Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-450-M2024/it202-module-6-milestone-1-2024/grade/mm2849

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

Submissions:

Submission Selection

1 Submission [active] 7/10/2024 1:16:29 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

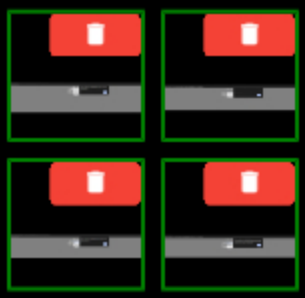
Details:
Thoughtful CSS should be applied to all parts (must differ from the "ugly" CSS given via the lessons).
The Heroku dev URL must be present in all screenshots of the site.

#1) Screenshot of the form

Caption (required) ✓
Describe/highlight what's being shown
Showing Screenshot of Form with data filled in.

URL (required) ✓
Prod link to the registration page
https://it202-mm2849-

#2) Demonstrate JavaScript



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

#3) Demonstrate email

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating email already in use and not available.

#4) Demonstrate username

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating username already in use and not available.

prod-3453894141df.herokuapp.com/project/register.php

#5) Demonstrate user-friendly

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating user-friendly message of a new account being created.

^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

Caption (required) ✓
Describe/highlight what's being shown
Showing html of the form.

Explanation (required)

✓

Briefly explain the html for each field including the chosen attributes

PREVIEW RESPONSE

The code provided in the screenshot above and onto user registration serves several purposes. It has a user registration form that requires an email address, a username, a password, and a confirmation password. It also adds javascript validation for TODO1. The user registration form is then processed by validating and sanitizing the submitted data. When the user is prompted on the website to provide an email address, a username, a password, and a password confirmation, and when all the information meets the length and requirements, then the information is added into the database. The maximum number of characters allowed in the username is set to 30, while the minimum length of the password is set to 8. When the user is successfully registered, it will throw a success message, if there is an error, it will throw an error exception.

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

Caption (required) ✓
Describe/highlight what's being shown
Showing javascript validation of the form.

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

PREVIEW RESPONSE

The code displayed in the screenshot above and on TODO1 in User registration performs a variety of functions. It examines the value of the email, username, password, and confirm password and throws an error if any of the specified fields are not filled out, if they are, it returns true and proceeds to the next page.

#2) Show the PHP validations

Caption (required) ✓
Describe/highlight what's being shown
Showing PHP Validations.

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

PREVIEW RESPONSE

The code displayed in the screenshot above and on the PHP section in User registration examines the value of all the given field in PHP format and then shows the messge after the field is filled out and submitted.

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

Caption (required) ✓
Describe/highlight what's being shown
Showing valid data per the checklist.

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. Prompts the user to submit the form.
2. User has to type in the valid email, username, password and confirm password.
3. Email and username cannot match with any other of the same email in the data base.
4. Then the data is retreived from the form to the database including php.
5. If the form is unable to submit it will throw a error, since we have intialized the error in the code. (`$hasError = false`;)
6. Then the form checks for any errors incase one or none of the fields are filled out, if not filled out then that would throw an error.

7. If no errors are found then the password gets hashed so that the orignal password dosent go in the database, an hashpassword is generate and stored in database.
8. Then the database connection is established and with the sql statement all the values get inserted to the correct table in the Database.
9. After the user is in the database, it can be given a specific role by the host, or another admin can also assign the roles.

COLLAPSE

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/mm2849/mm2849-IT202-450/pull/14

User Login (2 pts.)

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

Caption (required) ✓
Describe/highlight what's being shown
Attaching Two Screenshot of the form (one with valid email data filled and one with

#2) Demonstrate JavaScript

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating JavaScript Validation.

#3) Demonstrate user-friendly

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

#4) Demonstrate user-friendly

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating a user-friendly message of when an password

valid username data filled).

URL (required) ✓
Prod link to the login page
https://it202-mm2849-prod-3453894141df.herokuapp.com/Project/login.php

which an account doesn't exist.

which an password doesn't match with what's on the database.

#5) Demonstrate successful

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating successful login message and the destination page which is the home page.

#6) Demonstrate session data

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating session data being set. Screenshot captured from Heroku 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 html of the form

#2) Show the JavaScript validations

#3) Show PHP validations

Caption (required) ✓
Describe/highlight what's being shown
Showing user-login html form.

Explanation (required)
✓
Briefly explain the html for each field including the chosen attributes
PREVIEW RESPONSE
First, the form prompts the user to log in using their email address or username and password. If the username or email form is filled out but the password is wrong, it will return an error message stating "Invalid password." If the email address is incorrect, the error message will read "email not found". If the user is logged in, they will land on the main page. The form will then check for the user's roles, if any.

Caption (required) ✓
Describe/highlight what's being shown
Showing the Java-Script Validation.

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The code displayed in the screenshot above and on TODO1 in login performs a variety of functions. It examines the value of the email/username and password and throws an error if any of the specified fields are not filled out, if they are, it returns true and proceeds to the next page.

Caption (required) ✓
Describe/highlight what's being shown
Adding PHP Validations.

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
When a user logs in, php validations are displayed, along with their email address and the ability to change their password. Sanitzers.php Checks if all fields are in valid format and sanitized the email. Checks to see if the username matches the pattern and if the password is the minimal length.

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. Prompts user to login
2. Gets connection to the database
3. While using the stmt–> executle statement, all the user data will be fetched from the database
4. Verifies the Username, email and password with whats on the database.
5. Stores user data from database to the session.
6. Next, the User Roles are fetched from the database to the session.
7. If everything is successfull then a welcome 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/mm2849/mm2849-IT202-450/pull/15

User Logout (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Capture the following screenshots

#1) Screenshot of the

Caption (required) ✓
Describe/highlight what's being shown
Attaching Screenshot of the navigation when logged in.

#2) Screenshot of the

Caption (required) ✓
Describe/highlight what's being shown
Attaching Screenshot of the redirect to login with the user-friendly logged-out message.

#3) Screenshot of the

Caption (required) ✓
Describe/highlight what's being shown
Attaching Screenshot of the logout-related code showing the session is destroyed.

Explanation (required)

Explain in concise steps how this logically works

PREVIEW RESPONSE

The first thing the logout page does is initiate the session, so the user is not prompted to log out before logging in. The "session_start();" command is what initiates the session. The ("session_unset();") will clear all variables. Finally, the "session_destroy();" function will be used to delete the session and clear it up on the server. Once logged out, the user will be 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/mm2849/mm2849-IT202-450/pull/15

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

Caption (required) ✓
Describe/highlight what's being shown
Screenshot of the function that checks if a user is logged in.

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This function makes sure that a user is logged in to the session. When the user logs in they will get redirected to the profile page. If the user is not logged in then a error will be thrown telling the user to login.

#2) Screenshot of the login

Caption (required) ✓
Describe/highlight what's being shown
Screenshot of the login check function being used.

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This function checks if a user is logged in to the database and it validated the username. If the user is not validated then an error is thrown and if its logged in then the is_logged_in is set to true.

#3) Demonstrate the user-

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating the user-friendly message of trying to manually access a login-protected page while being logged out.

COLLAPSE

Task #2 - Points: 1
Text: Authorization Screenshots

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

#1) Screenshot of the

#2) Screenshot of the role

#3) Demonstrate the user-

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

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

PREVIEW RESPONSE

The code above shows that if a user is logged in but does not have a specified role, a message will be thrown stating "user does not have permission to view this page." If the user has a role, the system will check the database and provide the user the permissions specified in the role.

Caption (required) ✓
Describe/highlight what's being shown
Screenshot of the role check function being used.

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

PREVIEW RESPONSE

The above code checks the roles. When an administrator tries to create a role with the same name, the database checks to determine if the role already exists. If the same role has already been created in the database, an error notice will appear stating that the role already exists.

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating the user-friendly message of trying to manually access a role-protected page while being logged out.

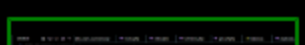
^COLLAPSE ^

Task #3 - Points: 1

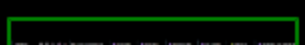
Text: Screenshots of UserRoles and Roles Tables

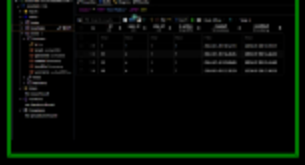
Details:
Ensure left panel or database name is present in each table screenshot (should contain your ucid)

#1) UserRoles table with at



#2) Roles table with at least one





Caption (required) ✓
Describe/highlight what's being shown
Showing UserRoles table with at least one valid entry.

Caption (required) ✓
Describe/highlight what's being shown
Showing Roles table with at least 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 will function as an association table for many-to-many relationships. This basically means that multiple users can have different types of roles. It is also not required that all users have a role. A role will define additional permissions and functions.

#2) Roles

Explanation (required) ✓
How does Roles.is_active differ from UserRoles.is_active?
PREVIEW RESPONSE
This table will record the name and description of a role, as well as its unique identifier. For example, I created many roles, including administrator, publisher, and reviewer. All of these roles will be saved in roles, and when a user with administrative privileges offers a role to a non-admin user, the roll will then be assigned to that specific 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/mm2849/mm2849-IT202-450/pull/20

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

Caption (required) ✓
Describe/highlight what's being shown
Showing the 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. Prompts user to login
2. Gets connection to the database
3. While using the stmt–> executle statement, all the user data will be fetched from the database.
4. Checks for the username, email and password.
5. Allows the user to select data from the table.
6. Allows users to update email, username and password.
7. When password is changed it will be submitted as hashed in the database.
8. If everything is successfull then a Updated Profile message is shown.

^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

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating with before and after of a username change.

#2) Demonstrate the success

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating the success message of updating password.

#3) Demonstrate all

Caption (required) ✓
Describe/highlight what's being shown
Showing java script validation.

#4) Demonstrate PHP user-

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating PHP user-friendly validation message.

#5) Demonstrate PHP user-

#6) Demonstrate PHP user-

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating PHP user-friendly validation message Desired username is not available.

Caption (required) ✓
Describe/highlight what's being shown
Demonstrating PHP user-friendly validation message Current password doesn't match what's in the DB.

COLLAPSE

Task #4 - Points: 1

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

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

#1) Updating Username/Email

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
First, it will prompt the user for their current username or email. If the valid username or email is supplied, the user will be allowed to set a new username or email and confirm it. If the email is incorrect, there will be an error. If the email and username is correct then it will be successfully updated with a success message.

#2) Updating password

Explanation (required) ✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
First, it will prompt the user for their current password. If the right password is supplied, the user will be allowed to set a new password and confirm it. If the password is incorrect, there will be an error. Then it will check to see if the passwords match. If the passwords match, the password will be successfully updated, along with a success message.

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

https://github.com/mm2849/mm2849-IT202-450/pull/21

Misc (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of wakatime

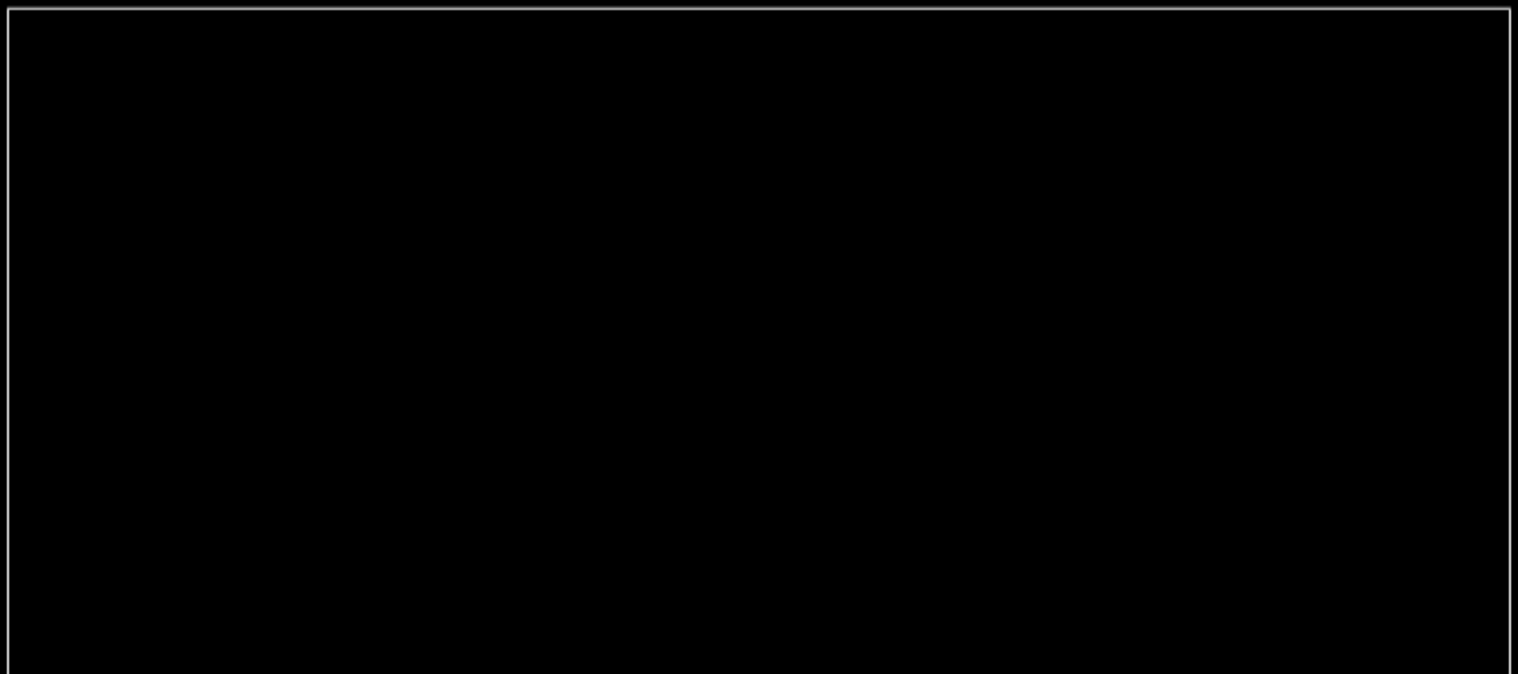
Details:

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



Adding screenshot of Wakatime

COLLAPSE

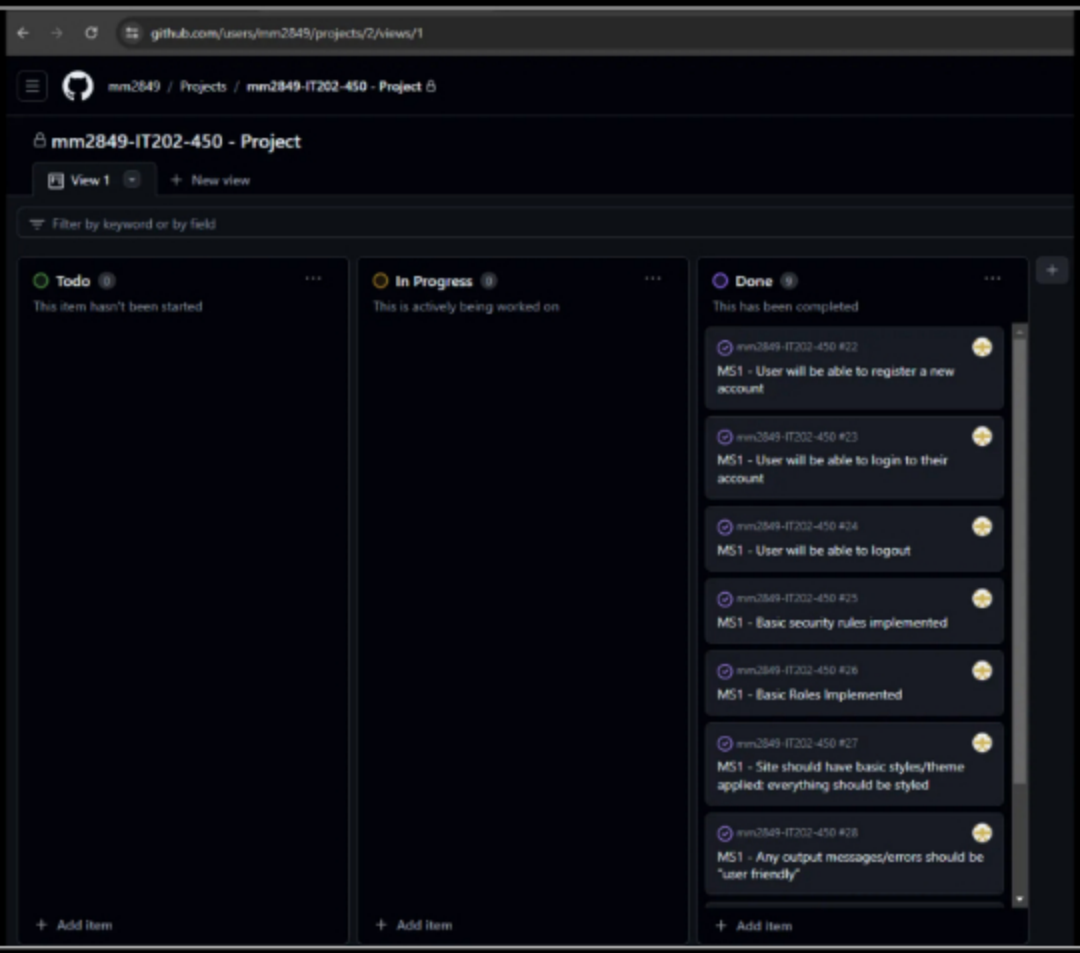
Task #2 - Points: 1

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



Adding 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/mm2849/projects/2/views/1

End of Assignment