Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/jn426

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

Submissions:

Submission Selection

1 Submission [active] 4/1/2024 10:34:35 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

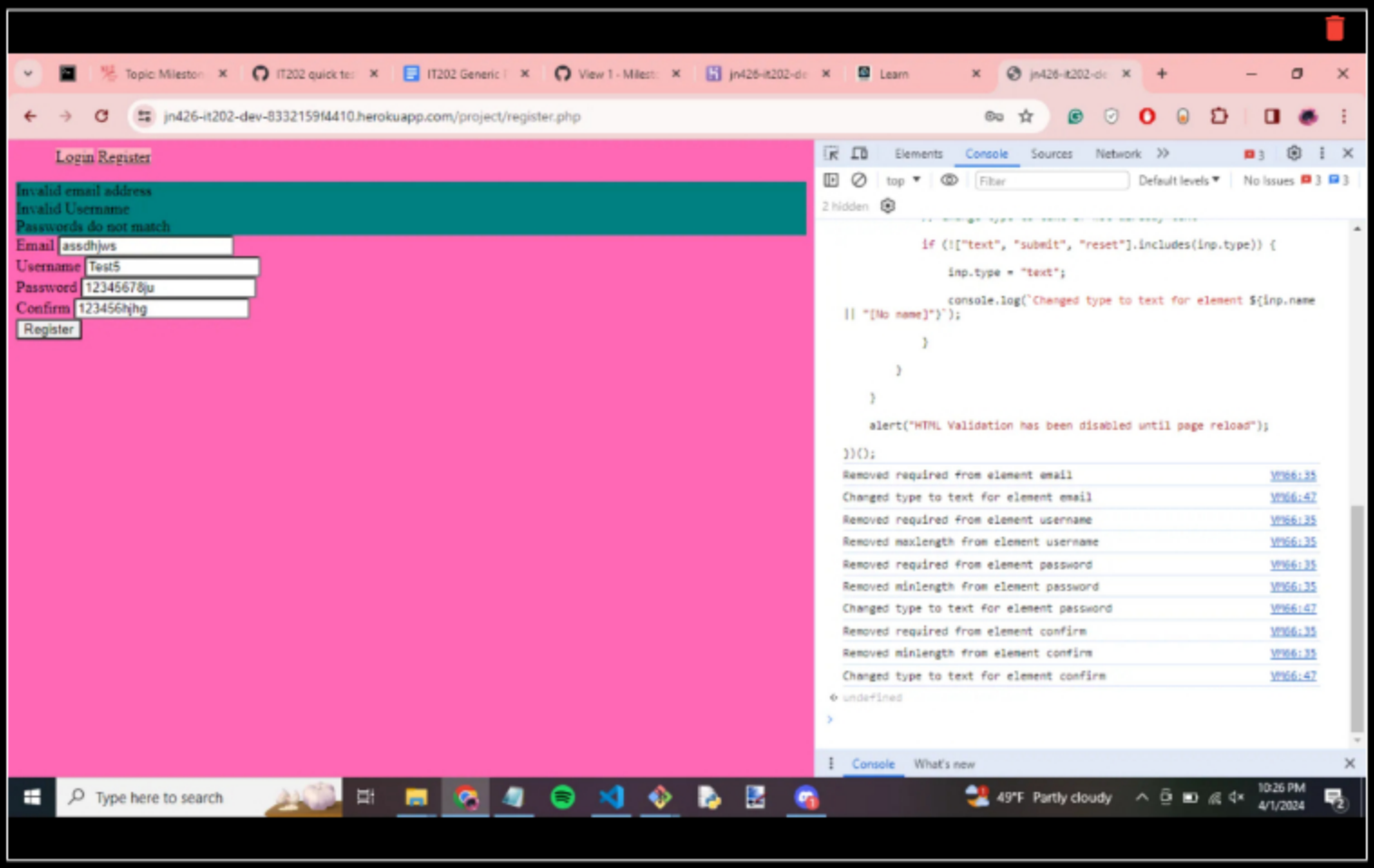
*The checkboxes are for your own tracking

#	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



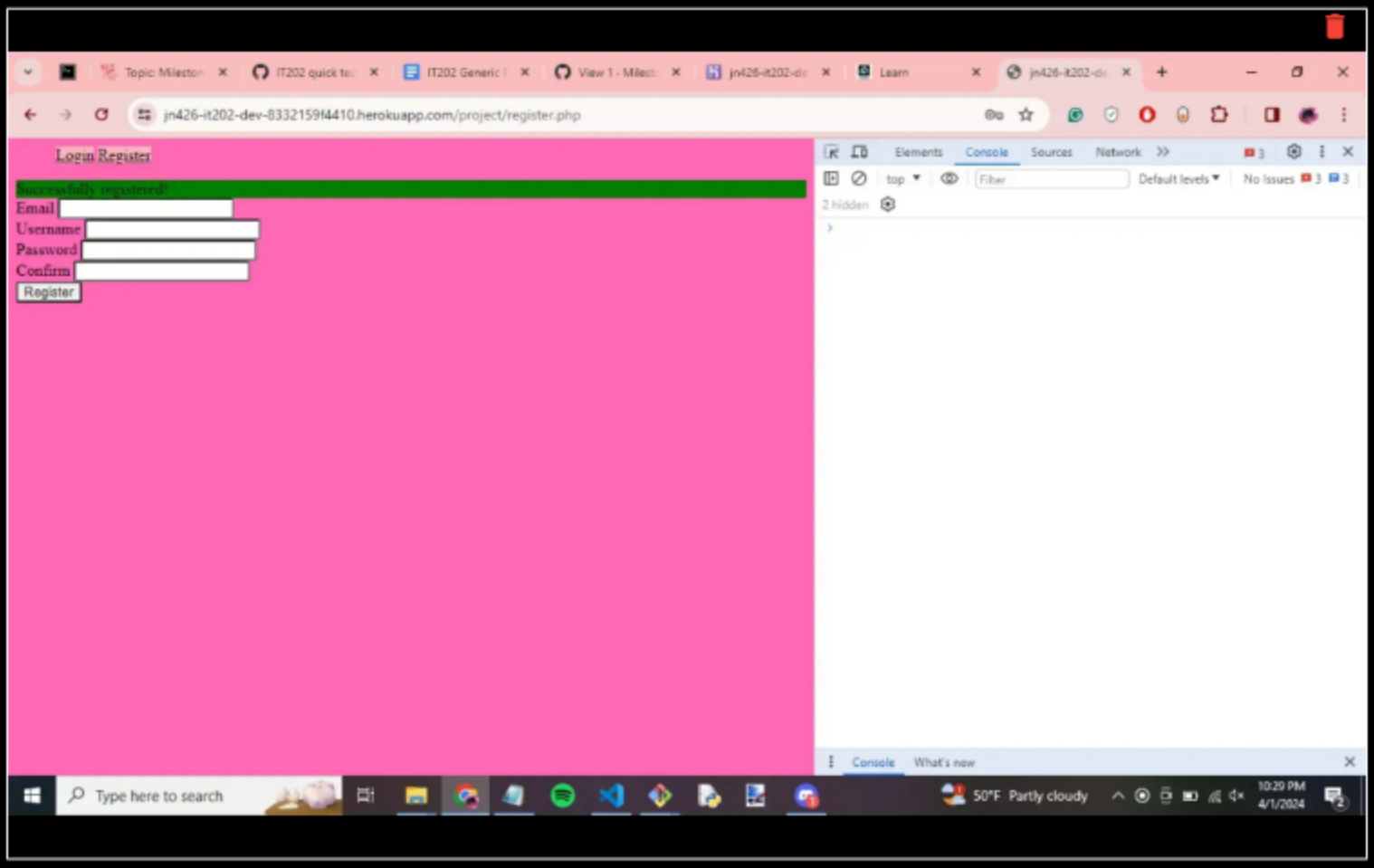
JS Validation for #3

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

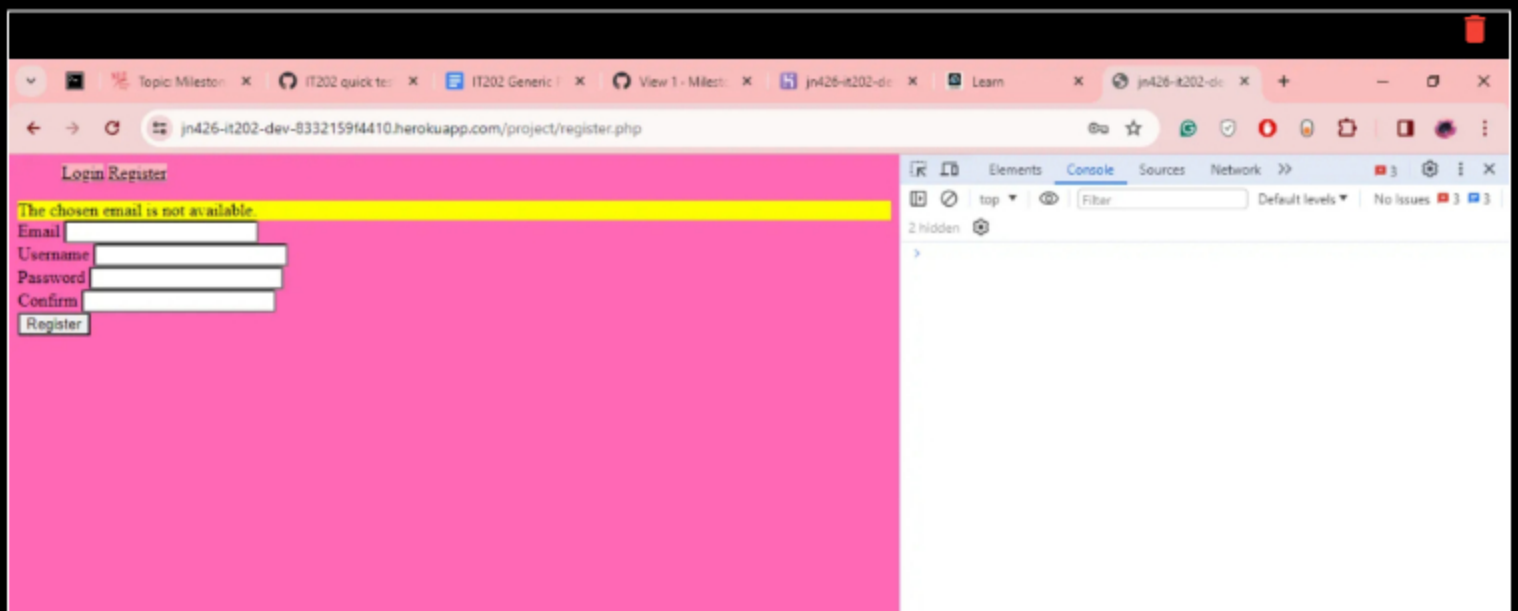
#3 Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required



user friendly msg of acc being made

Checklist Items (1)

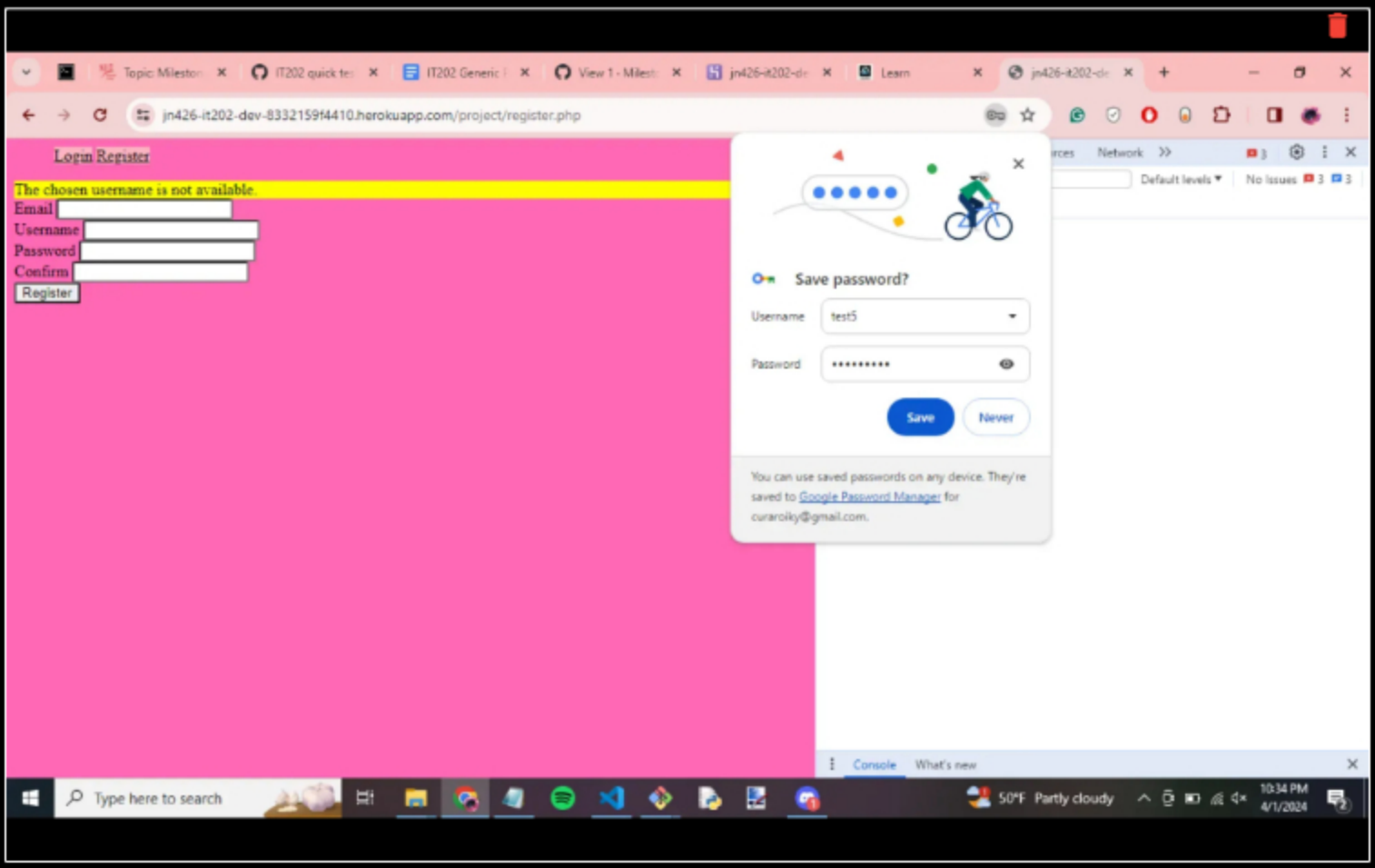
#6 Demonstrate user-friendly message of new account being created



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)

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the form code

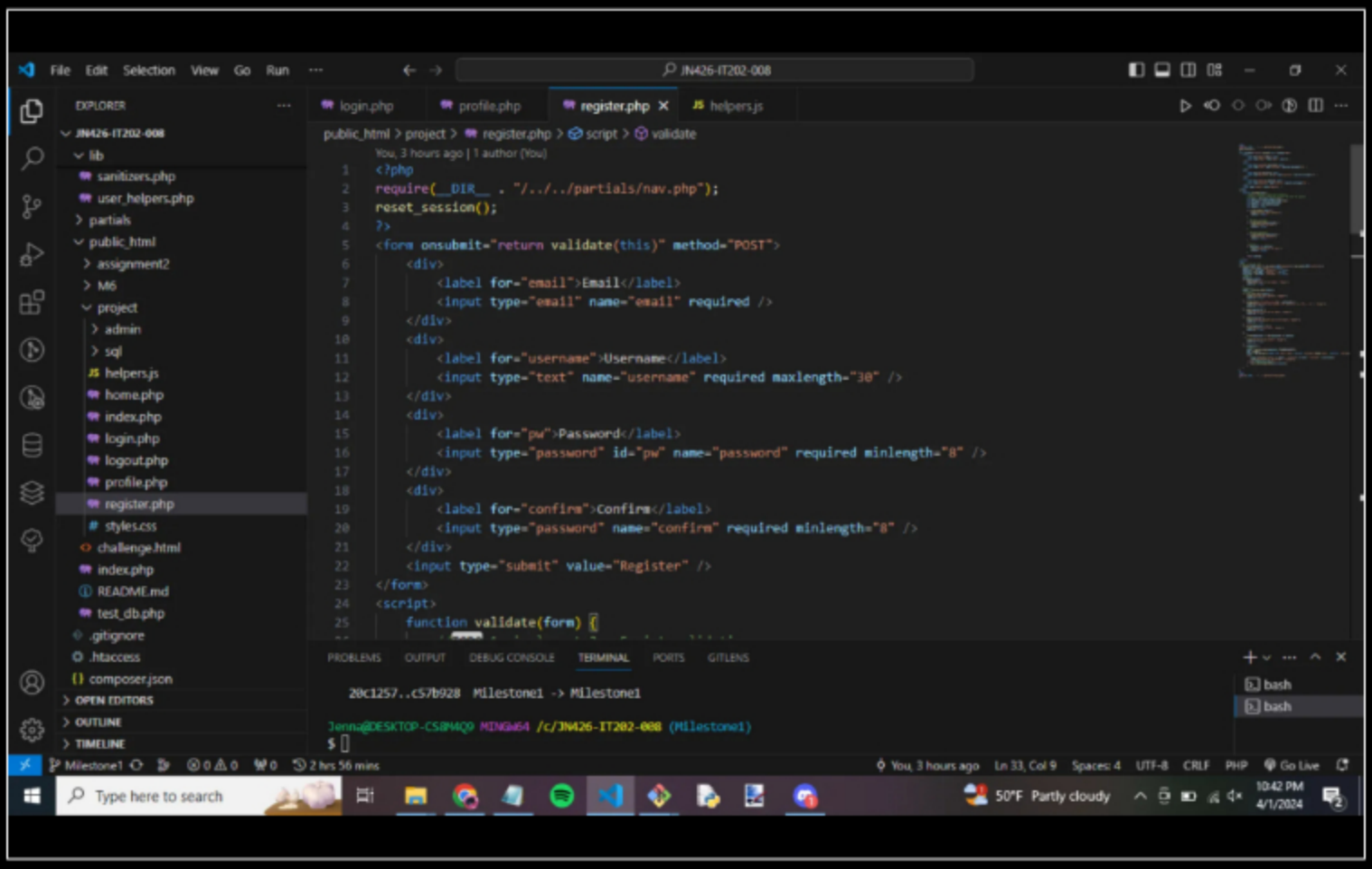
Details:

Should have appropriate input types for the fiel

Task Screenshots:

Gallery Style: Large View

Small Medium Large



register form

COLLAPSE

Task #3 - Points: 1

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

Checklist

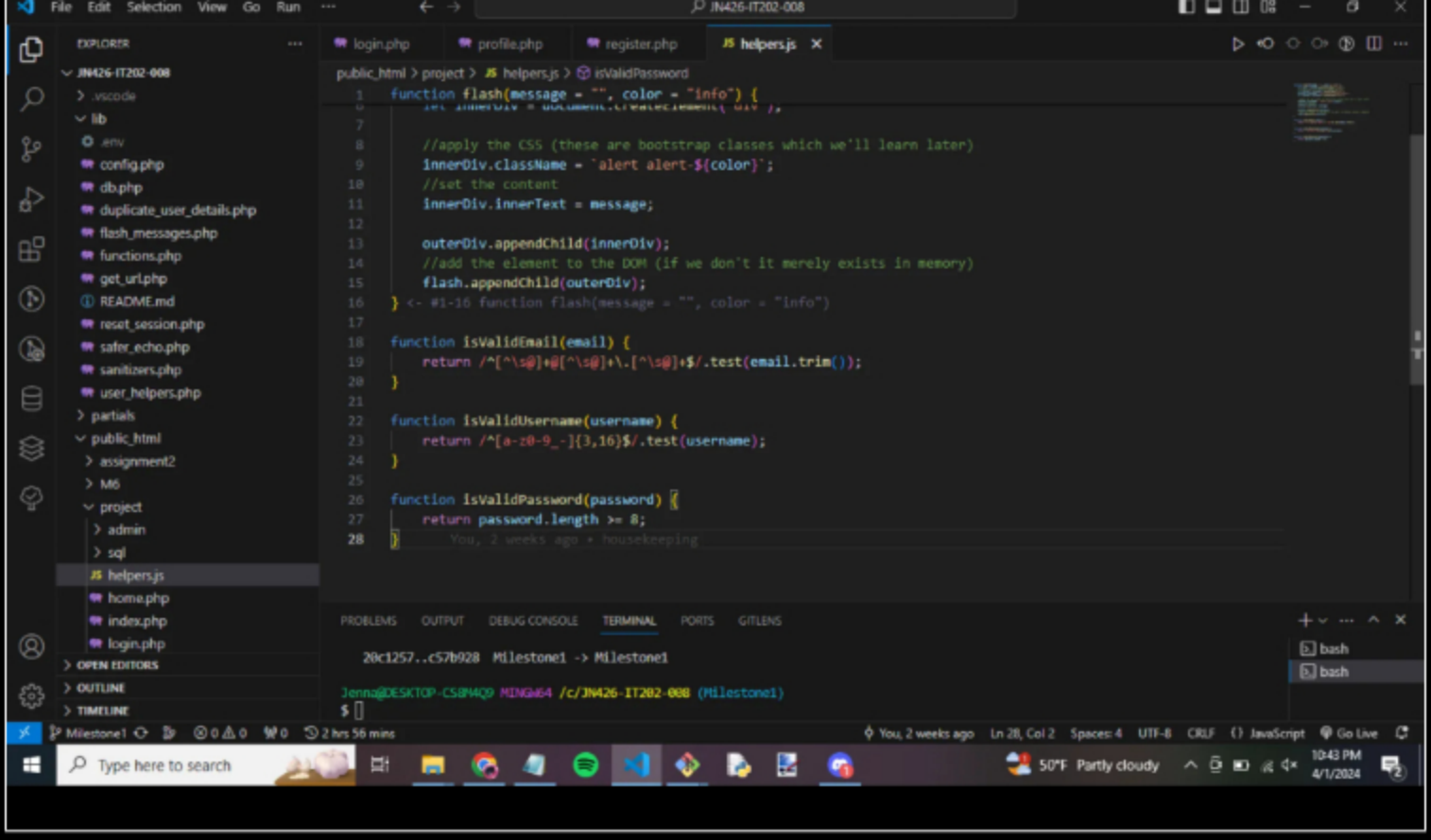
*The checkboxes are for your own tracking

#	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

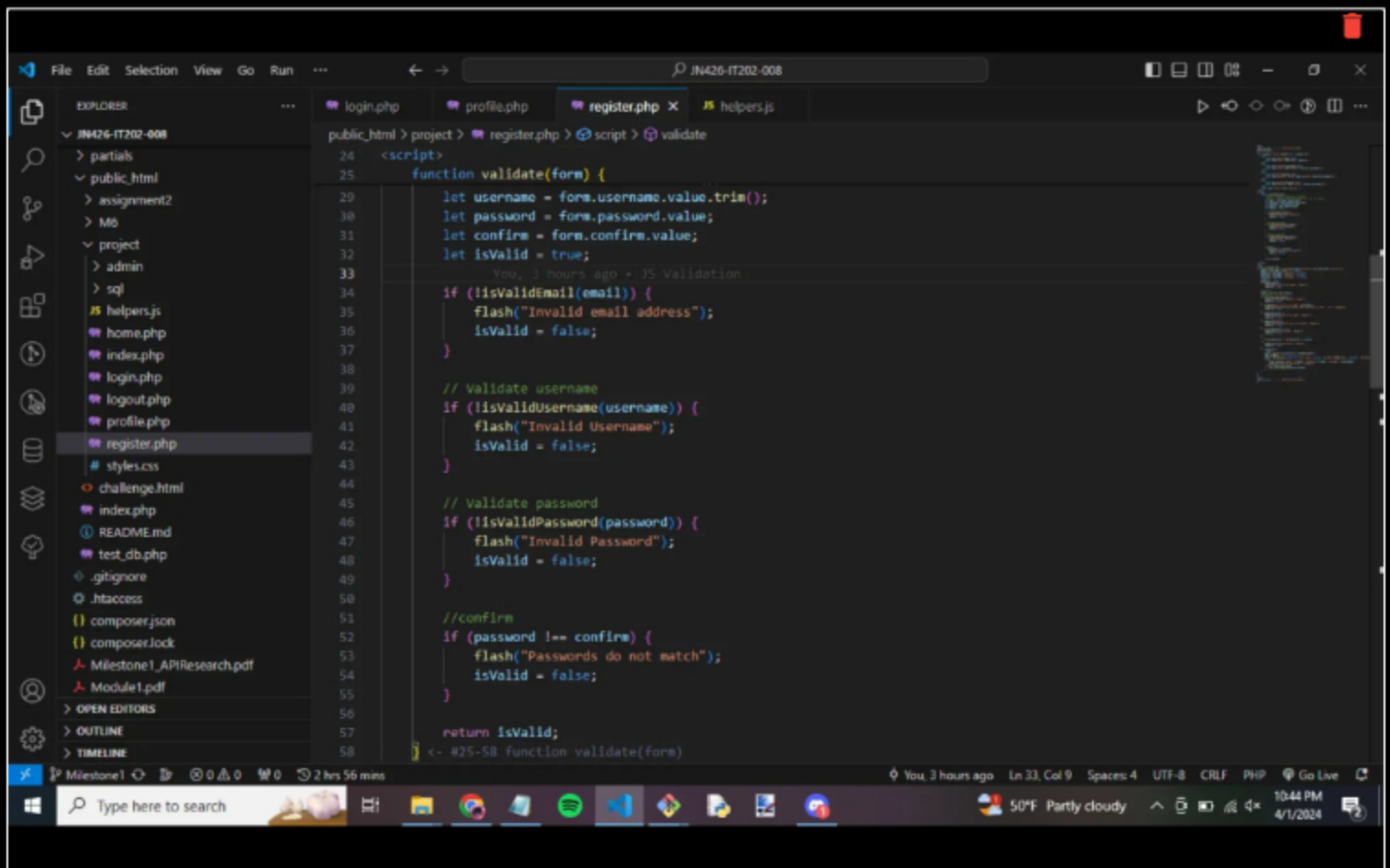
Small Medium Large



Extra file for JS validation

Checklist Items (1)

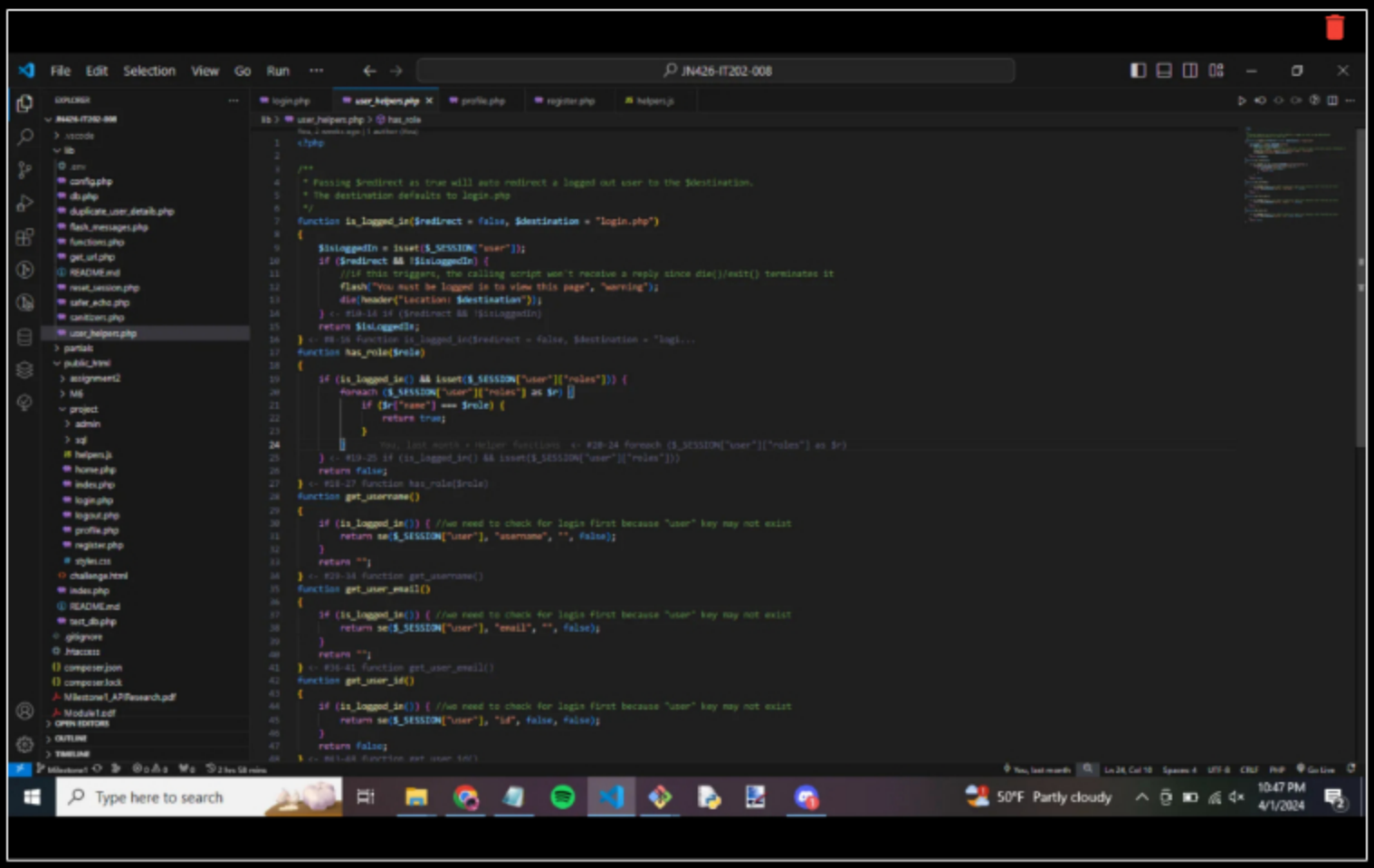
#1 Show the JavaScript validations (include any extra files related)



register JS validation

Checklist Items (1)

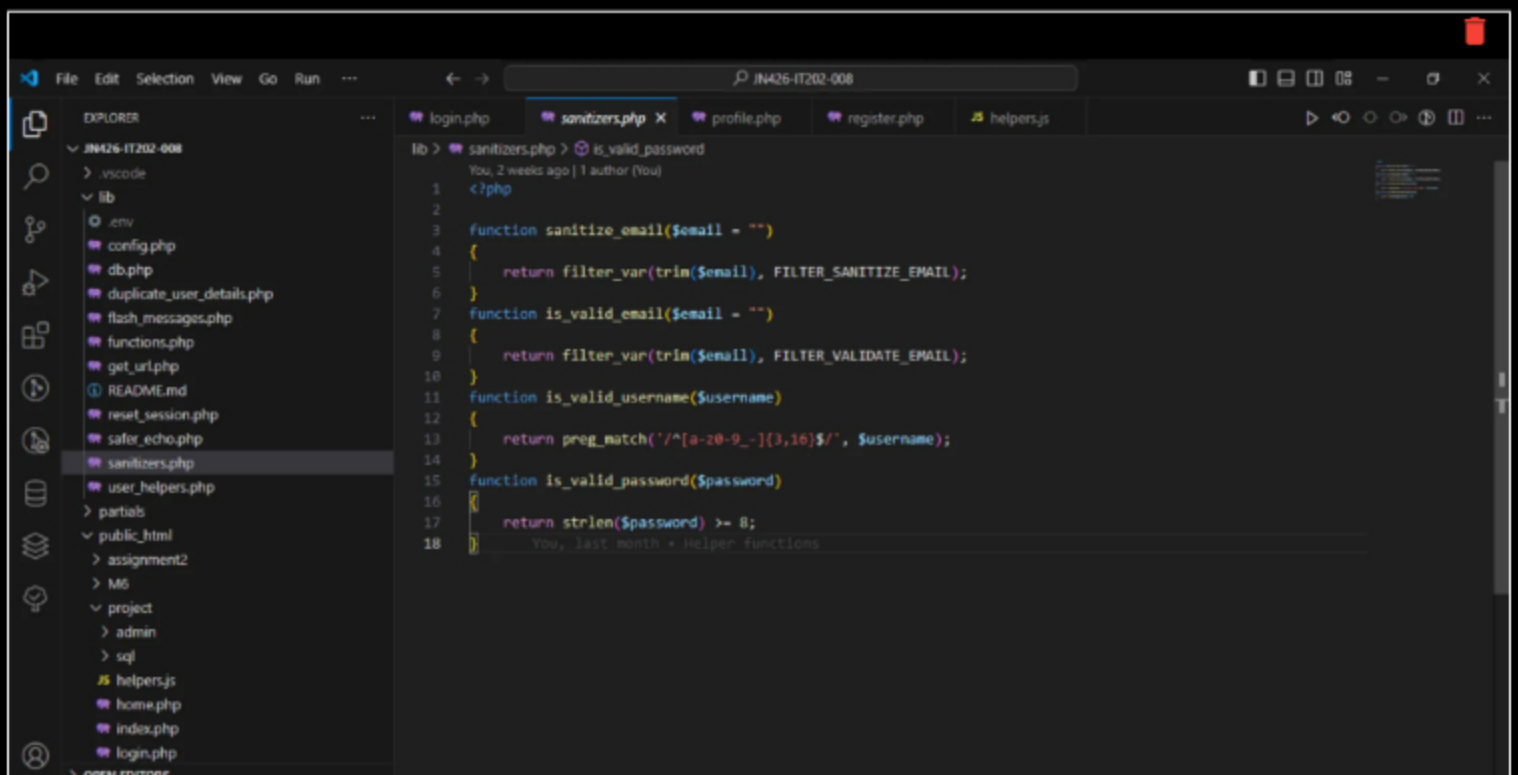
#1 Show the JavaScript validations (include any extra files related)



User helpers (lib file1)

Checklist Items (1)

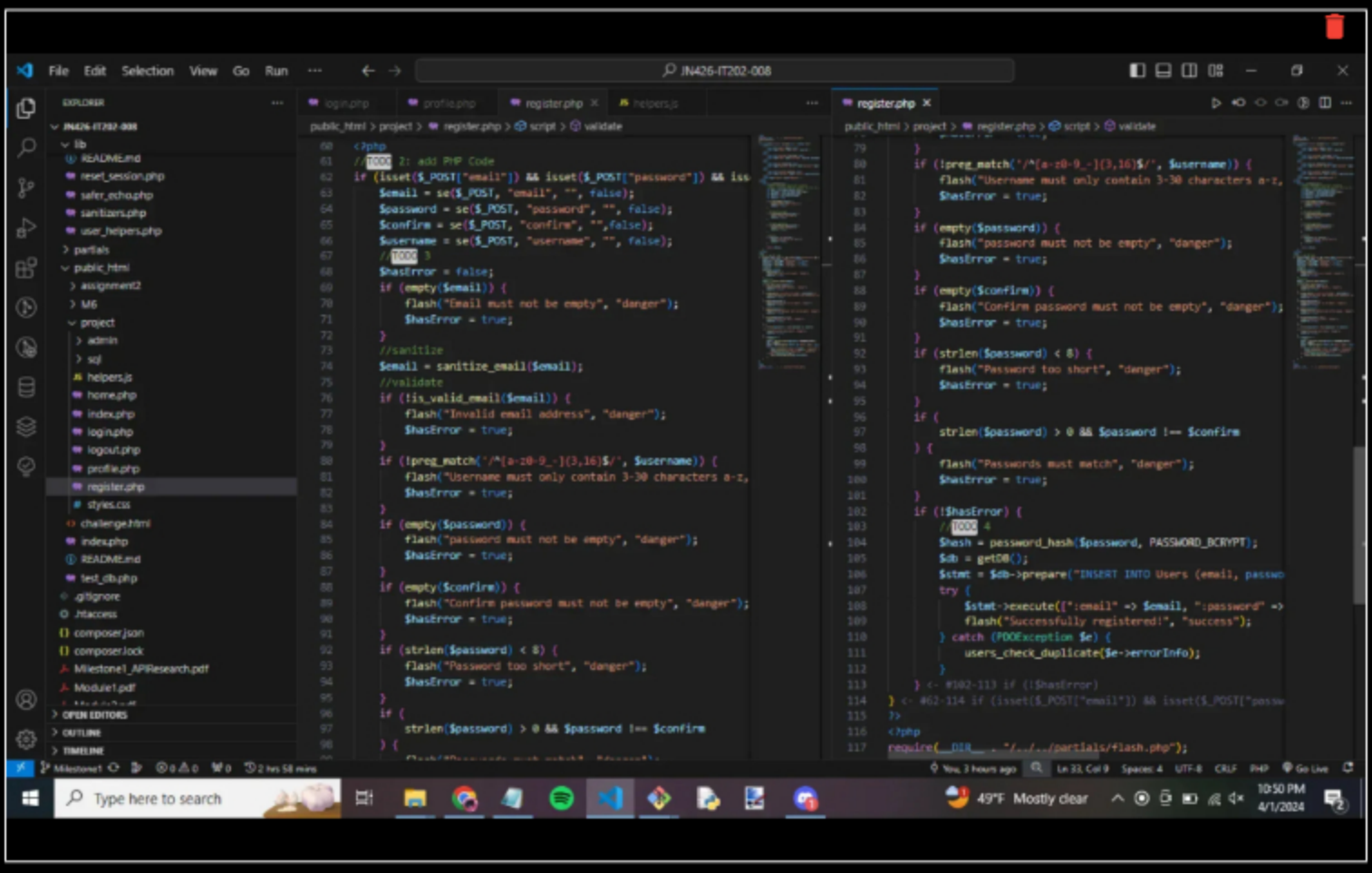
#2 Show the PHP validations (include any lib content)



lib file 2

Checklist Items (1)

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



php register

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

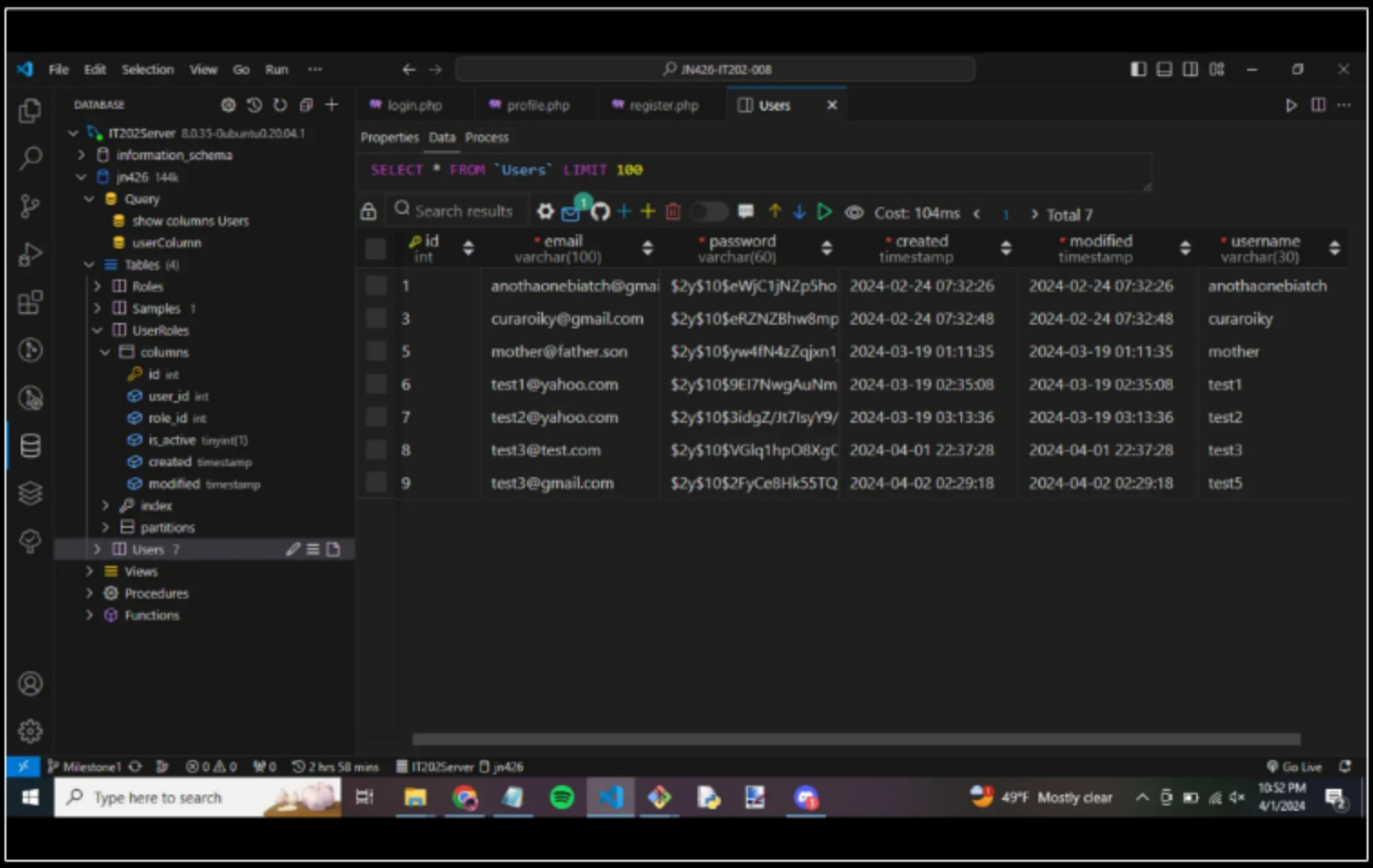
*The checkboxes are for your own tracking

#	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

Checklist Items (0)

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:

1. Page Load: HTML form with JavaScript validation and PHP handling.

2. Form Submission:

- JavaScript validates email, username, password, and confirmation.
- If validation fails, error messages are shown.

- If validation passes, data is submitted to the server.

3. Server-side Validation (PHP):

- PHP validates and sanitizes form data.
- Checks for empty fields, valid email format, and username constraints.
- Password and confirmation are checked for match.
- Flash messages display errors or success.

3. Database Interaction:

- If validation passes, password is hashed.
- Database connection established, user data inserted.
- Success or error messages flashed accordingly.

4. Final Result:

- Success message displayed if registration is successful.
- Error messages prompt user to correct input.

COLLAPSE

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/jennatnguyen/JN426-IT202-008/pull/24

User Login (2 pts.)

COLLAPSE

EXPAND

Task #1 - Points: 1

Text: Screenshot of form on website page

EXPAND

Task #2 - Points: 1

Text: Screenshot of the form code

EXPAND

Task #3 - Points: 1

Text: Explain the login logic step-by-step from when the page loads to when the data is fetched from the DB and stored in the session

EXPAND

Task #4 - Points: 1

Text: Include pull request links related to this feature

User Logout (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Capture the following screenshots

Checklist

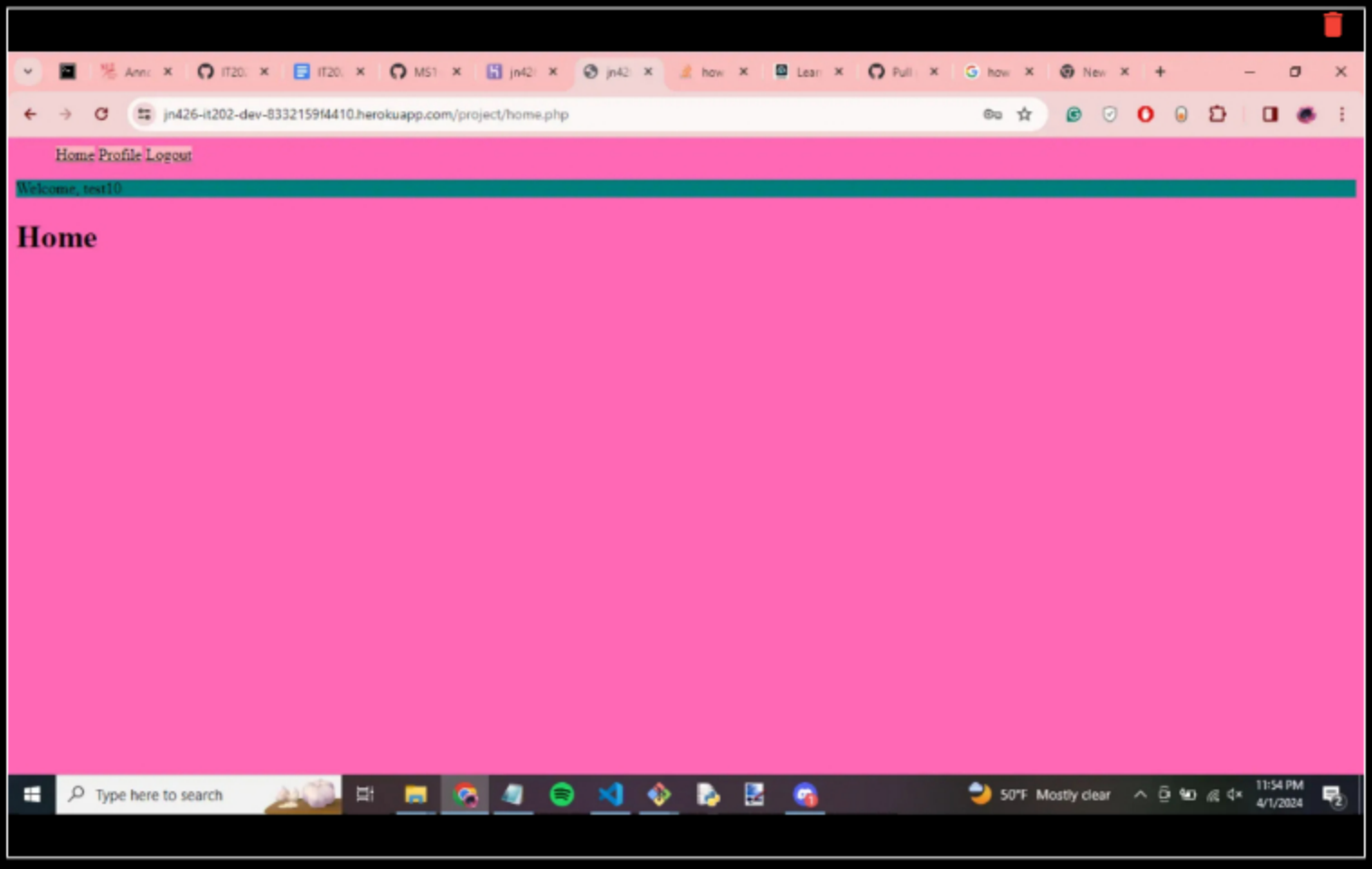
*The checkboxes are for your own tracking

#	Points	Details
#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:

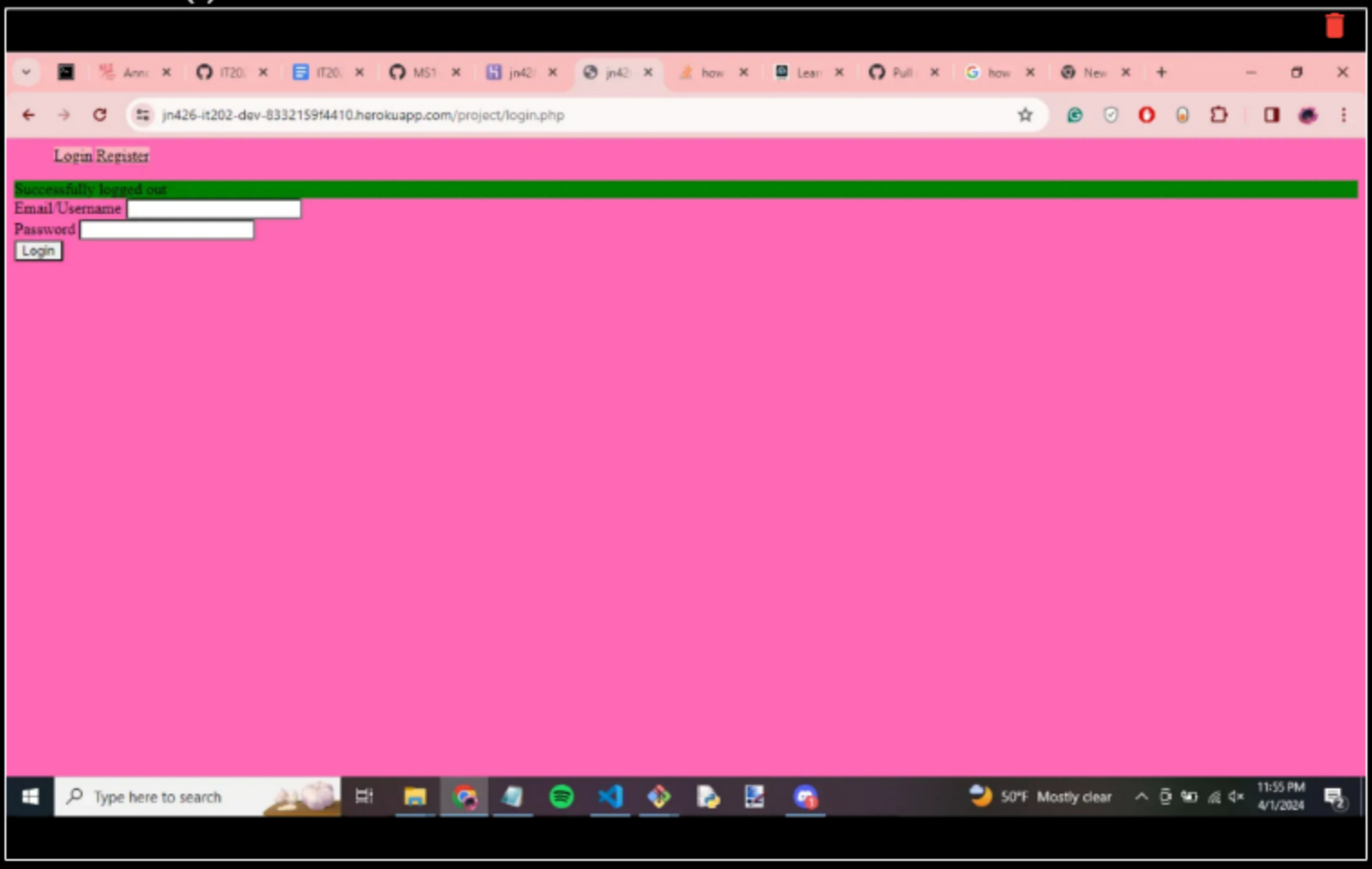
Gallery Style: Large View

Small Medium Large



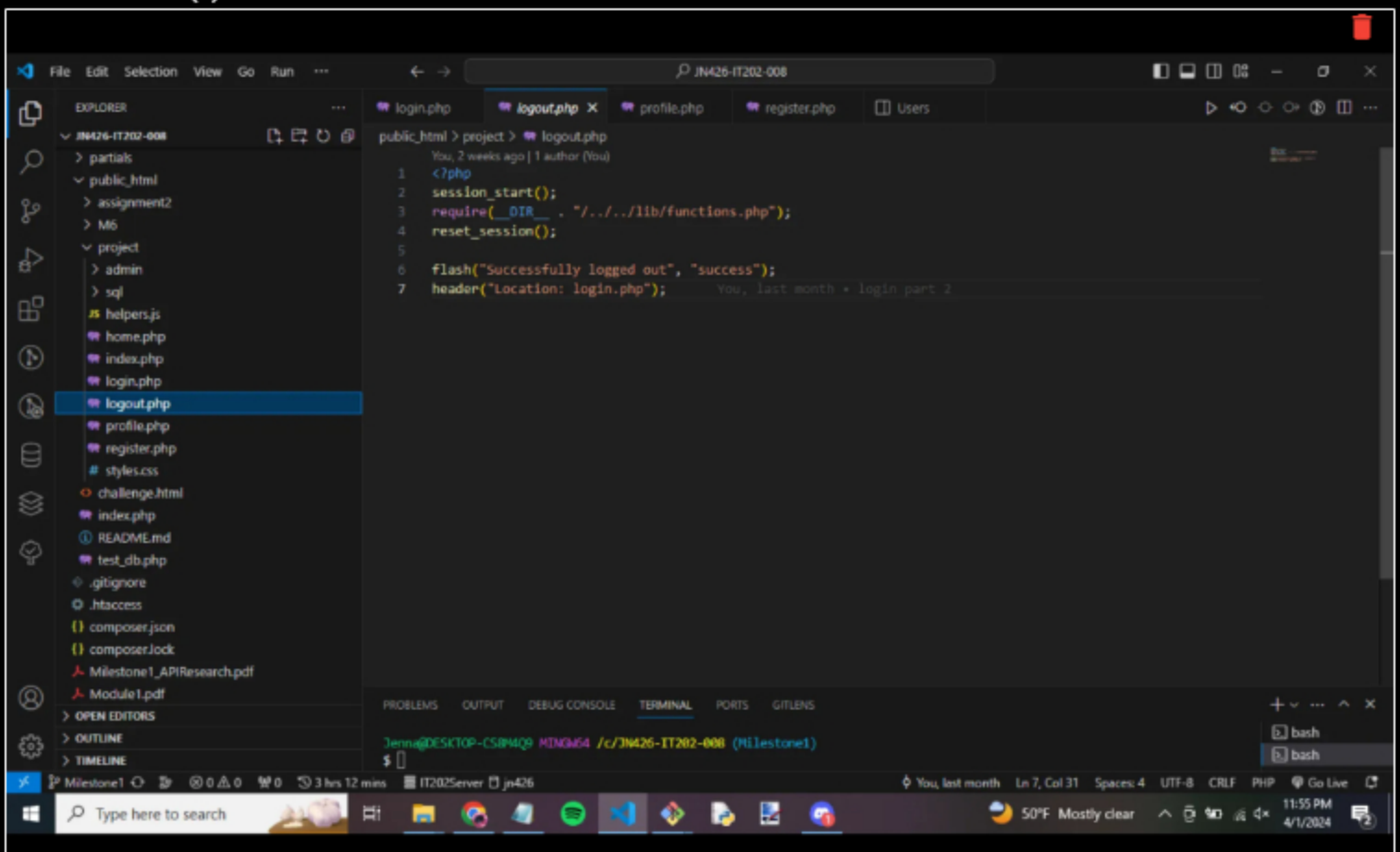
login

Checklist Items (0)



log out

Checklist Items (0)



logout code

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/jennatnguyen/JN426-IT202-008/pull/38

Basic Security Rules and Roles (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Authentication Screenshots

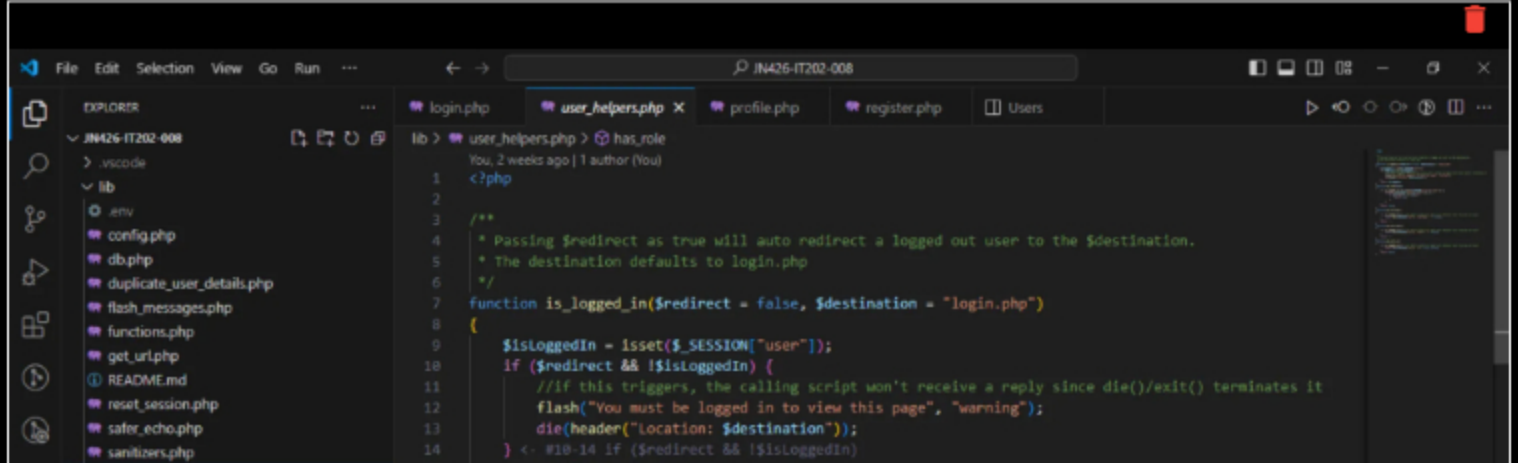
Checklist *The checkboxes are for your own tracking

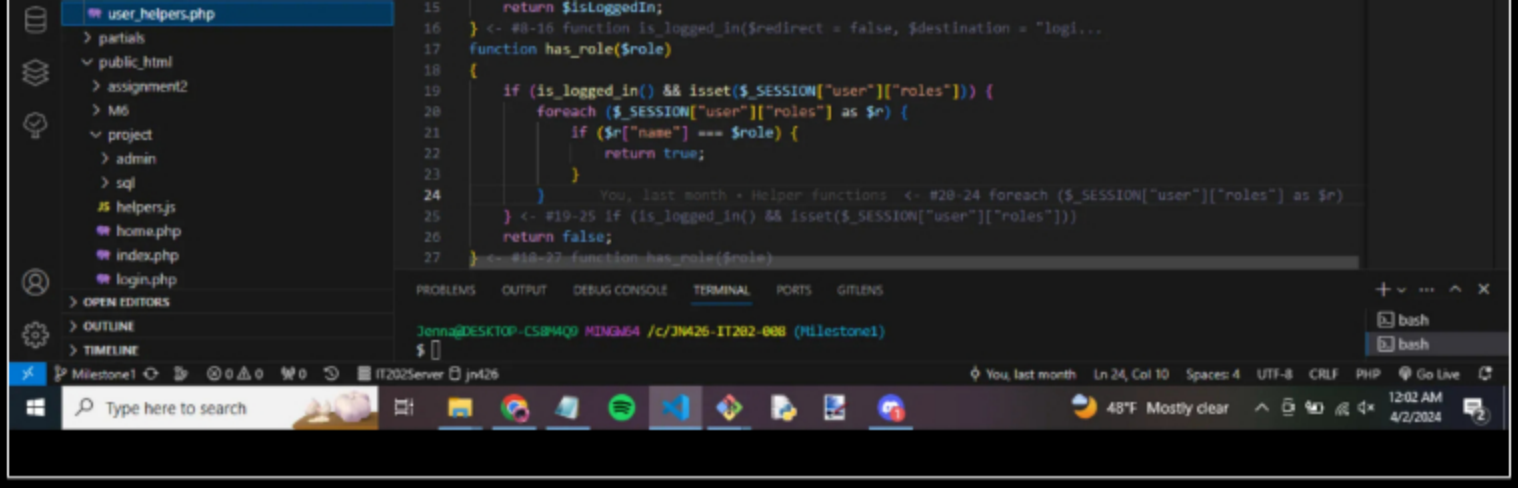
#	Points	Details
#1	1	Screenshot of the function that checks if a user is logged in
#2	1	Screenshot of the login check function being used (i.e., profile likely). Also caption what pages it's used on
#3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)
#4	1	Demonstrate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

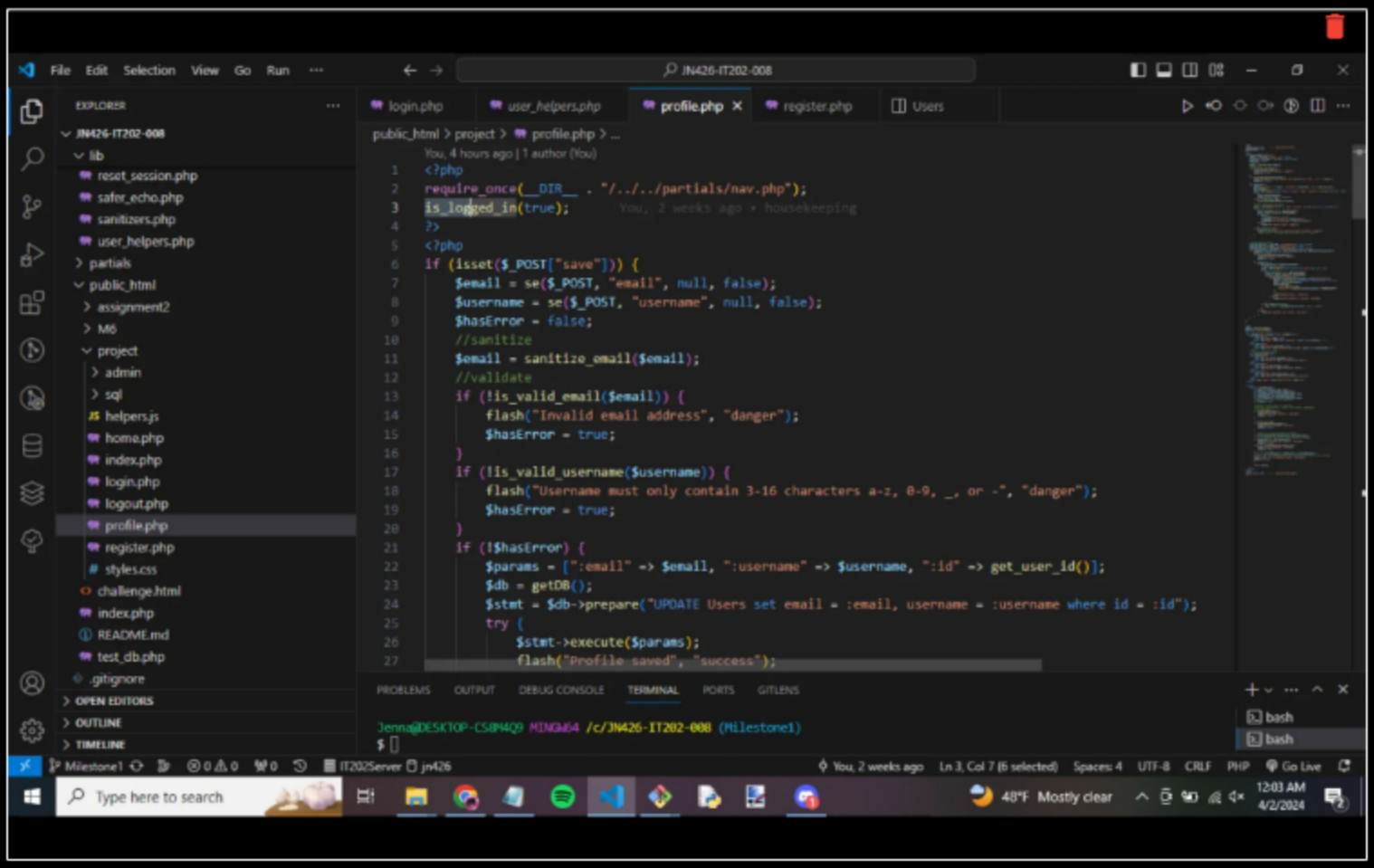




checks if user is logged in

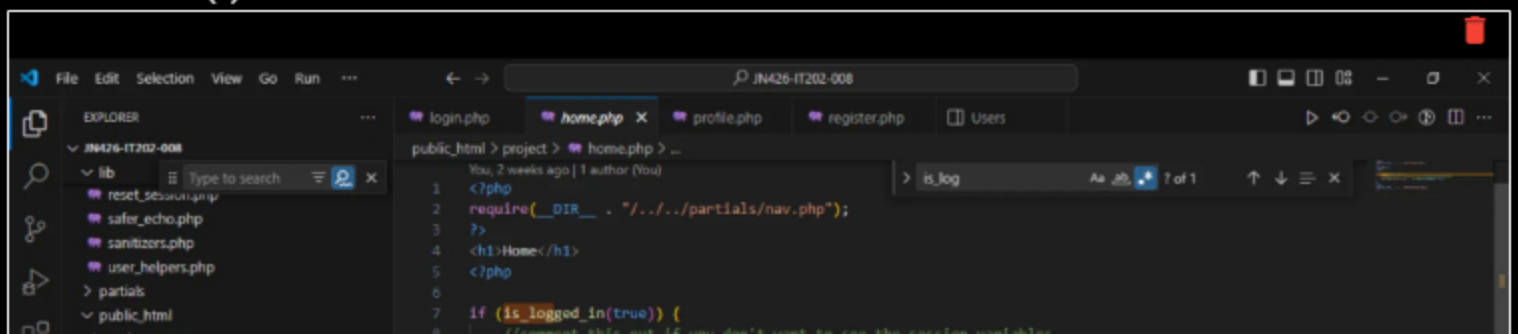
Checklist Items (1)

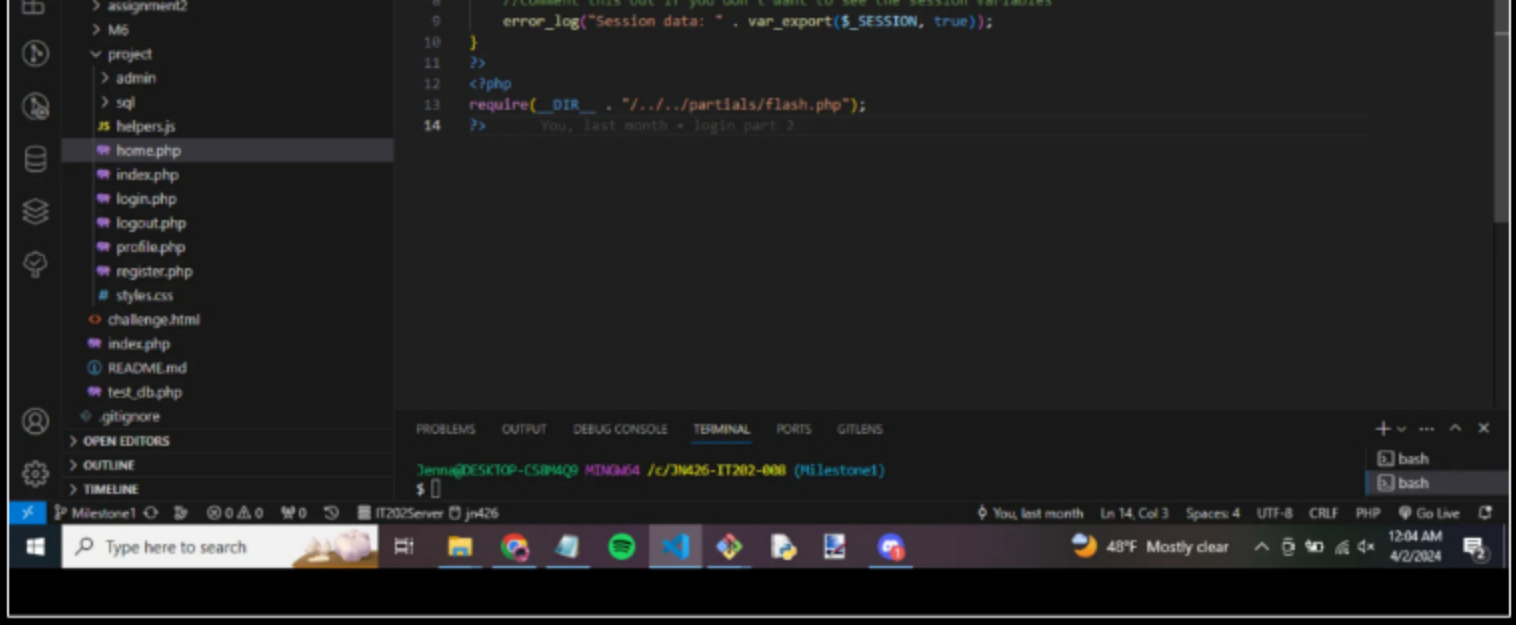
#1 Screenshot of the function that checks if a user is logged in



function in profile

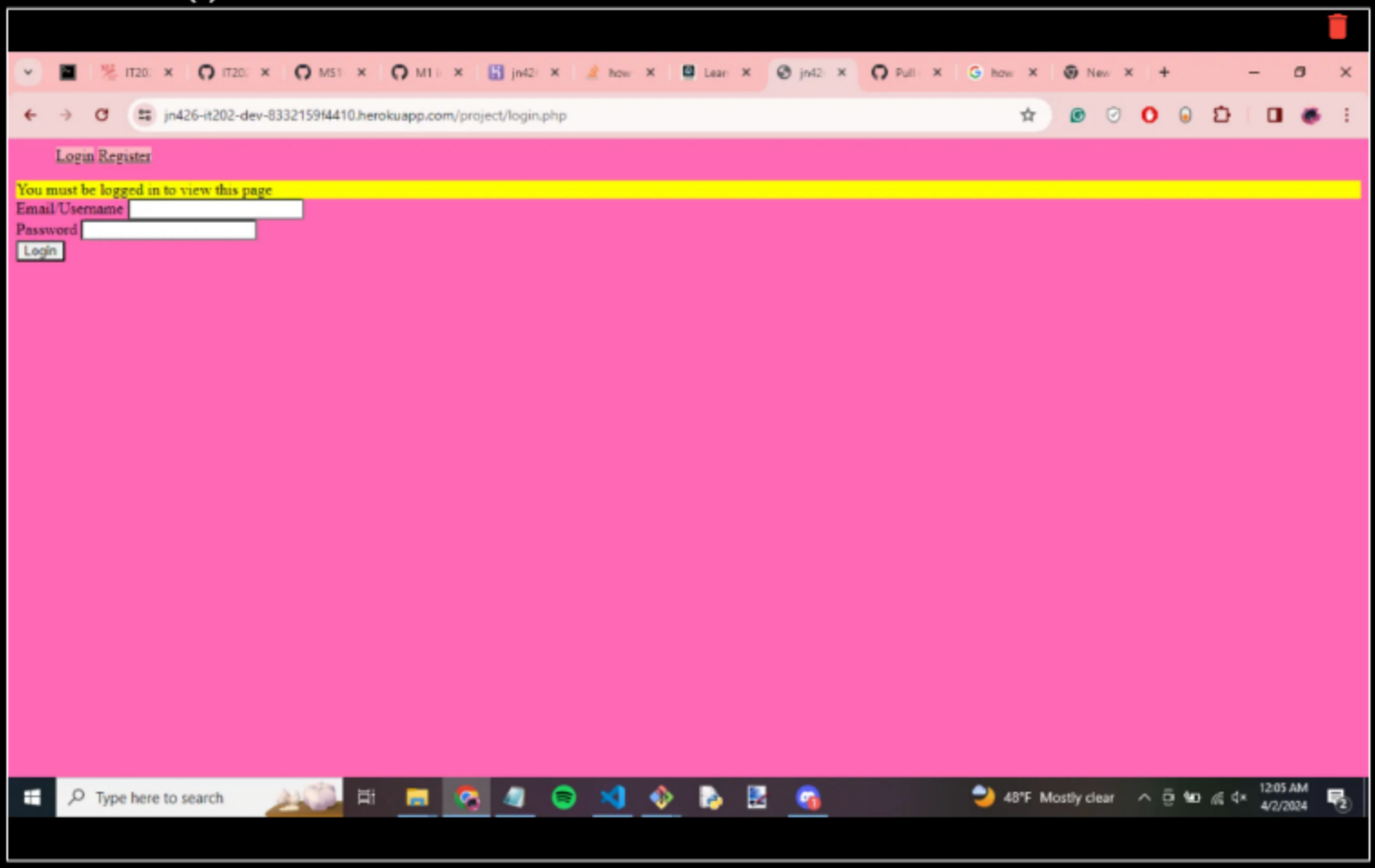
Checklist Items (0)





function in home

Checklist Items (0)



user friendly msg

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Authorization Screenshots

Checklist

*The checkboxes are for your own tracking

Checklist

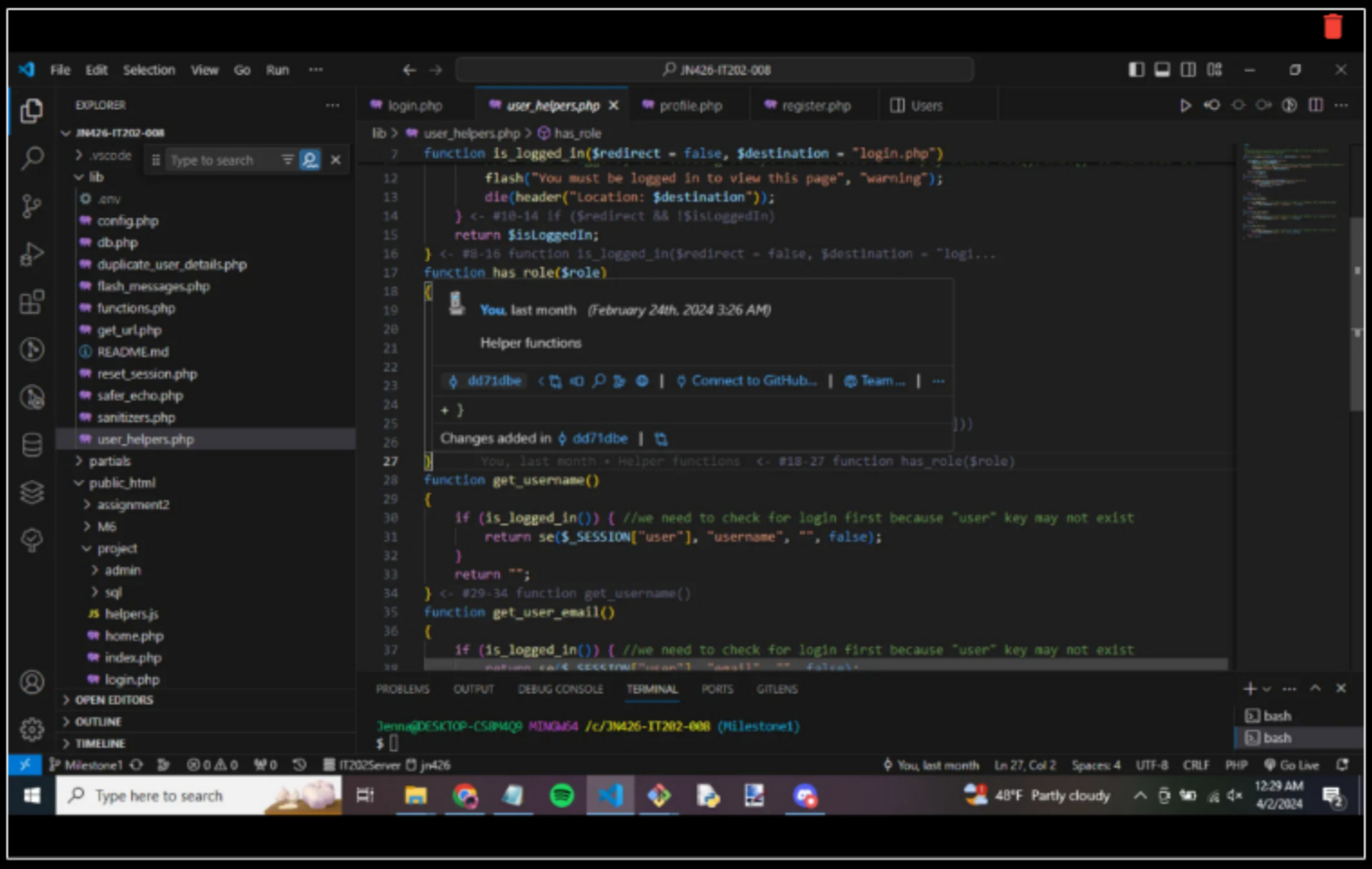
The checkboxes are for your own tracking

#	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:

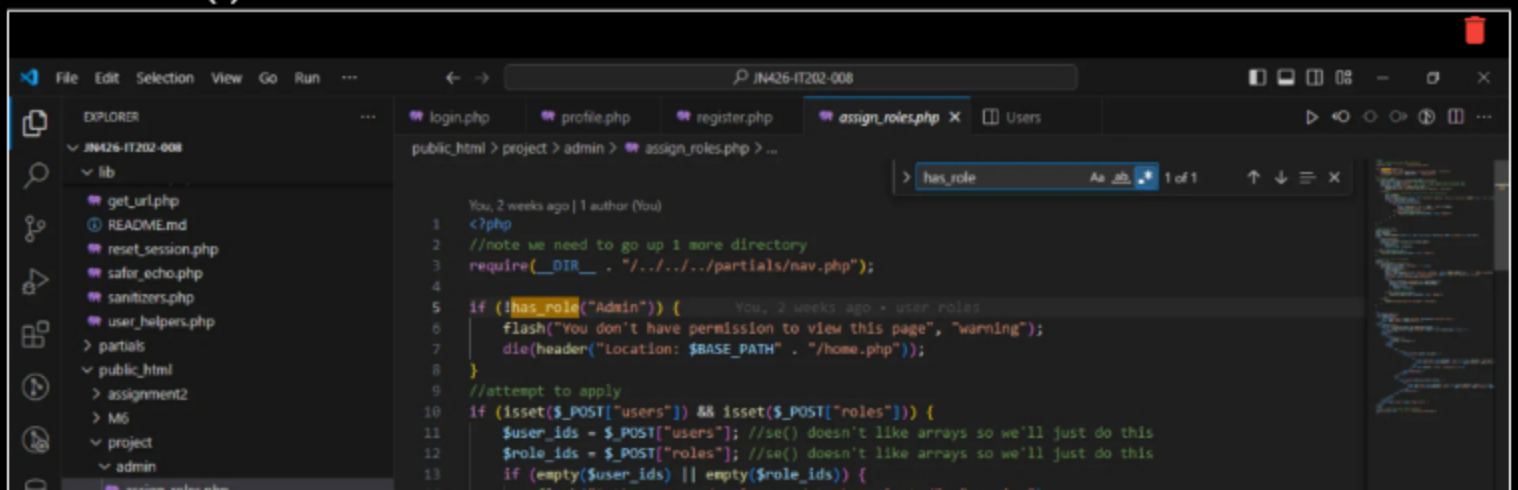
Gallery Style: Large View

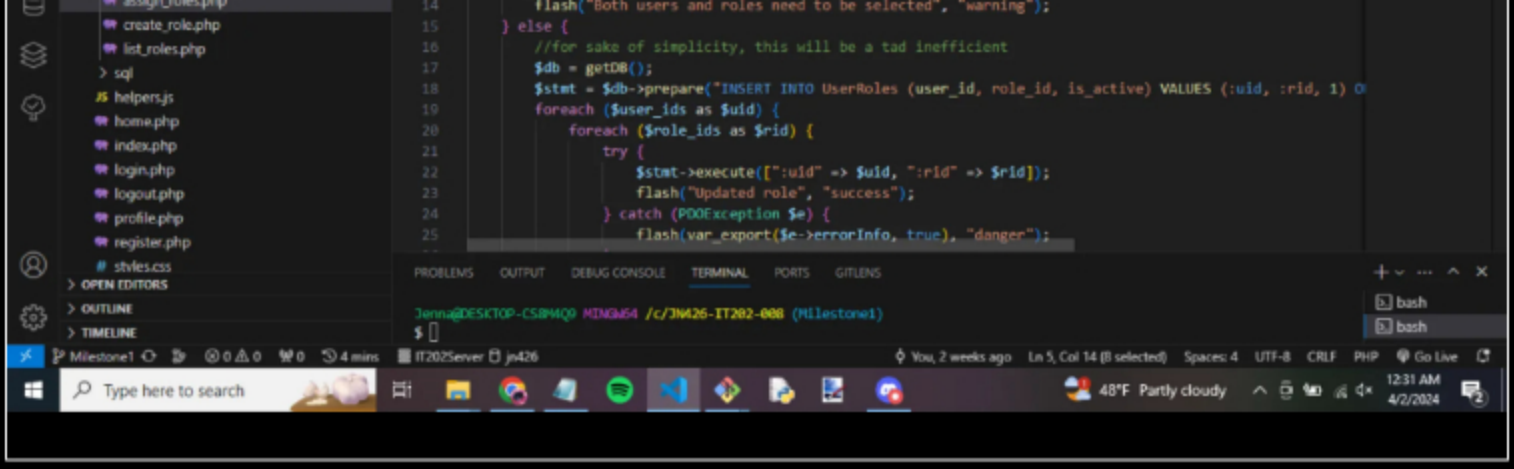
Small Medium Large



has roles function

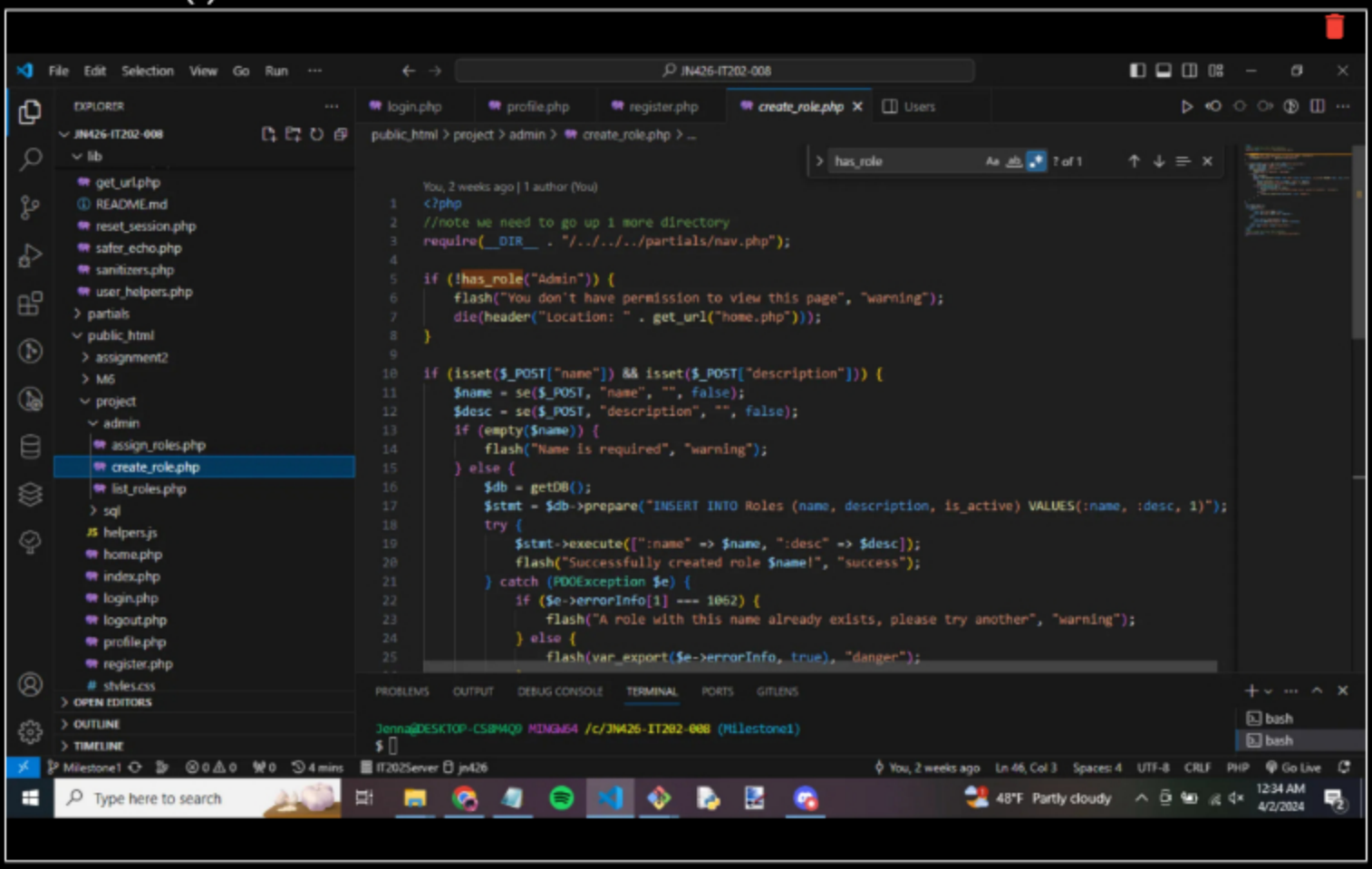
Checklist Items (0)





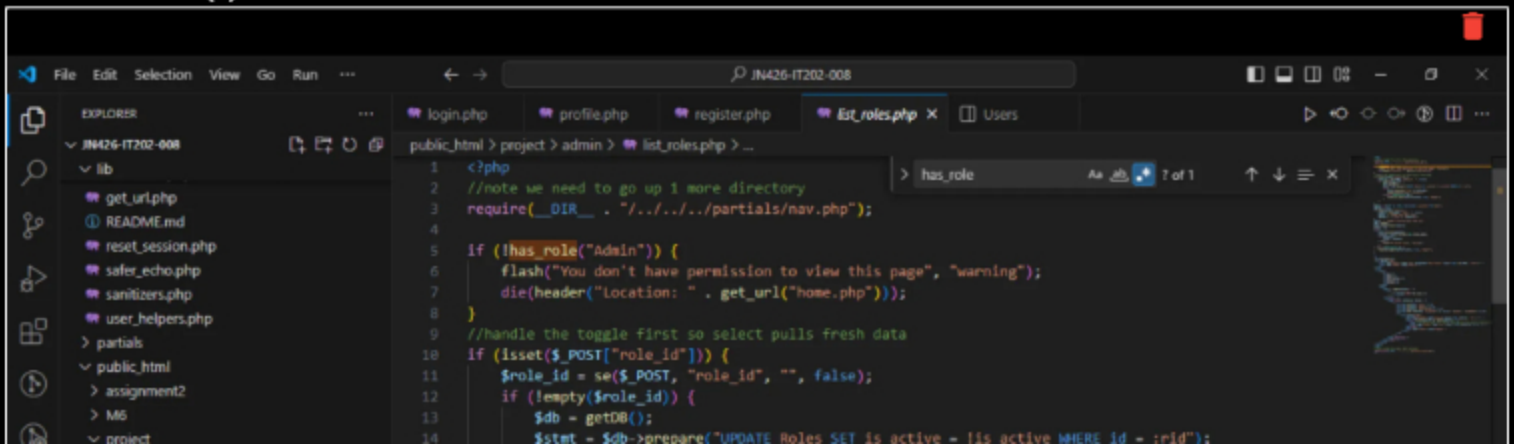
hasroles in assign_roles

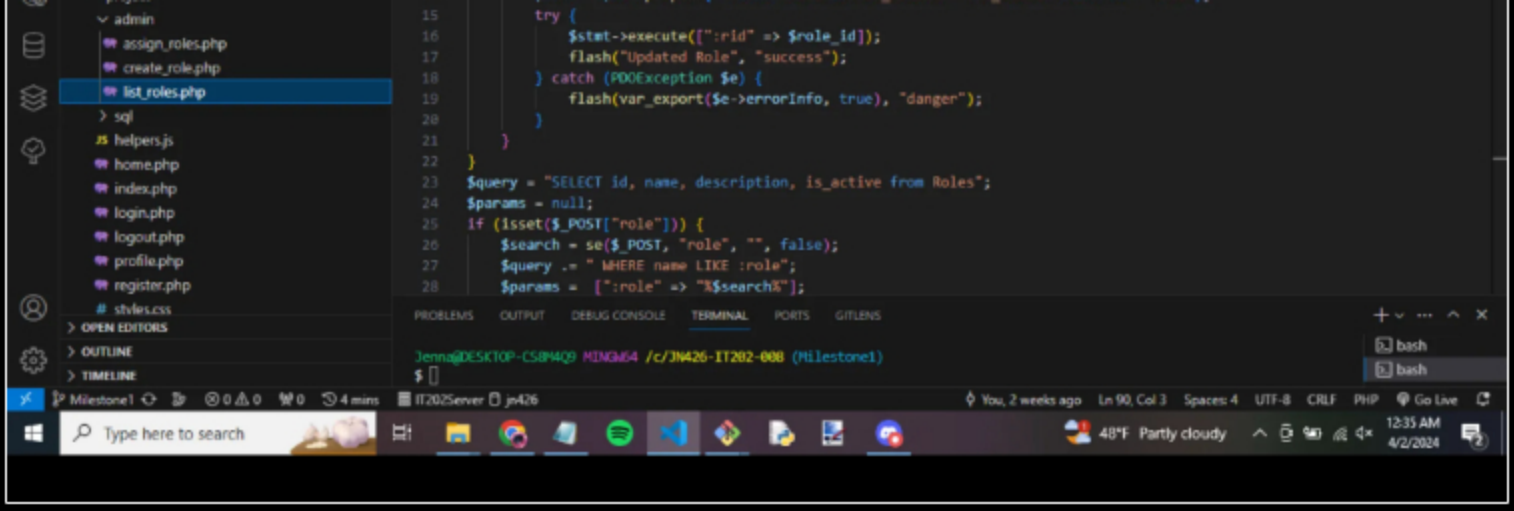
Checklist Items (0)



has role in create_role

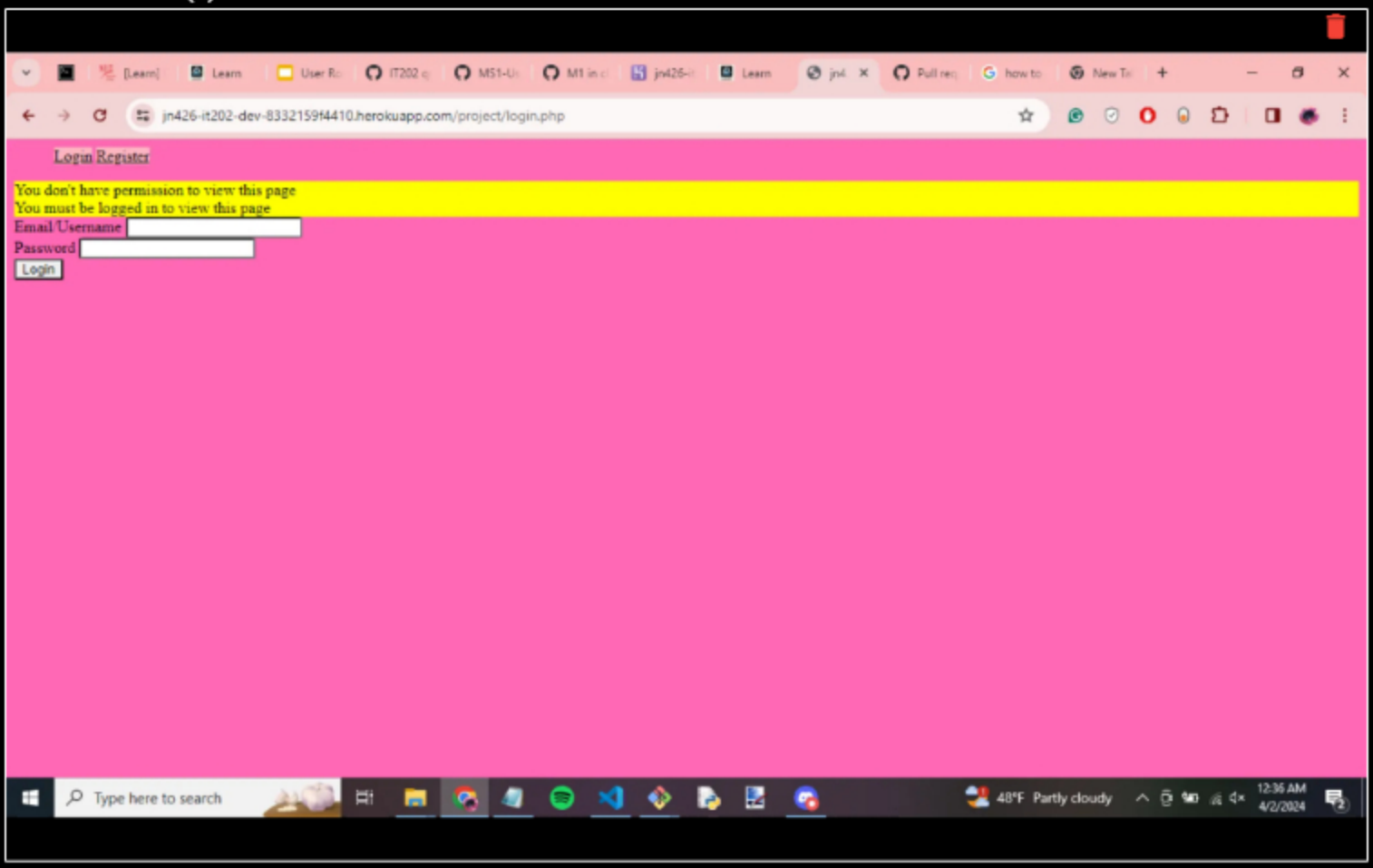
Checklist Items (0)





has role in list role

Checklist Items (0)



user friendly message

Checklist Items (0)

COLLAPSE

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

Checklist

*The checkboxes are for your own tracking

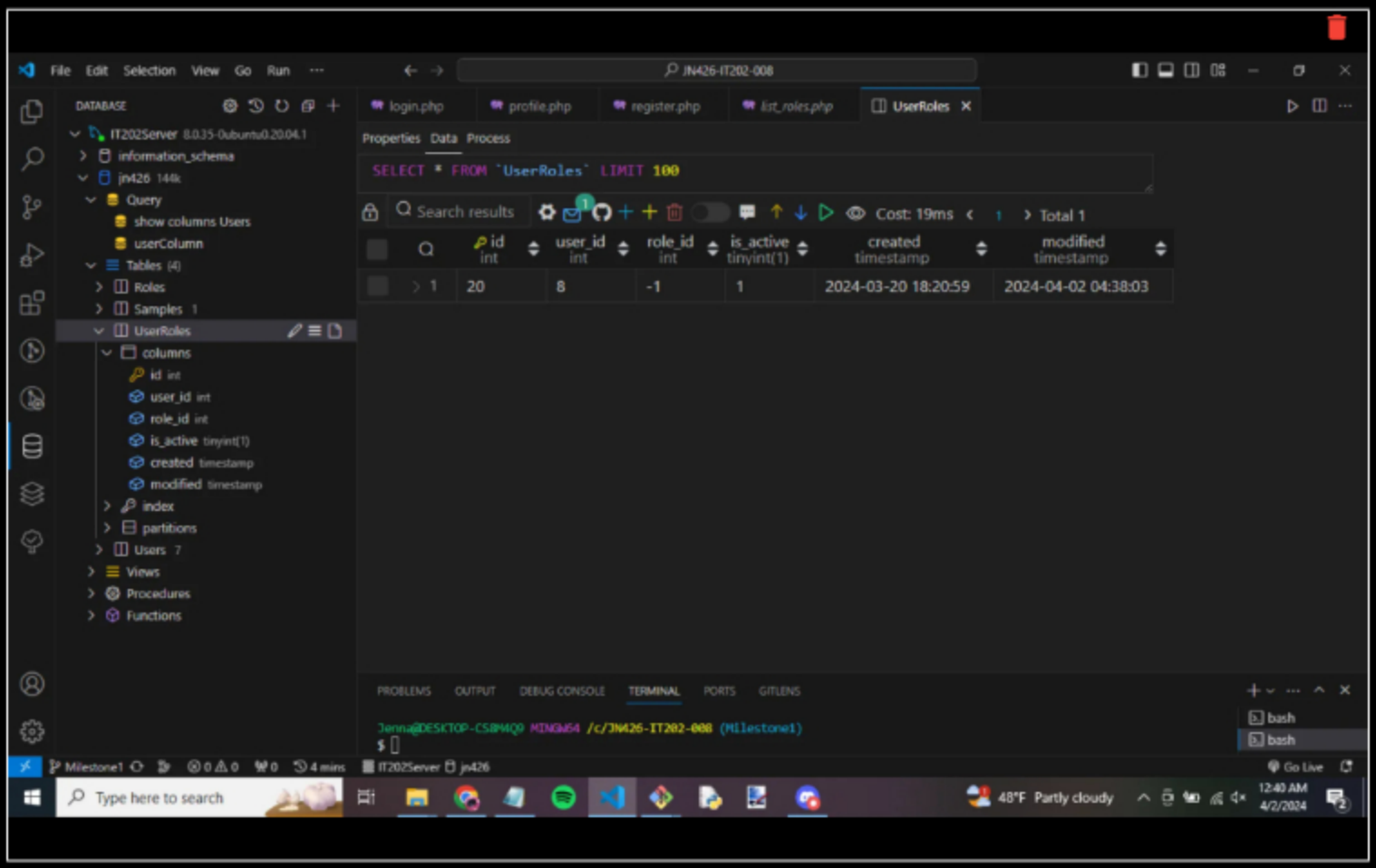
#	Points	Details

#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:

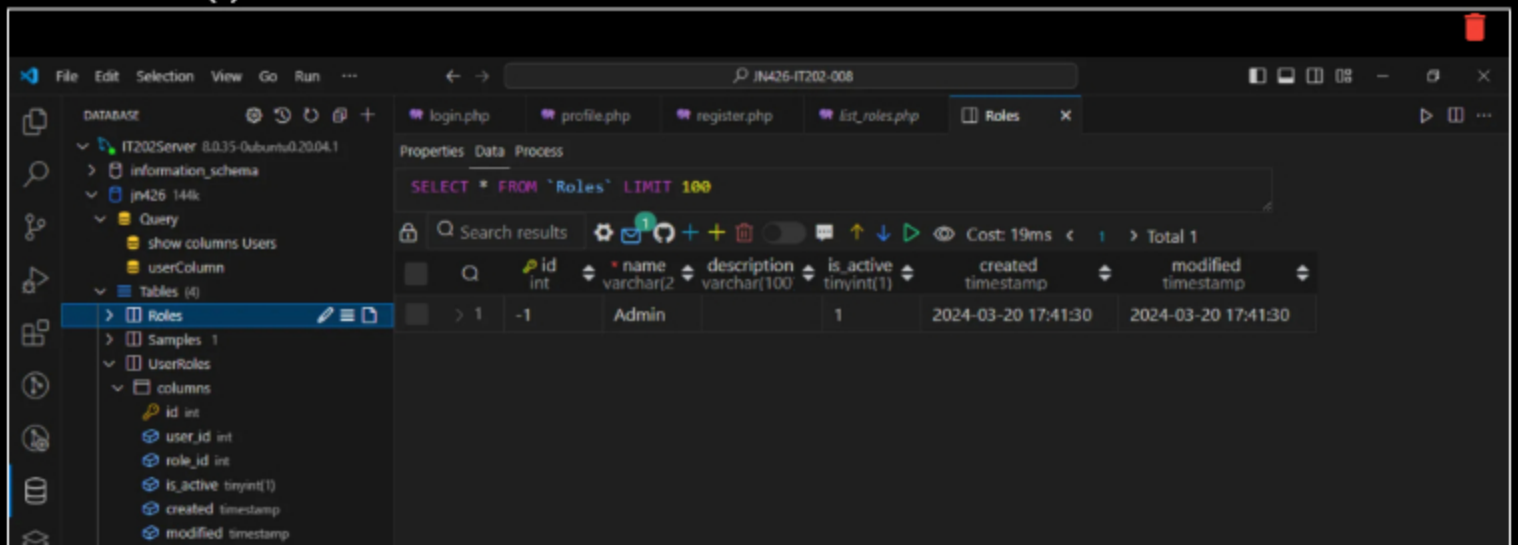
Gallery Style: Large View

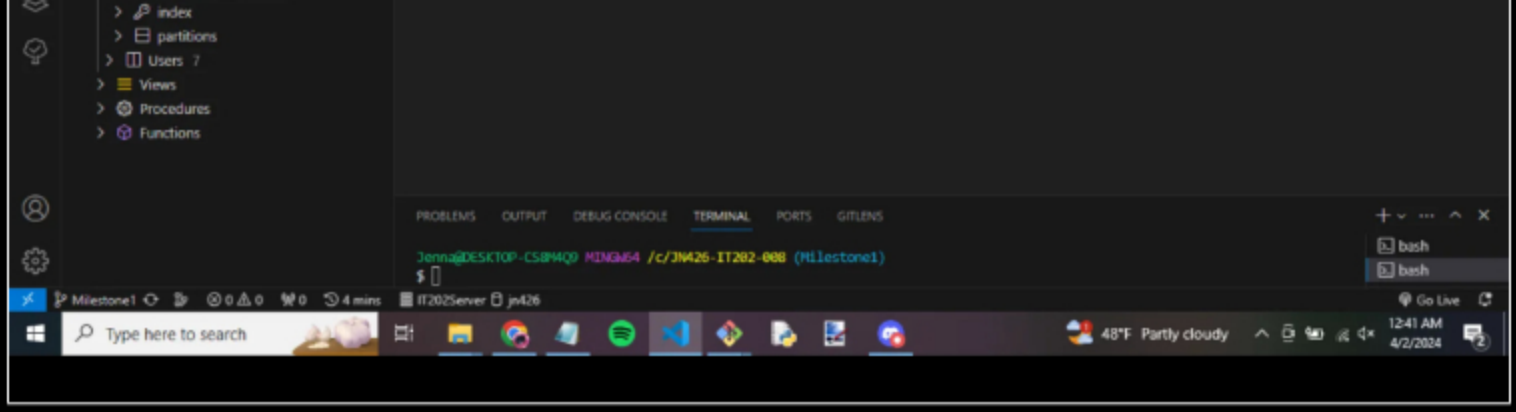
Small Medium Large



UserRoles table checklist 1 and 2

Checklist Items (0)





Roles table 3 & 4

Checklist Items (0)

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 establish a many to many relationship. A role can have many users and a user can have many roles. It allows for certain users to have different permissions. Roles.is_active is a global on and off switch for the role, and UserRoles.is_active is a user level active switch for the 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/jennatnguyen/JN426-IT202-008/pull/39

User Profile (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: View Profile Website Page

Checklist

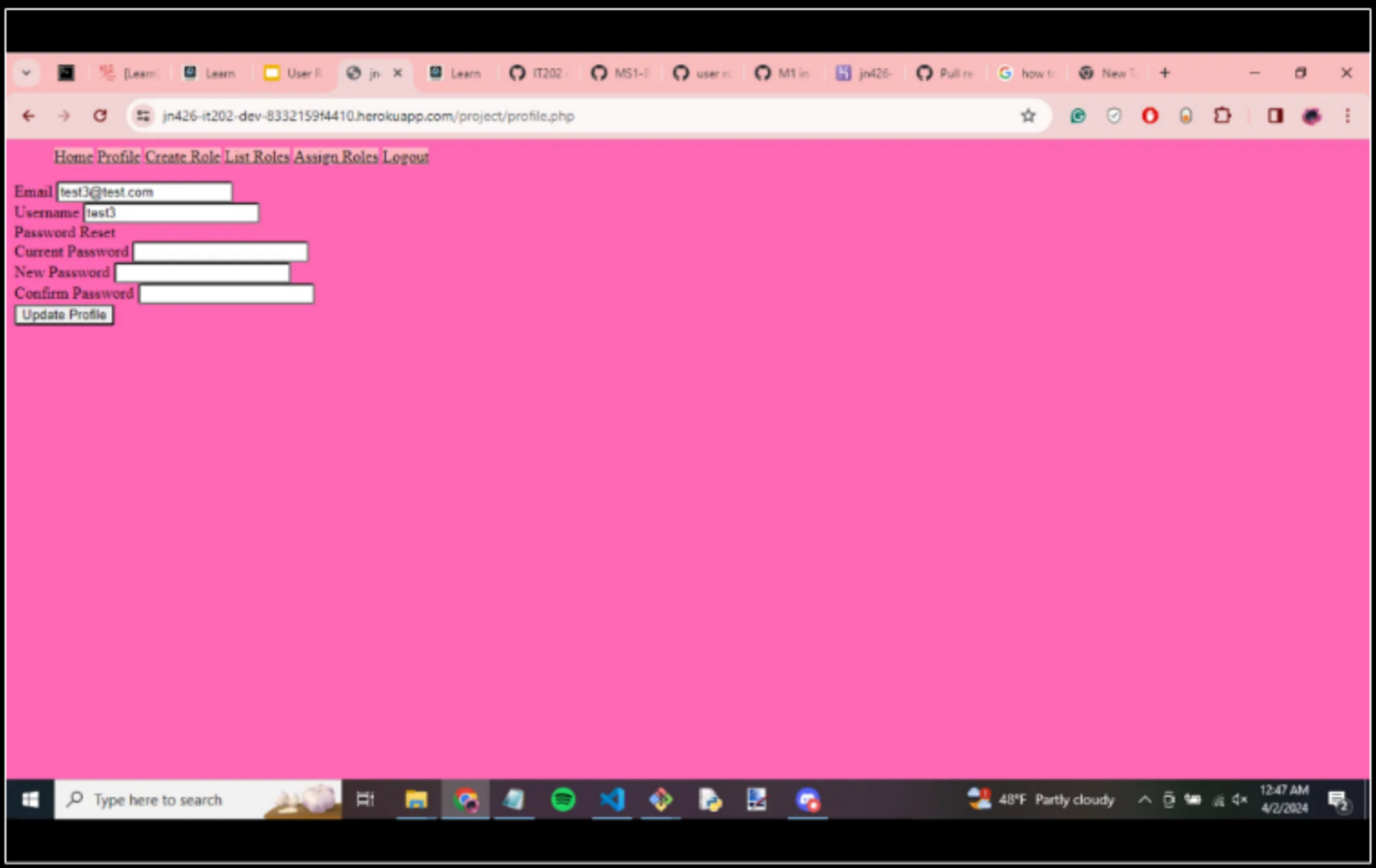
*The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



profile

Checklist Items (0)

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:

1. Require files and check login status: The code includes necessary files and ensures the user is logged in.

2. Handle form submission: If the form is submitted, it updates the user's email, username, and password if provided, after validating the inputs.

3. Load user data: It retrieves the current email and username of the logged-in user to pre-fill the form fields.

4. Client-side validation: JavaScript validates form inputs before submission to provide instant feedback to the user.

5. Display flash messages: Any feedback or error messages are shown to the user after form submission or client-side validation.

COLLAPSE

Task #3 - Points: 1

Text: Edit Profile Website Page

Checklist

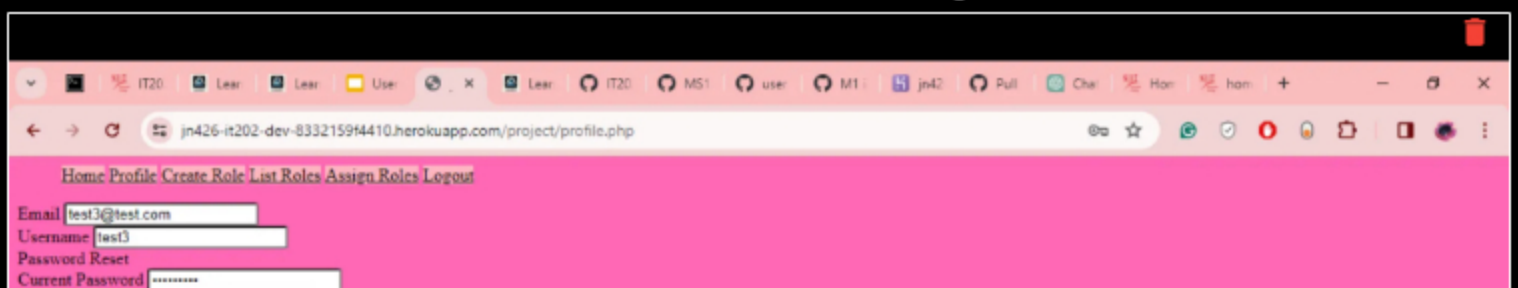
*The checkbox are for your own tracking

#	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)

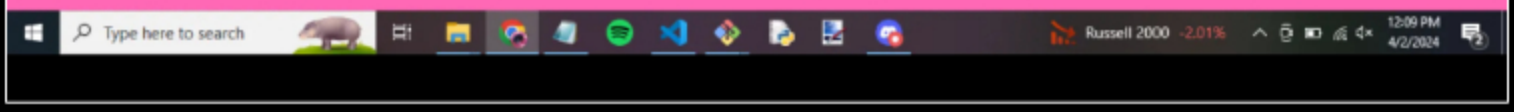
Task Screenshots:

Gallery Style: Large View

Small Medium Large

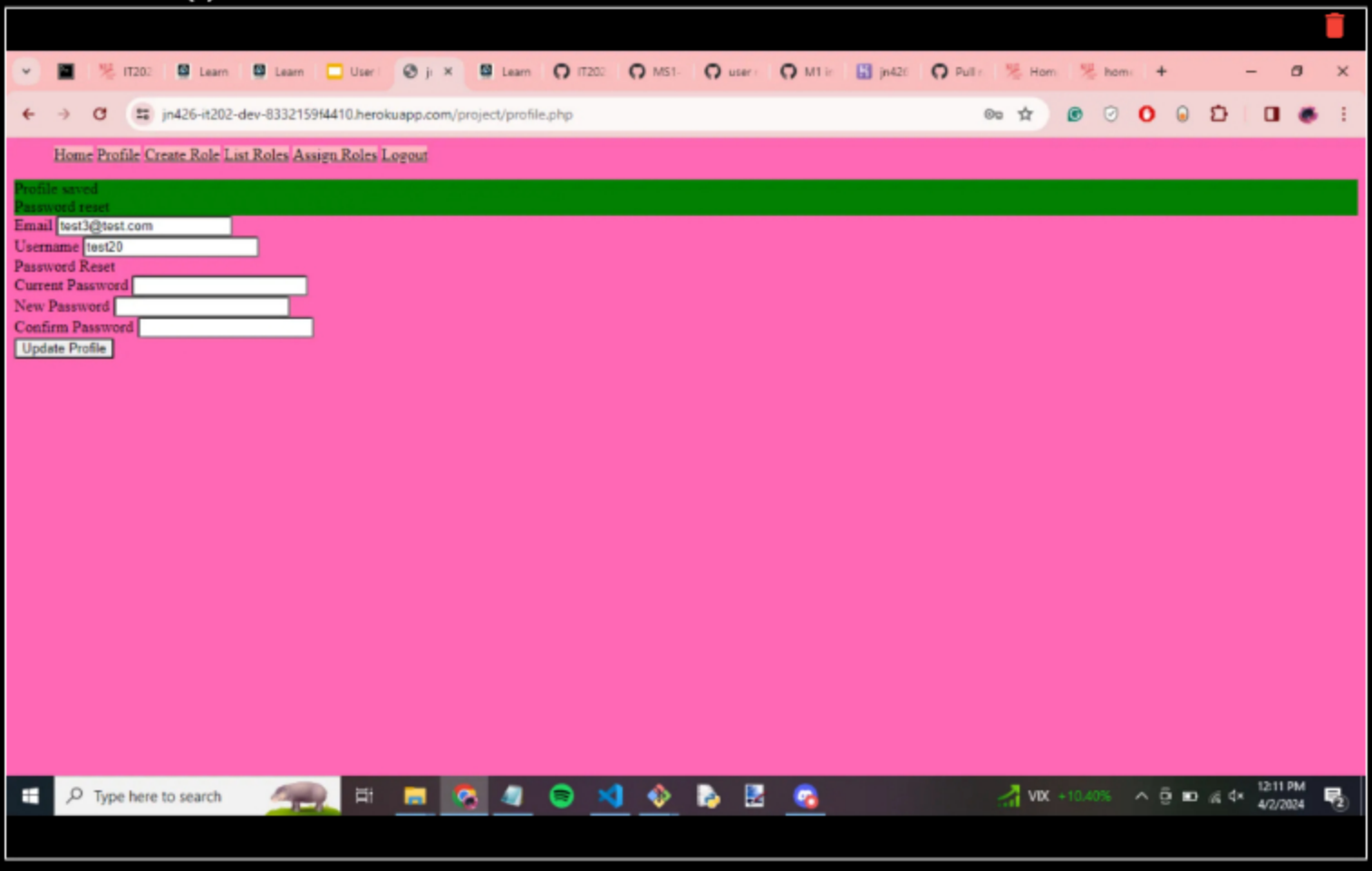


New Password
Confirm Password
Update Profile



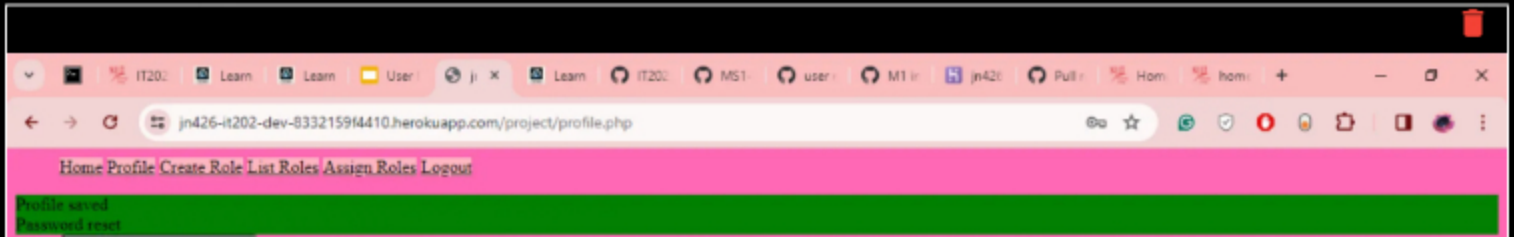
before username/email change

Checklist Items (0)

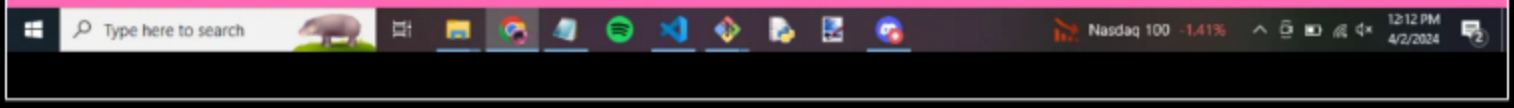


Username changed

Checklist Items (0)

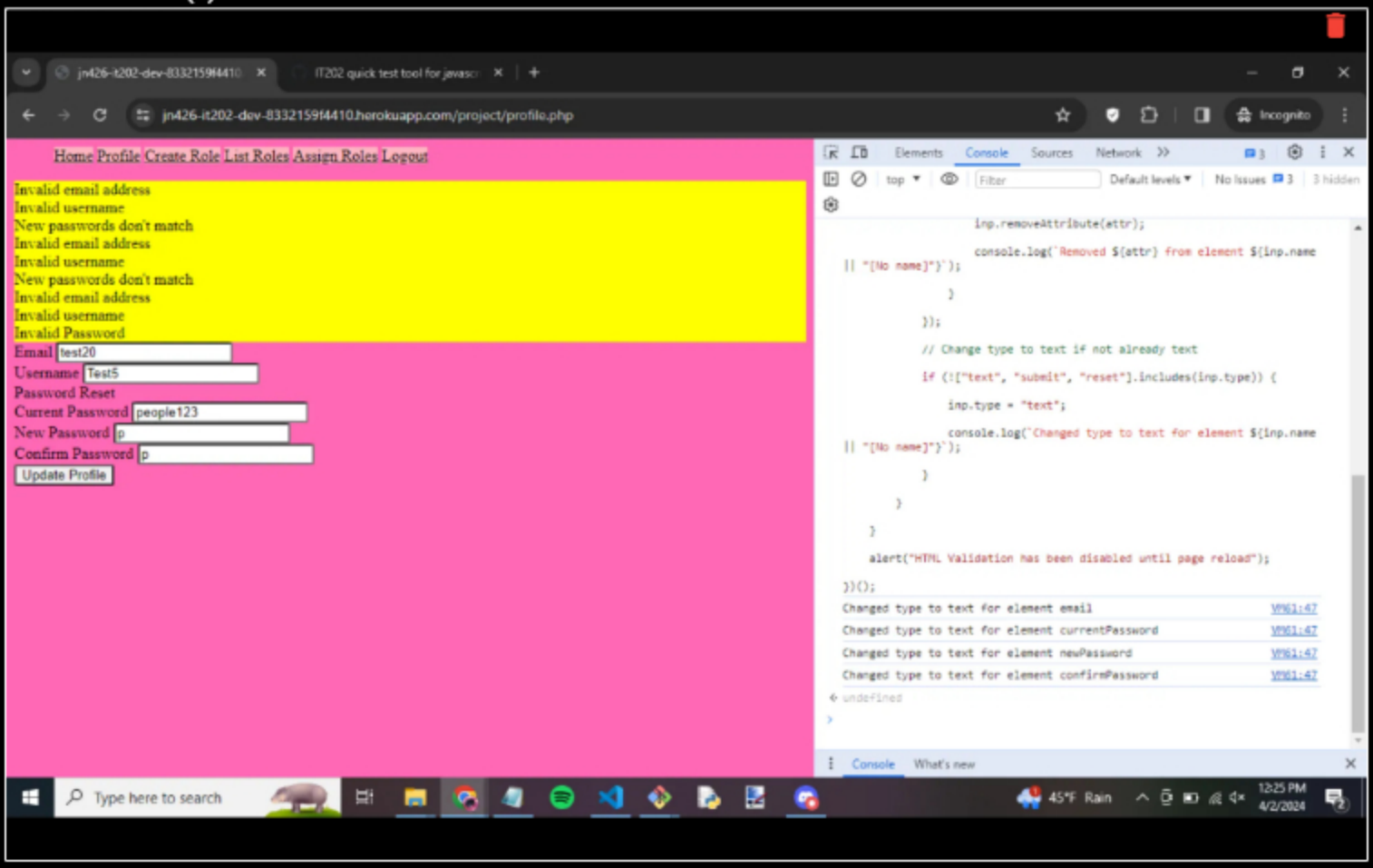


Email test20@test.com
Username test20
Password Reset
Current Password
New Password
Confirm Password
Update Profile



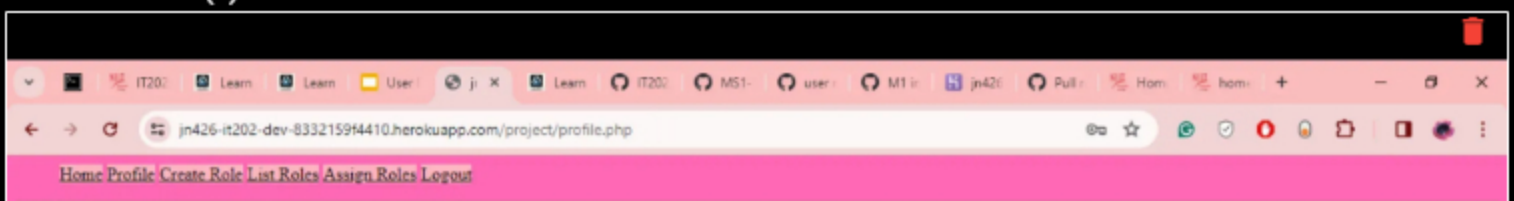
email changed

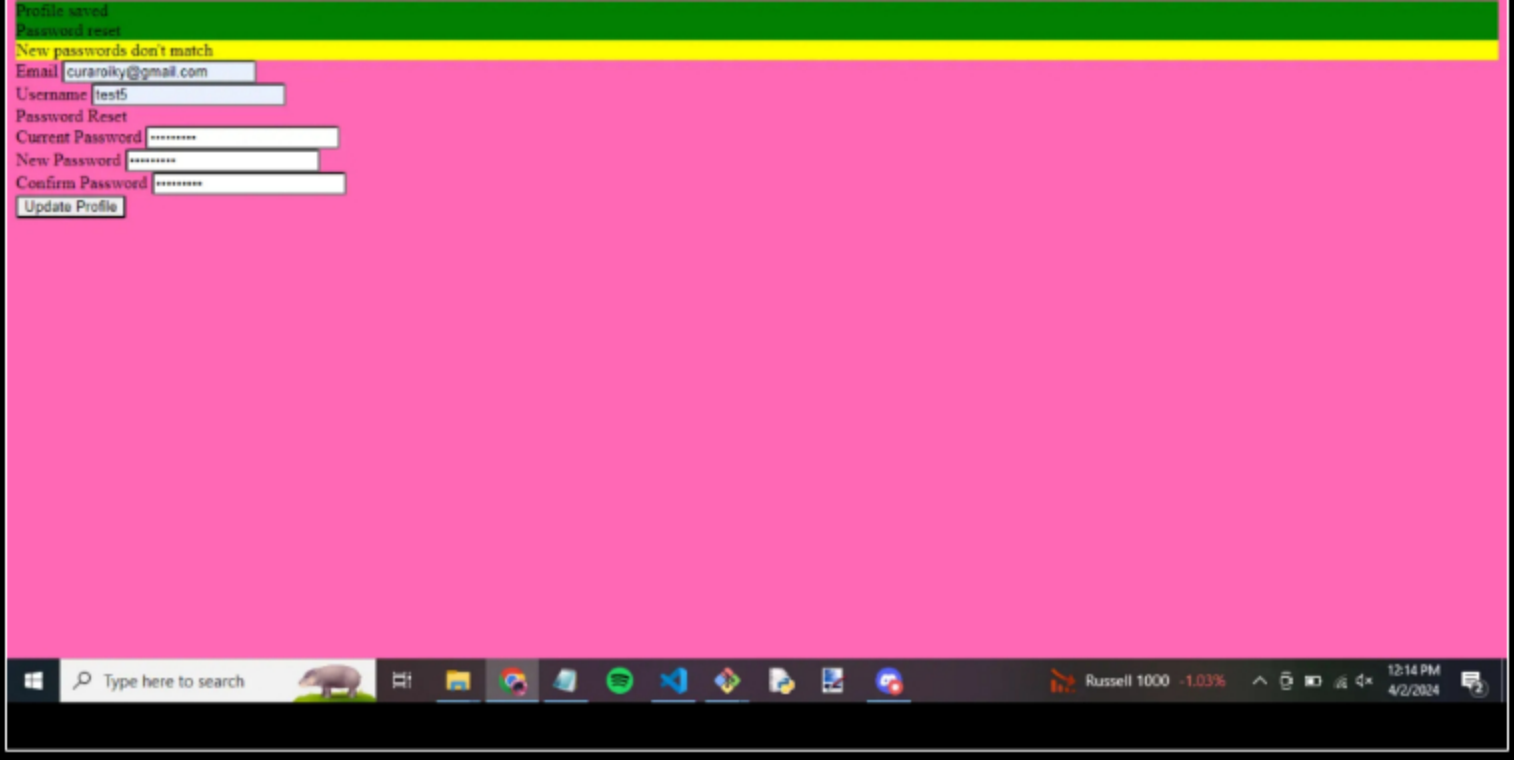
Checklist Items (0)



JS validation messages

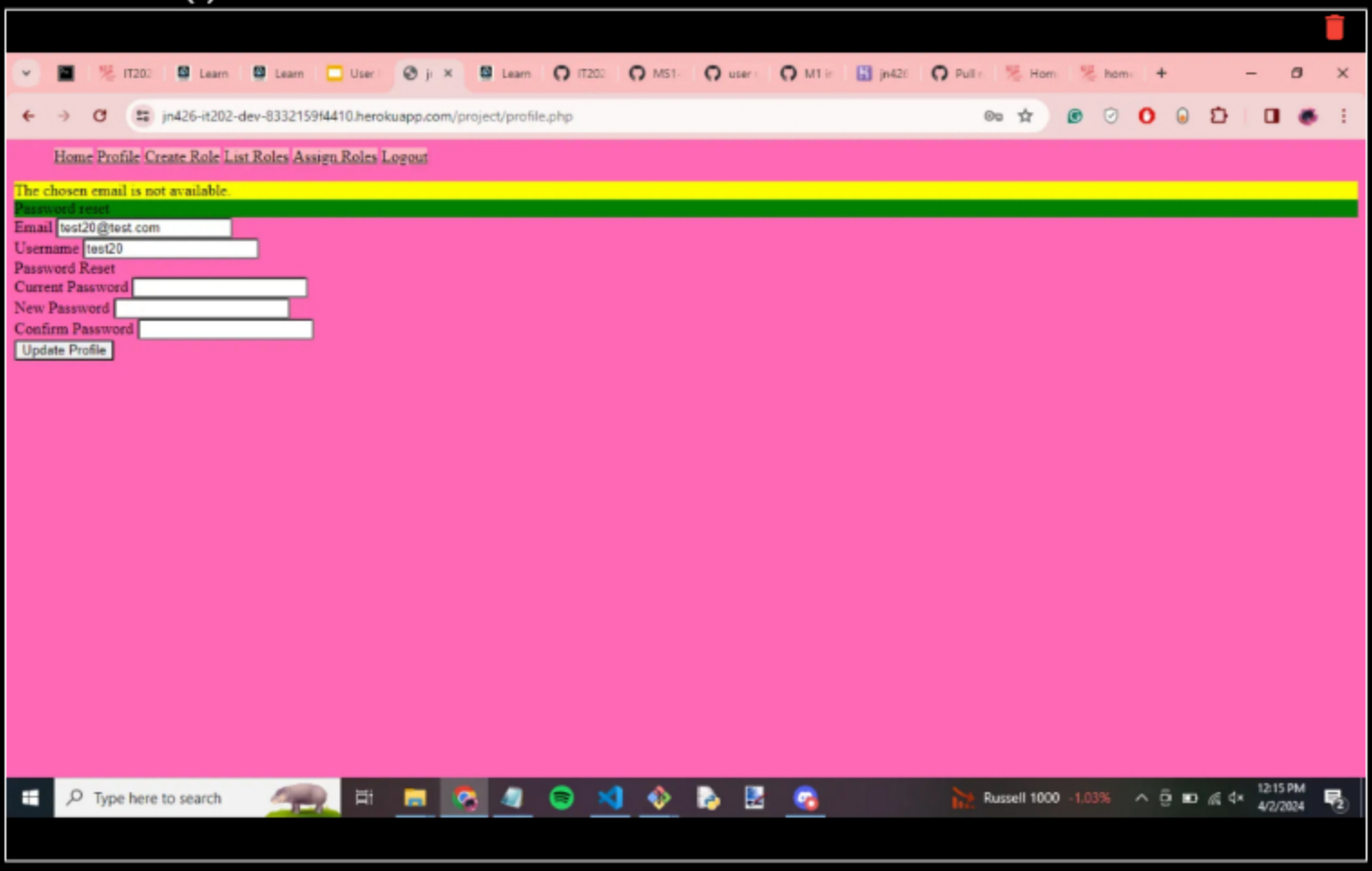
Checklist Items (0)





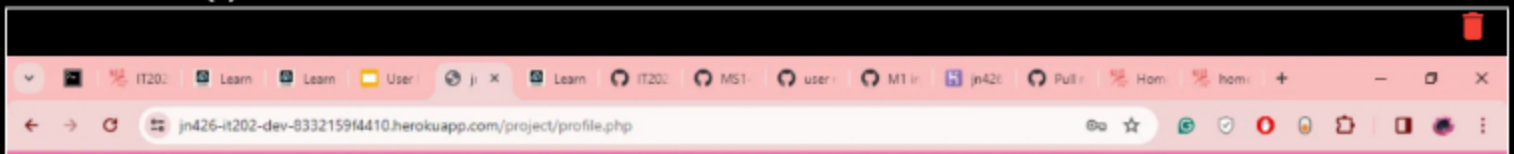
php validation passwords dont match

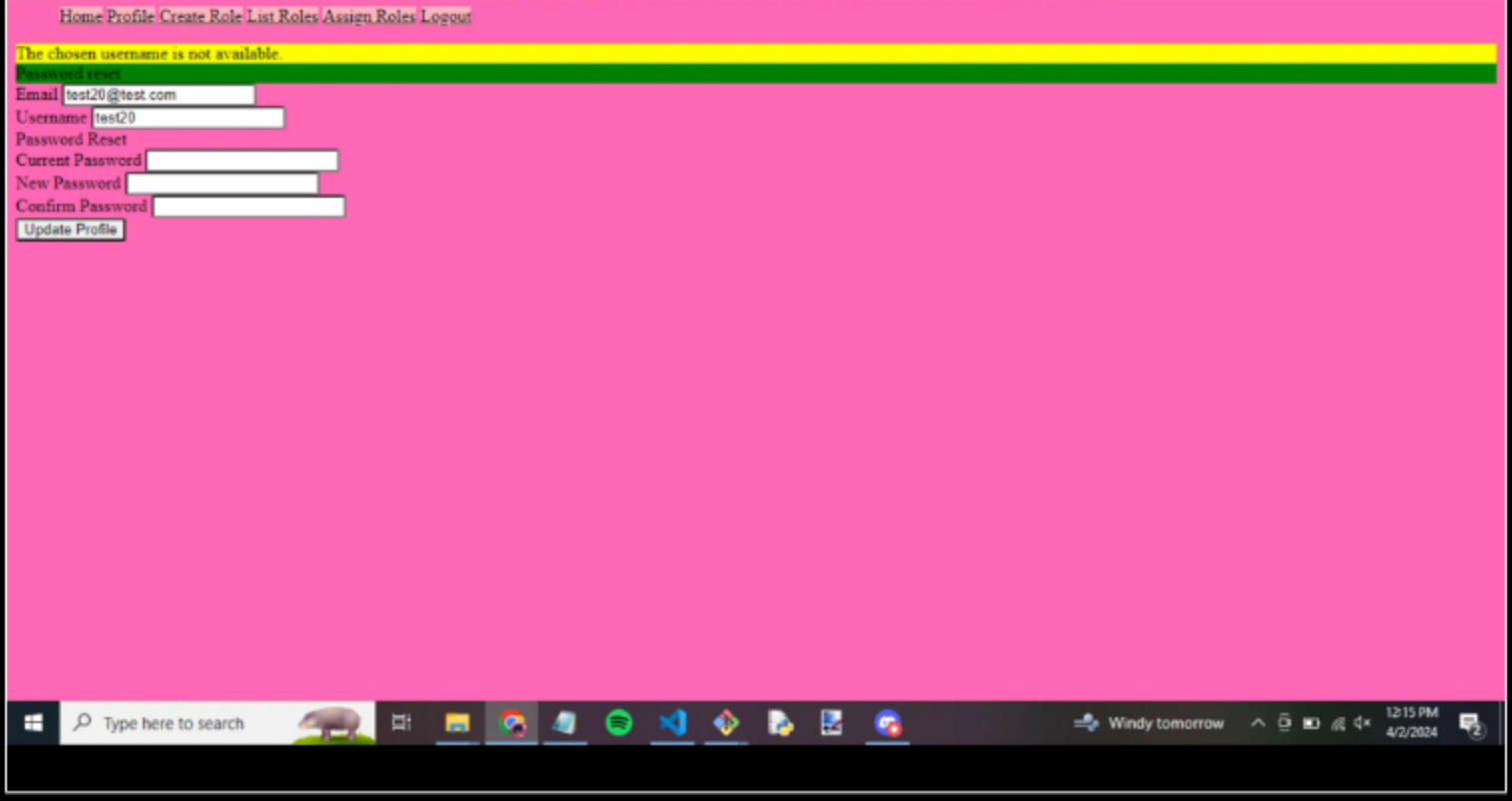
Checklist Items (0)



php validation chosen email not available

Checklist Items (0)





php username not available

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:

Updating Username/Email:
1. Form Submission: When the user submits the form by clicking the "Update Profile" button, the PHP code first checks if the form was submitted

2. Data Retrieval: It retrieves the entered email and username from the POST data submitted by the form.

3. Data Sanitization and Validation:

The email is sanitized and checked for validity using sanitize_email() and is_valid_email() functions respectively.
The username is checked for validity using is_valid_username() function.

3. Error Handling:

If the email or username is invalid, it sets a flash message indicating the error

If the email or username is invalid, it sets a flash message indicating the error.
If there are no errors, it proceeds to update the user's profile.

4. Database Update:

It prepares an SQL statement to update the user's email and username in the database.
It then executes the SQL statement with the provided parameters.

5. Session Update:

After successfully updating the profile in the database, it updates the session data with the new email and username.

Updating Password:
1. Form Submission: Similar to the previous scenario, the form submission is detected.

2. Data Retrieval:

It retrieves the current password, new password, and confirm password from the form.

3. Validation:

It checks if all password fields are filled.
If they are, it proceeds to further validation.
If not, it sets a flash message to prompt the user to fill all password fields.

4. Error Handling:

If the new passwords don't match, it sets a flash message indicating the mismatch.
If the new password is invalid, it sets a flash message.
If the current password is incorrect, it sets a warning flash message.

5. Database Update:

If all validation passes, it checks the current password's validity by querying the database.
If the current password is correct, it updates the password in the database with the new hashed password.
Flash Message Display: Flash messages are displayed to inform the user about the success or failure of the password update.

6. JavaScript Validation: The JavaScript function validate() provides client-side validation for email, username, and password fields, ensuring a smoother user experience.

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1

https://github.com/jennatnguyen/JN426-IT202-008/pull/33

Misc (1 pt.)

^COLLAPSE^

^COLLAPSE^

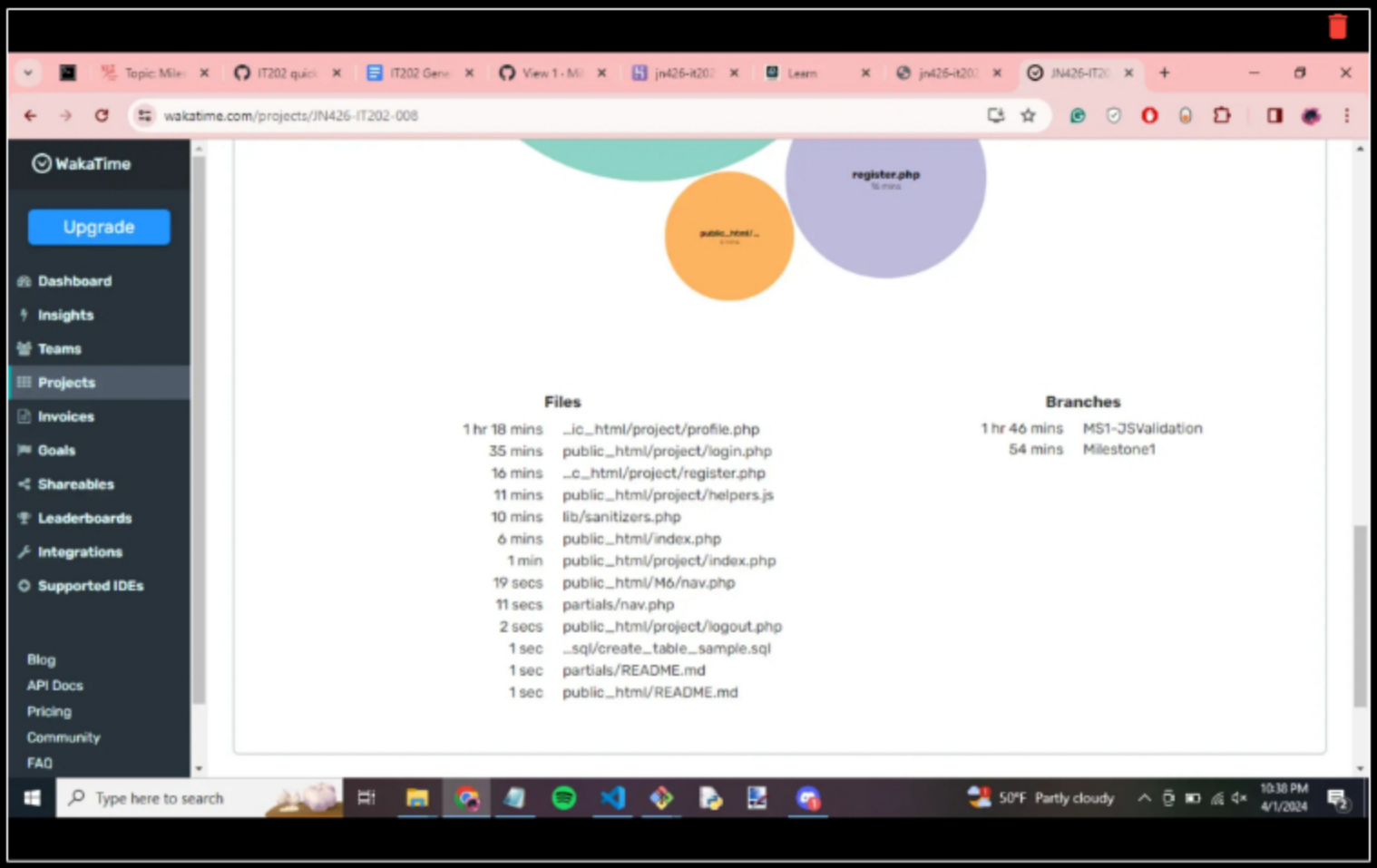
Task #1 - Points: 1

Text: Screenshot of wakatime

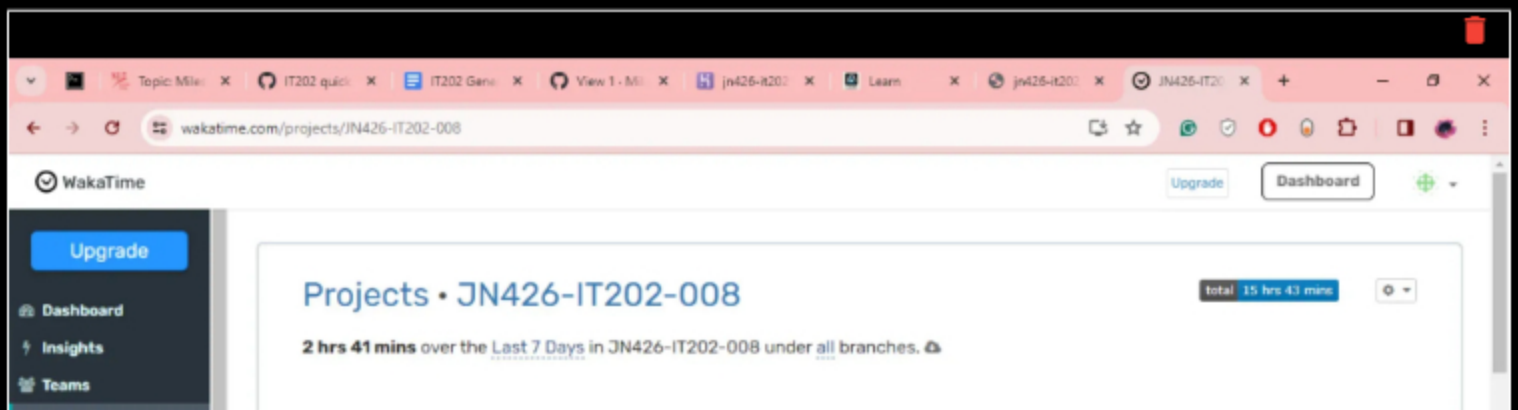
Task Screenshots:

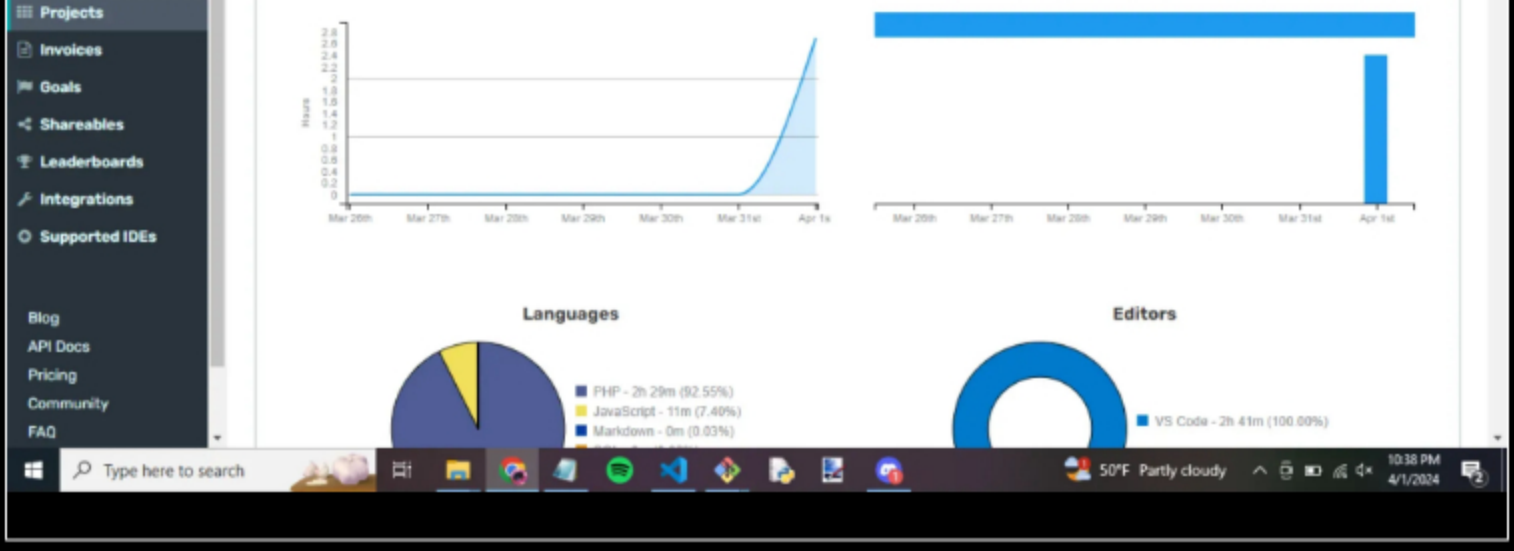
Gallery Style: Large View

Small Medium Large



wakatime from this week





wakatime showing total (15hrs)

COLLAPSE

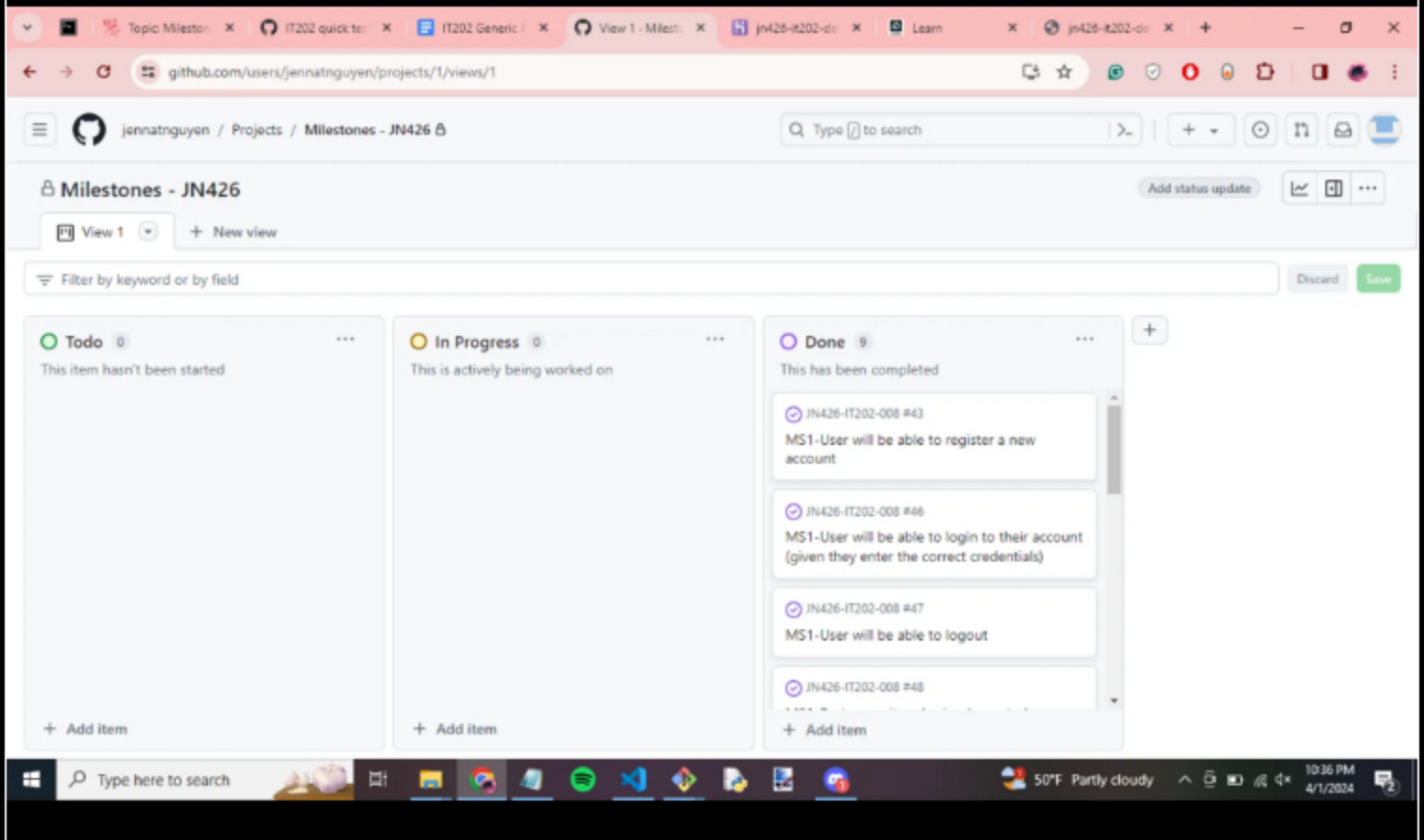
Task #2 - Points: 1

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



proj board

COLLAPSE

Task #3 - Points: 1

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

URL #1

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

COLLAPSE

Task #4 - Points: 1

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

URL #1

https://jn426-it202-prod-a94514875a20.herokuapp.com/project/login.php

End of Assignment