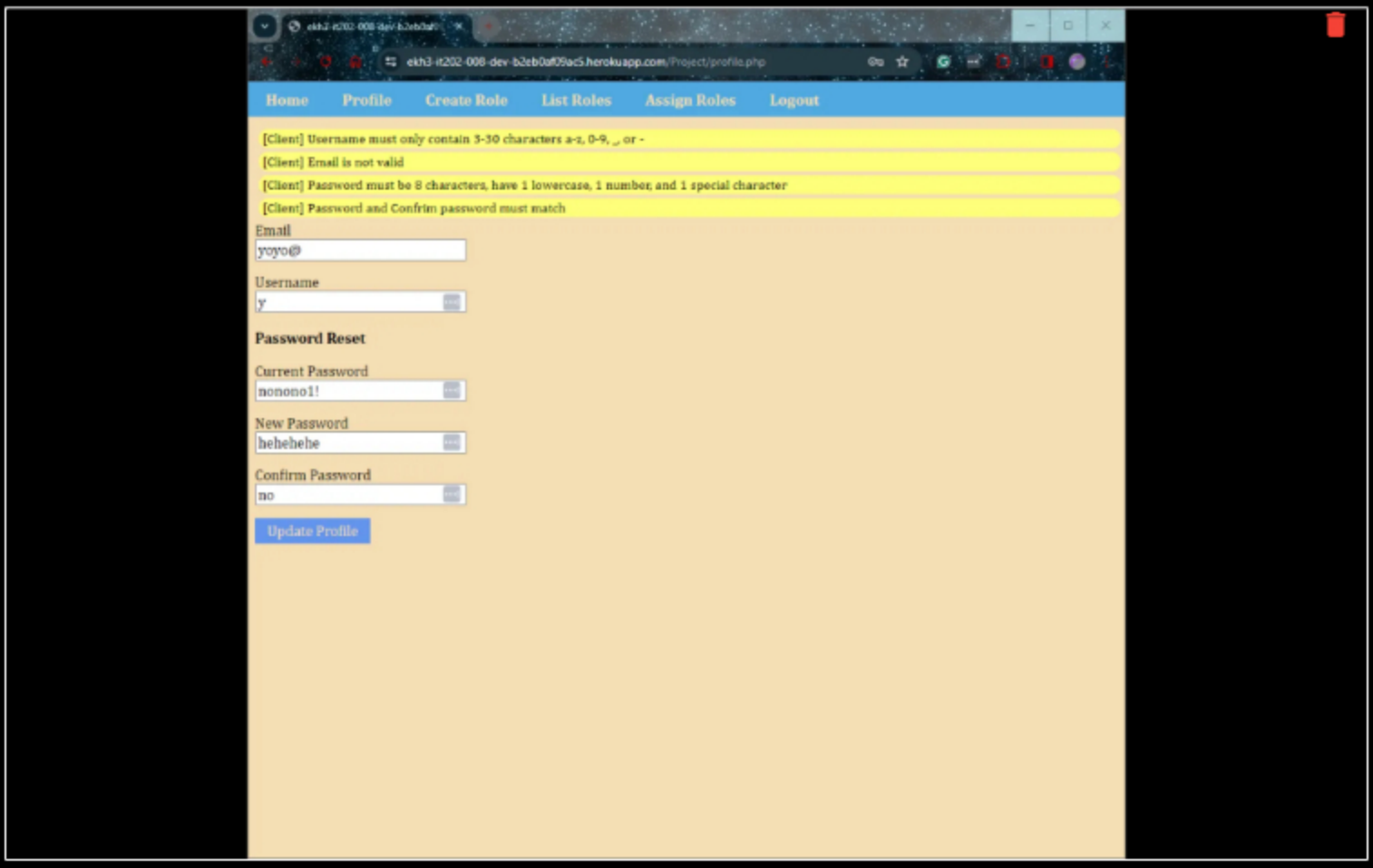
Profile after password change

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

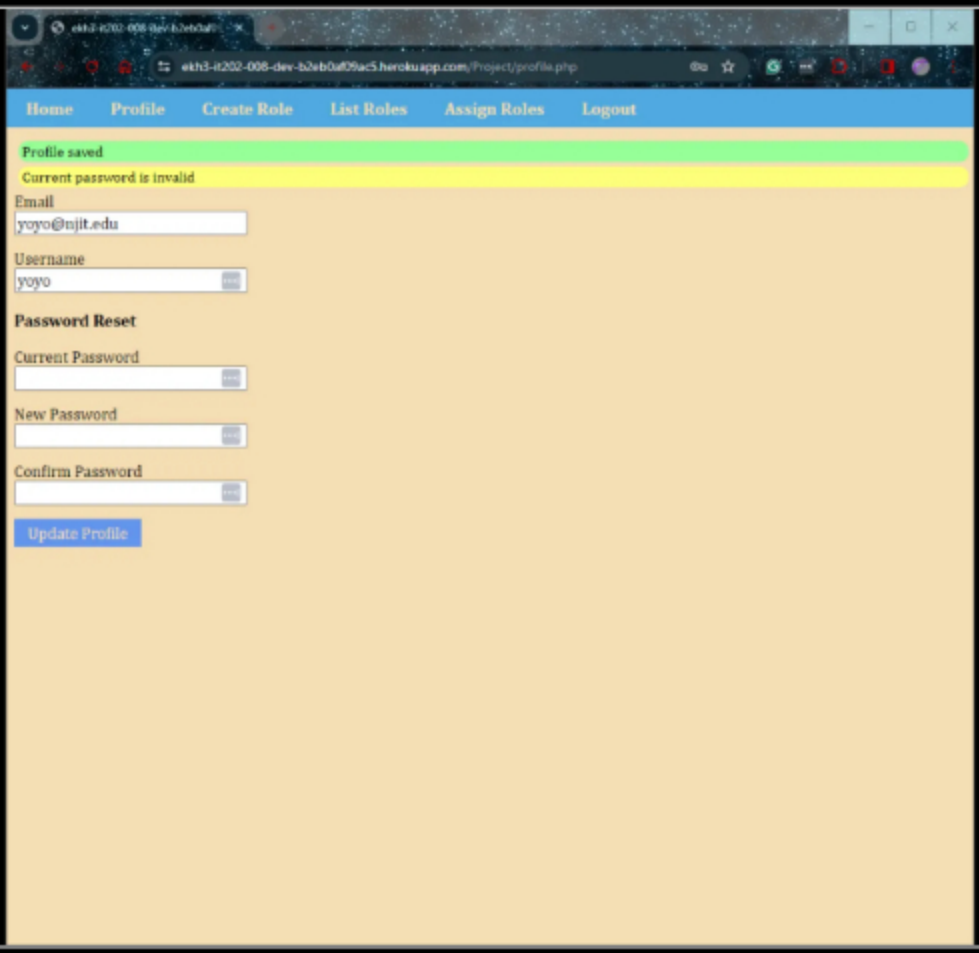
#5 Demonstrate the success message of updating password



Profile JS validation

Checklist Items (1)

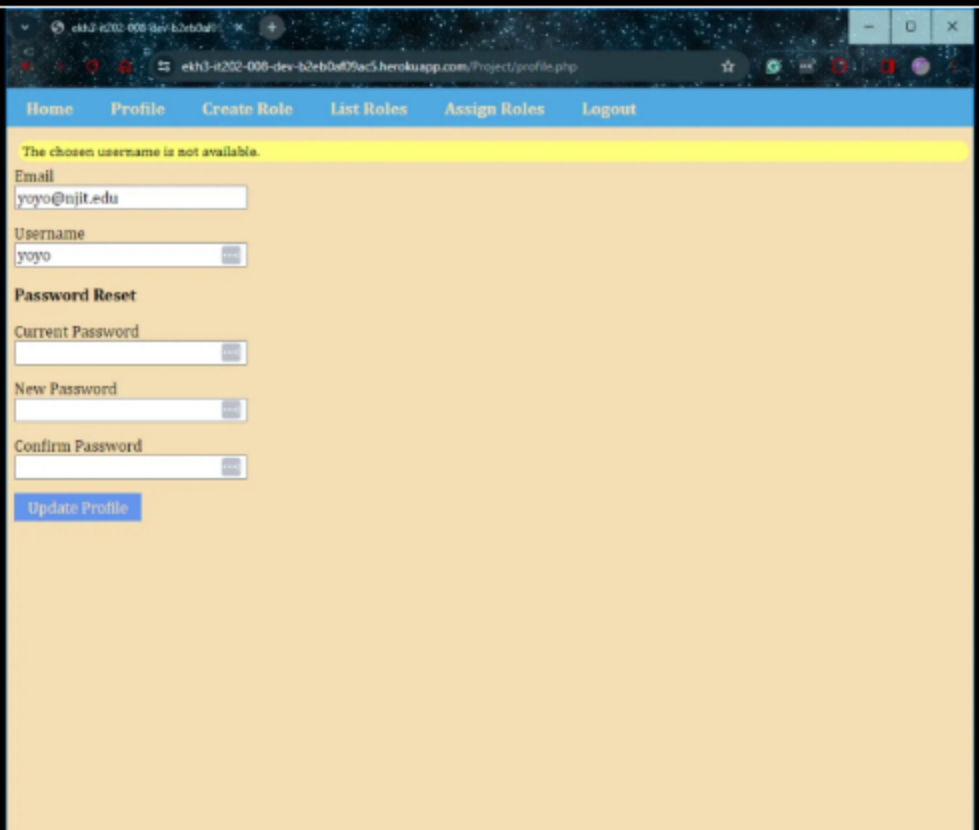
#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)



PHP current password invalid

Checklist Items (1)

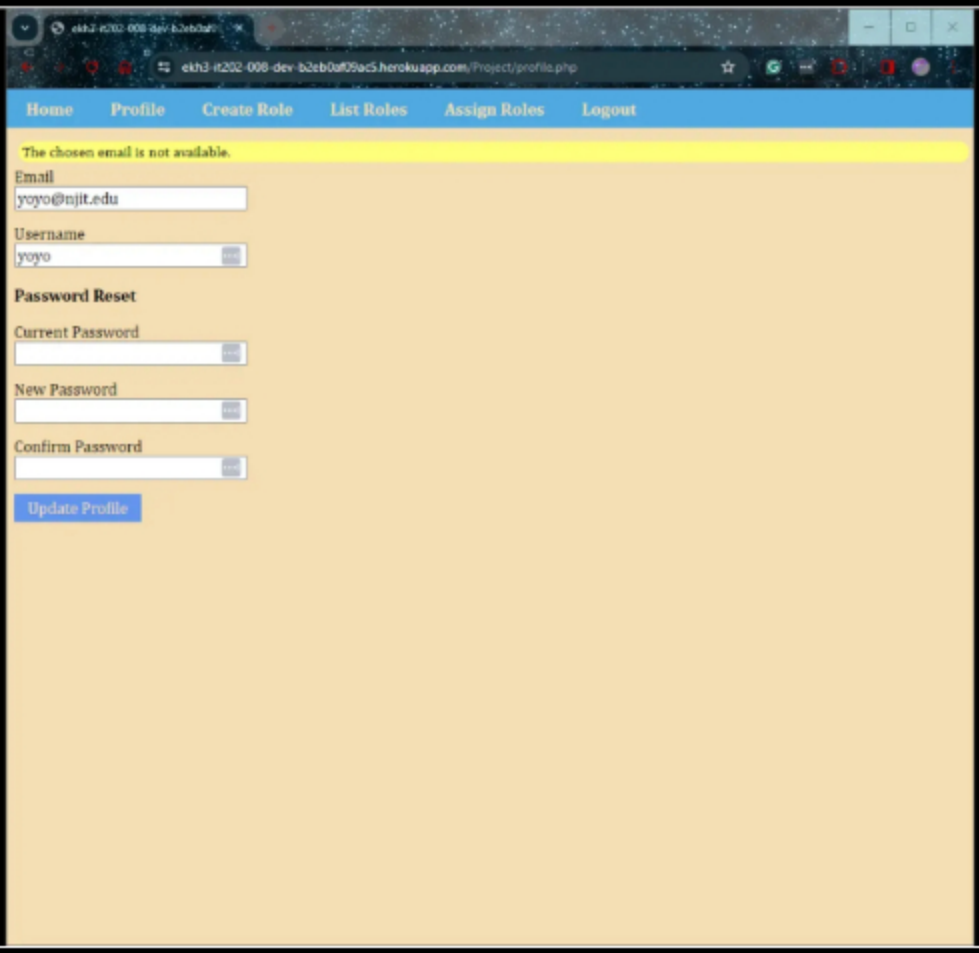
#7 Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)



Profile - PHP invalid username (in use)

Checklist Items (1)

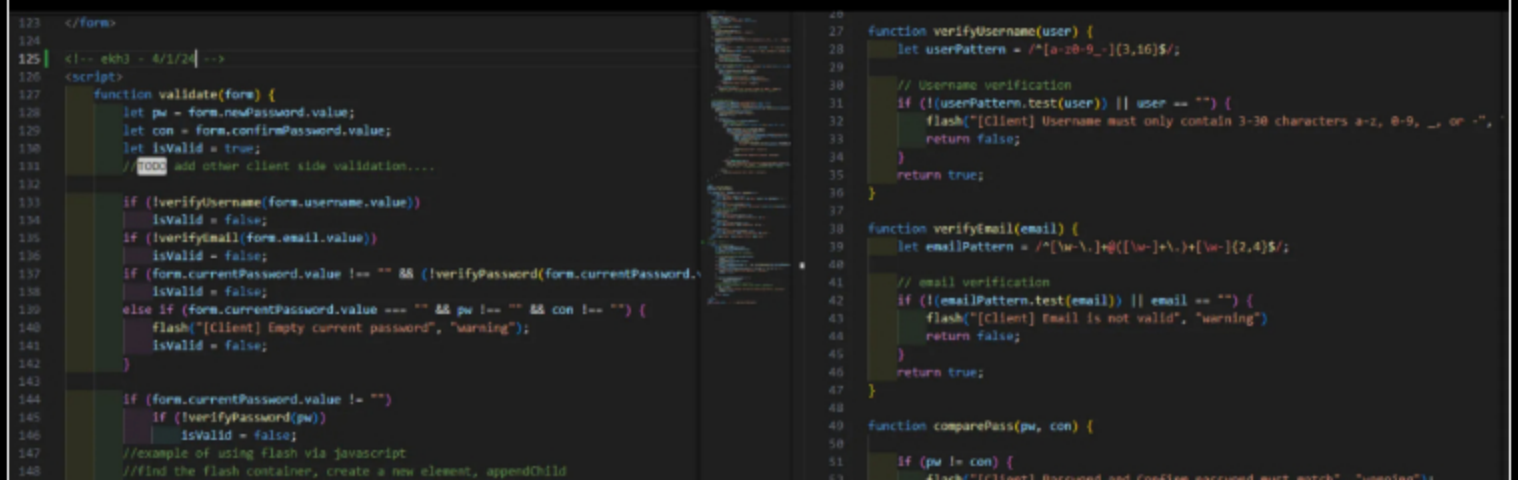
#7 Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)

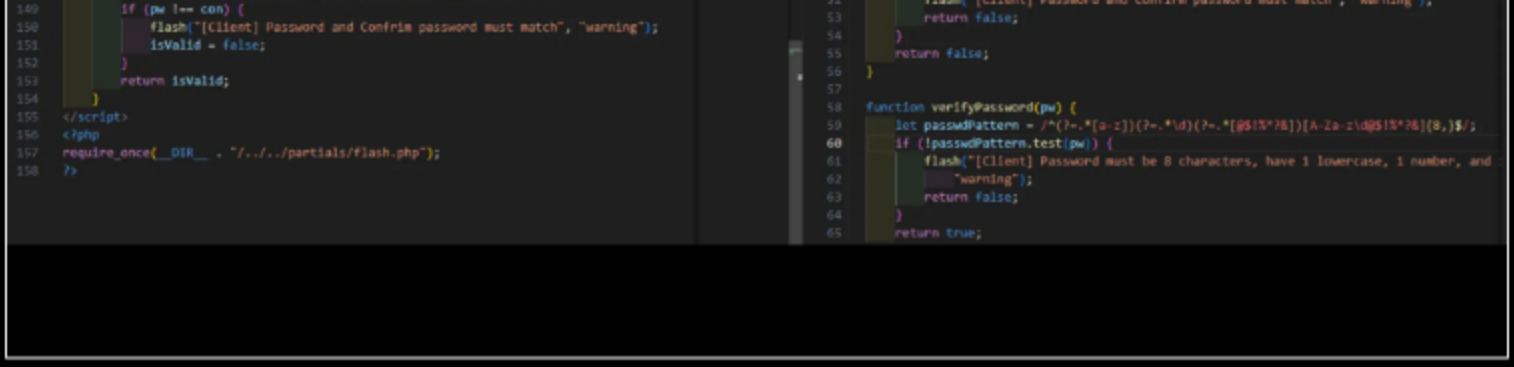


Profile - invalid email (in use)

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)





Profile validation code for JS (if needed)

Checklist Items (0)

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:

For updating the email and username, the process is roughly the same. When the user enters the profile page the current values are presented. If edited, and there is nothing in the password fields, there would be a JS validation for the username and/or email using JS regex.test(). If everything seems in check it would submit to the PHP side, otherwise it would send flashes and prevent submission. The PHP would handle its validations with a similar regex and then pass it to an update query. If there is a duplicate error for either, there would be a caught exception that would be sent to the users_check_duplicate() exception statement and provide the user with a flash that it cannot be used. It would then update the saved session info so it persists.

For the password, it would check if the current password field is filled, otherwise, it would ignore the rest or prompt the user to fill in the current password if the rest of the password fields are filled. If all three fields are filled, then the confirmation will be checked with the new password to make sure they match. If they are not, then prompt the user using a flash, otherwise send it to PHP to validate. The PHP would then check if the current password matches the hash it has stored, if it does it makes the change. If it does not match the stored password it would alert the 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/WokFriedE/ekh3-it202-008/pull/20

Misc (1 pt.)

COLLAPSE

COLLAPSE

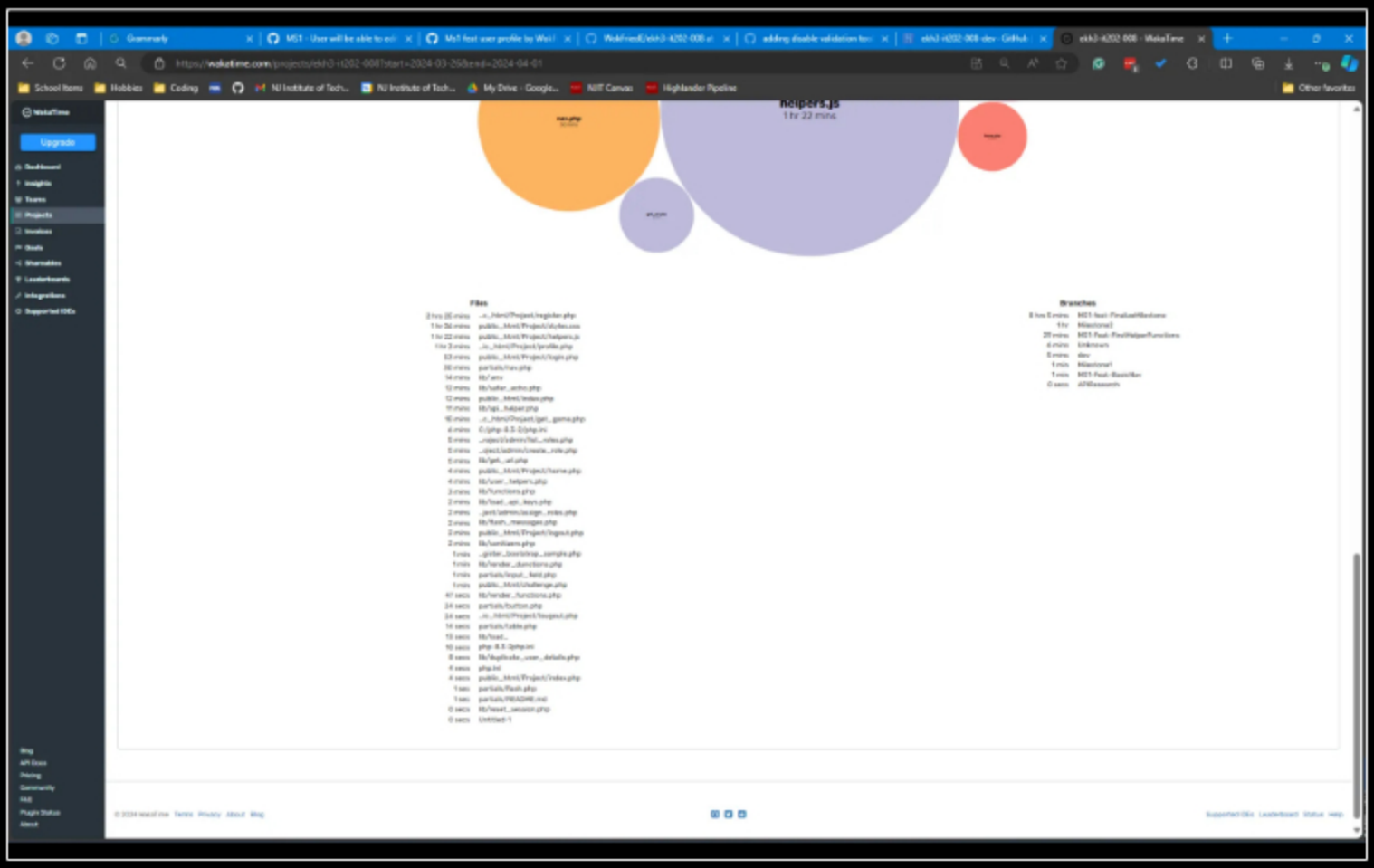
Task #1 - Points: 1

Text: Screenshot of wakatime

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Wakatime (pardon the time, left my computer on)

COLLAPSE

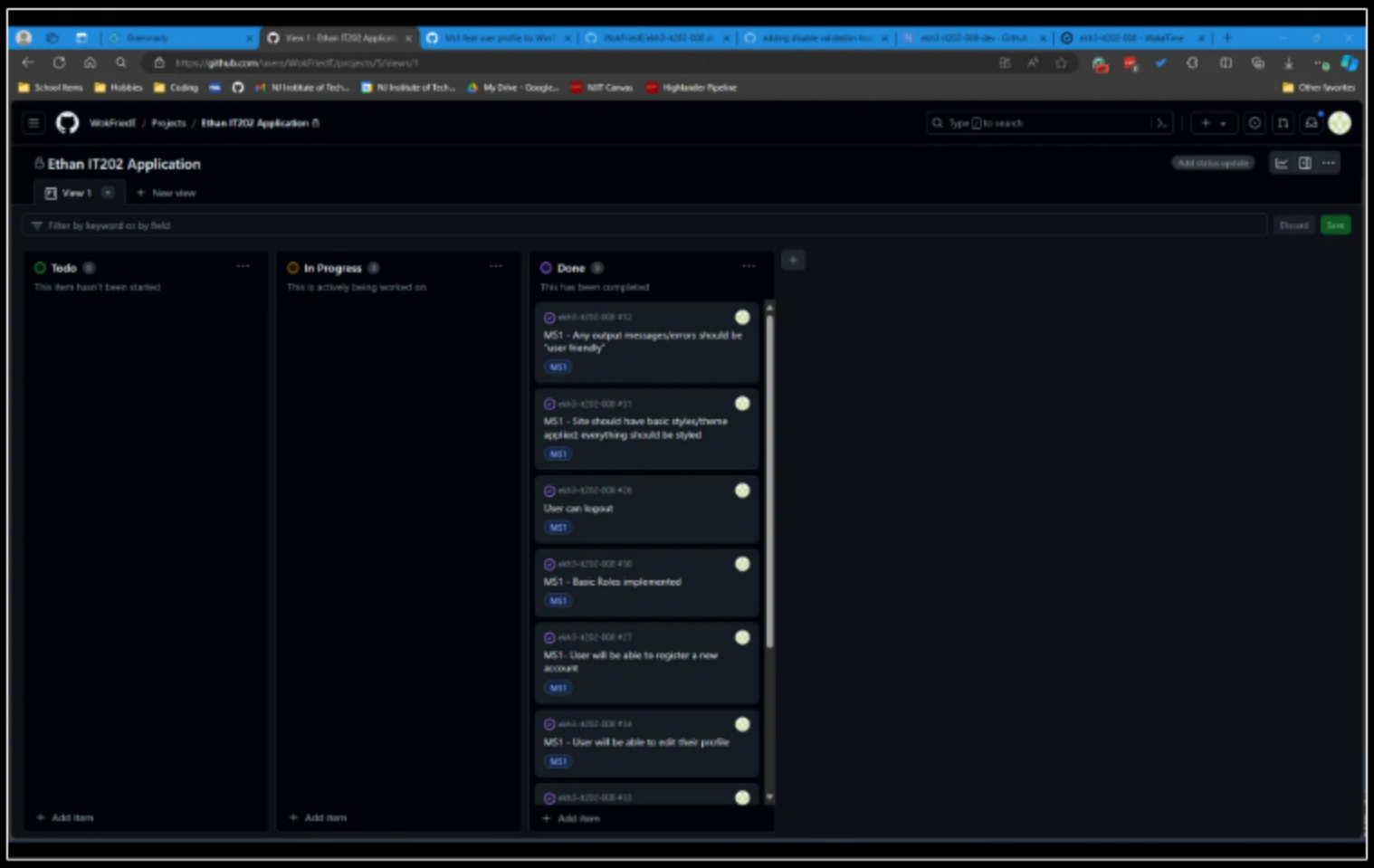
Task #2 - Points: 1

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Github project board

COLLAPSE

Task #3 - Points: 1

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

URL #1

https://github.com/users/WokFriedE/projects/5/views/1

COLLAPSE

Task #4 - Points: 1

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

URL #1

https://ekh3-it202-008-prod-e9bbcd10cf36.herokuapp.com/Project/login.php

End of Assignment