Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-450-M2024/it202-module-6-milestone-1-2024/grade/jed56

IT202-450-M2024 - [IT202] Module 6 Milestone 1 2024

Submissions:

Submission Selection

1 Submission [active] 7/9/2024 12:08:08 AM

Instructions

COLLAPSE

Overview Video: https://youtu.be/V7oHa8KKtss

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
- Create a Project board on GitHub (if you haven't yet)
    - Add each major item from the proposal doc as an Issue item
    - Invite the grader(s) and myself as collaborators on the board (they're separate from your repository)
        - [ ] See Canvas announcements for the Usernames or check your collab list on your repo
- 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, Login, and Profile (include "[Client]" in the output messages to differentiate between server-side validations)

Instructions:

1. Make sure you're in Milestone1 with the latest changes pulled
2. Ensure Milestone1 has been deployed to heroku dev
3. Gather the requested evidence and fill in the explanations per each prompt
4. Save the submission and generate the output PDF
5. Put the output PDF into your local repository folder

6. add/commit/push it to GitHub
7. Merge Milestone1 into dev
8. Locally checkout dev and pull the changes
9. Create and merge a pull request from dev to prod to deploy Milestone1 to prod
10. Upload this output PDF to Canvas

Branch name: Milestone1

Tasks: 25 Points: 10.00

User Registration (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of form on website page

Details:
Thoughtful CSS should be applied to all parts (must differ from the "ugly" CSS given via the lessons).
The Heroku dev URL must be present in all screenshots of the site.

#1) Screenshot of the form (ensure valid data is filled in)

Caption (required) ✓
Describe/highlight what's being shown
showing screenshot of the form for registration

URL (required) ✓
Prod link to the registration page
https://it202-jed56-

#2) Demonstrate JavaScript validation for each field [can be

Caption (required) ✓
Describe/highlight what's being shown
showing all error messages from javascript validation for email format, password format, username format, and passwords

#3) Demonstrate email already in use message (message text doesn't

Caption (required) ✓
Describe/highlight what's being shown
showing email already in use message

prod-553fb5e41cd0.herokuapp.com/

matching

#4) Demonstrate username already in use message (message text

Caption (required) ✓
Describe/highlight what's being shown
showing username already in use message

#5) Demonstrate user-friendly message of new account being created

Caption (required) ✓
Describe/highlight what's being shown
showing new account created message

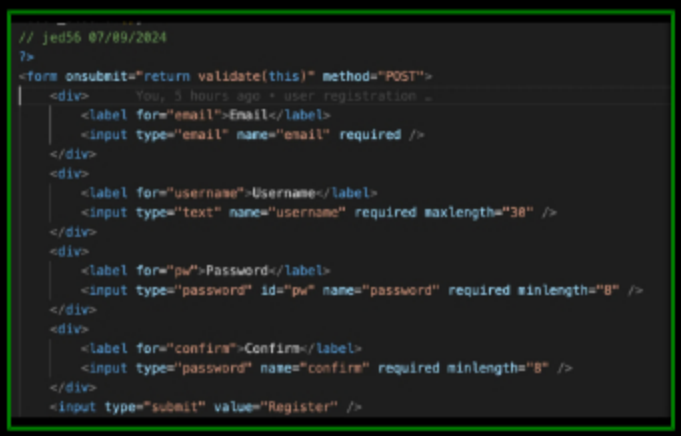
COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the form code

Details:
Should have the appropriate type attributes for the fields.
Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

#1) Show the html of the form



```
// jed56 07/09/2024
?>
<form onsubmit="return validate(this)" method="POST">
    <div>      You, 5 hours ago • user registration _
        <label for="email">Email</label>
        <input type="email" name="email" required />
    </div>
    <div>
        <label for="username">Username</label>
        <input type="text" name="username" required maxlength="30" />
    </div>
    <div>
        <label for="pw">Password</label>
        <input type="password" id="pw" name="password" required minlength="8" />
    </div>
    <div>
        <label for="confirm">Confirm</label>
        <input type="password" name="confirm" required minlength="8" />
    </div>
    <input type="submit" value="Register" />
```

Caption (required) ✓
Describe/highlight what's being shown
showing html form code

Explanation (required) ✓
Briefly explain the html for each field including the chosen attributes

PREVIEW RESPONSE

The email field collects the users email address and ensures it is in an email format, and makes it required. The username field gets the username, limits it to 30 characters and also makes it required. The password field gets the password and makes sure it is at least 8 characters and it is also required. Then the confirm password field gets the password again to confirm it, gives it a minimum length of 8 and makes it required as well. Finally the submit button submits the form data to our server.

COLLAPSE

Task #3 - Points: 1

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

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

#1) Show the JavaScript validations of the form (include any extra files related if you made separate files)

Caption (required) ✓
Describe/highlight what's being shown
showing javascript validations of the form

Explanation (required) ✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

The validate functions gets the values for email, username, password, and confirm password from the form and then it checks if the email is not empty and is

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

Caption (required) ✓
Describe/highlight what's being shown
showing php validations of the form

Explanation (required) ✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This PHP code takes the input values from the post request and goes through a function to sanitize the email. Then it validates the email, username, password,

in a valid email format. It does the exact same thing for the username and that it is also in appropriate format. It also makes sure the password field is not empty and has at least 8 characters. It also ensures that the password field and confirm password field are matching. If any of these things do not pass, then the flash functions displays an error on the screen and the function returns false to prevent the form from being submitted.

and confirm password using the helper functions. If any of these things do not pass, then the flash functions displays an error message, and the hasError flag is se to true. If all of these things pass however, then the password is hashed and the users information is set into our database. If there are any duplicates in the database it also handles that correctly by showing an error message.

COLLAPSE

Task #4 - Points: 1

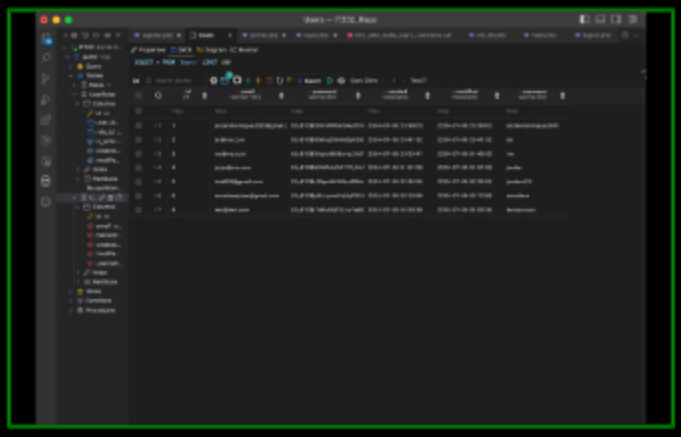
Text: Screenshot of the Users table with a valid user entry

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)

#1) Show valid data per the checklist



Caption (required) ✓

Describe/highlight what's being shown

showing valid data in SQL table

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 in concise steps.

May be worthwhile using a list.

Response:

1. The PHP requires all necessary files for external functions we use. When the form is submitted it takes all data using the "se" fucntion for safe extraction.
2. The email is sanitized using the :sanitize_email" function and each field is validated. Email is validated using the "is_valid_email" fucntion, username uses "is_valid_username", and password uses "is_valid_password". If any of these fail then the flash function is called to display and error message.
3. If there are no errors then the password is hashed using "password_hash" function. The database then gets connected by using getDB() function. The code then writes a SQL statement to inser the users information into the Users table. The SQL statement gets executed inside of a try-catch block. If it is inserted successfully then a success message is displayed. If a duplicate is found then the code checks which field is duplicated and displays an error message.
4. After processing the data from the form, the code will show any flash messages that were set during the entering fo the data and when the page reloads and registration was successful then the user can go to login page and login.

COLLAPSE

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/jordand2003/jed56-it202-450/pull/34

User Login (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of form on website page

Details:
Thoughtful CSS should be applied to all parts (must differ from the "ugly" CSS given via the lessons).
The Heroku dev URL must be present in all screenshots of the site.

#1) Two Screenshot of the form (one

Caption (required) ✓
Describe/highlight what's being shown
showing screenshots of the login page

URL (required) ✓
Prod link to the login page
https://it202-jed56-prod-553fb5e41cd0.heroku

#2) Demonstrate JavaScript

Caption (required) ✓
Describe/highlight what's being shown
showing javascript validation for each field

#3) Demonstrate user-friendly

Caption (required) ✓
Describe/highlight what's being shown
showing user friendly message of when an account doesn't exist

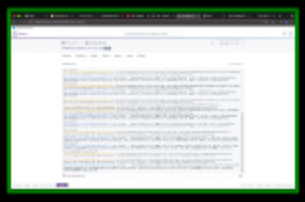
#4) Demonstrate user-friendly

Caption (required) ✓
Describe/highlight what's being shown
showing user friendly message of when password is incorrect

#5) Demonstrate successful

Caption (required) ✓
Describe/highlight what's being shown
showing successful login message

#6) Demonstrate session data



Caption (required) ✓
Describe/highlight what's being shown
showing server logs

COLLAPSE

Task #2 - Points: 1
Text: Screenshot of the form code

Details:
Should have the appropriate type attributes for the fields.
Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

#1) Show the

#2) Show the

#3) Show PHP

html of the form

Caption (required) ✓
Describe/highlight what's being shown
showing the html code for the form

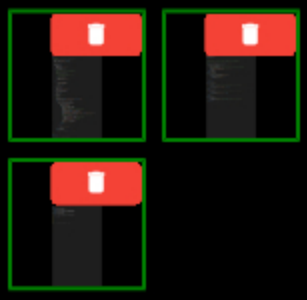
Explanation (required)
✓
Briefly explain the html for each field including the chosen attributes
PREVIEW RESPONSE
The email/username field takes in text to allow both emails and usernames as input. The name attribute is used to submit the form and server side processing. It is also a required field. The password field takes a type of password to makes sure the input is hidden, and the id is given as "pw" to give it a unique identifier. The name attribute is set to password for form submission and server processing and it is also a required field. The minlength attribute also makes it so that the password cannot be less than 8 characters.

JavaScript validations of

Caption (required) ✓
Describe/highlight what's being shown
showing the javascript validation code

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
When the form is submitted the validate function is called. It takes in the values. If the email/username field is empty it shows an error. The appropriate pattersonf or username and email are also defined. It then checks if the input matches the email or username patterns and if they do not thenn it displays an error. It then checks if password field is empty and if it is it displays a error message. If everythign passes then the function returns true.

validations (include any lib



Caption (required) ✓
Describe/highlight what's being shown
showing the PHP validation code

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The form data is taken from the POST request by using the se function. It then cheks if the username/email and password fields are not empty. Then it sanitizes the email/username and validates it using the is_valid functions for each field. If any field fails, then an error message is displayed using the flash function. If there are no errors, then the database is connected and a SQL query is created to get the user details from the users table. The code then goes through with the query and gets the users data and if a user is found then it verified the password with the hashed password stored in the database by using

the password_verify function. If the password is correct then the user is logged in. If login does not work for any reason then the appropriate error message is shown.

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

Details:
Don't just show code, translate things to plain English in concise steps.
Explain how the session works and why/how it's used

May be worthwhile using a list.

Response:

1. The PHP code includes all necessarry files with external functions. It then checks if the form is submitted by looking for the POST of email and password. The data from the form is then retirved from the POST request using the se function.
2. The code checks if the email/username is empty and displays an error message if it is. If not, then it sanitizes the email/username using the santize function for each.
3. The code then checks if the password is empty and if it is it shows an error message. If not then the code goes on to do the database query by establishing the connection and executing the query. THe user data gets fetched from that query and if a user is found then the code verifies the password with the hashed password and if they match the user is logged in and redirected to the home page.
4. If no user is found or the password is wrong then the correct error message is shown.

COLLAPSE

Task #4 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/jordand2003/jed56-it202-450/pull/35

User Logout (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Capture the following screenshots

#1) Screenshot of the navigation

Caption (required) ✓

Describe/highlight what's being shown

showing screenshot of navigation when logged in

#2) Screenshot of the redirect to login with

Caption (required) ✓

Describe/highlight what's being shown

showing screenshot of user logged out

#3) Screenshot of the logout-related code

Caption (required) ✓

Describe/highlight what's being shown

showing screenshot of log out related code showing the session is destroyed

Explanation (required)

✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The session is initialized, then it includes the functions.php file which has all necessary functions, the session is then reset using the reset_session function to clear all the sessions and log the user out. It then displays a flash message saying they were successfully logged out. Lastly they are redirected to the login page.

COLLAPSE

Task #2 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/jordand2003/jed56-it202-450/pull/36

COLLAPSE

Basic Security Rules and Roles (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Authentication Screenshots

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

#1) Screenshot of the function that checks if a

Caption (required) ✓
Describe/highlight what's being shown
showing screenshot of function that checks if user is logged in

Explanation (required) ✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This function takes the

#2) Screenshot of the login check function

Caption (required) ✓
Describe/highlight what's being shown
screenshot of is-Logged_in function being used

Explanation (required) ✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

The functions is called

#3) Demonstrate the user-

Caption (required) ✓
Describe/highlight what's being shown
showing screenshot of trying to access login protected page while logged out

parameters $redirect and $destination and checks if the user session variable is set or not to check if the user is logged in. If redirect is true then the user is not logged in and the error message is shown. It then redirects the user to the specified destination and terminates the code. Finally the function returns the login status.

with redirect paramter set to true. If the user is not logged in then they are redirected to the login page, and if they are logged in then their data is logged.

COLLAPSE

Task #2 - Points: 1

Text: Authorization Screenshots

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

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

Caption (required) ✓
Describe/highlight what's being shown
showing screenshot of function that checks for a specific role

Explanation (required) ✓

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

Caption (required) ✓
Describe/highlight what's being shown
showing screenshot of the role being checked

Explanation (required) ✓
Explain in concise steps how this logically works

Explain in concise steps how this logically works

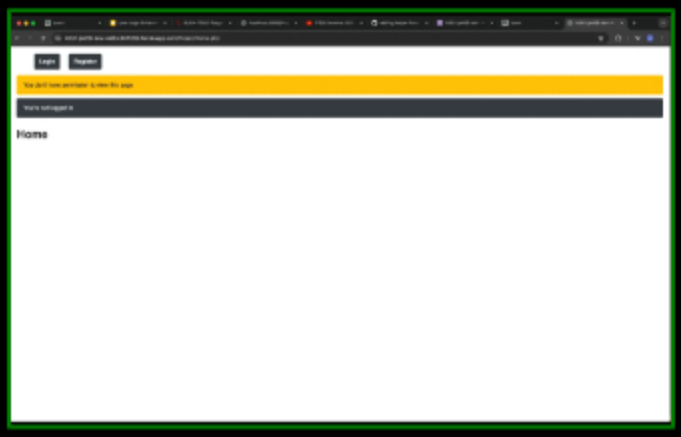
PREVIEW RESPONSE

The has role function takes one paramter of $role. It checks if the user is logged in using the is_logged_in function and verifies if the roles array is set in the user data. It then iterates over every role in the array and checks if the current role matches the given role and if so it return true. If no match is foudn then it retruns false.

PREVIEW RESPONSE

The function is being used here by checking if a user has a role of admin. However it is being negated so we are checking if the user does not have. role of admin. If they do not have this role then the message is displayed that they dont have access and they are brought to the home page.

#3) Demonstrate the user-friendly message of trying to manually access a role-protected page while being logged out (must show the



Caption (required) ✓

Describe/highlight what's being shown

showing user friendly message of trying to access role protected page while being logged out

COLLAPSE

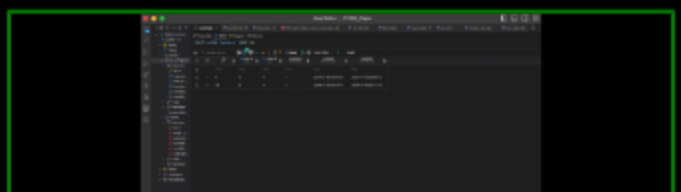
Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

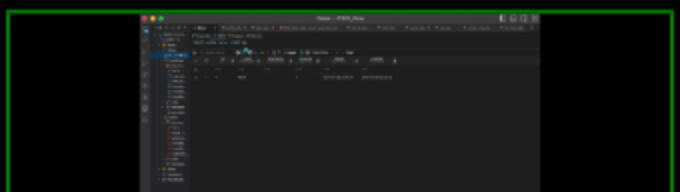
Details:

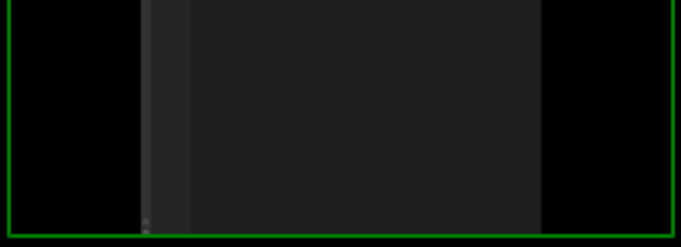
Ensure left panel or database name is present in each table screenshot (should contain your ucid)

#1) UserRoles table with at least one valid entry (Table should have id, user_id, role_id, is_active, created, and modified columns)

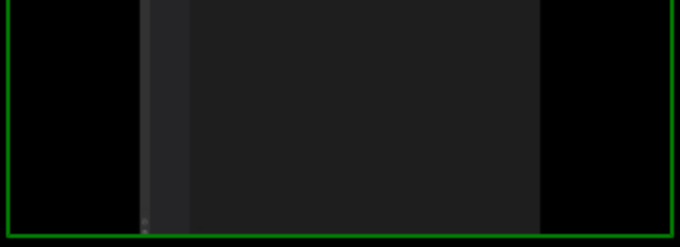


#2) Roles table with at least one valid entry (Table should have id, name, description, is_active, modified, and created columns)





Caption (required) ✓
Describe/highlight what's being shown
showing userroles table with one valid entry



Caption (required) ✓
Describe/highlight what's being shown
showing roles table with one valid entry

COLLAPSE

Task #4 - Points: 1

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

#1) UserRoles

Explanation (required) ✓
What's the purpose of the UserRoles table?
PREVIEW RESPONSE
The UserRoles table is sort of a join table between Users and Roles. It stores the relationships between a user and whatever role they have and links user id's from the user table and the role_id from the roles table. the is_active column in userroles determines whether or not a users role is currently active.

#2) Roles

Explanation (required) ✓
How does Roles.is_active differ from UserRoles.is_active?
PREVIEW RESPONSE
Roles.is_active and UserRoles.is_active are different because Roles.is_active tells us whether or not a role is actually active in the system to be assigned or not, while UserRoles.is_active tells us whether or not a particular user has that role active rather than if the role itself is active for use in general.

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/jordand2003/jed56-it202-450/pull/37
URL #2
https://github.com/jordand2003/jed56-it202-450/pull/39

User Profile (2 pts.)

COLLAPSE

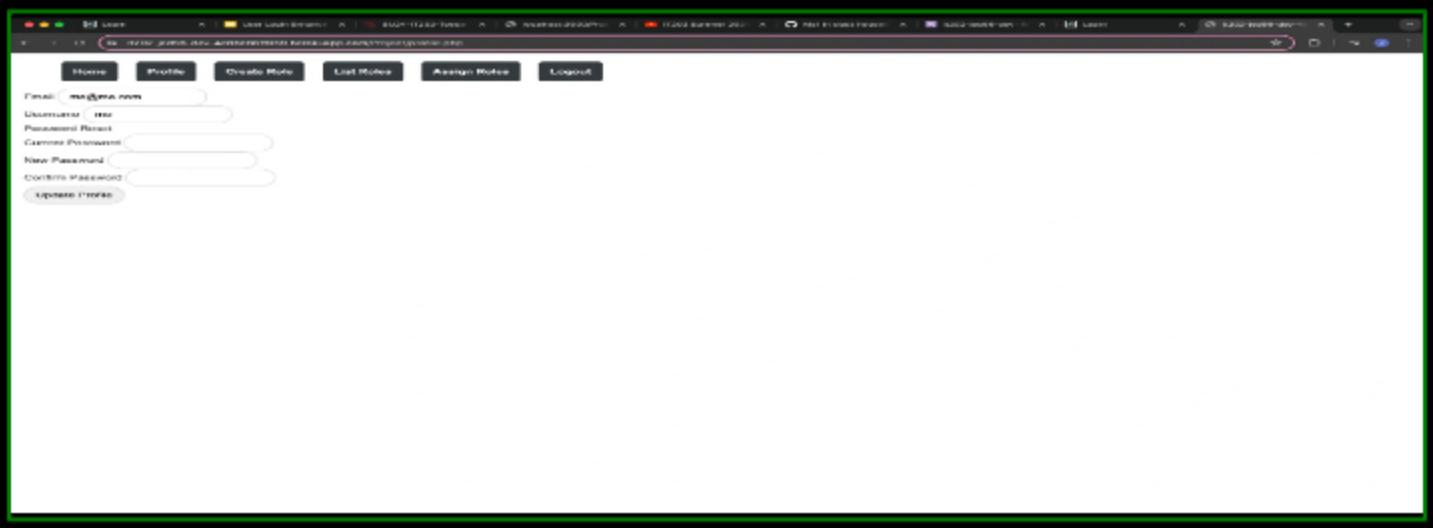
COLLAPSE

Task #1 - Points: 1

Text: View Profile Website Page

Details:
Thoughtful CSS should be applied to all parts (must differ from the "ugly" CSS given via the lessons).
The Heroku dev URL must be present in all screenshots of the site.

#1) Show the profile form correctly populated on page load (username, email) Form should have the following fields: username, email, current password, new password, confirm password (or similar)



Caption (required) ✓
Describe/highlight what's being shown
showing profile form correctly populated on page load

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 in concise steps.

May be worthwhile using a list.

Response:

1. The appropriate functions and nav files are included at the top. Th code then checks if the user is logged in and if they are not they are redirected to the login page.
2. The users data is then retrieved using the get user email and get username functions. The users data is then displayed as a pre filled out form with the users username and email and also has some other fields for updating the username, and resetting the password. The client side javascript function validate then checks that the email and username are not empty and meet the requirements. If the user wants to reset their password then their old password must match whats int he database and the new password and cofirmed new password have to match.
3. The code then checks if the form has been submitted and retrieves all the updated infromation and updates it in the database. It displayes any appropriate error message if there is any of a success message if everythign was updated successfully.

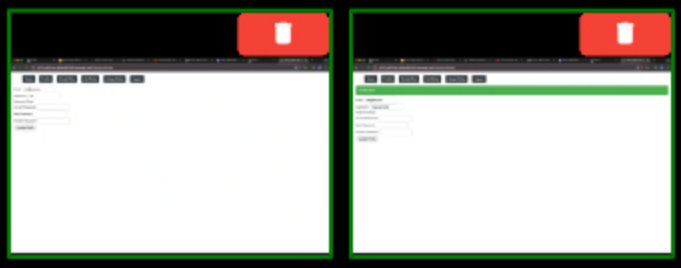
COLLAPSE

Task #3 - Points: 1

Text: Edit Profile Website Page

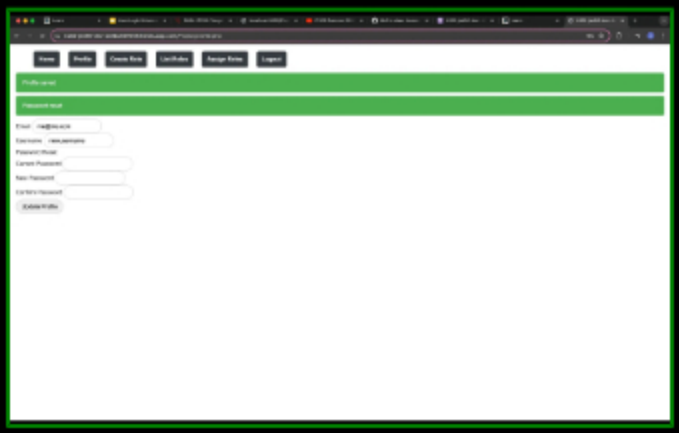
Details:
Thoughtful CSS should be applied to all parts (must differ from the "ugly" CSS given via the lessons).
The Heroku dev URL must be present in all screenshots of the site.

#1) (Two Screenshots) Demonstrate with before and after of a username change (including success message)



Caption (required) ✓
Describe/highlight what's being shown
showing screenshots of before and after username change

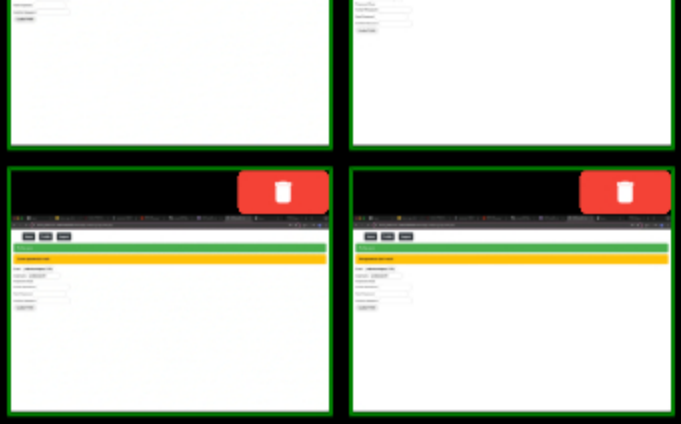
#2) Demonstrate the success message of updating password



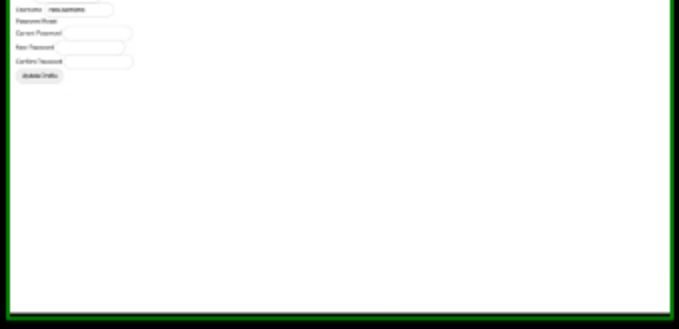
Caption (required) ✓
Describe/highlight what's being shown
showing success message after updating password

#3) Demonstrate all JavaScript user-friendly validation messages [can be combined] (email format, username format, password

#4) Demonstrate PHP user-friendly validation message (desired email is already in use)



Caption (required) ✓
Describe/highlight what's being shown
showing javascript user validation messages



Caption (required) ✓
Describe/highlight what's being shown
showing PHP user validation message for email in use

#5) Demonstrate PHP user-friendly validation message (Desired username is already in use)

Caption (required) ✓
Describe/highlight what's being shown
showing PHP user validation message for username in use

#6) Demonstrate PHP user-friendly validation message (Current password doesn't match what's in the DB)

Caption (required) ✓
Describe/highlight what's being shown
showing PHP user validation message for password incorrect

COLLAPSE

Task #4 - Points: 1

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

ℹ Details:
Don't just show code, translate things to plain English

#1) Updating Username/Email

#2) Updating password

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The code checks whether or not the form wa ssubmitted by checking if the POST request is set. The updated username or email are then taken from the form using the se function and an array of paramters is created including the new email, username, and user id. The database then gets connected and a SQL update query is created wherever the user id's match. The query then gets executed. It handles any duplicate entries by checking if the new email and password are already in use. Finally a new SQL select query is run and gets the updated user information.

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

The code checks if the POST request has been set. It then gets the current password, new password, and confirmed new password from the form through the se function. It then checks if all of the password field are fileld and verifies that the new password and confrim new password are matching. Then it has to get the current password from the datavase through a select query and verify that it is accurate. If the password is correct then the new password is hashed and a SQL update query is run to change the password in the database.

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

https://github.com/jordand2003/jed56-it202-450/pull/38

Misc (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of wakatime

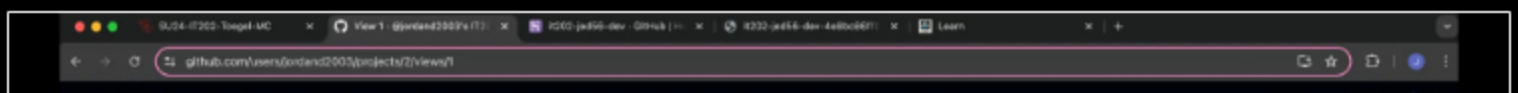
Details:

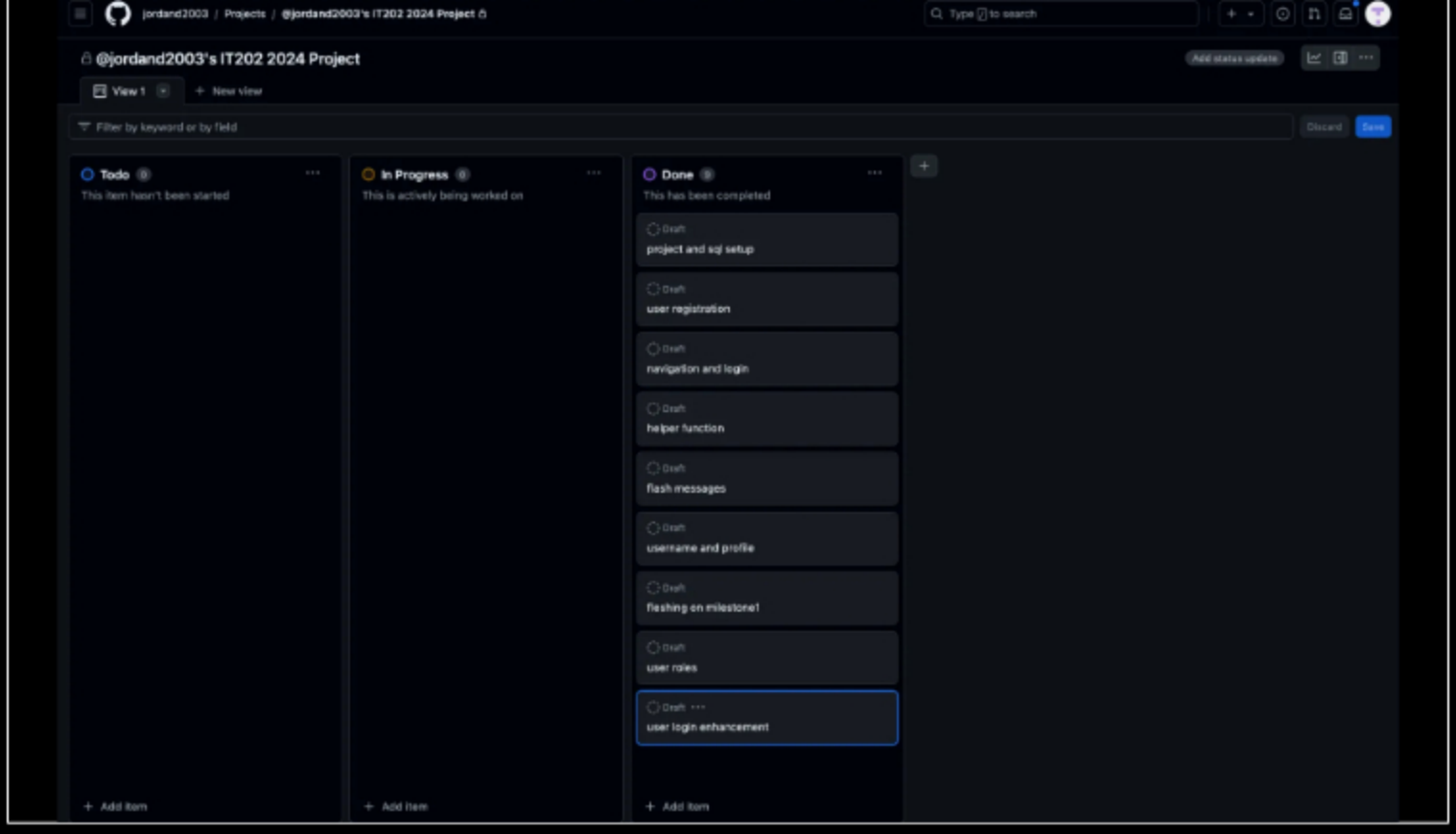
Note: The duration of time isn't directly related to the grade, the goal is to just make sure time is being tracked

Task Screenshots:

Gallery Style: Large View

Small Medium Large





screenshot of WakaTime

COLLAPSE

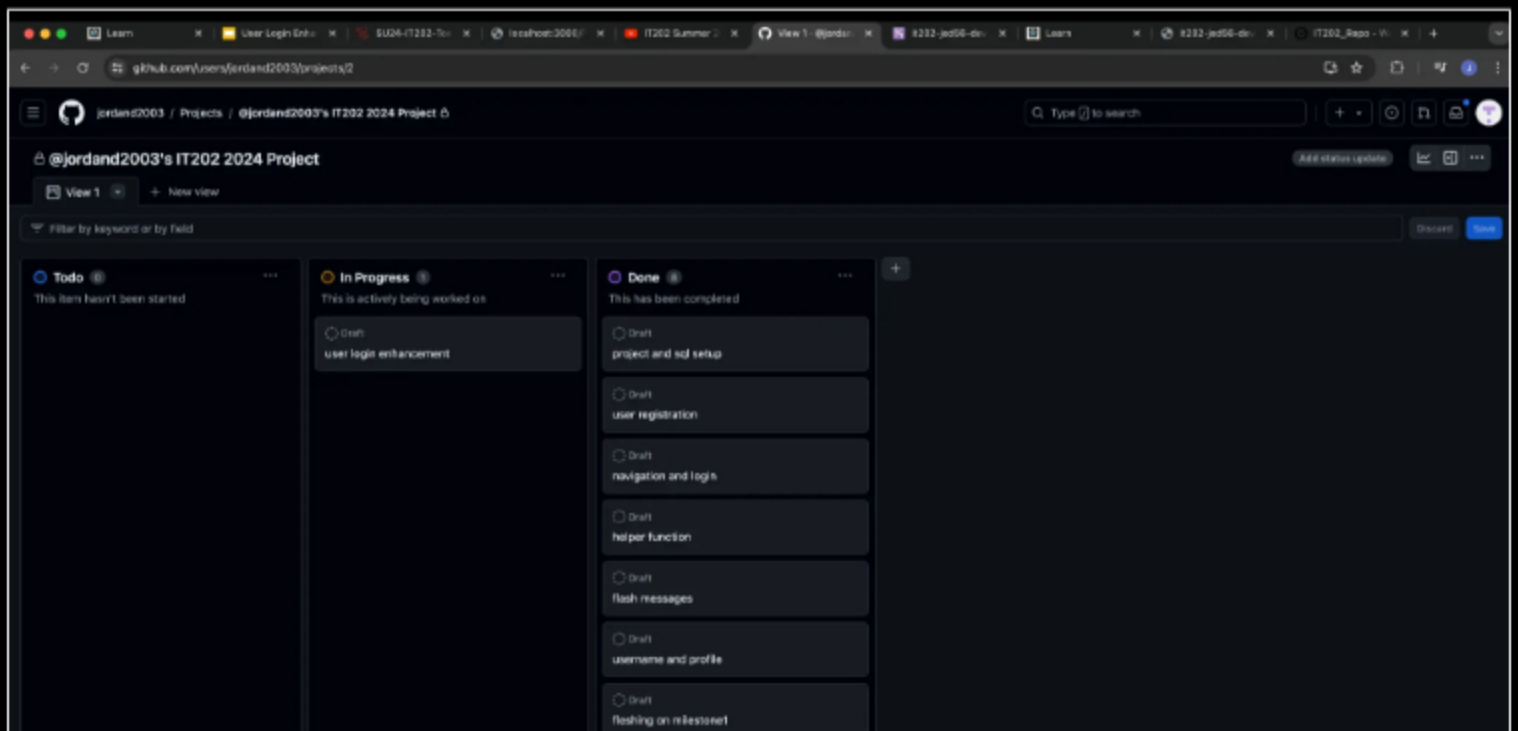
Task #2 - Points: 1

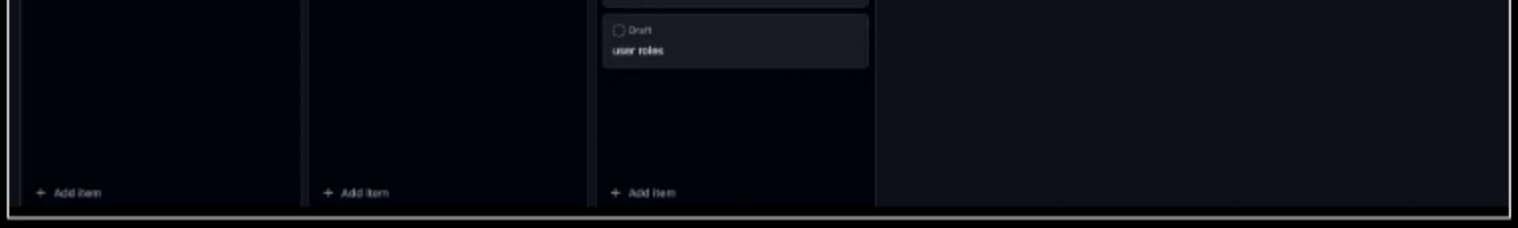
Text: Screenshot of your project board from GitHub (tasks should be in the proper column)

Task Screenshots:

Gallery Style: Large View

Small Medium Large





screenshot of project board from github

COLLAPSE

Task #3 - Points: 1

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

URL #1

https://github.com/users/jordand2003/projects/2/views/1

End of Assignment