

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/ak2774>

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

Submissions:

Submission Selection

1 Submission [active] 4/1/2024 5:44:11 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:

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: 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
<input type="checkbox"/> #1	1	Heroku dev url should be present in the address bar
<input type="checkbox"/> #2	1	Should have thoughtful CSS applied
<input type="checkbox"/> #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)
<input type="checkbox"/> #4	1	Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)
<input type="checkbox"/> #5	1	Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)
<input type="checkbox"/> #6	1	Demonstrate user-friendly message of new account being created

Task Screenshots:

Gallery Style: Large View

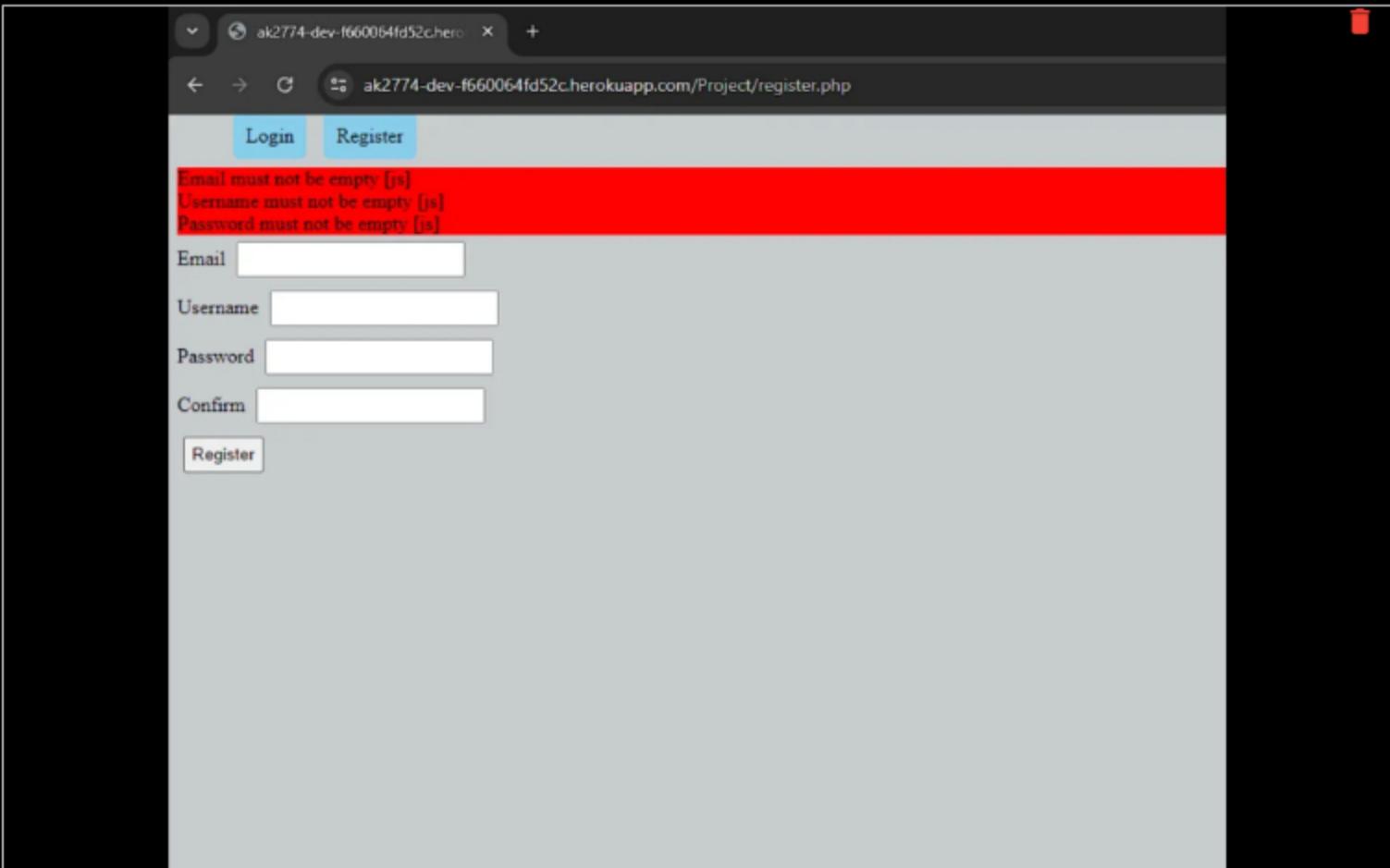
[Small](#)[Medium](#)[Large](#)

The screenshot shows a web browser window with the following details:

- Address Bar:** ak2774-dev-f660064fd52c.herokuapp.com/Project/register.php
- Form Fields:**
 - Email: [Input Field]
 - Username: [Input Field]
 - Password: [Input Field]
 - Confirm: [Input Field]
- Buttons:**
 - Login Tab (inactive)
 - Register Tab (active)
 - Register Button

Main page when clicked on register

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

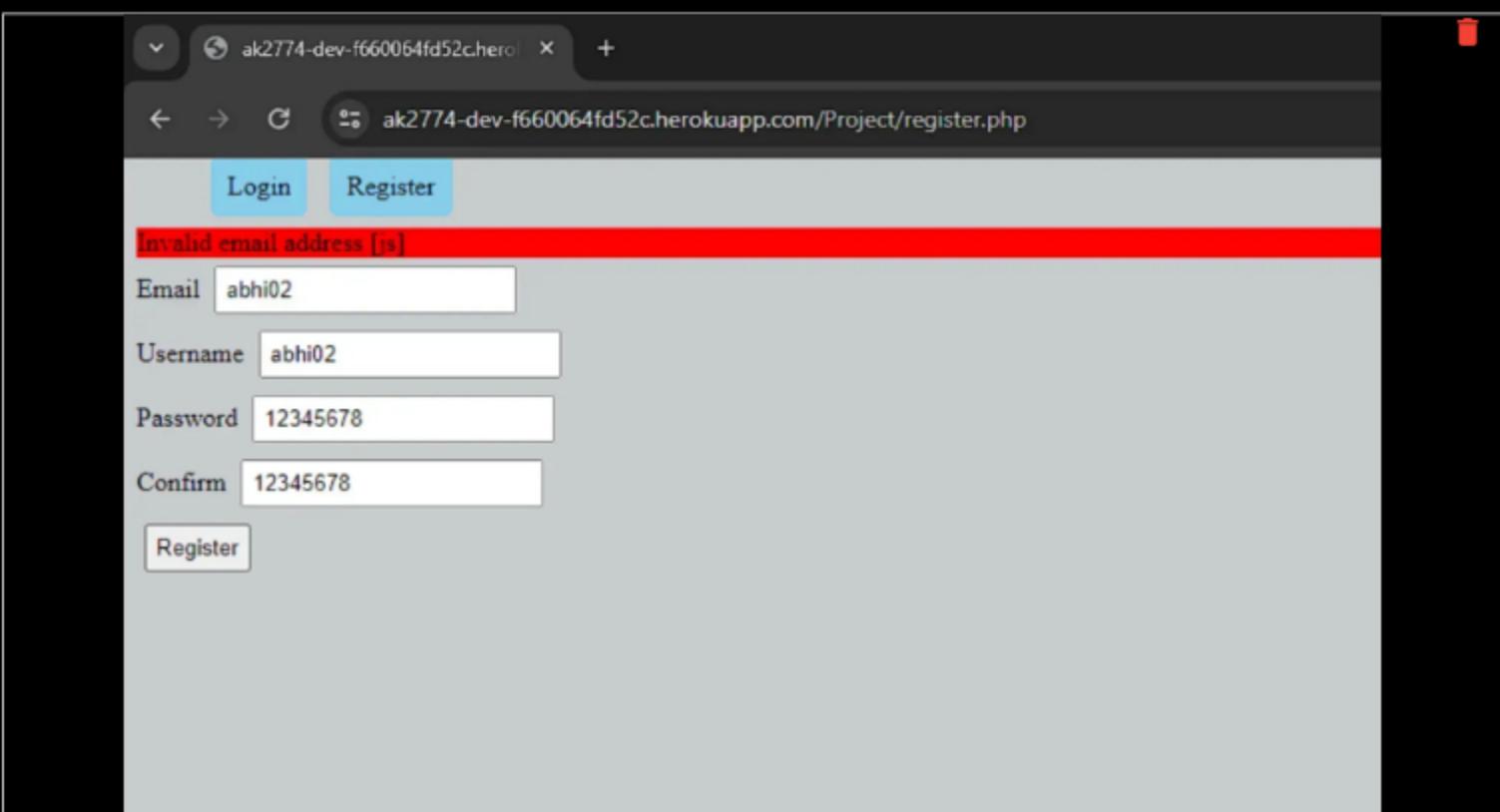


The screenshot shows a web browser window with the URL `ak2774-dev-f660064fd52c.herokuapp.com/Project/register.php`. The page displays a registration form with four input fields: Email, Username, Password, and Confirm. Above the input fields, there are three red horizontal bars containing validation error messages: "Email must not be empty [js]", "Username must not be empty [js]", and "Password must not be empty [js]". Below the input fields is a "Register" button.

JS validation showing each field is required

Checklist Items (1)

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



The screenshot shows a web browser window with the URL `ak2774-dev-f660064fd52c.herokuapp.com/Project/register.php`. The page displays a registration form with four input fields: Email, Username, Password, and Confirm. Above the Email input field, there is a red horizontal bar containing the validation error message "Invalid email address [js]". The Email input field contains the value "abhi02". The other input fields (Username, Password, Confirm) are empty. Below the input fields is a "Register" button.

JS validation showing email format

Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL: ak2774-dev-f660064fd52c.herokuapp.com/Project/register.php. The page contains a registration form with four input fields: Email, Username, Password, and Confirm. Below the form is a 'Register' button. A red horizontal bar at the top of the page displays the validation message: "Username must only contain 3-16 characters a-z, 0-9, _, or - [js]". The 'Email' field contains the value "abhi02@test.com". The 'Username' field contains the value "ak". The 'Password' and 'Confirm' fields both contain the value "12345678".

JS validation showing username format

Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL: ak2774-dev-f660064fd52c.herokuapp.com/Project/register.php. The page contains a registration form with four input fields: Email, Username, Password, and Confirm. Below the form is a 'Register' button. A red horizontal bar at the top of the page displays the validation message: "Password must be at least 8 characters long [js]". The 'Email' field contains the value "abhi02@test.com". The 'Username' field contains the value "abhi02". The 'Password' field contains the value "12345". The 'Confirm' field contains the value "12345".

JS validation showing password format

Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL: ak2774-dev-f660064fd52c.herokuapp.com/Project/register.php. The page contains a registration form with four input fields: Email, Username, Password, and Confirm. Below the form, there is a red horizontal bar with the text "Passwords do not match [js]".

Email	abhi02@test.com
Username	abhi02
Password	1234567899
Confirm	1234567890

JS validation showing passwords don't match

Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL: ak2774-dev-f660064fd52c.herokuapp.com/Project/register.php. The page contains a registration form with four input fields: Email, Username, Password, and Confirm. Below the form, there is a yellow horizontal bar with the text "The chosen email is not available.".

Email	<input type="text"/>
Username	<input type="text"/>
Password	<input type="text"/>
Confirm	<input type="text"/>

Email already in use message

Checklist Items (0)

The screenshot shows a web browser window with the URL `ak2774-dev-f660064fd52c.herokuapp.com/Project/register.php`. The page contains a registration form with fields for Email, Username, Password, and Confirm. A yellow horizontal bar at the top displays the error message: "The chosen username is not available." Below the form, there is a "Register" button.

Email

Username

Password

Confirm

Register

Username already in use message

Checklist Items (0)

The screenshot shows a web browser window with the URL `ak2774-dev-f660064fd52c.herokuapp.com/Project/register.php`. The page contains a registration form with fields for Email, Username, Password, and Confirm. A green horizontal bar at the top displays the success message: "Successfully registered!" Below the form, there is a "Register" button.

Successfully registered!

Email

Username

Password

Confirm

Register

New account being created message

Checklist Items (0)

Task #2 - Points: 1

Text: Screenshot of the form code

Details:

Should have appropriate input types for the field

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```
<form onsubmit="return validate(this)" method="POST">
  <div>
    <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" />
</form>
```

Screenshot of form code

^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
<input type="checkbox"/> #1	1	Show the JavaScript validations (include any extra files related)
<input type="checkbox"/> #2	1	Show the PHP validations (include any lib content)
<input type="checkbox"/> #3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

Task Screenshots:

Gallery Style: Large View

Small

Medium

Large

```
function validate(form) {
    // TODO 1: implement JavaScript validation
    // ensure it returns false for an error and true for success
    var email = form.email.value;
    var username = form.username.value;
    var password = form.password.value;
    var confirm = form.confirm.value;
    let isValid=true;
    var emailPattern = /^[^@\s]+@[^\s]+\.\w+$/;
    var usernamePattern = /^[a-zA-Z0-9_-]{3,16}$/;

    if (email === "") {
        flash("Email must not be empty [js]", "danger");
        isValid=false;
    }
    else if (!emailPattern.test(email)) {
        flash("Invalid email address [js]", "danger"); //ak2774
        isValid=false; //4/1/2024
    }
    if (username === "") {
        flash("Username must not be empty [js]", "danger");
        isValid=false;
    }
    else if (!usernamePattern.test(username)) {
        flash("Username must only contain 3-16 characters a-z, 0-9, _, or - [js]", "danger");
        isValid=false;
    }
    if (password === "") {
        flash("Password must not be empty [js]", "danger");
        isValid=false;
    }
    else if (password.length < 8 || confirm.length < 8) {
        flash("Password must be at least 8 characters long [js]", "danger");
        isValid=false;
    }
    if (password != confirm) {
        flash("Passwords do not match [js]", "danger");
        isValid=false;
    }

    return isValid;
} <- #25-82 function validate(form)
```

Client-side validation

Checklist Items (0)

```
if (isset($_POST["email"]) && isset($_POST["password"]) && isset($_POST["confirm"]) && isset($_POST["username"])) {
    $email = se($_POST, "email", "", false);
    $password = se($_POST, "password", "", false);
    $confirm = se($_POST, "confirm", "", false);
    $username = se($_POST, "username", "", false);
    // TODO 3
    $hasError = false;
    if (empty($email)) {
        flash("Email must not be empty", "danger");
        $hasError = true;
    }
    //sanitize
    $email = sanitize_email($email);
    //validate
    if (!is_valid_email($email)) {
        flash("Invalid email address", "danger");
        $hasError = true;
    }
}
```

```

        }
        if (!is_valid_username($username)) {
            flash("Username must only contain 3-16 characters a-z, 0-9, _, or -", "danger");
            $hasError = true;
        }
        if (empty($password) || empty($confirm)) {
            flash("password must not be empty", "danger"); //ak2774
            $hasError = true; //4/1/2024
        }

        if (!is_valid_password($password)) {
            flash("Password too short", "danger");
            $hasError = true;
        }
        if (strlen($password) > 0 && $password !== $confirm) {
            flash("Passwords must match", "danger");
            $hasError = true;
        }
        if (!$hasError) {
            //1000 4
            $hash = password_hash($password, PASSWORD_BCRYPT);
            $db = getDB();
            $stmt = $db->prepare("INSERT INTO Users (email, password, username) VALUES(:email, :password, :username)");
            try {
                $stmt->execute([":email" => $email, ":password" => $hash, ":username" => $username]);
                flash("Successfully registered!", "success");
            } catch (Exception $e) {
                users_check_duplicate($e->errorInfo);
            }
        } <- #123-134 If (!$hasError)
} <- #06-135 If (isset($_POST["email"]) && isset($_POST["password"])) && is...

```

Server-side validation

Checklist Items (0)

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
<input type="checkbox"/> #1	1	Password should be hashed
<input type="checkbox"/> #2	1	Should have email, password, username (unique), created, modified, and id fields
<input type="checkbox"/> #3	1	Ensure left panel or database name is present (should contain your ucid)

Task Screenshots:

Gallery Style: Large View

Small Medium Large

The screenshot shows a MySQL Workbench interface with the following details:

- File Edit Selection View Go Run Terminal Help**
- DATABASE** AbhiServer 8.0.35-Ubuntu020.04.1
- Tables** (3) Roles 2, UserRoles 6, Users 5
- Properties Data Process**
- Query** SELECT * FROM `Users` LIMIT 100
- Search results** Total 5
- Columns** id int, email varchar(100), password varchar(60), created timestamp, modified timestamp, username varchar(30)
- Data**

	id	email	password	created	modified	username
1	1	abhi@test.com	\$2y\$10\$B9h7Q4fl2cT.tr	2024-02-28 21:31:07	2024-04-01 17:22:02	abhi
2	2	abhi2@test.com	\$2y\$10\$SLmG5sk/qEFH	2024-02-29 01:03:12	2024-02-29 01:03:12	abhi2
3	5	abhi3@test.com	\$2y\$10\$KymJnSku/dzI	2024-02-29 01:04:32	2024-02-29 01:04:32	abhi3
4	7	abhikart02@gmail.com	\$2y\$10\$.CQZYJjrxClqr	2024-03-29 17:54:25	2024-04-01 18:08:37	abhikart02
5	11	abhi02@test.com	\$2y\$10\$ZWVIOO3yBYO	2024-04-01 21:52:28	2024-04-01 21:52:28	abhi02

Screenshot of Users table

Checklist Items (0)

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

1 Details:

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

Response:

Page Load:

The PHP code at the beginning includes necessary files and resets the session. The HTML form for user registration is displayed, including fields for email, username, password, and password confirmation. JavaScript code for client-side validation is included in the page. This code defines a validate() function that checks if the input values are valid before submitting the form.

Client-Side Validation (JavaScript):

When the form is submitted, the validate() function is called. This function retrieves input values for email, username, password, and confirmation. It validates each field according to defined criteria, such as non-empty, valid email format, valid username format, minimum password length, and password confirmation. If any validation fails, an error message is displayed using the flash() function, and isValid is set to false. If all validations pass, isValid remains true.

Server-Side Validation (PHP):

After the client-side validation, if the form is submitted, the PHP code checks if the required POST variables are set. It retrieves the submitted values for email, username, password, and confirmation. Server-side validation checks are performed to ensure that the submitted data is valid: Email must not be empty, and it must be a valid email format. Username must not be empty, and it must follow the specified pattern. Password and confirmation must not be empty, and password must be at least 8 characters long. Password and confirmation must match. If any validation fails, error messages are flashed using the flash() function, and \$hasError is set to true.

Database Insertion:

If all validations pass (both client-side and server-side), \$hasError remains false. The password is hashed using password_hash() function for security. A database connection is established, and a prepared statement is used to insert the user's data (email, hashed password, username) into the database table Users. If the insertion is successful,

a success message is flashed to the user. If there's an error during the database operation, such as a duplicate entry, it's handled appropriately, and an error message is flashed.



^COLLAPSE ^

Task #6 - Points: 1

Text: Include pull request links related to this feature



Details:
Should end in /pull/#

URL #1

<https://github.com/abhinavkarthikn/ak2774-it202-008/pull/29>



User Login (2 pts.)

^COLLAPSE ^

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 ^

Task #1 - Points: 1

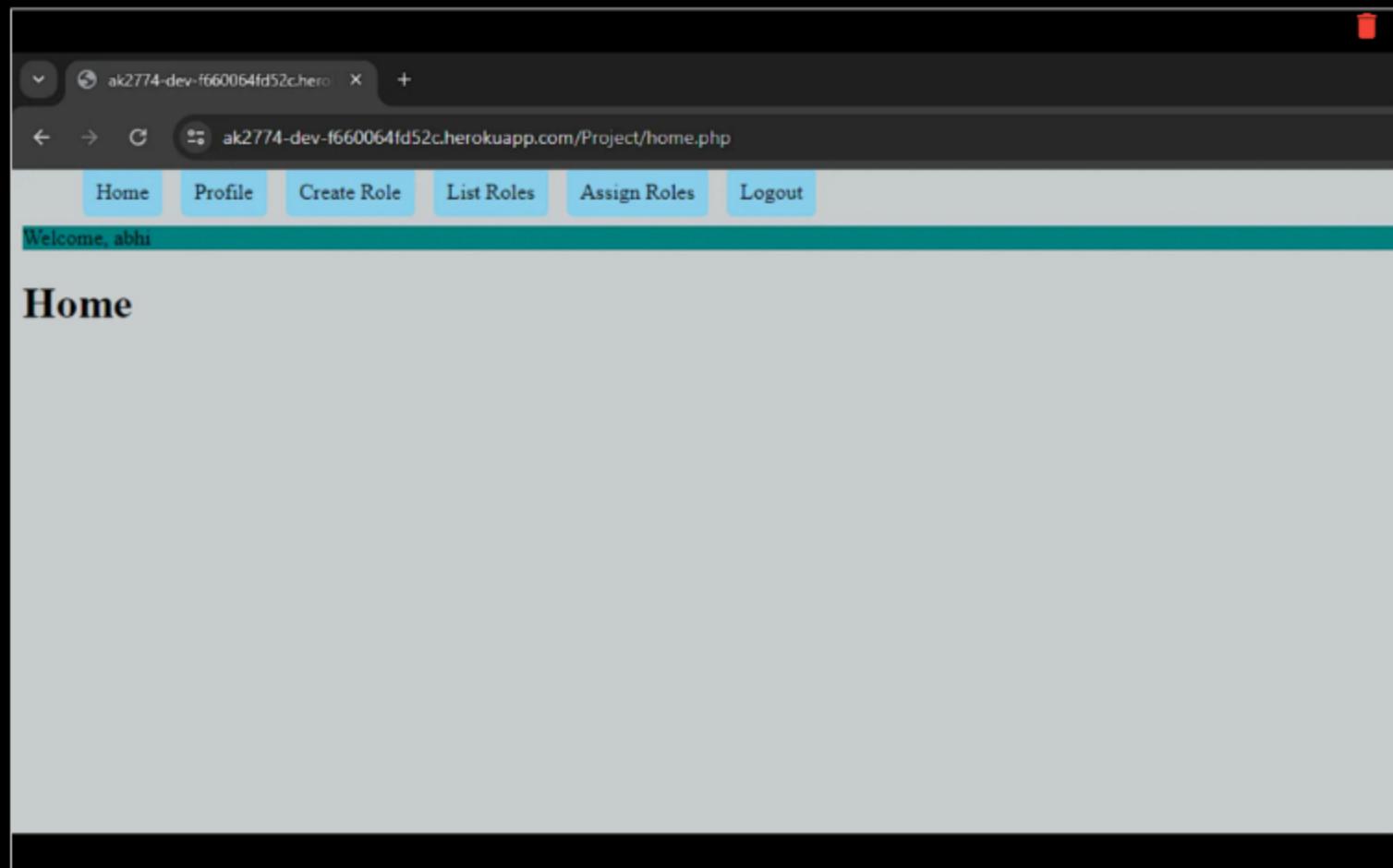
Text: Capture the following screenshots

#	Points	Details
<input type="checkbox"/> #1	1	Screenshot of the navigation when logged in (site)
<input type="checkbox"/> #2	1	Screenshot of the redirect to login with the user-friendly logged-out message (site)
<input type="checkbox"/> #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



Screenshot of page when logged in

Checklist Items (0)

 A screenshot of a web browser window. The address bar shows the URL "ak2774-dev-f660064fd52c.herokuapp.com/Project/login.php". The page title is "Login". The header includes links for "Login" and "Register". A green banner at the top says "Successfully logged out". Below the banner are input fields for "Email/Username" and "Password", and a "Submit" button.

Login

Screenshot of login page with log out message

Checklist Items (0)



```
public_html > Project > logout.php
...
1  <?php
2  session_start();
3  require(__DIR__ . "/../../lib/functions.php");      //ak2774
4  reset_session();                                    //4/1/2024
5
6  flash("Successfully logged out", "success");
7  header("Location: login.php");
```

Screenshot of code

Checklist Items (0)

Task #2 - Points: 1

COLLAPSE

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

https://github.com/abhinavkarthikn/ak2774_it202_008/pull/32

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



```
function is_logged_in($redirect = false, $destination = "login.php")
{
    $isLoggedIn = isset($_SESSION["user"]);
    if ($redirect && !$isLoggedIn) [
        //if this triggers, the calling script won't receive a reply since die()/exit() terminates it
        flash("You must be logged in to view this page", "warning"); //ak2774
        die(header("Location: $destination")); //4/1/2024
    ] <- #10-14 if ($redirect && !$isLoggedIn)
    return $isLoggedIn;
} <- #8-16 function is_logged_in($redirect = false, $destination = "logi...
```

Screenshot of is_logged_in function

Checklist Items (0)

```
<?php
require_once(__DIR__ . "/../../partials/nav.php"); // ak2774, 4/1/2024
is_logged_in(true);
?>
<?php
```

Used on profile.php

Checklist Items (0)

```
<?php
require(__DIR__ . "/../../partials/nav.php");
?>
<h1>Home</h1>
<?php

if (is_logged_in(true)) {
    //comment this out if you don't want to see the session variables
    error_log("Session data: " . var_export($_SESSION, true)); //ak2774, 4/1/2024
}
?>
<?php
require(__DIR__ . "/../../partials/flash.php");
?>
```

Used on home.php

Checklist Items (0)

You must be logged in to view this page

Email/Username

Password

Login

Message of manual access

Checklist Items (0)

Task #2 - Points: 1

Text: Authorization Screenshots

Checklist *The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Screenshot of the function that checks for a specific role
<input type="checkbox"/> #2	1	Screenshot of the role check function being used. Also caption what pages it's used on
<input type="checkbox"/> #3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)
<input type="checkbox"/> #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

```
function has_role($role)
{
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {
        foreach ($_SESSION["user"]["roles"] as $r) {
            if ($r["name"] === $role) { //ak2774, 4/1/2024
                return true;
            }
        } #20-24
    } #19-25
    if (is_logged_in() && isset($_SESSION["user"]["roles"]))
        return false;
} #18-27
```

Screenshot of has_role function

Checklist Items (0)

```
if (!has_role("Admin")) {
    flash("You don't have permission to view this page", "warning"); //ak2774, 4/1/2024
    die(header("Location: $BASE_PATH" . "/home.php"));
}
```

Used in assign_roles

Checklist Items (0)

```
if (!has_role("Admin")) {  
    flash("You don't have permission to view this page", "warning"); //ak2774, 4/1/2024  
    die(header("Location: " . get_url("home.php")));  
}
```

Used in `create_role`

Checklist Items (0) 

```
if (!has_role("Admin")) {  
    flash("You don't have permission to view this page", "warning"); //ak2774, 4/1/2024  
    die(header("Location: " . get_url("home.php")));  
}
```

Used in `list_roles`

Checklist Items (0) 

ak2774-dev-f660064fd52c.herokuapp.com

Home Profile Logout

You don't have permission to view this page

Home

Message showing manually accessing a page user doesn't have access to

Checklist Items (0)

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

Checklist		*The checkboxes are for your own tracking
#	Points	Details
<input type="checkbox"/> #1	1	At least one valid and enabled User->Role reference (UserRoles table)
<input type="checkbox"/> #2	1	UserRoles Table should have id, user_id, role_id, is_active, created, and modified columns
<input type="checkbox"/> #3	1	Roles Table should have id, name, description, is_active, modified, and created columns
<input type="checkbox"/> #4	1	At least one valid and enabled Role (Roles table)
<input type="checkbox"/> #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



The screenshot shows the MySQL Workbench interface with the 'Roles' table selected. The table has columns: id, name, description, is_active, created, and modified. Two rows are present: one for 'Admin' and one for 'Test Role'.

	id	name	description	is_active	created	modified
	Q	P	U	D	C	A
>	1	-1	Admin	1	2024-03-27 21:03:07	2024-03-28 23:40:31
>	2	2	Test Role	1	2024-03-27 22:09:41	2024-03-27 22:12:29

Screenshot of roles table

Checklist Items (0)

The screenshot shows the MySQL Workbench interface with the 'userRoles' table selected. The table has columns: id, user_id, role_id, is_active, created, and modified. Six rows map users 1 through 5 to roles 1 through 3.

	id	user_id	role_id	is_active	created	modified
	Q	P	U	D	C	A
>	1	18	1	-1	1	2024-03-27 21:54:52
>	2	19	2	2	1	2024-03-27 22:15:18
>	3	21	1	2	0	2024-03-27 22:15:46
>	4	23	5	2	1	2024-03-27 22:15:47
>	5	29	7	-1	1	2024-03-29 17:57:36
>	6	30	5	-1	0	2024-03-30 21:01:34

Screenshot of userRoles table

Checklist Items (0)

Task #4 - Points: 1

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



Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	What's the purpose of the UserRoles table?
<input type="checkbox"/> #2	1	How does Roles.is_active differ from UserRoles.is_active?

Response:

The purpose of the UserRoles table is to show which user has which role or roles active. Roles.is_active shows whether a role is active or not and UserRoles.is_active shows whether a user has a role active or not

Task #5 - Points: 1**Text: Include pull request links related to this feature****i Details:**

Should end in /pull/#

URL #1<https://github.com/abhinavkarthikn/ak2774-it202-008/pull/37>**User Profile (2 pts.)****^COLLAPSE ^****Task #1 - Points: 1****Text: View Profile Website Page****Checklist**

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Heroku dev url should be present in the address bar
<input type="checkbox"/> #2	1	Should have thoughtful CSS applied
<input type="checkbox"/> #3	1	Show the profile form correctly populated on page load (username, email)
<input type="checkbox"/> #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**

ak2774-dev-f660064fd52c.herokuapp.com



[Home](#)[Profile](#)[Create Role](#)[List Roles](#)[Assign Roles](#)[Logout](#)Email Username

Password Reset

Current Password New Password Confirm Password [Update Profile](#)

Screenshot of profile page

Checklist Items (0)

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:

Navigation and Authentication:

The partials/nav.php file is required to include the navigation bar. The `is_logged_in(true)` function is called to ensure that the user is logged in. If the user is not logged in, they are redirected to the login page.

Handling Form Submission:

If the form is submitted (`isset($_POST["save"])`), the PHP code executes. It retrieves the submitted values for email and username using the `se()` function, which provides default values if the keys are not found in the `$_POST` array. It prepares a SQL query to update the user's email and username in the database based on their user ID. The query is executed using a prepared statement, and if successful, a success message is flashed to the user.

Error Handling for Database Operations:

If an error occurs during the database operation, it is caught by a `try-catch` block. If the error code indicates a duplicate

If an error occurs during the database operation, it is caught by a try-catch block. If the error code indicates a duplicate entry (1062), it checks the error message for details about which field caused the duplicate entry. Depending on the type of error, appropriate error messages are flashed to the user.

Fetching Updated User Data:

After updating the user's email and username, a new SQL query is prepared to fetch the updated user data from the database based on their user ID. If the user data is successfully fetched, it is stored in the `$_SESSION["user"]` array, updating the email and username values. If the user doesn't exist in the database, an error message is flashed to the user.

Handling Password Reset:

The current password, new password, and confirm password fields are retrieved from the form submission. If all fields are not empty, the current password is verified against the hashed password stored in the database using `password_verify()`. If the current password is valid, the new password is hashed using `password_hash()` and updated in the database. If the current password is invalid or the new passwords don't match, appropriate error messages are flashed to the user.

Form Population:

The user's current email and username are fetched using `get_user_email()` and `get_username()` functions, respectively. These values are preloaded into the corresponding input fields in the HTML form.

Client-Side Validation (JavaScript):

The `validate()` function is defined to perform client-side validation before form submission. It checks if the email and username fields are empty and if they match the required patterns. It also validates the new password fields for length and matching confirmation. If any validation fails, error messages are flashed using the `flash()` function.

Task #3 - Points: 1

Text: Edit Profile Website Page

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input type="checkbox"/> #1	1	Heroku dev url should be present in the address bar
<input type="checkbox"/> #2	1	Should have thoughtful CSS applied
<input type="checkbox"/> #3	1	Demonstrate with before and after of a username change (including success message)
<input type="checkbox"/> #4	1	Demonstrate with a before and after of an email change (including success message)
<input type="checkbox"/> #5	1	Demonstrate the success message of updating password
<input type="checkbox"/> #6	1	Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)
<input type="checkbox"/> #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

The screenshot shows a web browser window with the following details:

- Address Bar:** ak2774-dev-f660064fd52c.herokuapp.com/Project/profile.php
- Page Title:** Profile
- Navigation:** Home, Profile, Create Role, List Roles, Assign Roles, Logout
- Form Fields:**
 - Email: abhi@test.com
 - Username: abhi
 - Current Password: (empty)
 - New Password: (empty)
 - Confirm Password: (empty)
- Buttons:** Update Profile

Before username change

Checklist Items (0)

The screenshot shows a web browser window with the following details:

- Address Bar:** ak2774-dev-f660064fd52c.herokuapp.com/Project/profile.php
- Page Title:** Profile
- Navigation:** Home, Profile, Create Role, List Roles, Assign Roles, Logout
- Message Bar:** Profile saved (in green)
- Form Fields:**
 - Email: abhi@test.com
 - Username: abhi1102
 - Current Password: (empty)
 - New Password: (empty)
 - Confirm Password: (empty)
- Buttons:** Update Profile

After username change

Checklist Items (0)

A screenshot of a web browser window titled "ak2774-dev-f660064fd52c.herokuapp.com". The URL in the address bar is "ak2774-dev-f660064fd52c.herokuapp.com/Project/profile.php". The page displays a user profile form with the following fields:

- Email: abhi@test.com
- Username: abhi
- Current Password: (empty input field)
- New Password: (empty input field)
- Confirm Password: (empty input field)

Below the form is a blue button labeled "Update Profile".

Before email change

Checklist Items (0)

A screenshot of a web browser window titled "ak2774-dev-f660064fd52c.herokuapp.com". The URL in the address bar is "ak2774-dev-f660064fd52c.herokuapp.com/Project/profile.php". The page displays a user profile form with the following fields:

- Email: abhi1102@test.com
- Username: abhi
- Current Password: (empty input field)
- New Password: (empty input field)
- Confirm Password: (empty input field)

Below the form is a blue button labeled "Update Profile". A green banner at the top of the page reads "Profile saved".

After email change

Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL `ak2774-dev-f660064fd52c.herokuapp.com/Project/profile.php`. The page title is "Profile". Below the title, there is a navigation menu with buttons for Home, Profile, Create Role, List Roles, Assign Roles, and Logout. A green success message box displays "Profile saved" and "Password reset". Below the message box, there are input fields for Email (containing "abhi@test.com"), Username (containing "abhi"), and several password fields (Current Password, New Password, Confirm Password). At the bottom left is a "Update Profile" button.

Password reset message

Checklist Items (0)

A screenshot of a web browser window. The address bar shows the URL `ak2774-dev-f660064fd52c.herokuapp.com/Project/profile.php`. The page title is "Profile". Below the title, there is a navigation menu with buttons for Home, Profile, Create Role, List Roles, Assign Roles, and Logout. A yellow validation message box displays "Invalid email address [js]". Below the message box, there are input fields for Email (containing "abhi"), Username (containing "abhi"), and several password fields (Current Password, New Password, Confirm Password). At the bottom left is a "Update Profile" button.

JS validation of email format

Checklist Items (0)

A screenshot of a web browser window titled "ak2774-dev-f660064fd52c.herokuapp.com". The URL in the address bar is "ak2774-dev-f660064fd52c.herokuapp.com/Project/profile.php". The page displays a profile update form with the following fields:

- Email: abhi@test.com (highlighted in red)
- Username: ab
- Password Reset (link)
- Current Password (input field)
- New Password (input field)
- Confirm Password (input field)
- Update Profile button

A yellow horizontal bar at the top of the form area contains the error message: "Username must only contain 3-16 characters a-z, 0-9, _, or - [js]".

JS validation of username format

Checklist Items (0)

A screenshot of a web browser window titled "ak2774-dev-f660064fd52c.herokuapp.com". The URL in the address bar is "ak2774-dev-f660064fd52c.herokuapp.com/Project/profile.php". The page displays a profile update form with the following fields:

- Email: abhi@test.com
- Username: abhi (highlighted in red)
- Password Reset (link)
- Current Password: 12345678
- New Password: 123456
- Confirm Password: 123456
- Update Profile button

A yellow horizontal bar at the top of the form area contains the error message: "New password must be at least 8 characters long [js]".

JS validation of new password format

Checklist Items (0)

ak2774-dev-f660064fd52c.herokuapp.com/Project/profile.php

Home Profile Create Role List Roles Assign Roles Logout

Password and Confirm password must match [js]

Email abhi@test.com

Username abhi

Current Password 12345678

New Password 1234567899

Confirm Password 1234567890

Update Profile

JS validation of new and confirm password matching

Checklist Items (0)

ak2774-dev-f660064fd52c.herokuapp.com/Project/profile.php

Home Profile Create Role List Roles Assign Roles Logout

The chosen email is not available.

Email abhi@test.com

Username abhi

Current Password

New Password

Confirm Password

[Update Profile](#)

Message of email already in use

Checklist Items (0)

The chosen username is not available.

Email abhi@test.com

Username abhi

Password Reset

Current Password

New Password

Confirm Password

[Update Profile](#)

Message of username already in use

Checklist Items (0)

Profile saved

Current password is invalid

Email abhi@test.com

Username abhi

Password Reset

Current Password

New Password

Confirm Password

[Update Profile](#)

Message that current password is not correct

Checklist Items (0)

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
<input type="checkbox"/> #1	1	Updating Username/Email
<input type="checkbox"/> #2	1	Updating password
<input type="checkbox"/> #3	1	Don't just show code, translate things to plain English

Response:

Updating Username/Email:

Form Submission Handling:

When the form is submitted, the PHP code checks if the save button is clicked (`isset($_POST["save"])`). It retrieves the submitted values for email and username from `$_POST`. It prepares parameters for a SQL query to update the user's email and username in the database.

Database Update Query:

The SQL query is prepared to update the user's email and username in the Users table based on their user ID. The prepared statement is executed with the provided parameters.

Error Handling:

If the query execution is successful, a success message is flashed to the user indicating that the profile has been saved. If there's a duplicate entry error (error code 1062), it checks the error message to determine which field caused the duplicate entry. Depending on the error, appropriate error messages are flashed to the user.

Fetching Updated User Data:

After successfully updating the email and username in the database, a new query is prepared to fetch the updated user data. The user's data is fetched based on their user ID. If the user data is found, it is updated in the session to reflect the changes.

Updating Password:

Form Submission Handling:

The current password, new password, and confirm password fields are retrieved from the form submission. It checks if all fields are not empty.

Current Password Verification:

If all password fields are filled, it prepares a query to fetch the current hashed password from the database based on the user ID. It verifies the current password by comparing the hashed password retrieved from the database with the submitted current password using `password_verify()`.

New Password Hashing and Update:

If the current password is verified successfully, the new password is hashed using `password_hash()`. A query is prepared to update the user's password in the database with the newly hashed password. The password update query is executed with the user ID and the new hashed password.

Error Handling:

If the current password is invalid, a warning message is flashed to the user. If the new passwords don't match, a warning message indicating that the passwords don't match is flashed.

Success Message:

If the password update is successful, a success message is flashed to the user indicating that the password has been reset.

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

<https://github.com/abhinavkarthikn/ak2774-it202-008/pull/33>

Misc (1 pt.)

[^COLLAPSE ^](#)

Task #1 - Points: 1

Text: Screenshot of wakatime

Task Screenshots:

Gallery Style: Large View

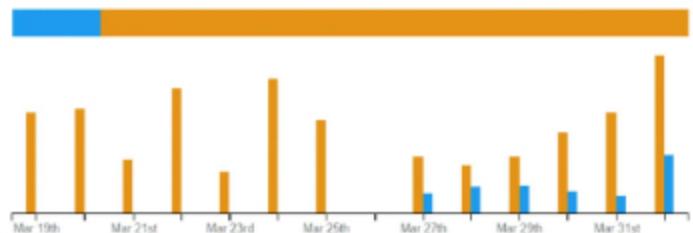
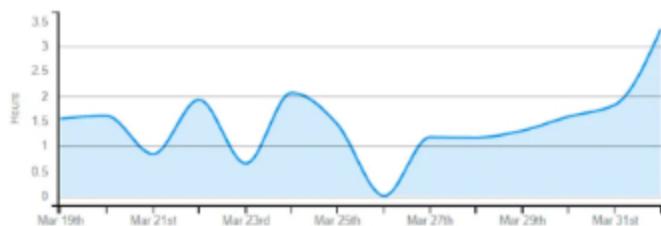
Small

Medium

Large

Projects • ak2774-it202-008

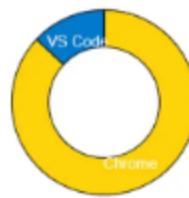
20 hrs 30 mins over the Last 14 Days in ak2774-it202-008 under all branches. ⏱



Languages



Editors



Screenshot of wakatime

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

Abhinav's it202 2024 project

View 1 + New view

Filter by keyword or by field

Todo

This item hasn't been started

In Progress

This is actively being worked on

Done

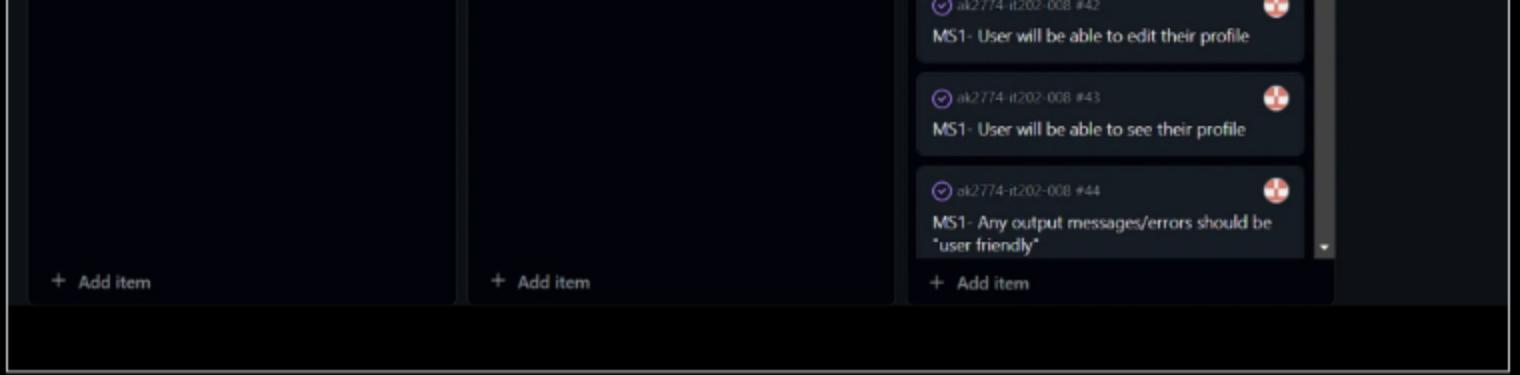
This has been completed

ak2774-it202-008 #39

MS1 - User will be able to logout

ak2774-it202-008 #41

MS1 - User will be able to register a new account



Screenshot of Project board part 1

This screenshot shows the project board after some work has been completed. The In Progress and Done columns now contain items. The Done column has been expanded to show five specific tasks, each with a checkmark, a link, and a red circular icon.

Todo	In Progress	Done
0	0	9
This item hasn't been started	This is actively being worked on	MS1- Any output messages/errors should be "user friendly" MS1- User will be able to login to their account MS1- Basic security rules implemented MS1- Basic Roles implemented MS1- Site should have basic styles/theme applied; everything should be styled

Screenshot of Project Board part 2



Task #3 - Points: 1

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

URL #1

<https://github.com/users/abhinavkarthikn/projects/1/views/1>



Task #4 - Points: 1

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

End of Assignment