Submission Worksheet

CLICK TO GRADE

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

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

Submissions:

Submission Selection

1 Submission [active] 4/1/2024 4:08:00 PM

Instructions

COLLAPSE

Prereqs:

10. Go through each lesson from "Project Setup and SQL" to "User Login Enhancement" and follow the branching names while gathering the code
    - Merge each into Milestone1 branch
    - Mark the related GitHub Issues items as "done"

    Implement your own custom CSS (something much different than the default "ugly" CSS given as an example)
    - Consider styling all forms/inputs, data output, navigation, etc

    Implement JavaScript validation on Register, Logout, and Profile (include "[Client]" in the output messages to differentiate between server-side validations)

Instructions:

Make sure you're in Milestone1 with the latest changes pulled
Ensure Milestone1 has been deployed to heroku dev
Gather the requested evidence and fill in the explanations per each prompt
Save the submission and generate the output PDF
Put the output PDF into your local repository folder
add/commit/push it to GitHub
Merge Milestone1 into dev
Locally checkout dev and pull the changes
Create and merge a pull request from dev to prod to deploy Milestone1 to prod
Upload this output PDF to Canvas

Branch name: Milestone1

Tasks: 26 Points: 10.00

User Registration (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of form on website page

Checklist

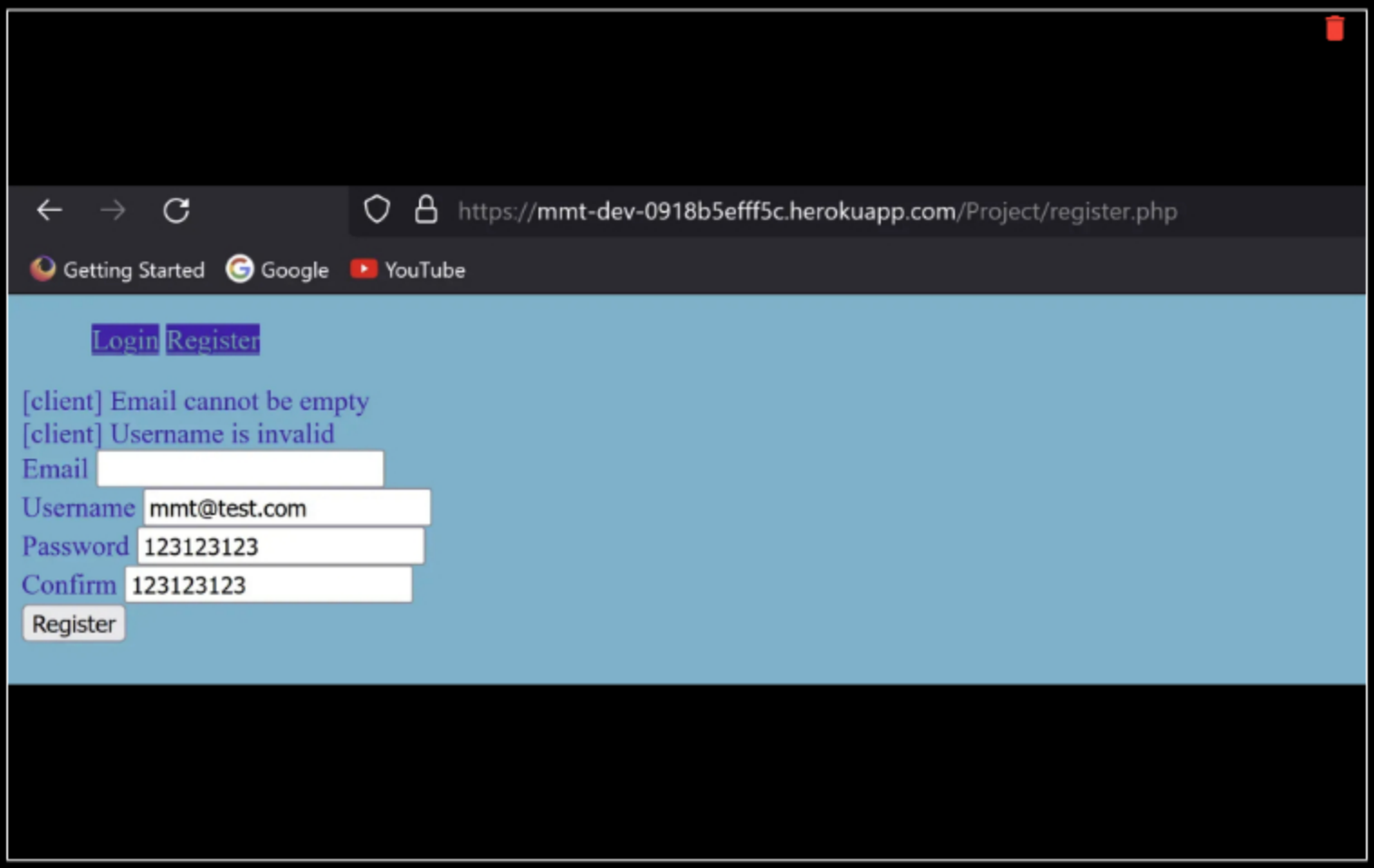
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required
#4	1	Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)
#5	1	Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)
#6	1	Demonstrate user-friendly message of new account being created

Task Screenshots:

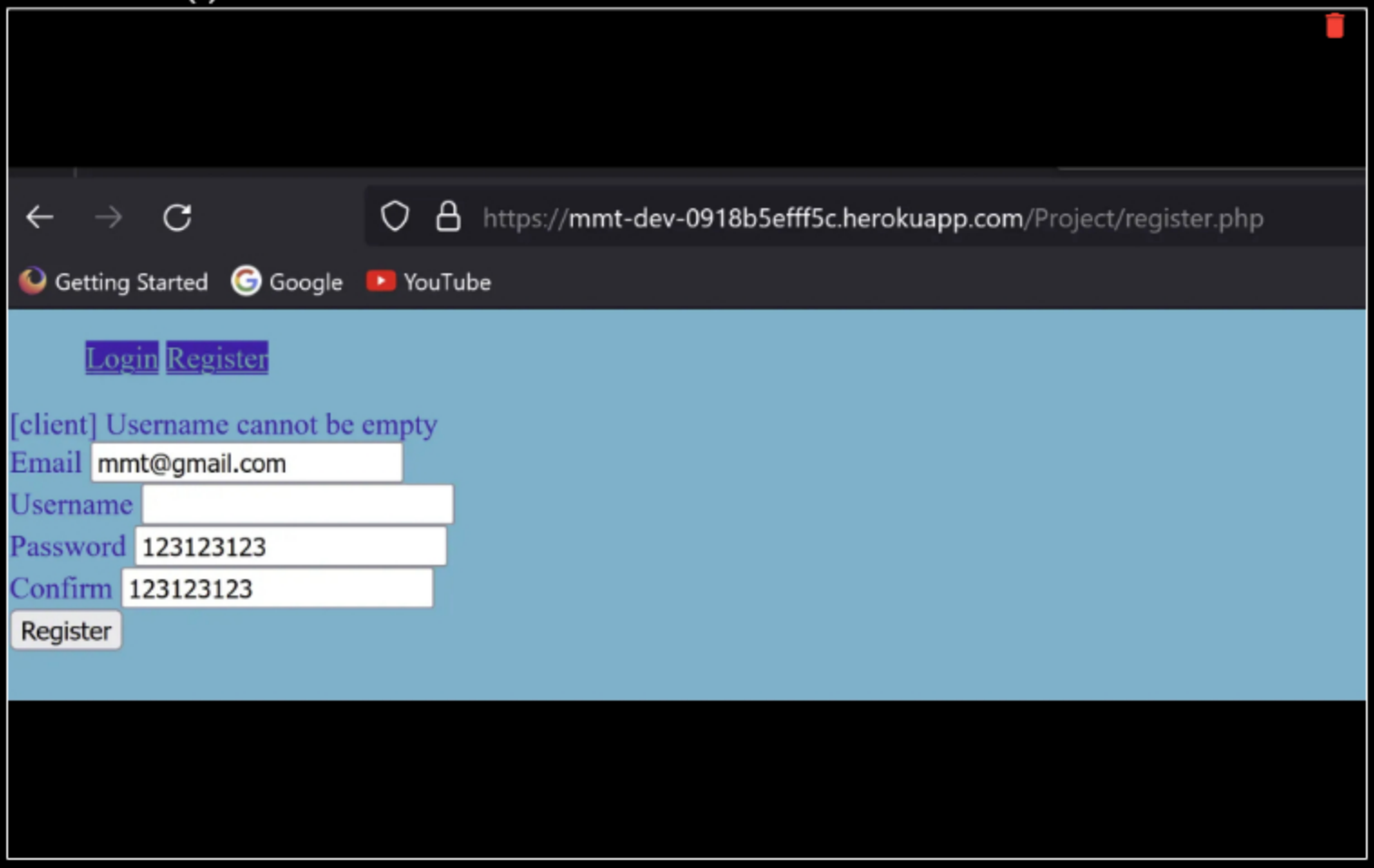
Gallery Style: Large View

Small Medium Large



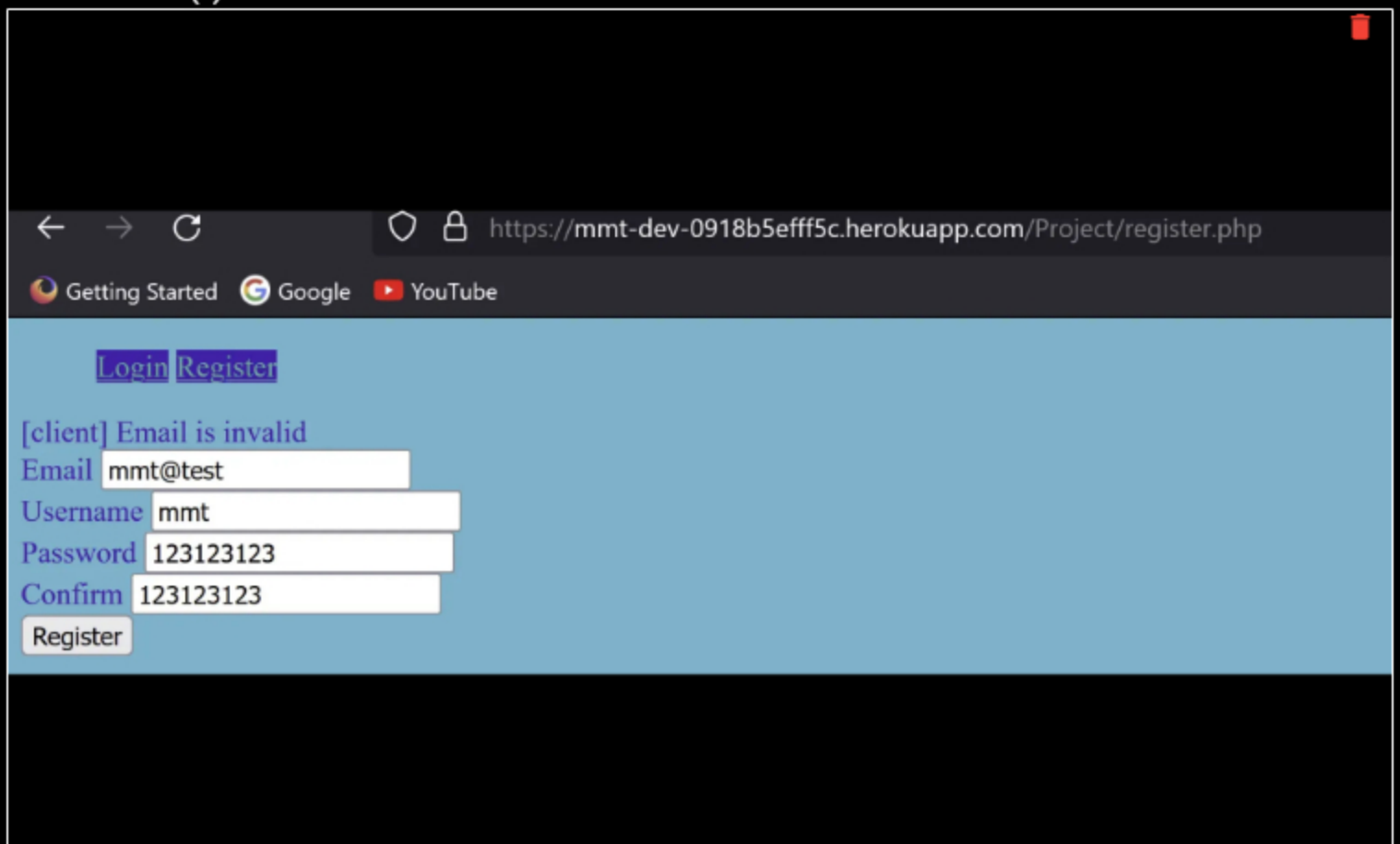
email cannot be empty & username is invalid (JS valid)

Checklist Items (0)



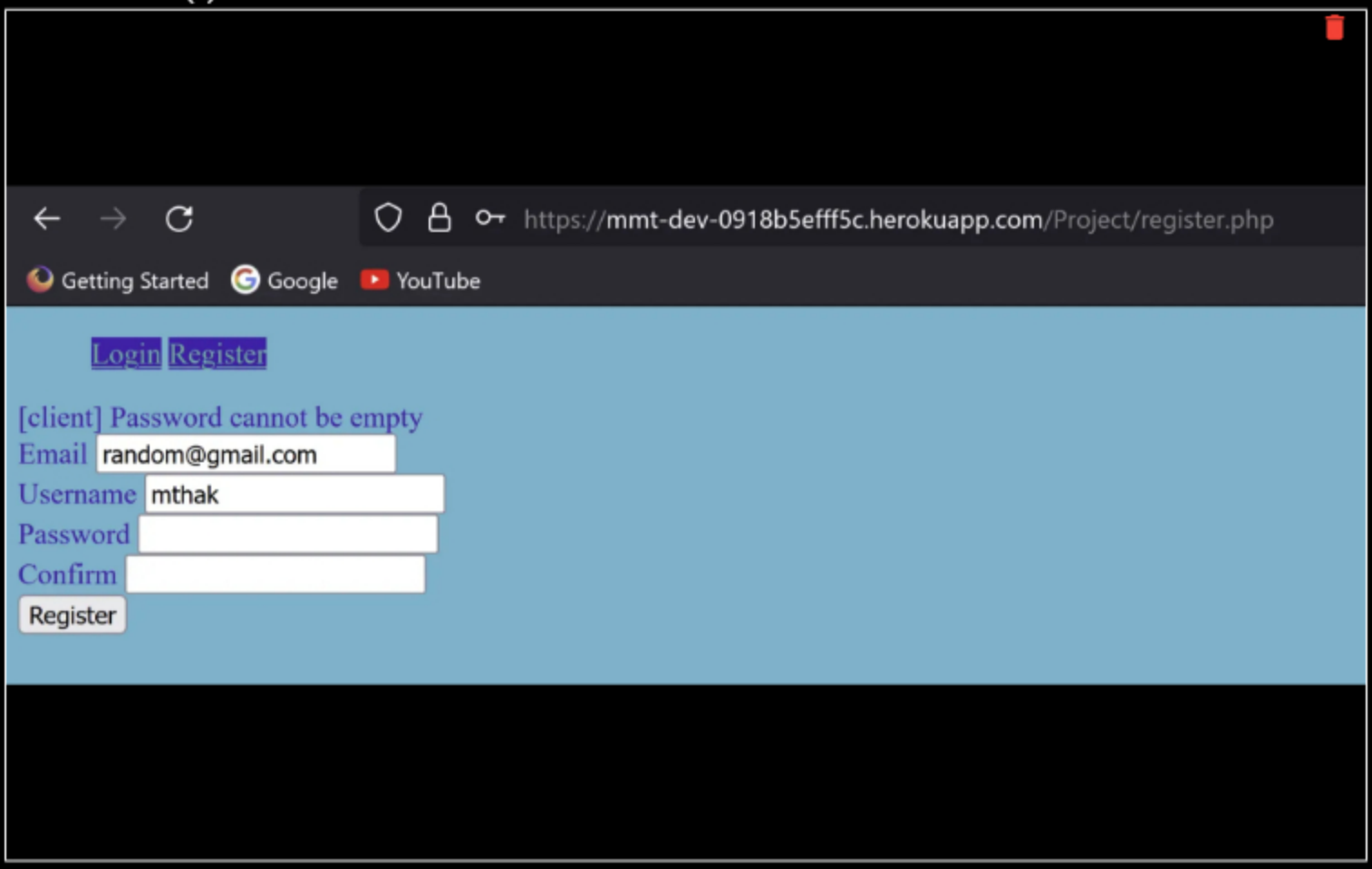
username cannot be empty (JS valid)

Checklist Items (0)



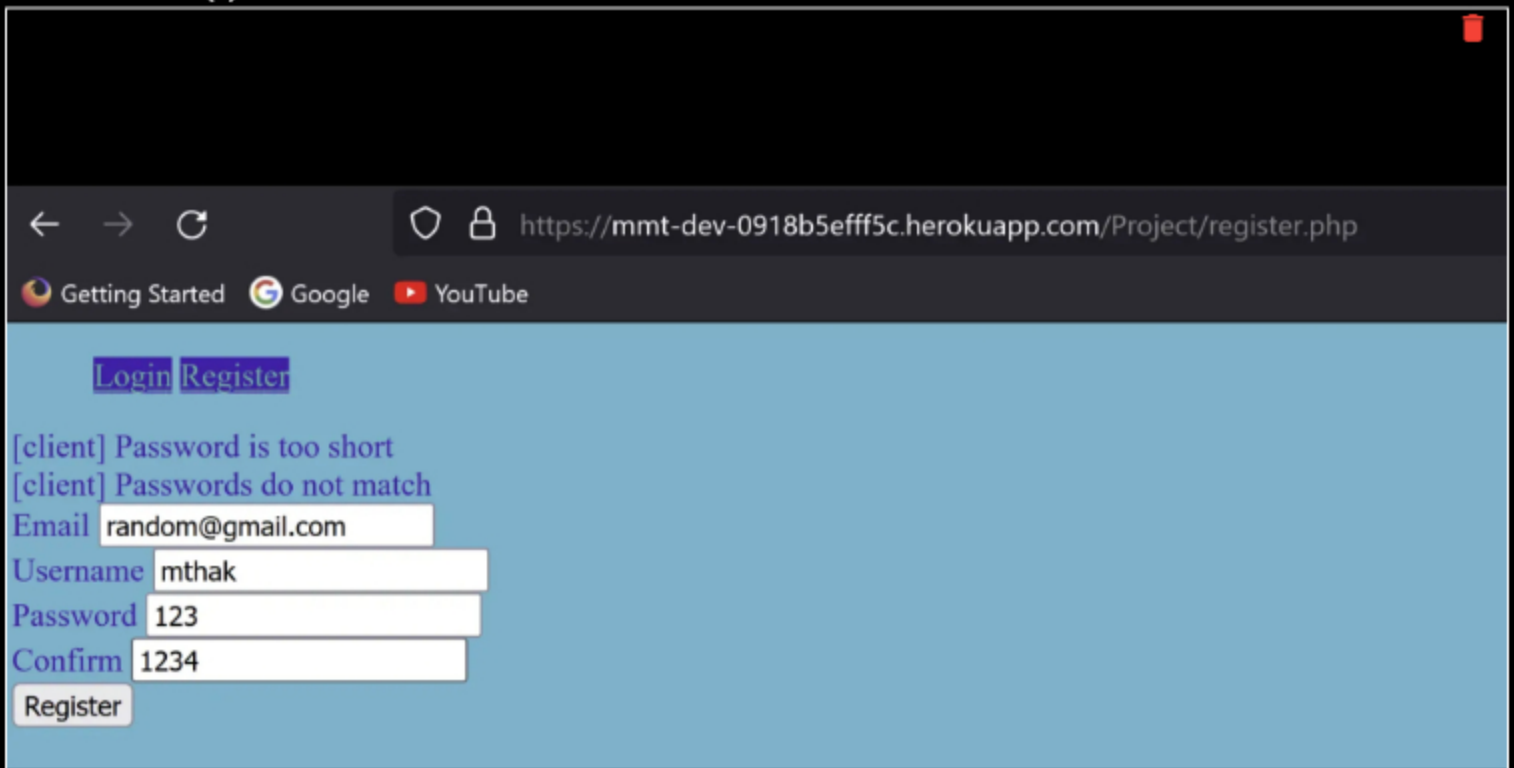
email is invalid (JS valid)

Checklist Items (0)



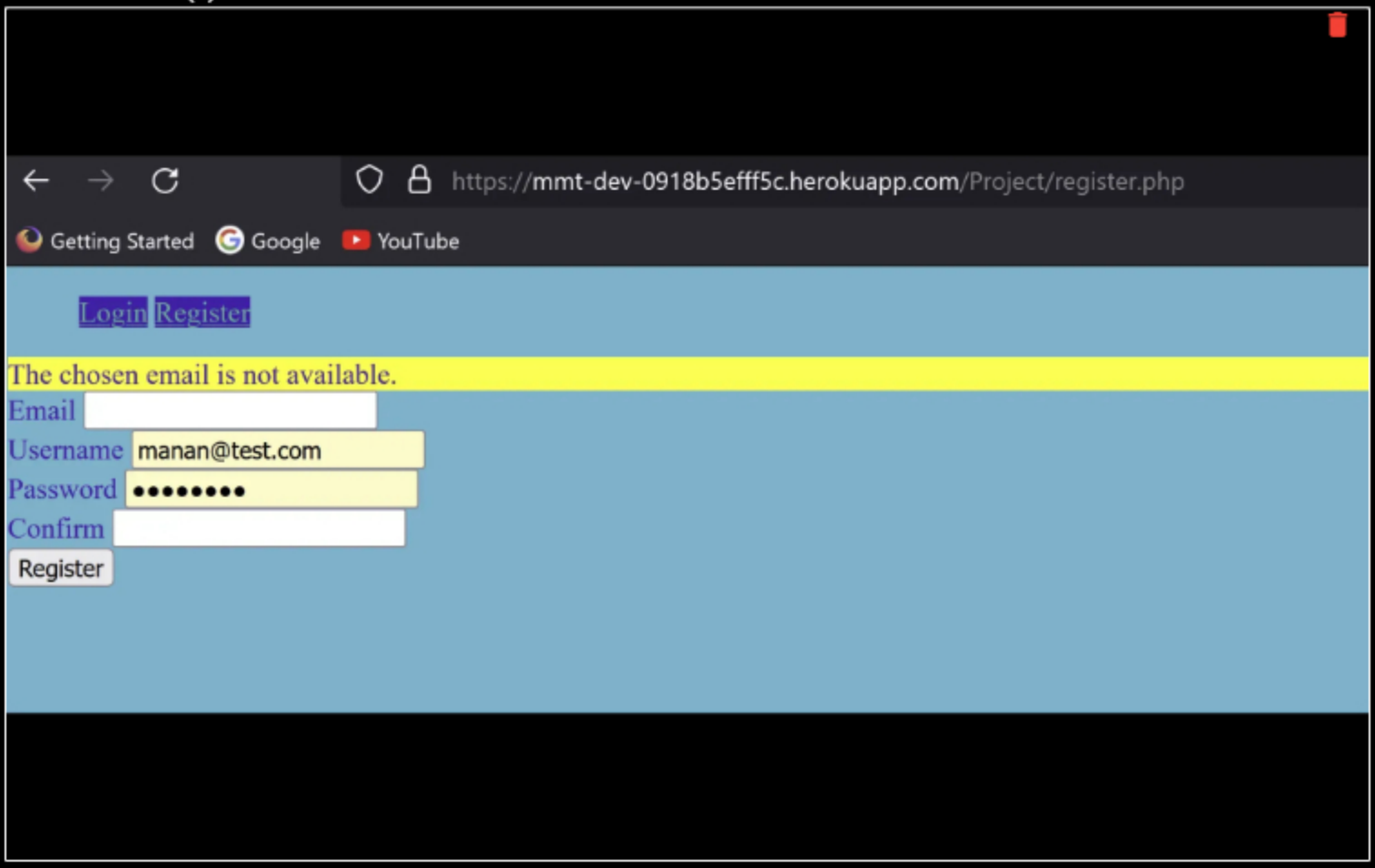
password cannot be empty (js valid)

Checklist Items (0)



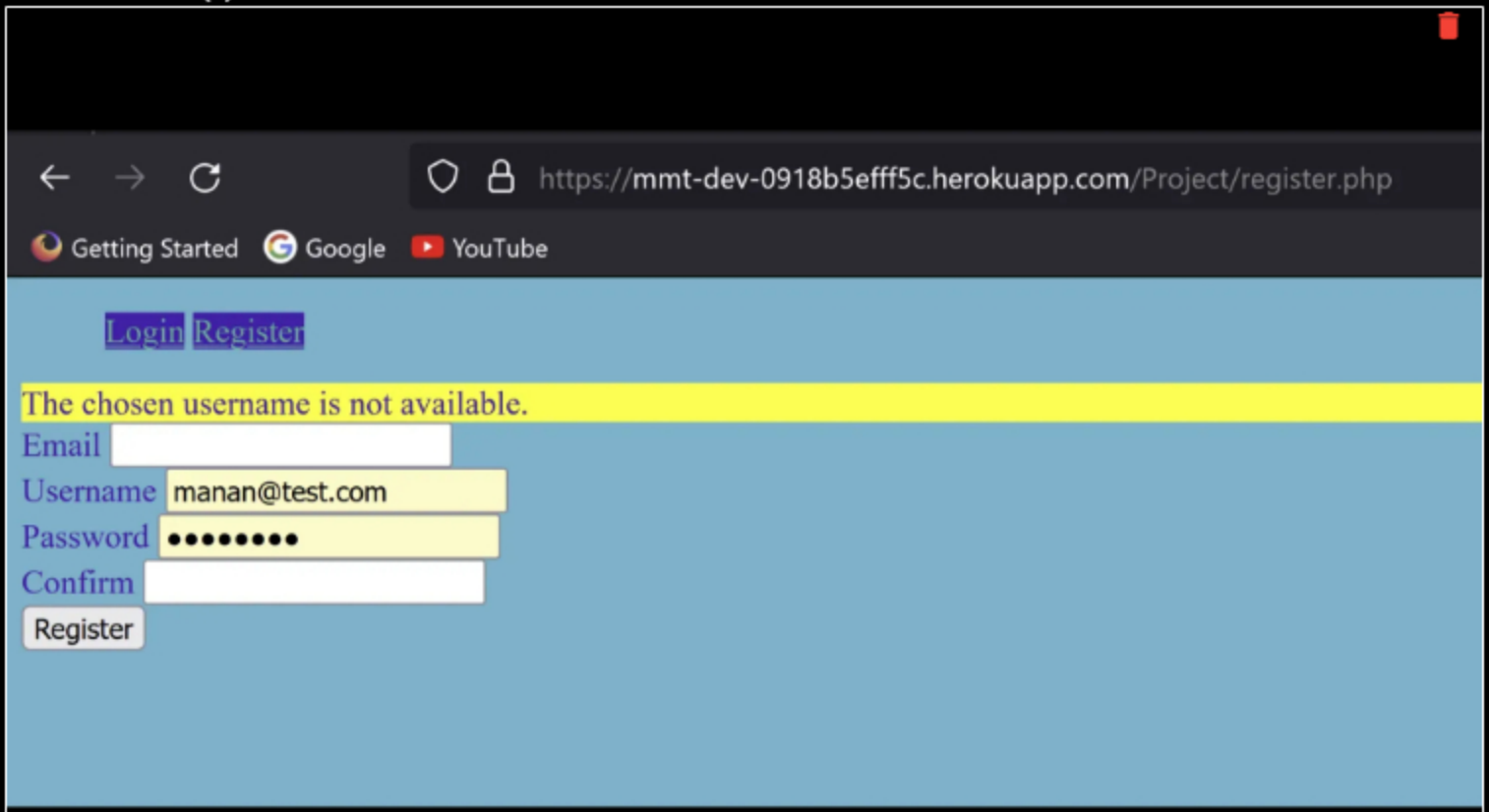
password is too short and passwords don't match (js valid)

Checklist Items (0)



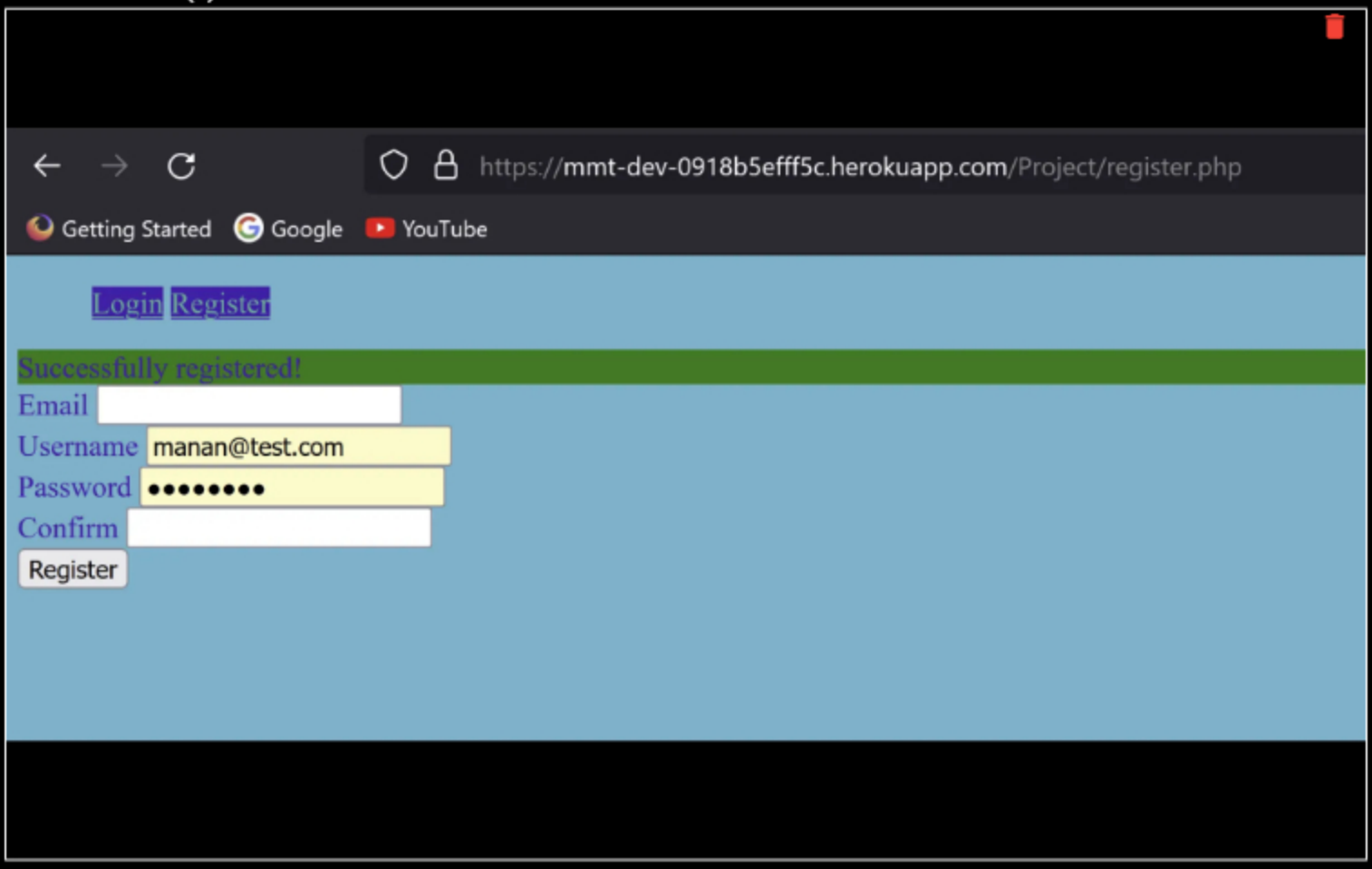
chosen email is unavailable

Checklist Items (0)



chosen username is unavailable

Checklist Items (0)



shows account being made along with user friendly message

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the form code

Details:

Should have appropriate input types for the fiel

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
...
1  <?php
2  require(__DIR__ . "/../../partials/nav.php");
3  // mmt 4/1/2024
```

```php
reset_session();
?>
<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>
<script>
```

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
■ #1	1	Show the JavaScript validations (include any extra files related)
☐ #2	1	Show the PHP validations (include any lib content)
■ #3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```javascript
function validate(form) {
    //TODO 1: implement JavaScript validation
    //ensure it returns false for an error and true for success
    // mmt 4/1/2024

    var Email = form.email.value;
    var Username = form.username.value;
    var Password = form.password.value;
    var Confirm = form.confirm.value;
```

```
var hasError = false;

function validUsername(Username) {
    const usernameRegex =/^[a-z0-9_-]{3,16}$/;
    return usernameRegex.test(Username);
}

function validEmail(Email) {
    const emailRegex = /^([a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6})*$/;
    return emailRegex.test(Email);
}

if (Email === "") {
    flash("[client] Email cannot be empty", "caution");
    hasError = true;
}
```

JS validation (first part)

Checklist Items (0)

```
public_html > Project > register.php > script > validate
25  <script>
26      function validate(form) {
51          else if (!validEmail(Email)) {
52              flash("[client] Email is invalid", "caution");
53              hasError = true;
54          }
55          if (Username === "") {
56              flash("[client] Username cannot be empty", "caution");
57              hasError = true;
58          }
59          else if (!validUsername(Username)) {
60              flash("[client] Username is invalid ", "caution");
61              hasError = true;
62          }
63          if (Password === "") {
64              flash("[client] Password cannot be empty", "caution");
65              hasError = true;
66          }
67          else if (Password.length < 8) {
68              flash("[client] Password is too short", "caution");
69              hasError = true;
70          }
71          if (Password !== Confirm) {
72              flash("[client] Passwords do not match", "caution");
73              hasError = true;      Manan Thakore, 6 hours ago • Milestone 1 changes & vali
74          }
75
76          return !hasError;
77          //mmt 4/1/2024
```

JS validation (second part)

Checklist Items (0)

```
81  //TODO 2: add PHP Code
82  if (isset($_POST["email"]) && isset($_POST["password"]) && isset($_POST["confirm"])) {
83      $email = se($_POST, "email", "", false);
84      $password = se($_POST, "password", "", false);
85      $confirm = se(
86          $_POST,
87          "confirm",
88          ""
```

```
        false
        // mmt 4/1/2024
    ); <- #83-88 $confirm = se
    $username = se($_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;
    }
```

php validation (first part)

Checklist Items (0)

```
    if (!is_valid_email($email)) {
        flash("Invalid email address", "danger");
        $hasError = true;
    }
    if (!preg_match('/^[a-z0-9_-]{3,16}$/', $username)) {
        flash("Username must only contain 3-30 characters a-z, 0-9, _, or -", "danger"); // mmt 4/1/2024
        $hasError = true;
    }
    if (empty($password)) {
        flash("password must not be empty", "danger");
        $hasError = true;
    }
    if (empty($confirm)) {
        flash("Confirm password must not be empty", "danger");
        $hasError = true;
    }
    if (strlen($password) < 8) {
        flash("Password too short", "danger");
        $hasError = true;
    }
    if (
        strlen($password) > 0 && $password !== $confirm
    ) {
        flash("Passwords must match", "danger");
        $hasError = true;
    }
    if (!$hasError) {
        //TODO 4
        $hash = password_hash($password, PASSWORD_BCRYPT);
```

php validation (second part)

Checklist Items (0)

```
    if (!$hasError) {
        //TODO 4 mmt 4/1/2024
```

```php
        $hash = password_hash($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 (PDOException $e) {
            users_check_duplicate($e->errorInfo);
        }
    } <- #125-136 if (!$hasError)
} <- #80-137 if (isset($_POST["email"]) && isset($_POST["password"]) && is...
?>
<?php
require(__DIR__ . "/../../partials/flash.php");
?>
```

php validation (third part)

Checklist Items (0)

COLLAPSE

Task #4 - Points: 1

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

Checklist

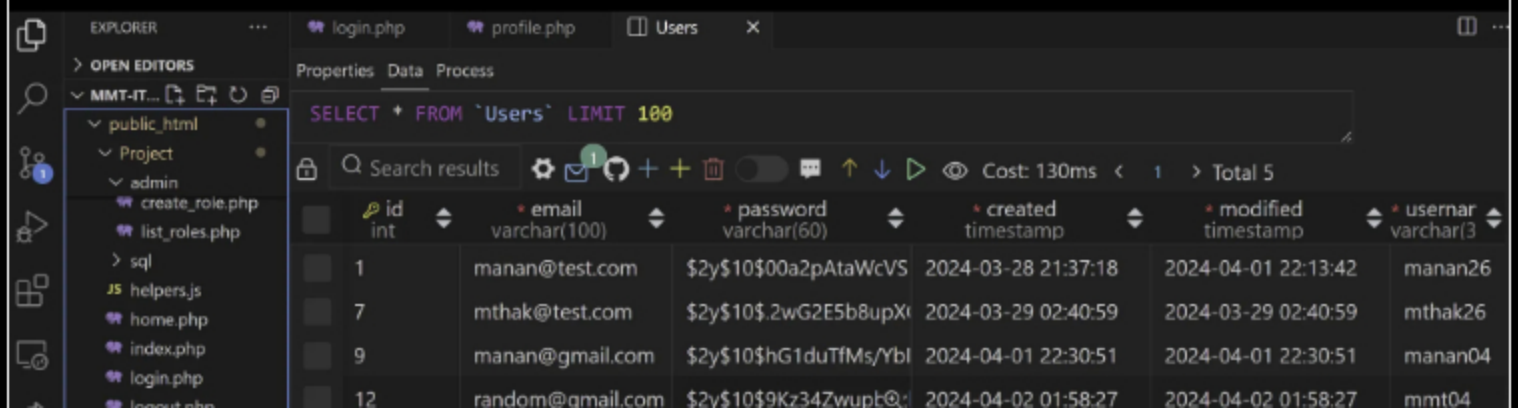
*The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



users table with all the requirements

Checklist Items (0)

COLLAPSE

Task #5 - Points: 1

Text: Explain the registration logic in a step-by-step manner from when the page loads to when the data is saved to the DB

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

Response:

The registration process begins with client-side validation through JavaScript, ensuring that all required fields are filled correctly, including email format, username format, password length, and password confirmation. After the form is submitted, server-side PHP validation is performed, where the data is sanitized and further validated, displaying appropriate error messages if necessary. If validation passes, the password is hashed using bcrypt, and the user's data is inserted into the database. Success or error messages are displayed using flash messages.

COLLAPSE

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/mananthakore/mmt-it202-008/pull/34

User Login (2 pts.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of form on website page

Checklist

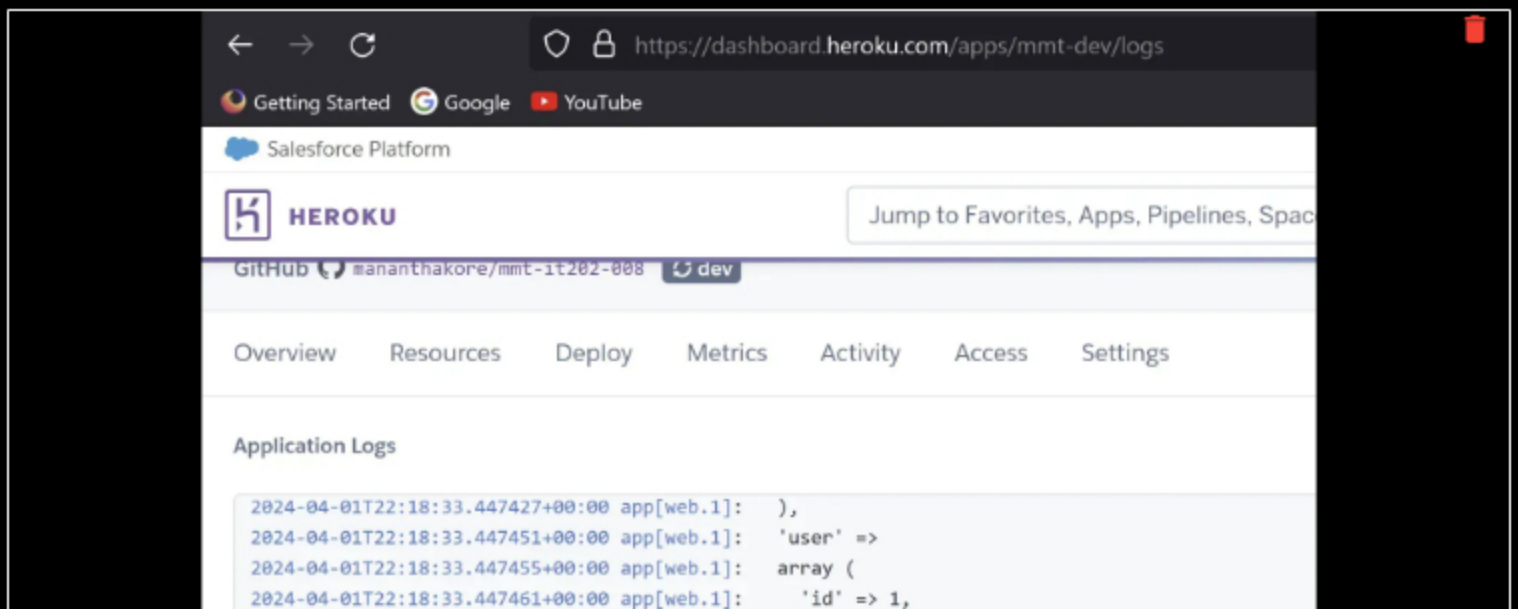
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Include correct data where username is used to login
#4	1	Include correct data where email is used to login
#5	1	Show success login message
#6	1	Demonstrate JavaScript validation for each field [can be combined] (username format, email format, password format, and each field being required)
#7	1	Demonstrate user-friendly message of when an account doesn't exist
#8	1	Demonstrate user-friendly message of when password doesn't match what's in the DB
#9	1	Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)
#10	1	Demonstrate session data being set (captured from server logs)

Task Screenshots:

Gallery Style: Large View

Small Medium Large

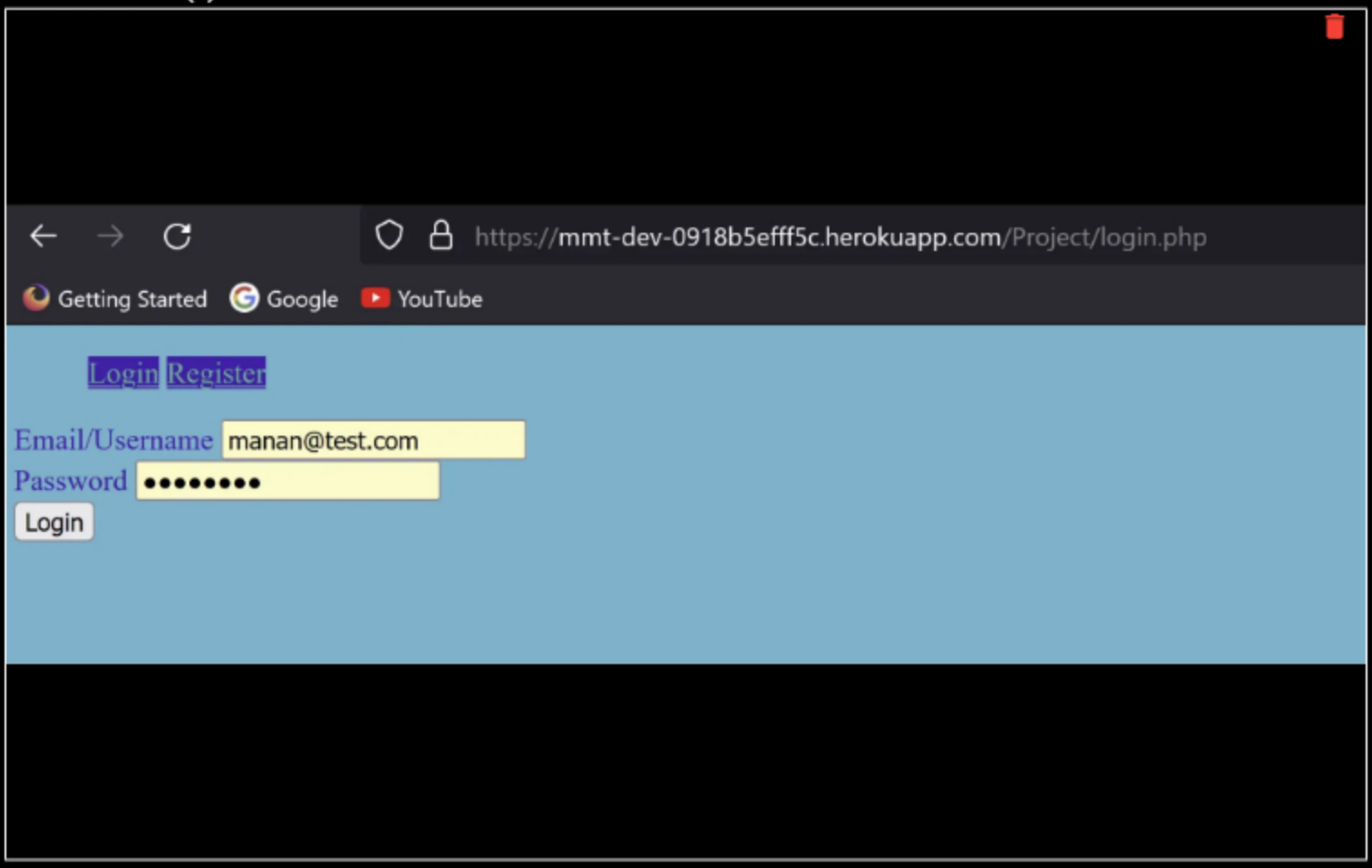


```
2024-04-01T22:18:33.447478+00:00 app[web.1]:     'email' => 'manan@test.com',
2024-04-01T22:18:33.447491+00:00 app[web.1]:     'username' => 'manan26',
2024-04-01T22:18:33.447500+00:00 app[web.1]:     'roles' =>
2024-04-01T22:18:33.447502+00:00 app[web.1]:     array (
2024-04-01T22:18:33.447516+00:00 app[web.1]:       0 =>
2024-04-01T22:18:33.447517+00:00 app[web.1]:       array (
2024-04-01T22:18:33.447532+00:00 app[web.1]:         'name' => 'Admin',
2024-04-01T22:18:33.447533+00:00 app[web.1]:       ),
2024-04-01T22:18:33.447544+00:00 app[web.1]:     ),
```

Autoscroll with output

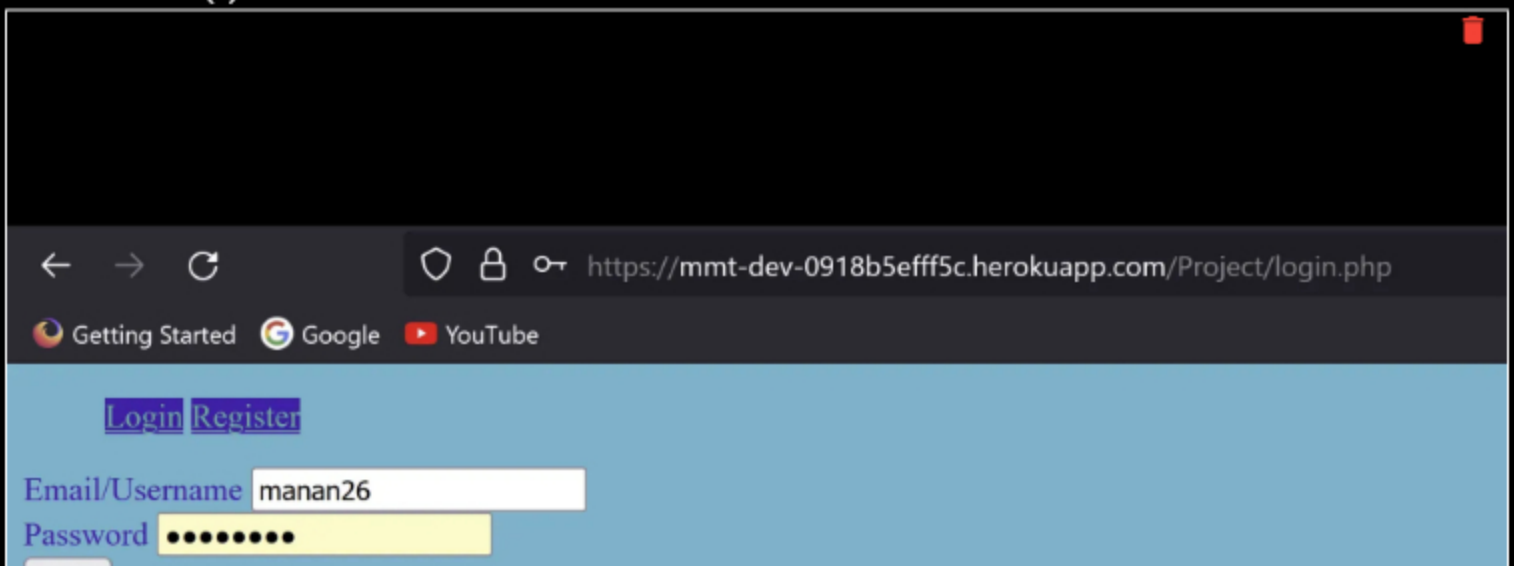
session data being set

Checklist Items (0)



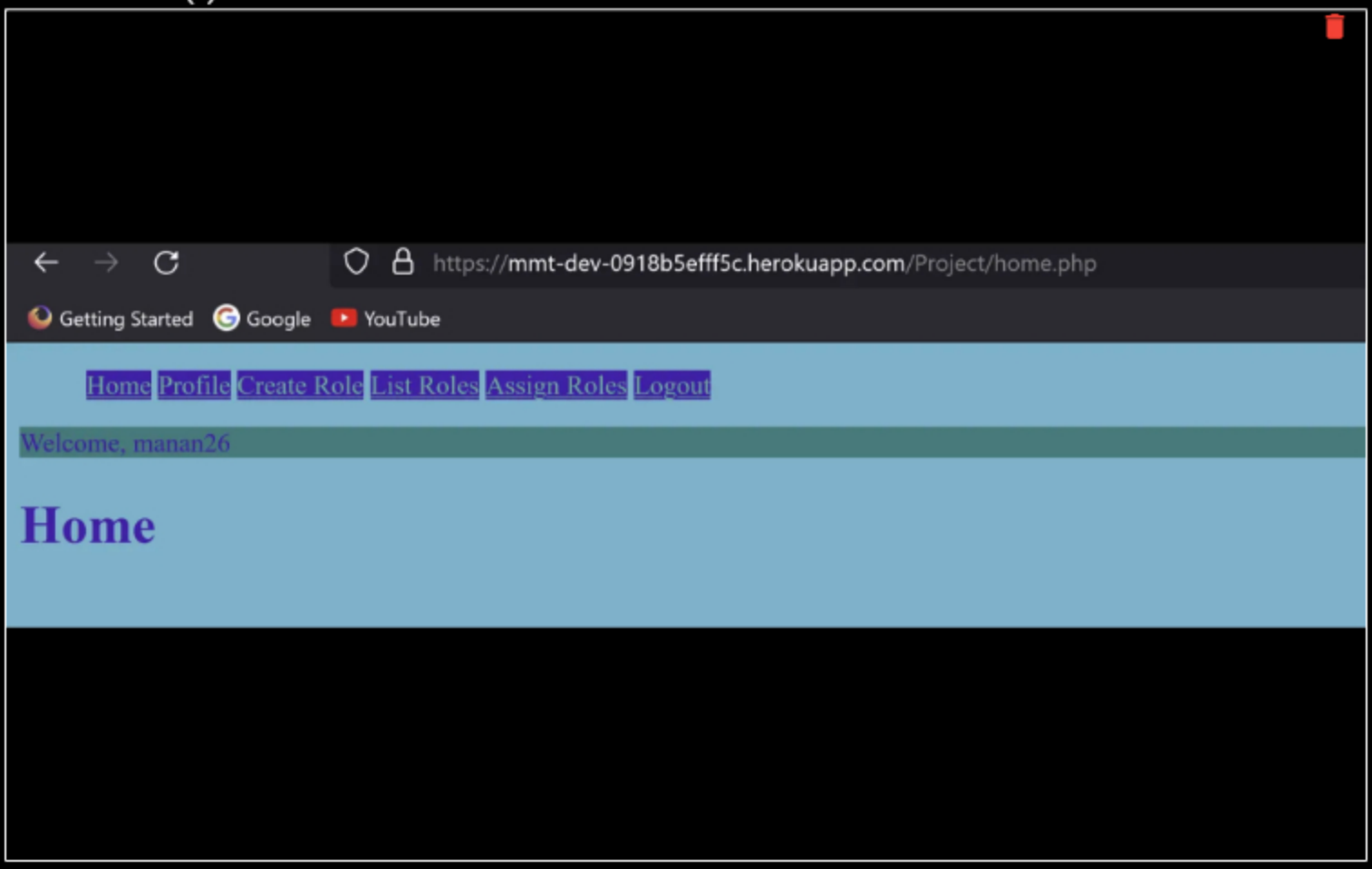
login page with email and password filled

Checklist Items (0)



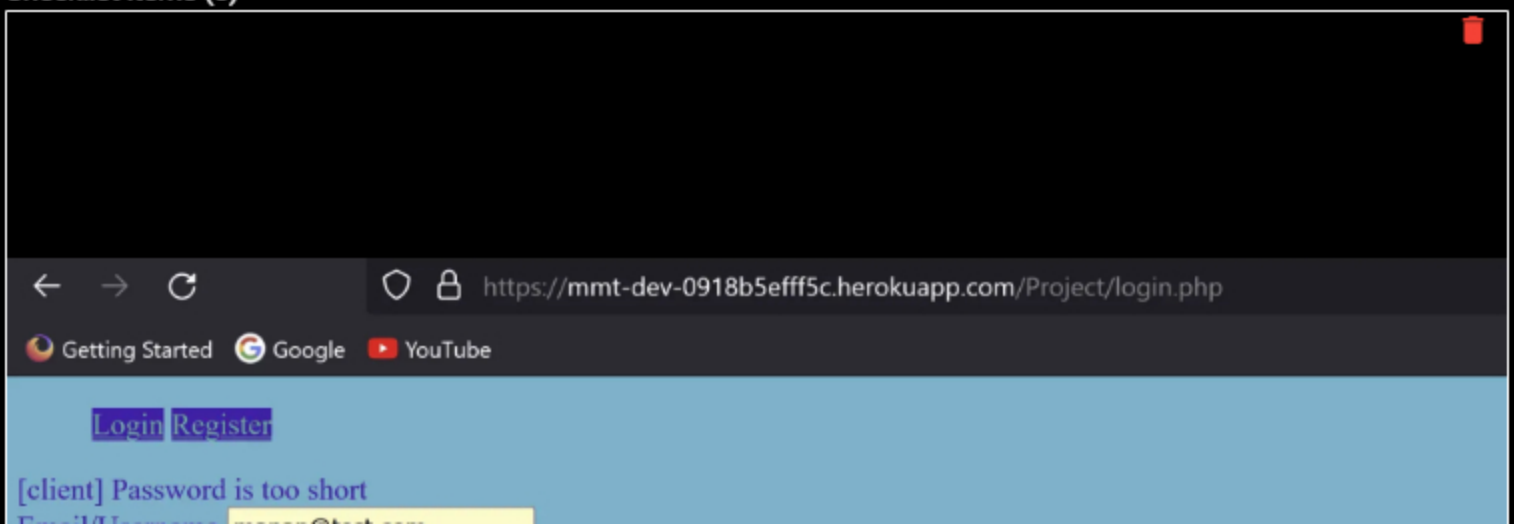
login page with username and password filled

Checklist Items (0)



success page after logging in

Checklist Items (0)



Email/Username manan@test.com
Password 123
Login

password is too short (JS valid)

Checklist Items (0)

https://mmt-dev-0918b5efff5c.herokuapp.com/Project/login.php
Getting Started Google YouTube

Login Register

[client] Password is too short
[client] Password cannot be empty
Email/Username manan@test.com
Password
Login

password cannot be empty (JS valid)

Checklist Items (0)

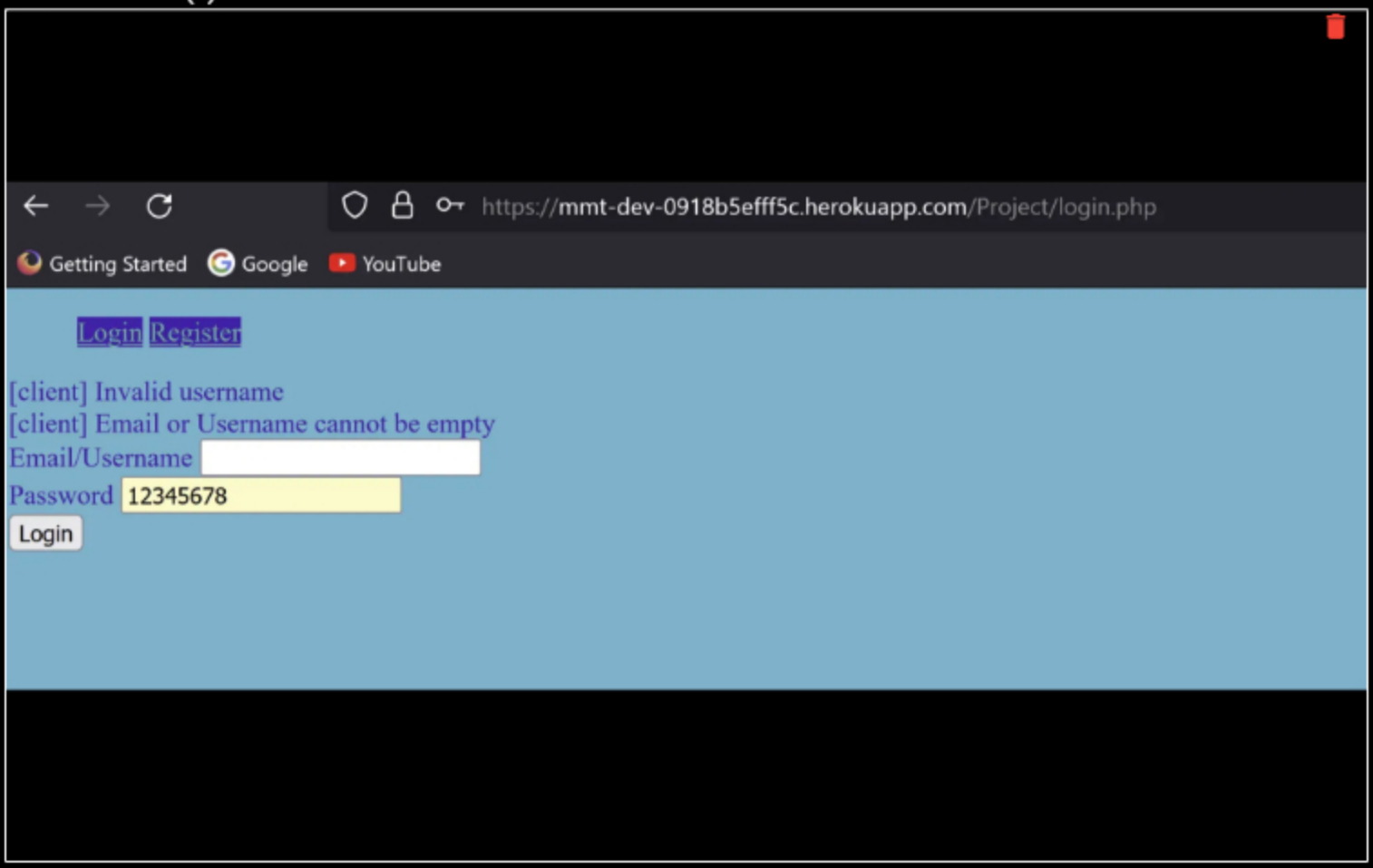
https://mmt-dev-0918b5efff5c.herokuapp.com/Project/login.php
Getting Started Google YouTube

Login Register

[client] Invalid email
Email/Username manan@test
Password 12345678
Login

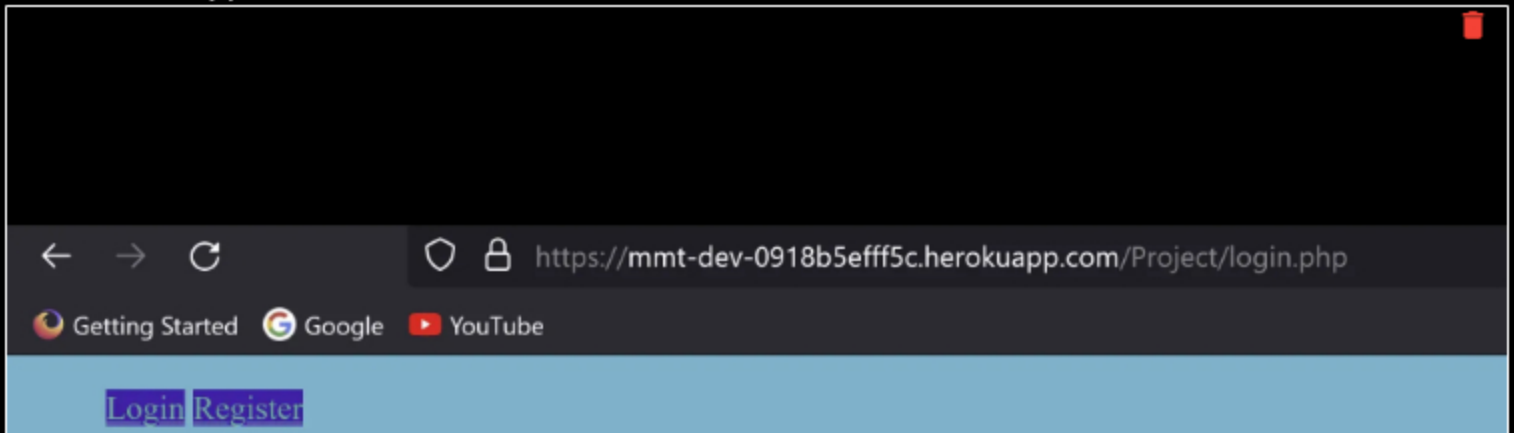
invalid email (JS valid)

Checklist Items (0)



invalid username and username cannot be empty (JS valid)

Checklist Items (0)



Email not found

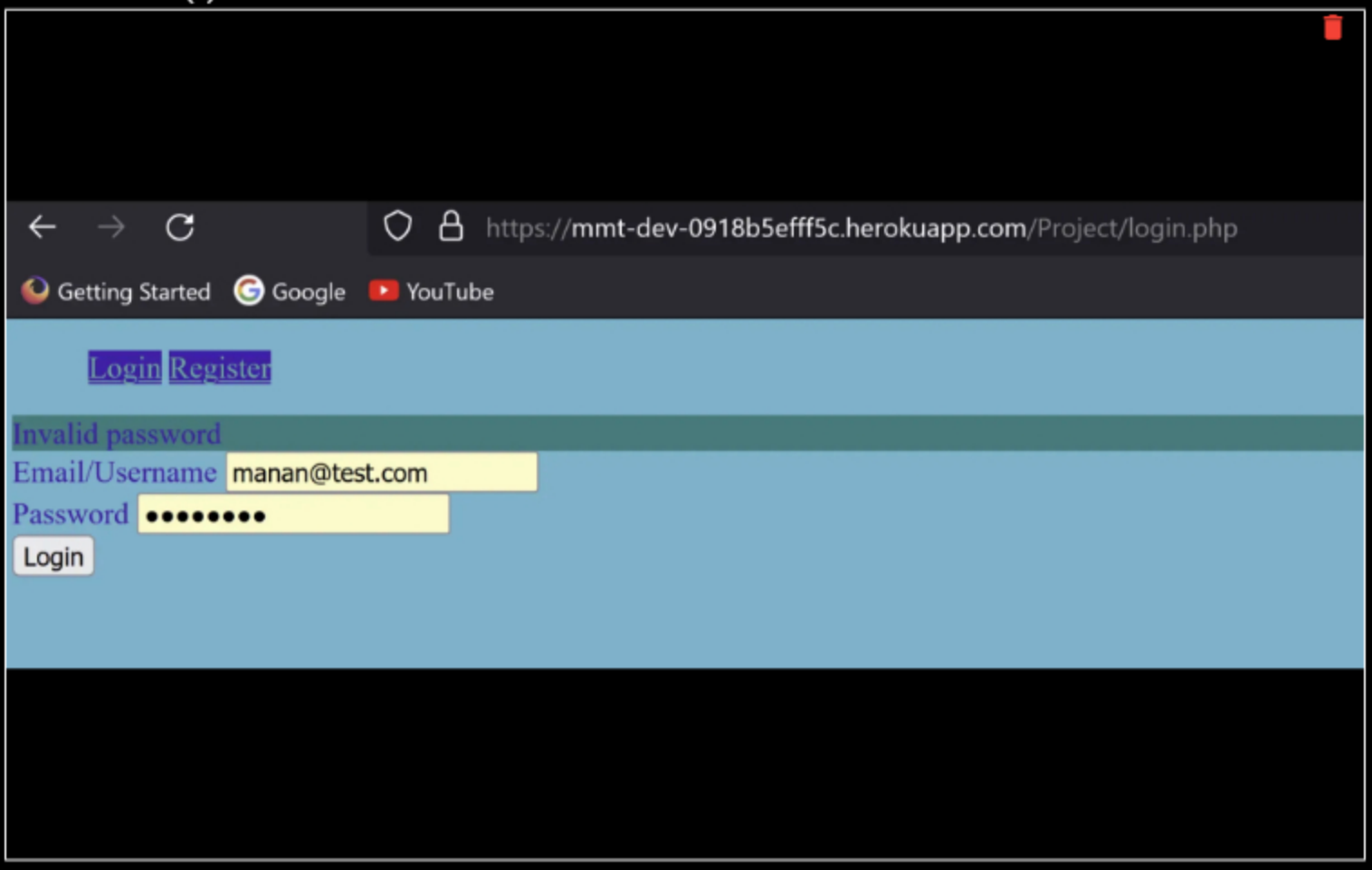
Email/Username manan@test.com

Password ••••••••

Login

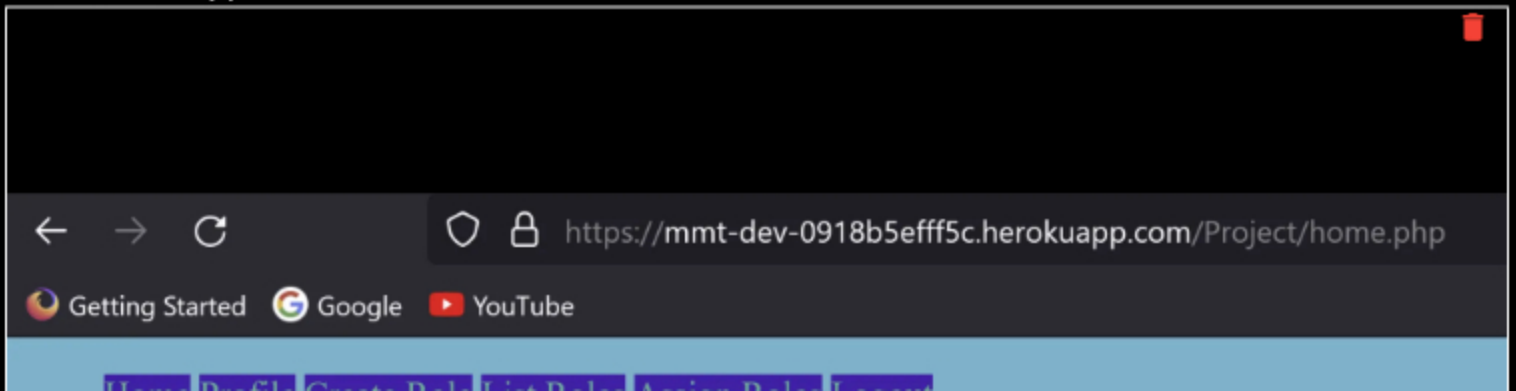
account doesnt exist

Checklist Items (0)



invalid password

Checklist Items (0)



Home Profile Create Role List Roles Assign Roles Logout

Welcome, manan26

Home

successful login page

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the form code

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Should have proper input types for the fields (note in the caption if you're using two fields for Username/Email or a combined field)
#2	1	Show JavaScript validations (include any extra files related)
#3	1	Show PHP validations (include any lib content)
#4	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
// mmt 4/1/2024
<?php
require(__DIR__ . "/../../partials/nav.php");
?>
<form onsubmit="return validate(this)" method="POST">
    <div>
        <label for="email">Email/Username</label>
        <input type="text" name="email" required />
```

```
    </div>      Manan Thakore, 5 days ago • login part1
    <div>
        <label for="pw">Password</label>
        <input type="password" id="pw" name="password" required minlength="8" />
    </div>
    <input type="submit" value="Login" />
</form>
<script>
```

form code

Checklist Items (0)

```
function validate(form) {
    // mmt 4/1/2024      You, 1 second ago • Uncommitted changes
    //TODO 1: implement JavaScript validation
    //ensure it returns false for an error and true for success

    let email = form.email.value;
    let password = form.password.value;
    let isValid = true;

    function validEmail(email) {
        const emailRegEx = /^([a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6})*$/;
        return emailRegEx.test(email);
    }

    function validUsername(username) {
        const usernameRegEx = /^[a-z0-9_-]{3,16}$/;
        return usernameRegEx.test(username);
    }


    if(password.length < 8) {
        flash("[client] Password is too short", "caution");
        isValid = false;
    }
```

js validation (first part)

Checklist Items (0)

```
    // mmt 4/1/2024
    if(password === "") {
        flash("[client] Password cannot be empty", "caution");
        isValid = false;
    }

    if(email.includes("@")) {
        if(!validEmail(email)) {
            flash("[client] Invalid email", "caution");
            isValid = false;
        }
```

```
} <- #48-53 if(email.includes("@"))
else {
    if(!validUsername(email)) {
        flash("[client] Invalid username", "caution");
        isValid = false;
    }
} <- #54-59 else

if (email === "") {
    flash("[client] Email or Username cannot be empty", "caution");
    isValid = false;
}
return isValid;
```

js validation part 2

Checklist Items (0)

```
<?php
//TODO 2: add PHP Code
// mmt 4/1/2024
if (isset($_POST["email"]) && isset($_POST["password"])) {
    $email = se($_POST, "email", "", false);
    $password = se($_POST, "password", "", false);

    //TODO 3
    $hasError = false;
    if (empty($email)) {
        flash("Email must not be empty");
        $hasError = true;
    }
    if (str_contains($email, "@")) {
        //sanitize
        $email = sanitize_email($email);
        //validate
        if (!is_valid_email($email)) {
            flash("Invalid email address");
            $hasError = true;
        }
    } else {
        if (!is_valid_username($email)) {
            flash("Invalid username");
            $hasError = true;
        }
    } <- #88-93 else
    if (empty($password)) {
        flash("password must not be empty");
```

php validation (first part)

Checklist Items (0)

```
// mmt 4/1/2024 <- #88-93 else
if (empty($password)) {
    flash("password must not be empty");      Manan Thakore, 3 days ago • housek
    $hasError = true;
}
if (!is_valid_password($password)) {
    flash("Password too short");
    $hasError = true;
}
if (!$hasError) {
    //flash("Welcome, $email");
```

```php
//TODO 4
$db = getDB();
$stmt = $db->prepare("SELECT id, email, username, password from Users
where email = :email or username = :email");
try {
    $r = $stmt->execute([":email" => $email]);
    if ($r) {
        $user = $stmt->fetch(PDO::FETCH_ASSOC);
        if ($user) {
            $hash = $user["password"];
            unset($user["password"]);
            if (password_verify($password, $hash)) {
                //flash("Weclome $email");
                $_SESSION["user"] = $user;
                try {
                    //Lookup potential roles
                    $stmt = $db->prepare("SELECT Roles.name FROM Roles
```

php validation (second part)

Checklist Items (0)

```php
                    // mmt 4/1/2024
                    JOIN UserRoles on Roles.id = UserRoles.role_id        Manan Thakore, 2 days ago •
                    where UserRoles.user_id = :user_id and Roles.is_active = 1 and UserRoles.is_activ
                    $stmt->execute([":user_id" => $user["id"]]);
                    $roles = $stmt->fetchAll(PDO::FETCH_ASSOC); //fetch all since we'll want mult
                } catch (Exception $e) {
                    error_log(var_export($e, true));
                }
                //save roles or empty array
                if (isset($roles)) {
                    $_SESSION["user"]["roles"] = $roles; //at least 1 role
                } else {
                    $_SESSION["user"]["roles"] = []; //no roles
                } //sets our session data from db
                flash("Welcome, " . get_username());
                die(header("Location: home.php"));
            } else {
                flash("Invalid password");
            }
        } else {
            flash("Email not found");
        }
    } <- #114-145 if ($user)
} catch (Exception $e) {
    flash("<pre>" . var_export($e, true) . "</pre>");
}
- #110-149 try
```

php validation (third part)

Checklist Items (0)

```
144                 }
145             } <- #114-145 if ($user)
146         } catch (Exception $e) {
```

```
147         flash("<pre>" . var_export($e, true) . "</pre>");
148         }       Manan Thakore, 5 days ago • login part1
149     } <- #110-149 try
150     // mmt 4/1/2024
151 } <- #108-151 $stmt = $db->prepare
152 ?>
153 <?php
154 require(__DIR__."/../../partials/flash.php"); <- #71-154 if (isset($_POST["email"
```

php validation (fourth part)

Checklist Items (0)

COLLAPSE

Task #3 - Points: 1

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

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Don't just show code, translate things to plain English
#2	1	Explain how the session works and why/how it's used

Response:

The login process begins with client-side validation through JavaScript, where the form data is checked for correctness, including password length and email or username format. Upon form submission, server-side PHP validation takes place, ensuring that the email or username is not empty and meets the required format, and that the password meets length criteria. If validation passes, the PHP code interacts with the database to fetch user data based on the provided credentials. If the user is found and the password matches, session variables are set to store user information, including roles fetched from the database. Flash messages are utilized throughout to provide feedback on the login attempt. Finally, upon successful authentication, the user is redirected to the home page

COLLAPSE

Task #4 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1
https://github.com/mananthakore/mmt-it202-008/pull/36

User Logout (1 pt.)
COLLAPSE

COLLAPSE

Task #1 - Points: 1
Text: Capture the following screenshots

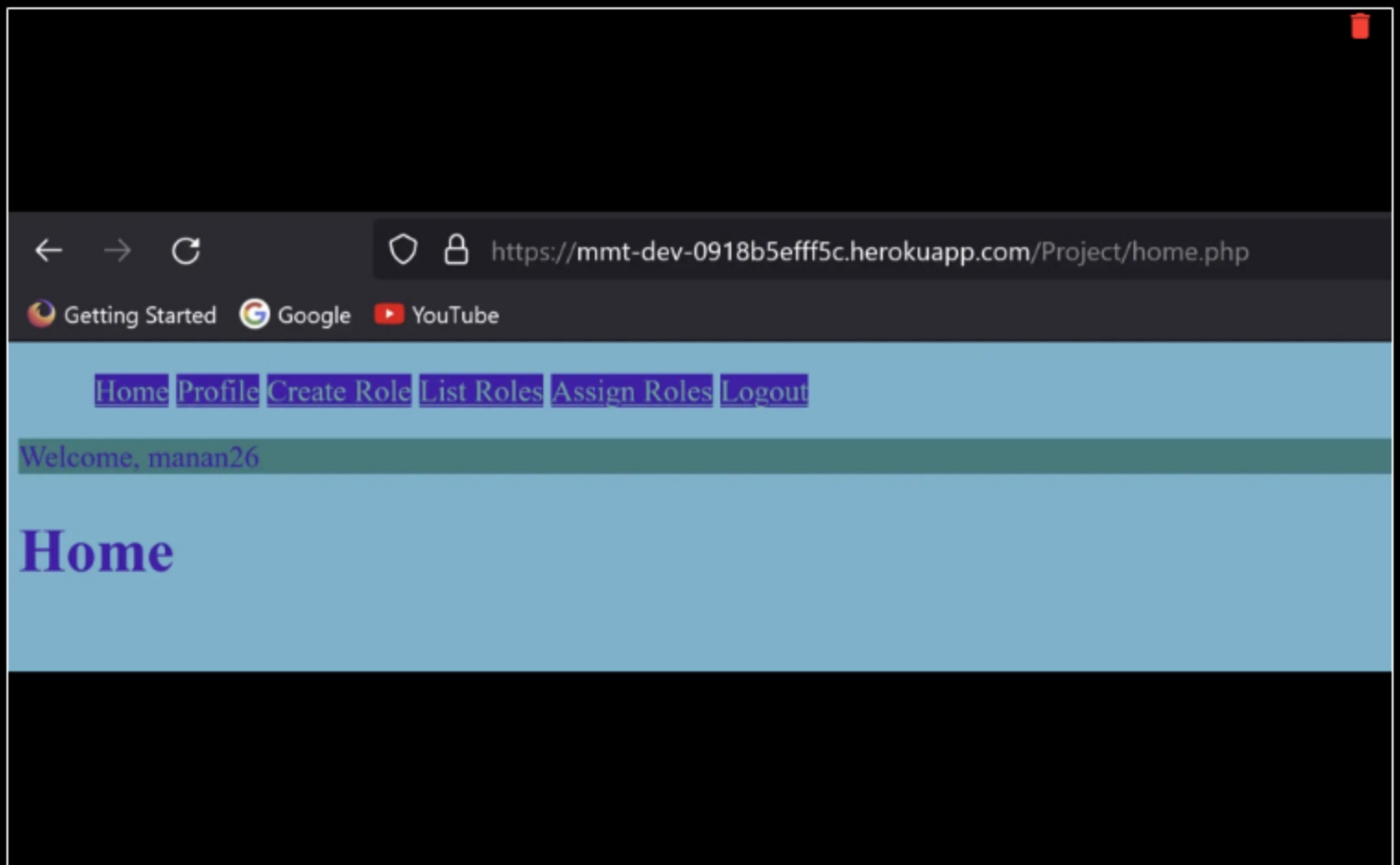
Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Screenshot of the navigation when logged in (site)
#2	1	Screenshot of the redirect to login with the user-friendly logged-out message (site)
#3	1	Screenshot of the logout-related code showing the session is destroyed (code). Ensure ucid/date comment is present.

Task Screenshots:

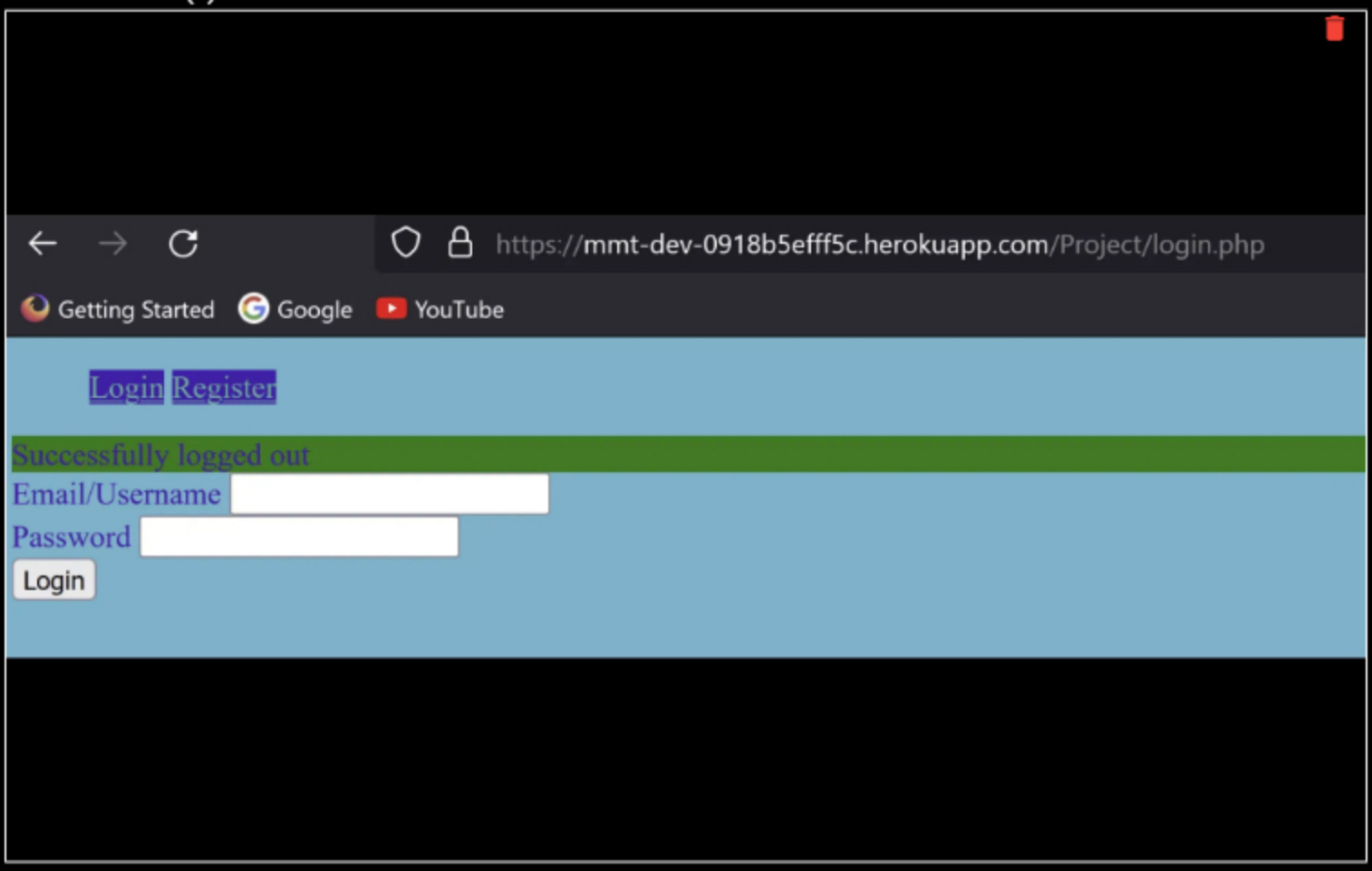
Gallery Style: Large View

Small Medium Large



screenshot of nav when logged in

Checklist Items (0)



screenshot of the redirect (with user-friendly message)

Checklist Items (0)

```php
// mmt 4/1/2024
<?php
session_start();
require(__DIR__ . "/../../lib/functions.php");
reset_session();
    Manan Thakore, 3 days ago • housekeeping stuff
flash("Successfully logged out", "success");
header("Location: login.php");
```

screenshot of logout code

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/mananthakore/mmt-it202-008/pull/34

Basic Security Rules and Roles (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Authentication Screenshots

Checklist

*The checkboxes are for your own tracking

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large

```
Manan Thakore, 3 hours ago | 1 author (Manan Thakore)
<?php
require_once(__DIR__ . "/../../partials/nav.php");
is_logged_in(true);        Manan Thakore, 3 days ago • housekeeping stuff
?>
<?php
if (isset($_POST["s
```

Manan Thakore, 3 days ago *(March 29th, 2024 11:45 AM)*

is_logged_in function being used in profile.php

Checklist Items (0)

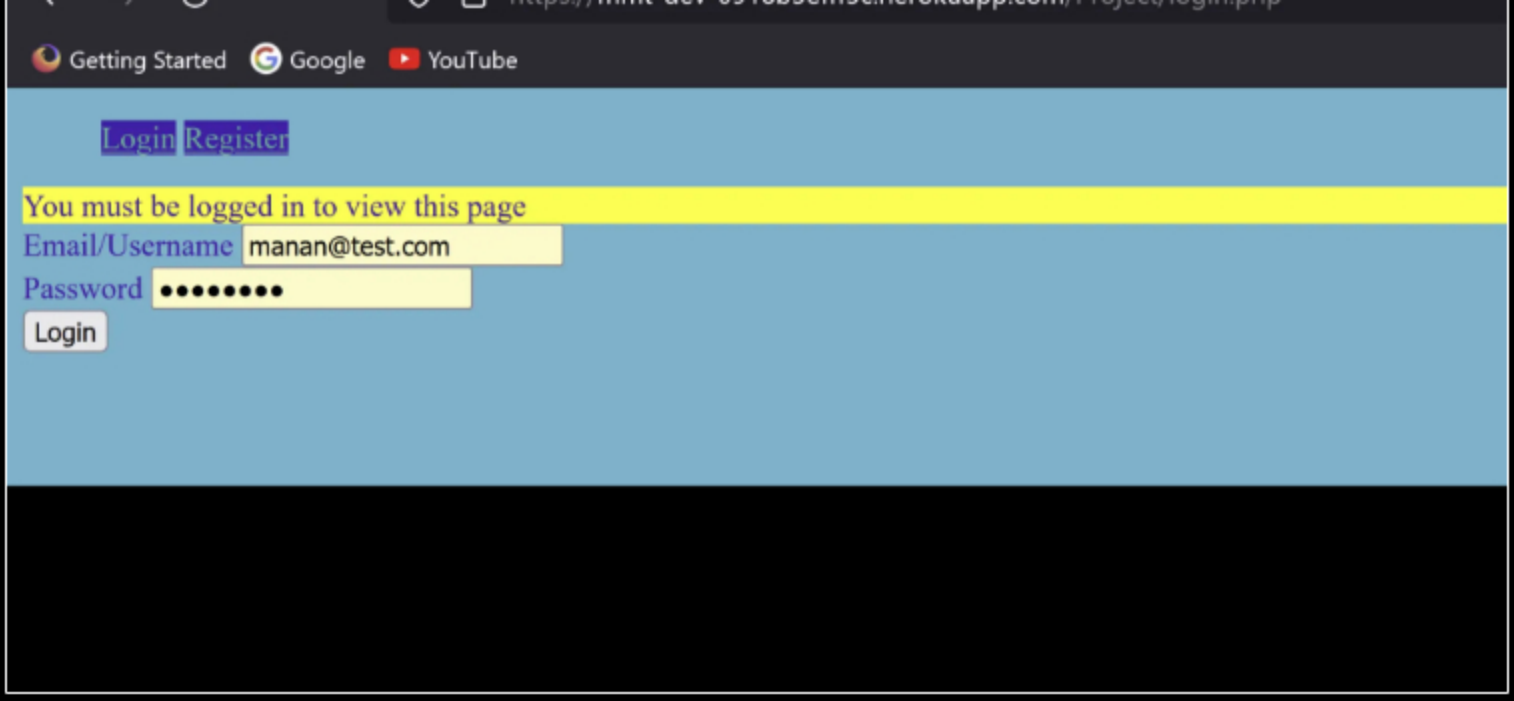
```php
<?php

/**
 * Passing $redirect as true will auto redirect a logged out user to the $destination.
 * The destination defaults to login.php
 * mmt 4/1/2024       You, 8 seconds ago • Uncommitted changes
 */
function is_logged_in($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() termin
        flash("You must be logged in to view this page", "warning");
        die(header("Location: $destination"));
    } <- #11-15 if ($redirect && !$isLoggedIn)
    return $isLoggedIn;
} <- #9-17 function is_logged_in($redirect = false, $destination = "logi...
```

is_logged_in function

Checklist Items (0)

https://mmt-dev-0918b5efff5c.herokuapp.com/Project/login.php



User-friendly message of trying to manually access a login-protected page while being logged out

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Authorization Screenshots

Checklist

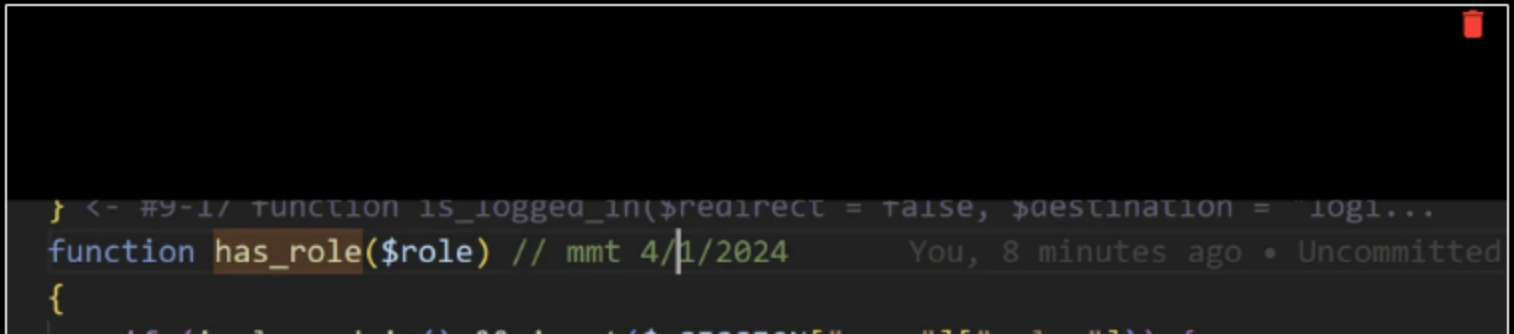
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Screenshot of the function that checks for a specific role
#2	1	Screenshot of the role check function being used. Also caption what pages it's used on
#3	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)
#4	1	Demonstrate the user-friendly message of trying to manually access a role-protected page while being logged out

Task Screenshots:

Gallery Style: Large View

Small Medium Large



```php
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {
        foreach ($_SESSION["user"]["roles"] as $r) {
            if ($r["name"] === $role) {
                return true;
            }
        } <- #21-25 foreach ($_SESSION["user"]["roles"] as $r)
    } <- #20-26 if (is_logged_in() && isset($_SESSION["user"]["roles"]))
    return false;
} <- #19-28 function has_role($role)
```

has_role function

Checklist Items (0)

```php
...
<?php
//note we need to go up 1 more directory
require(__DIR__ . "/../../../partials/nav.php");
// mmt 4/1/2024
if (!has_role("Admin")) {
    flash("You don't have permission to view this page", "warning");
    die(header("Location: $BASE_PATH" . "/home.php"));
}
```

has_role in assign_roles.php file

Checklist Items (0)

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

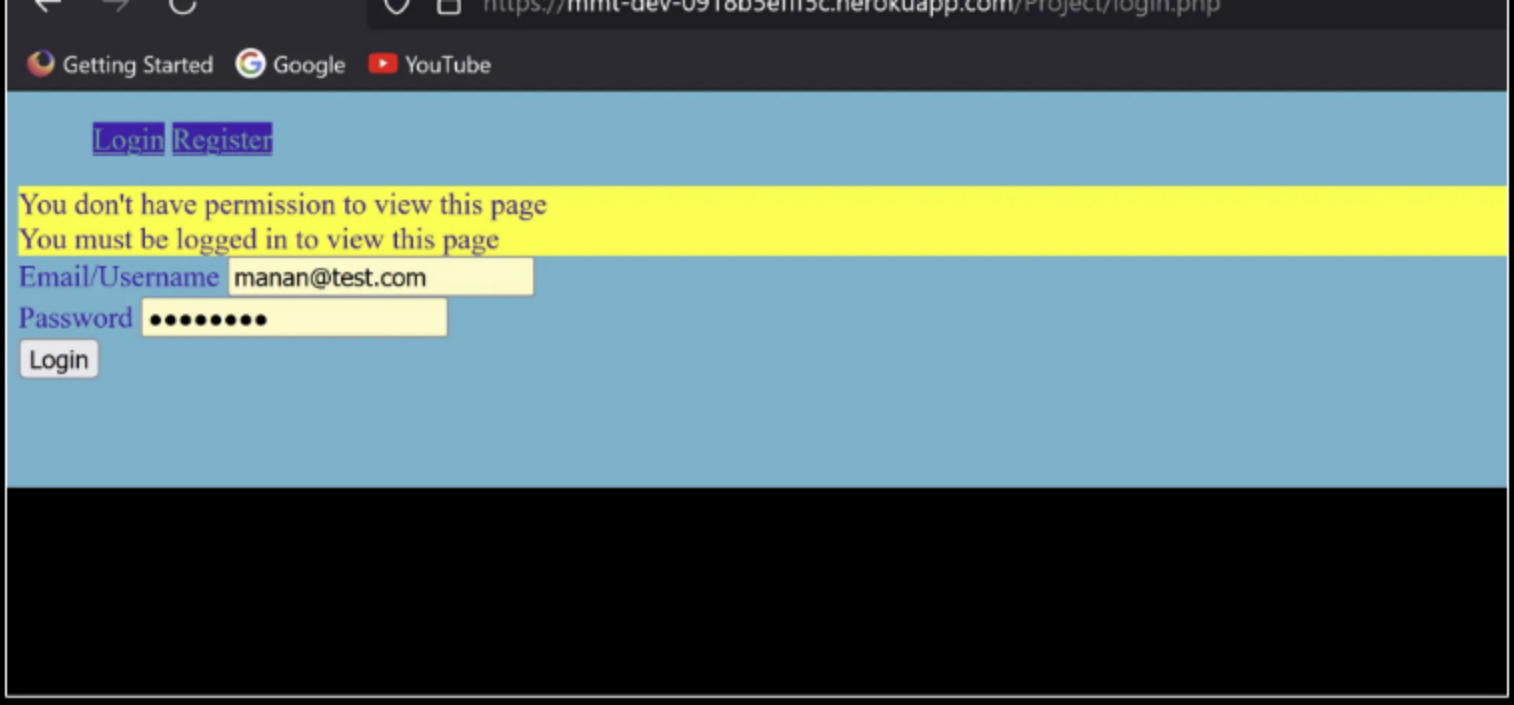
has_role in create_role.php

Checklist Items (0)

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

has_role function in list_roles.php

Checklist Items (0)



error message showing you need permission to access admin

Checklist Items (0)

COLLAPSE

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

Checklist

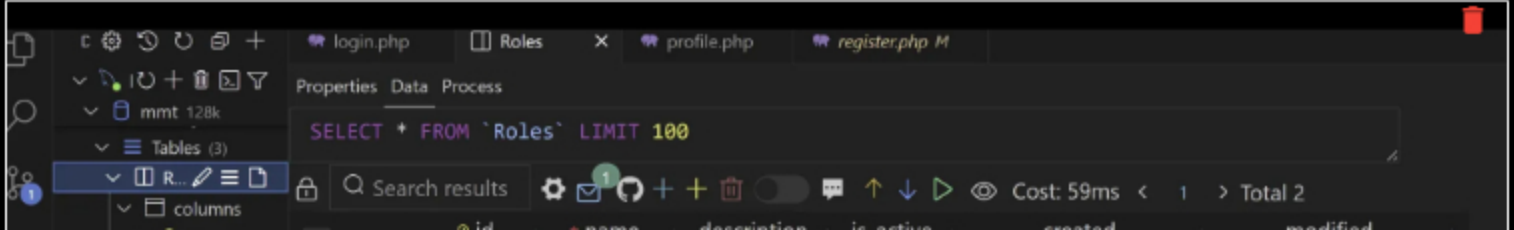
*The checkboxes are for your own tracking

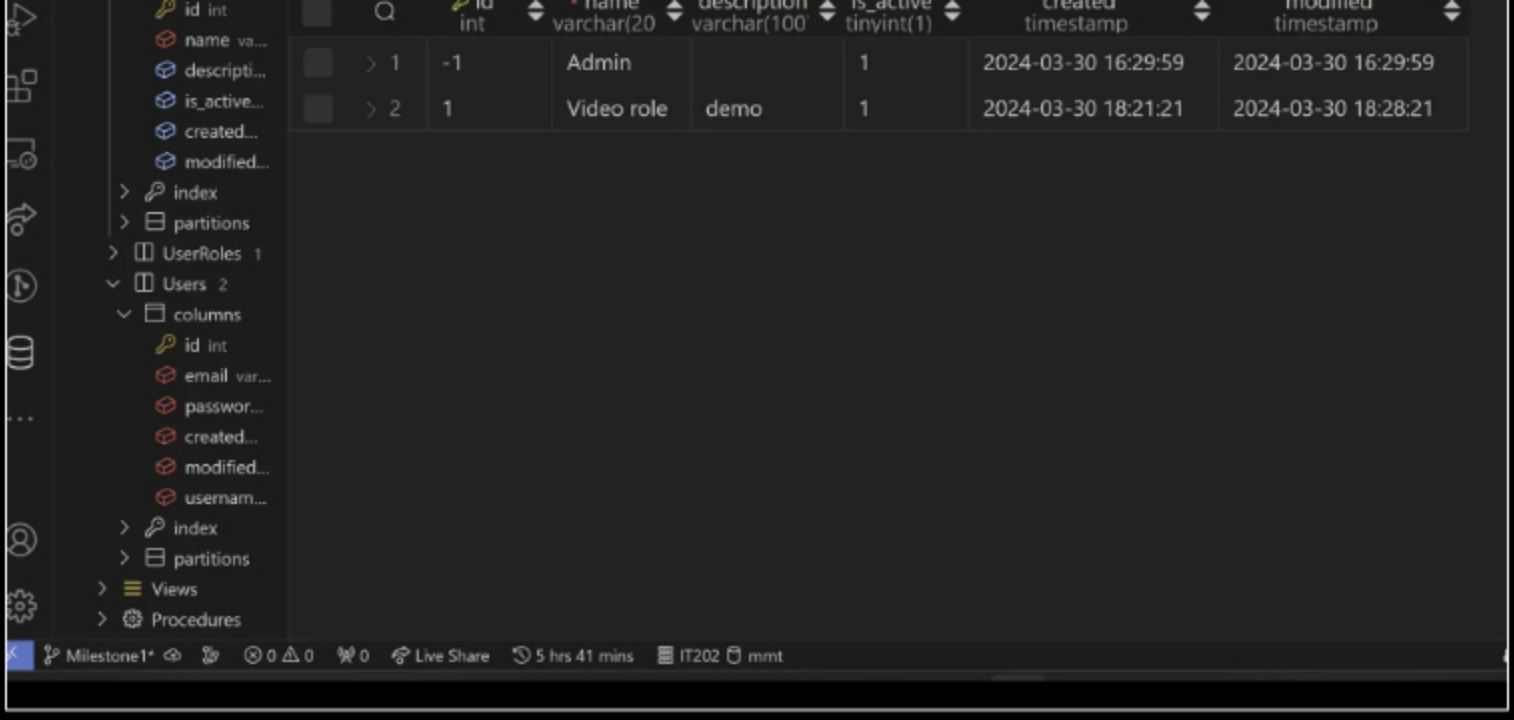
#	Points	Details
#1	1	At least one valid and enabled User->Role reference (UserRoles table)
#2	1	UserRoles Table should have id, user_id, role_id, is_active, created, and modified columns
#3	1	Roles Table should have id, name, description, is_active, modified, and created columns
#4	1	At least one valid and enabled Role (Roles table)
#5	1	Ensure left panel or database name is present in each table screenshot (should contain your ucid)

Task Screenshots:

Gallery Style: Large View

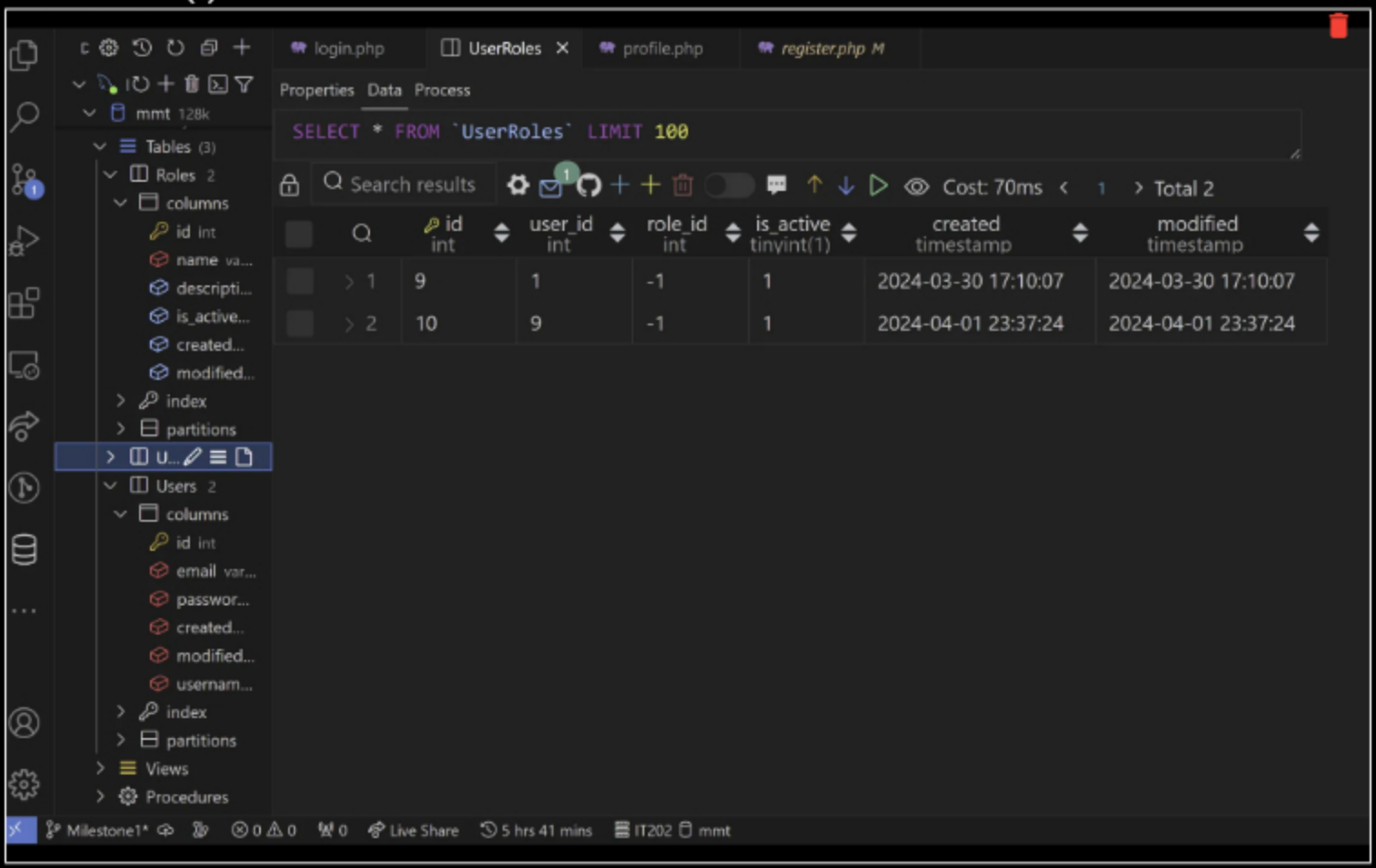
Small Medium Large





Roles table

Checklist Items (0)



UserRoles table

Checklist Items (0)

COLLAPSE

Task #4 - Points: 1

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

Checklist *The checkboxes are for your own tracking

#	Points	Details
■ #1	1	What's the purpose of the UserRoles table?
☐ #2	1	How does Roles.is_active differ from UserRoles.is_active?

Response:

All user-related data relating to accounts that have been registered is kept in the Users table, which mainly has fields for user ID, email address/username, and password. The roles table includes a default admin role that allows access to restricted areas (which are stored in the Roles table along with their names, IDs, descriptions, and activation status). The UserRoles database serves as a middleman in the interaction between Users and Roles, assigning and keeping track of particular roles for particular users. The Roles.is_active determines whether a particular role is currently active or not. The UserRoles.is_active is a flag within the UserRoles table, which shows the active status of the relationship between a user and a role.

● ^COLLAPSE^

Task #5 - Points: 1

Text: Include pull request links related to this feature

ⓘ Details:
Should end in /pull/#

URL #1
https://github.com/mananthakore/mmt-it202-008/pull/35

● User Profile (2 pts.)
^COLLAPSE^

● ^COLLAPSE^

Task #1 - Points: 1

Text: View Profile Website Page

Checklist *The checkboxes are for your own tracking

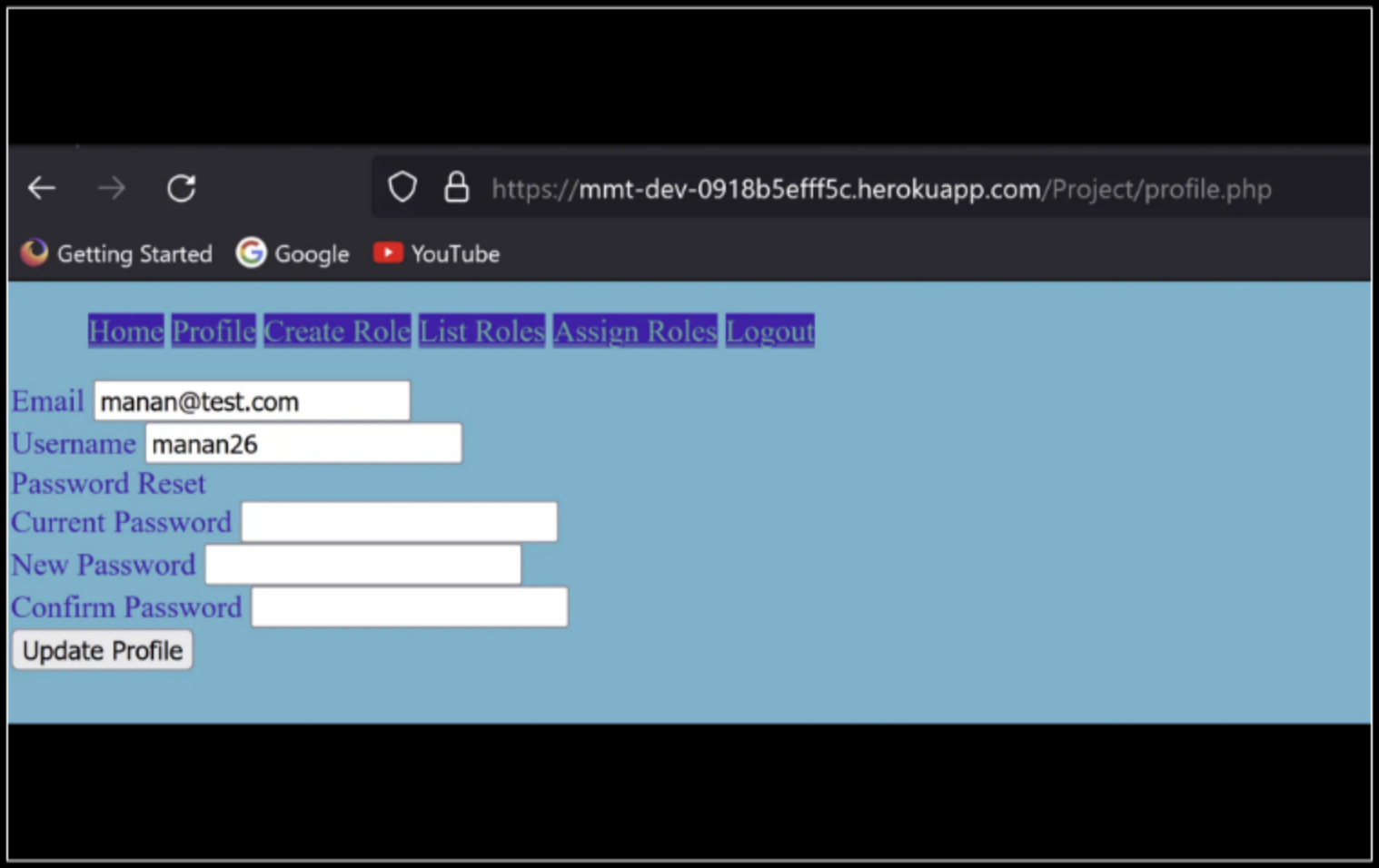
#	Points	Details
■ #1	1	Heroku dev url should be present in the address bar
■ #2	1	Should have thoughtful CSS applied
■ #3	1	Show the profile form correctly populated on page load (username, email)

#4	1	Should have the following fields: username, email, current password, new password, confirm password (or similar)

Task Screenshots:

Gallery Style: Large View

Small Medium Large



profile page

Checklist Items (0)

COLLAPSE

Task #2 - Points: 1

Text: Explain the logic step-by-step of how the data is loaded and populated when the profile page is visited

Details:

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

Response:

After a user logs in, the $_SESSION variable holds their user data, which includes their username, email address, and roles that have been assigned to them. As a result, PHP uses the get_user_email and get_username functions to retrieve user data from the $_SESSION variable whenever a user tries to access the profile page. The necessary form fields are then automatically filled in using the information that was collected.

COLLAPSE

Task #3 - Points: 1

Text: Edit Profile Website Page

Checklist

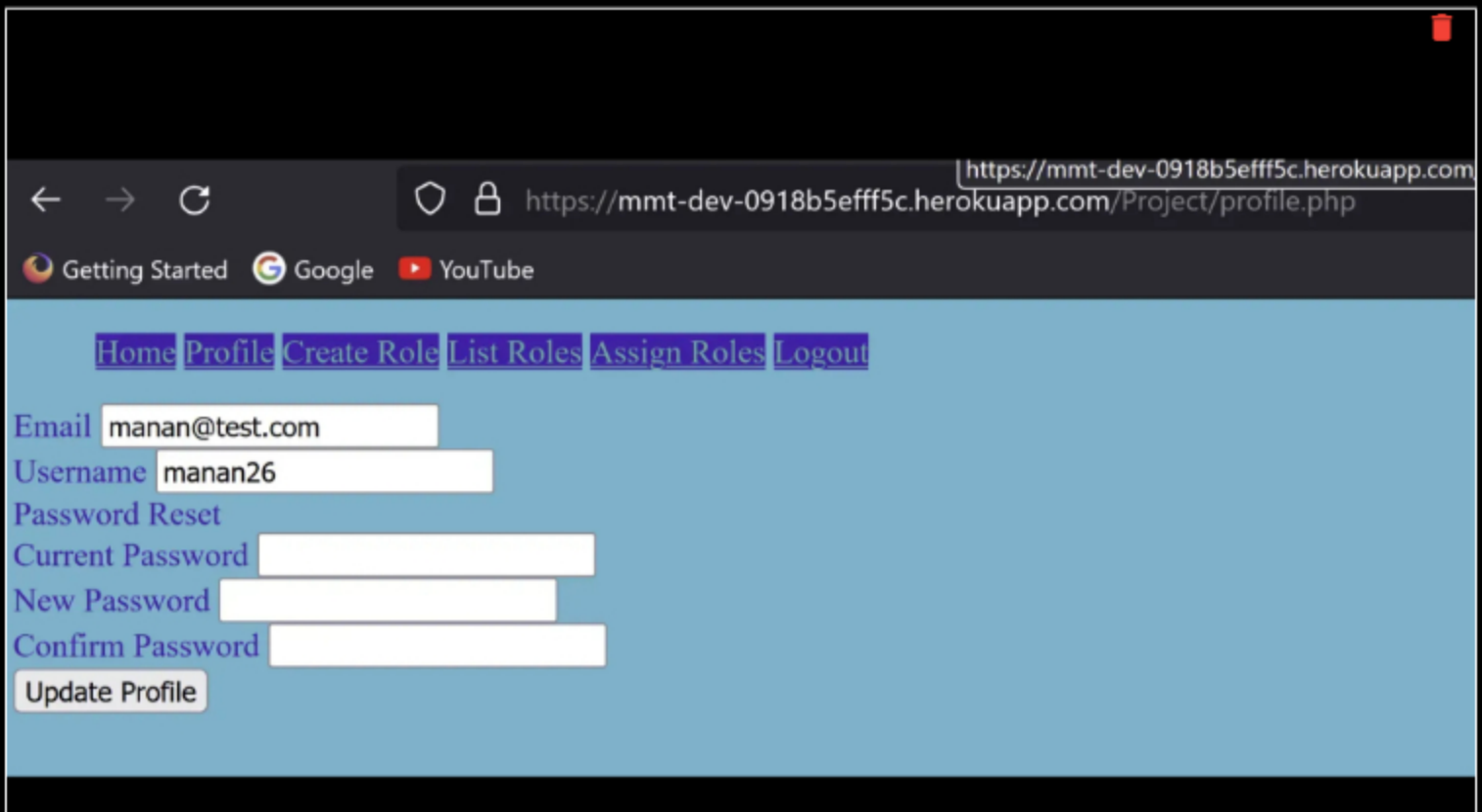
*The checkboxes are for your own tracking

#	Points	Details
#1	1	Heroku dev url should be present in the address bar
#2	1	Should have thoughtful CSS applied
#3	1	Demonstrate with before and after of a username change (including success message)
#4	1	Demonstrate with a before and after of an email change (including success message)
#5	1	Demonstrate the success message of updating password
#6	1	Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)
#7	1	Demonstrate PHP user-friendly validation messages (desired email is already in use, desired username is already in use, current password doesn't match what's in the DB)

Task Screenshots:

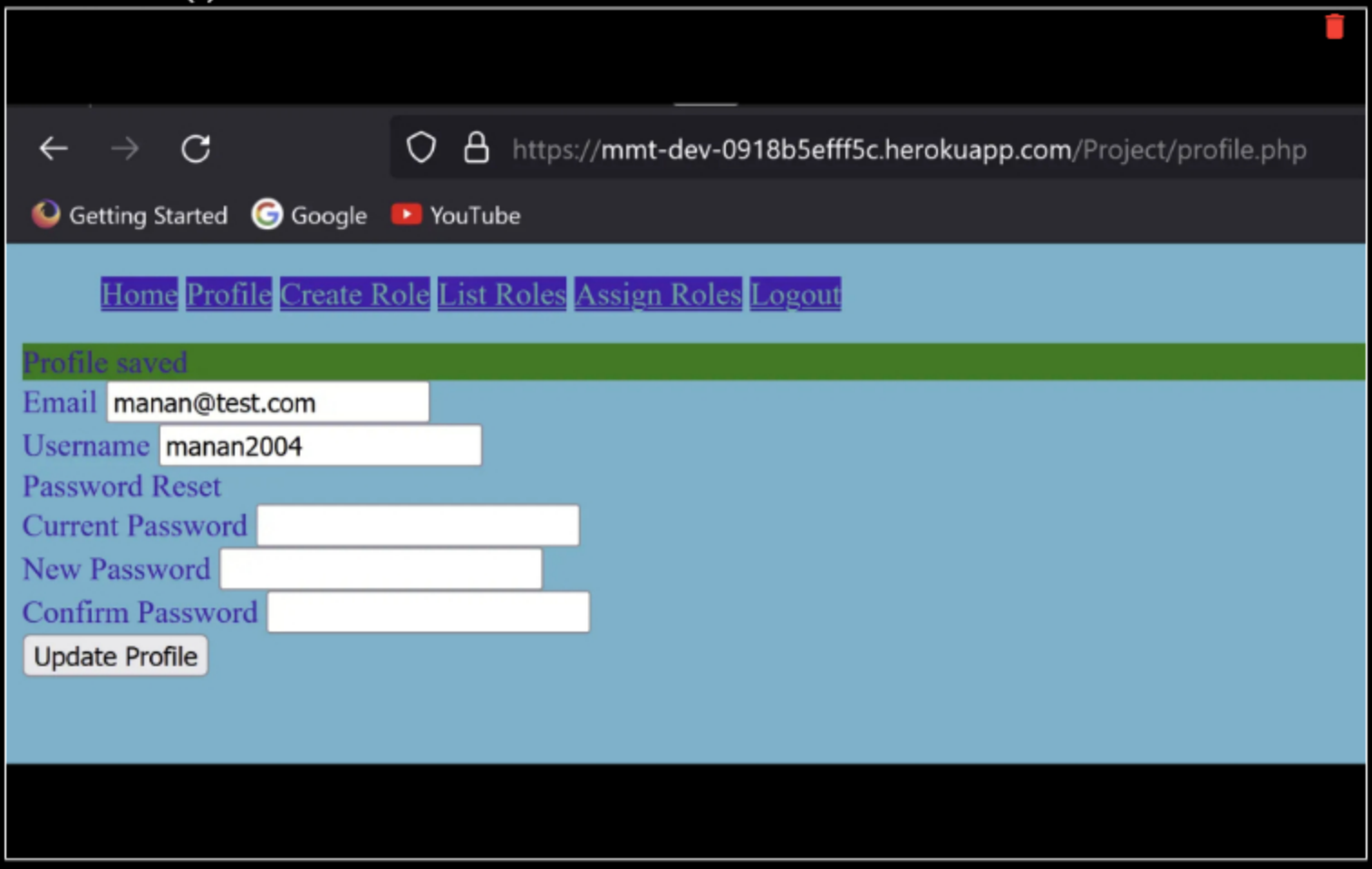
Gallery Style: Large View

Small Medium Large



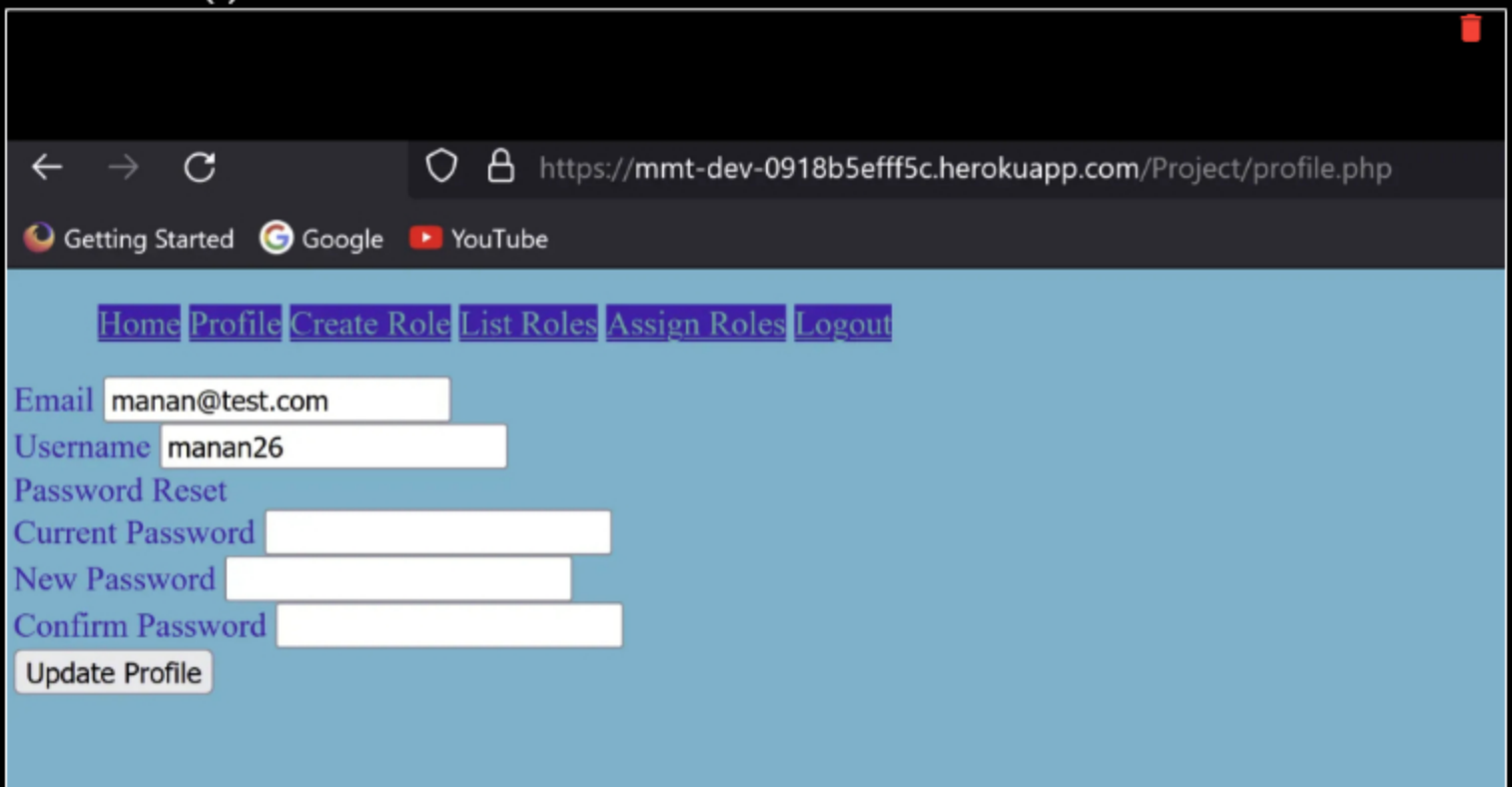
before username is changed

Checklist Items (0)



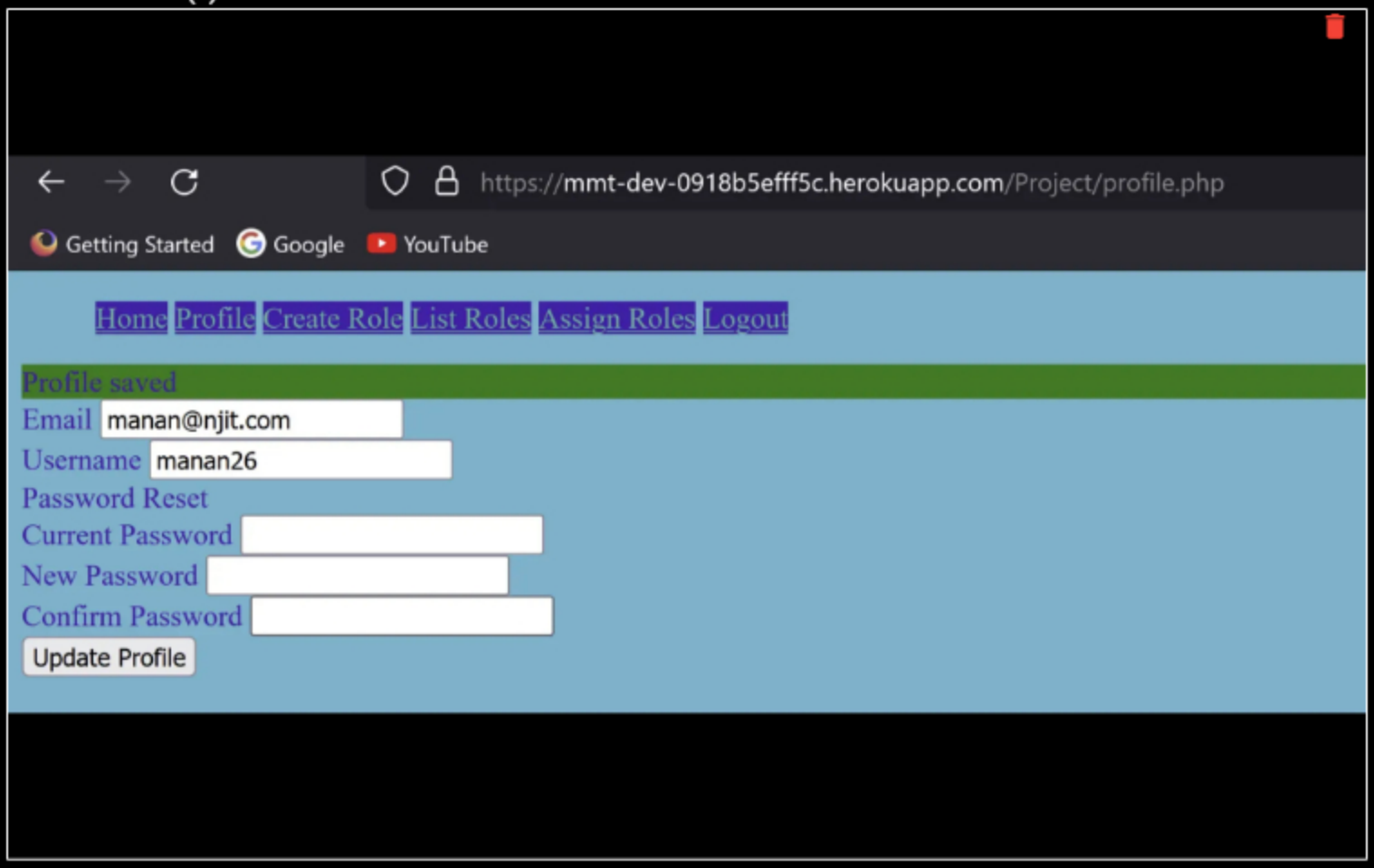
after username is changed

Checklist Items (0)



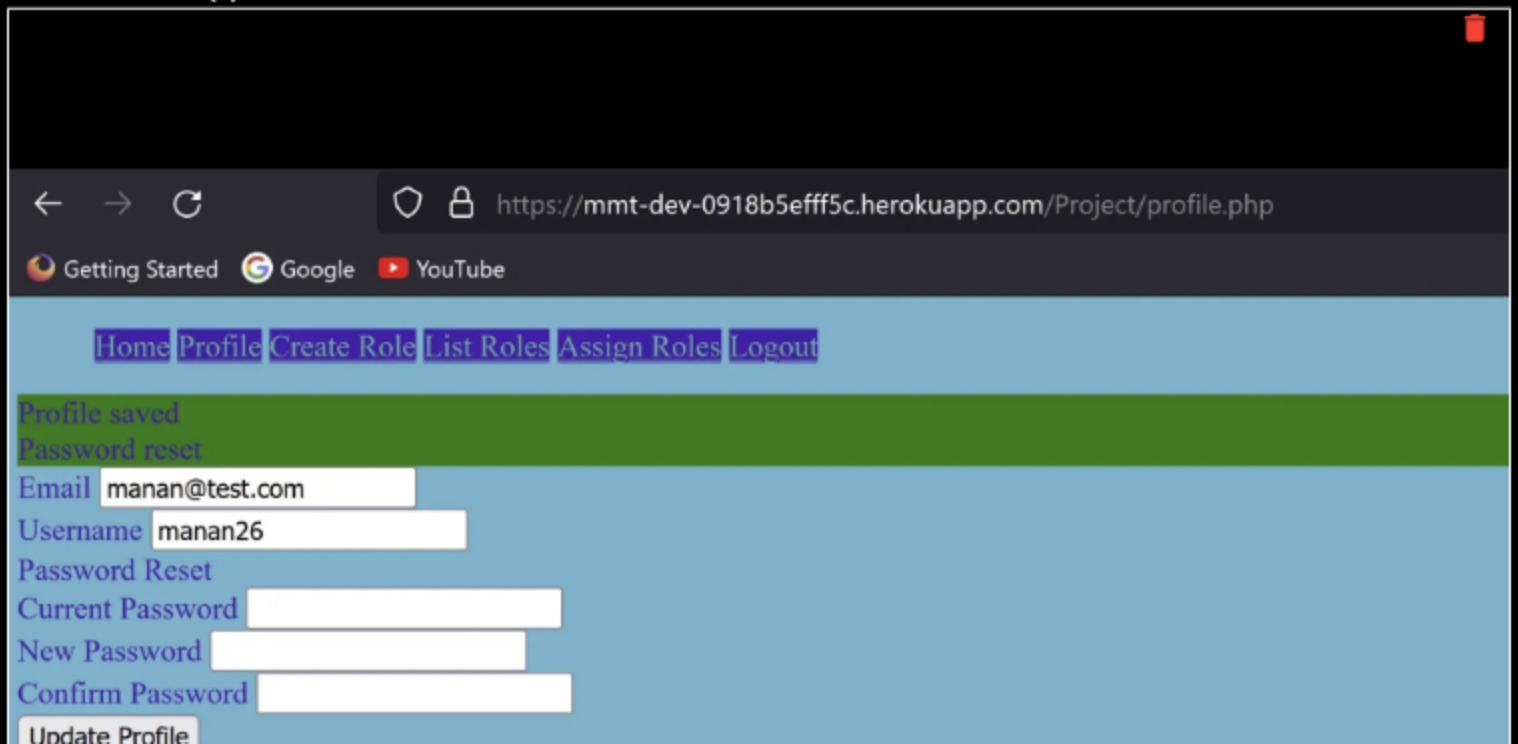
before email is changed

Checklist Items (0)



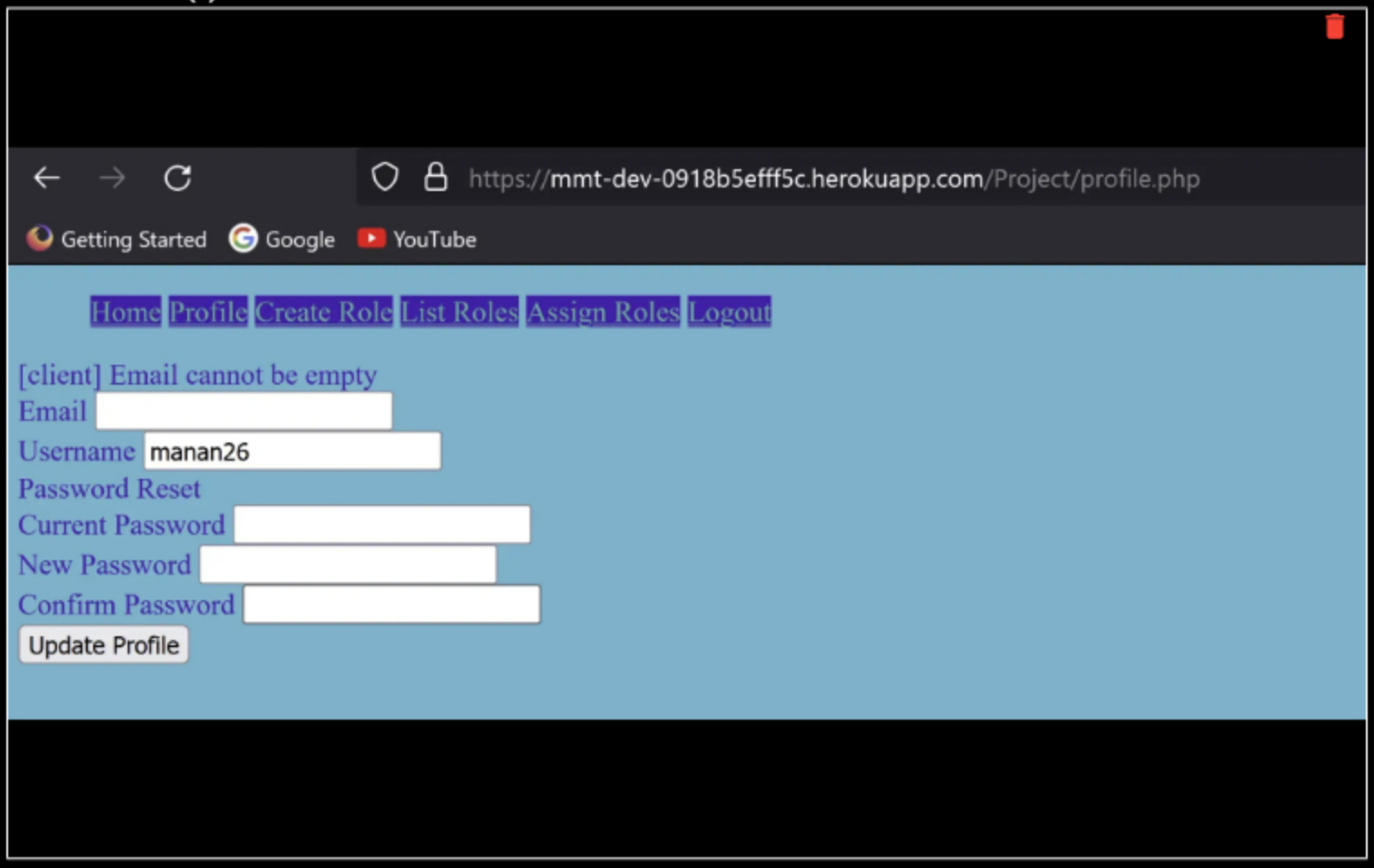
after email is changed

Checklist Items (0)



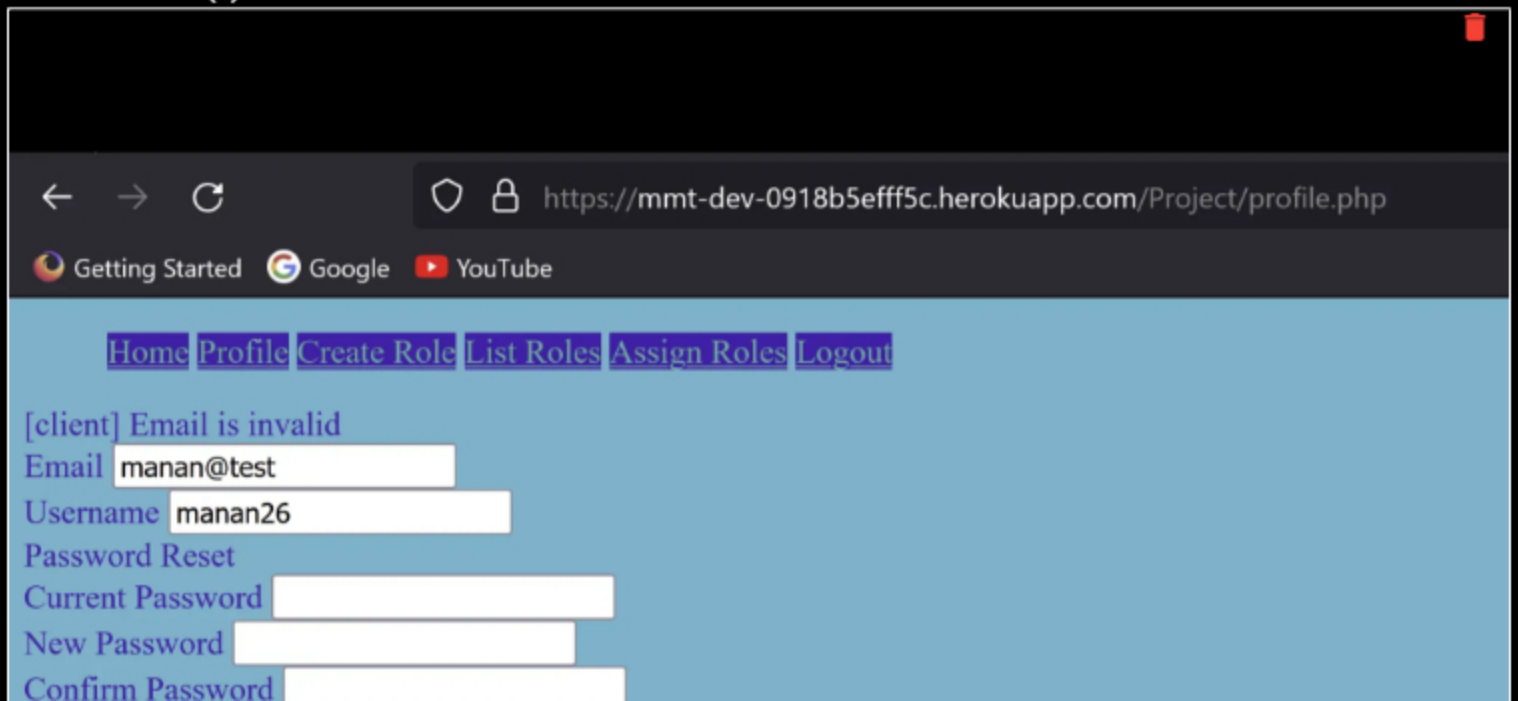
password updated

Checklist Items (0)



email cant be empty (JS valid)

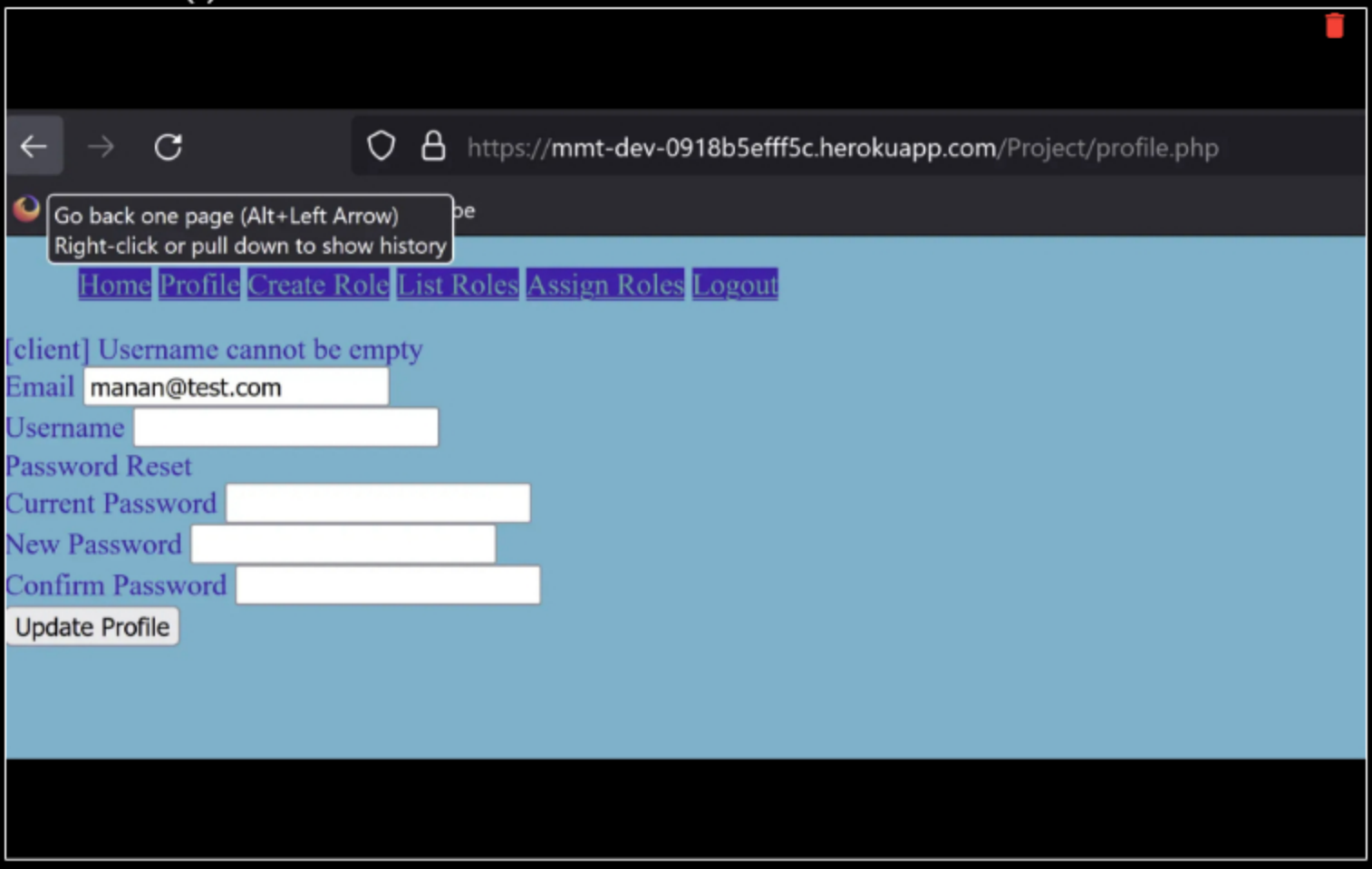
Checklist Items (0)



Update Profile

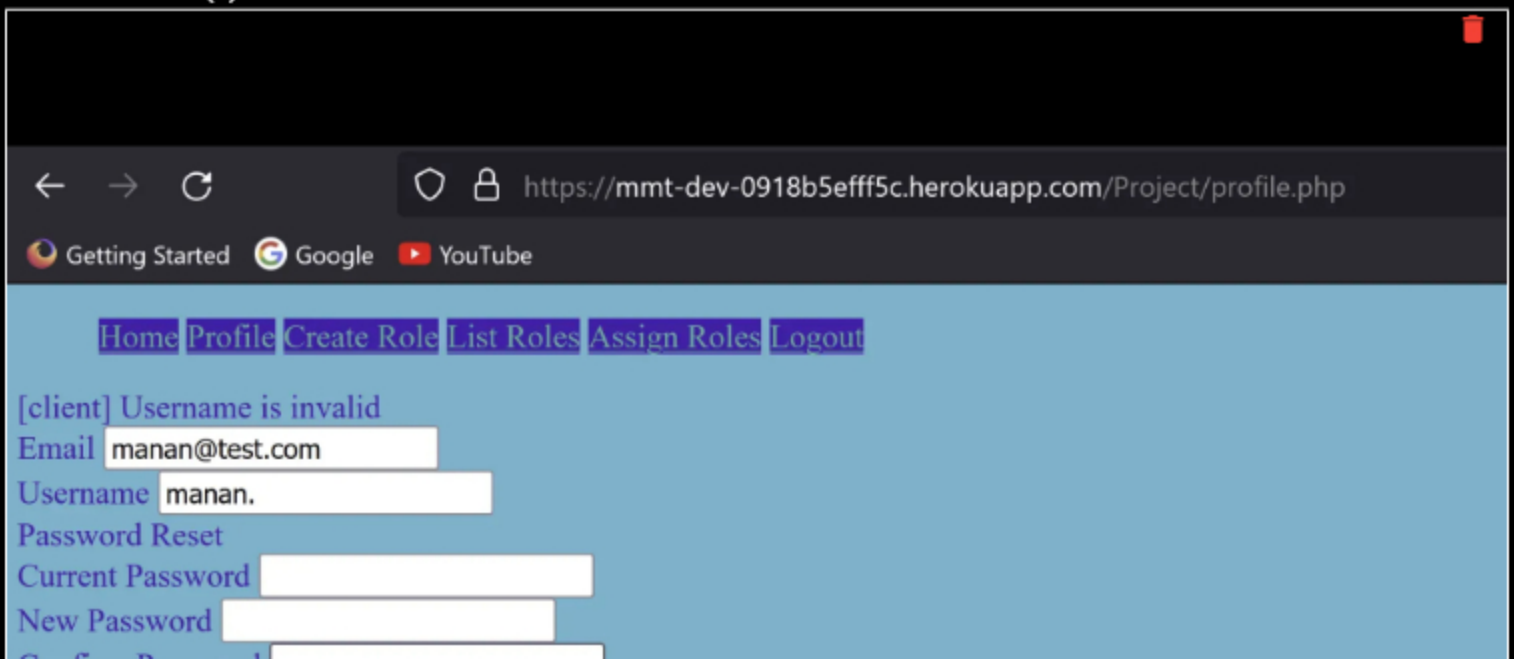
email is invalid (JS valid)

Checklist Items (0)



username cannot be empty (JS valid)

Checklist Items (0)

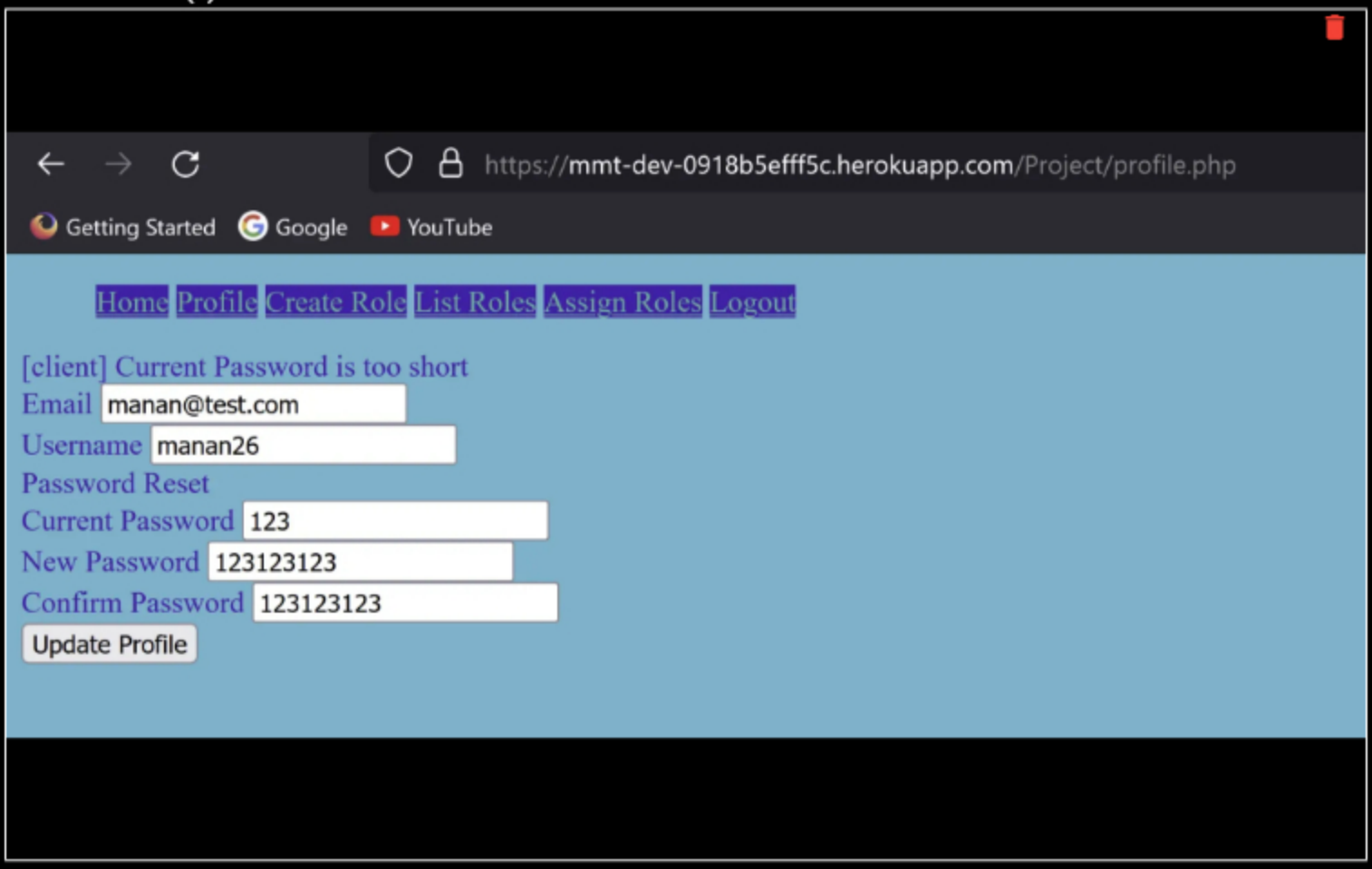


Confirm Password

Update Profile

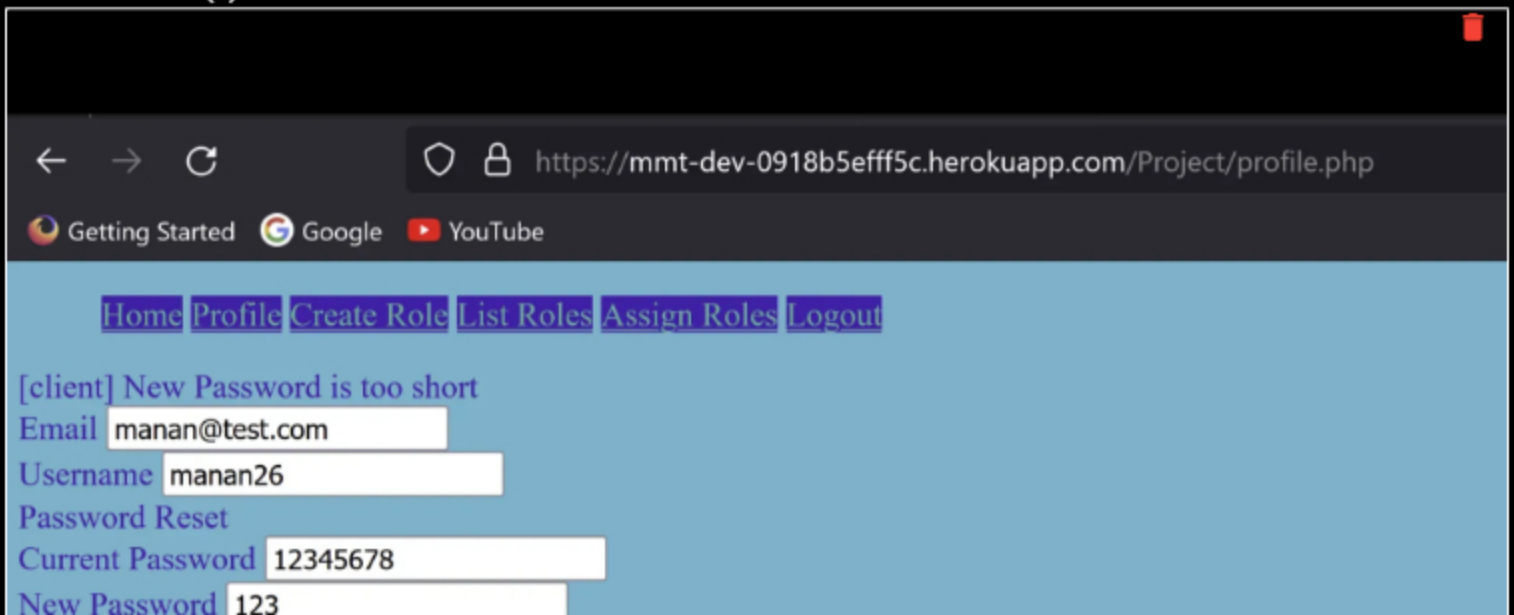
username is invalid (JS valid)

Checklist Items (0)



current password is too short (JS valid)

Checklist Items (0)

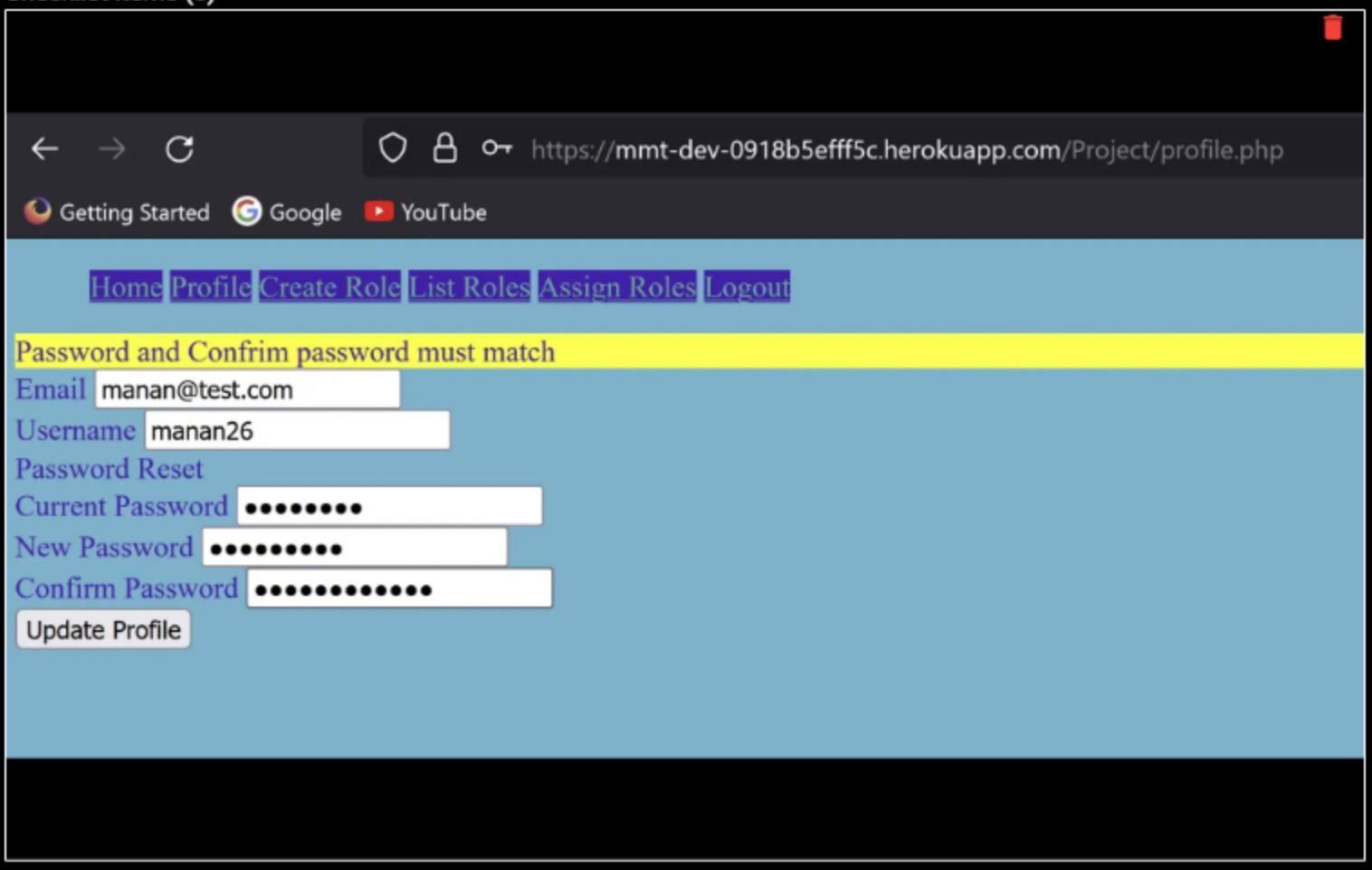


Confirm Password 123

Update Profile

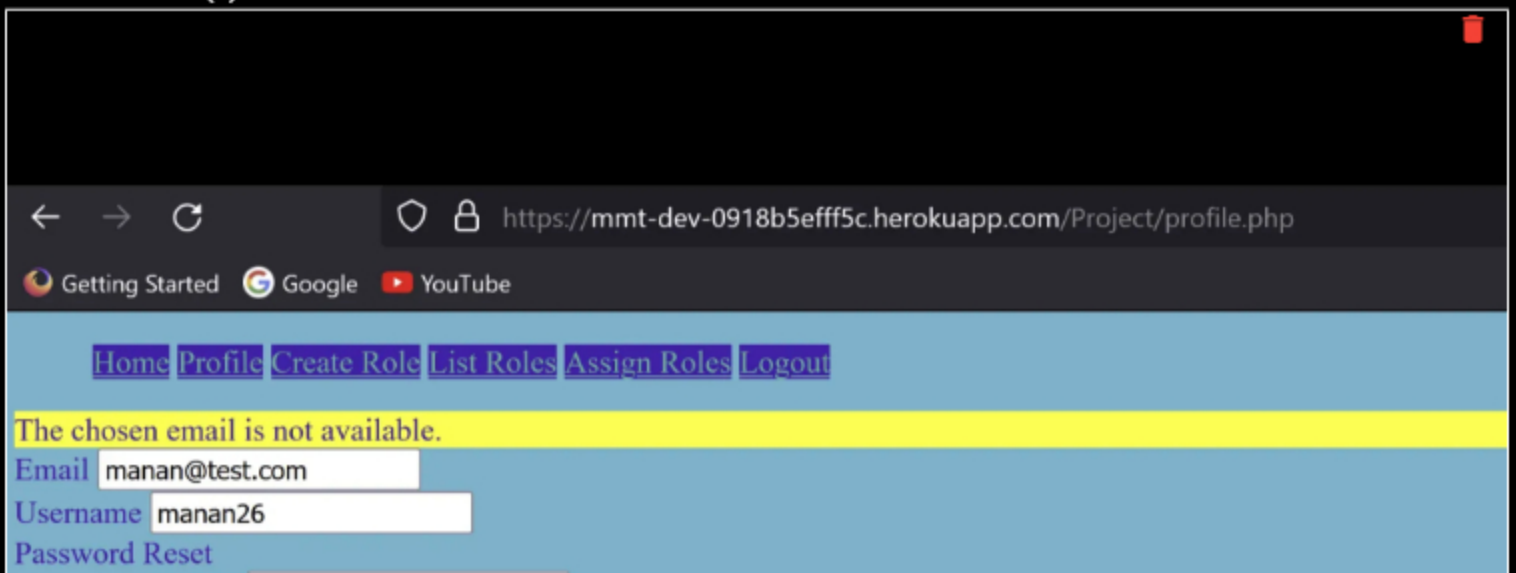
new password is too short (JS valid)

Checklist Items (0)



new pass and confirm pass dont match

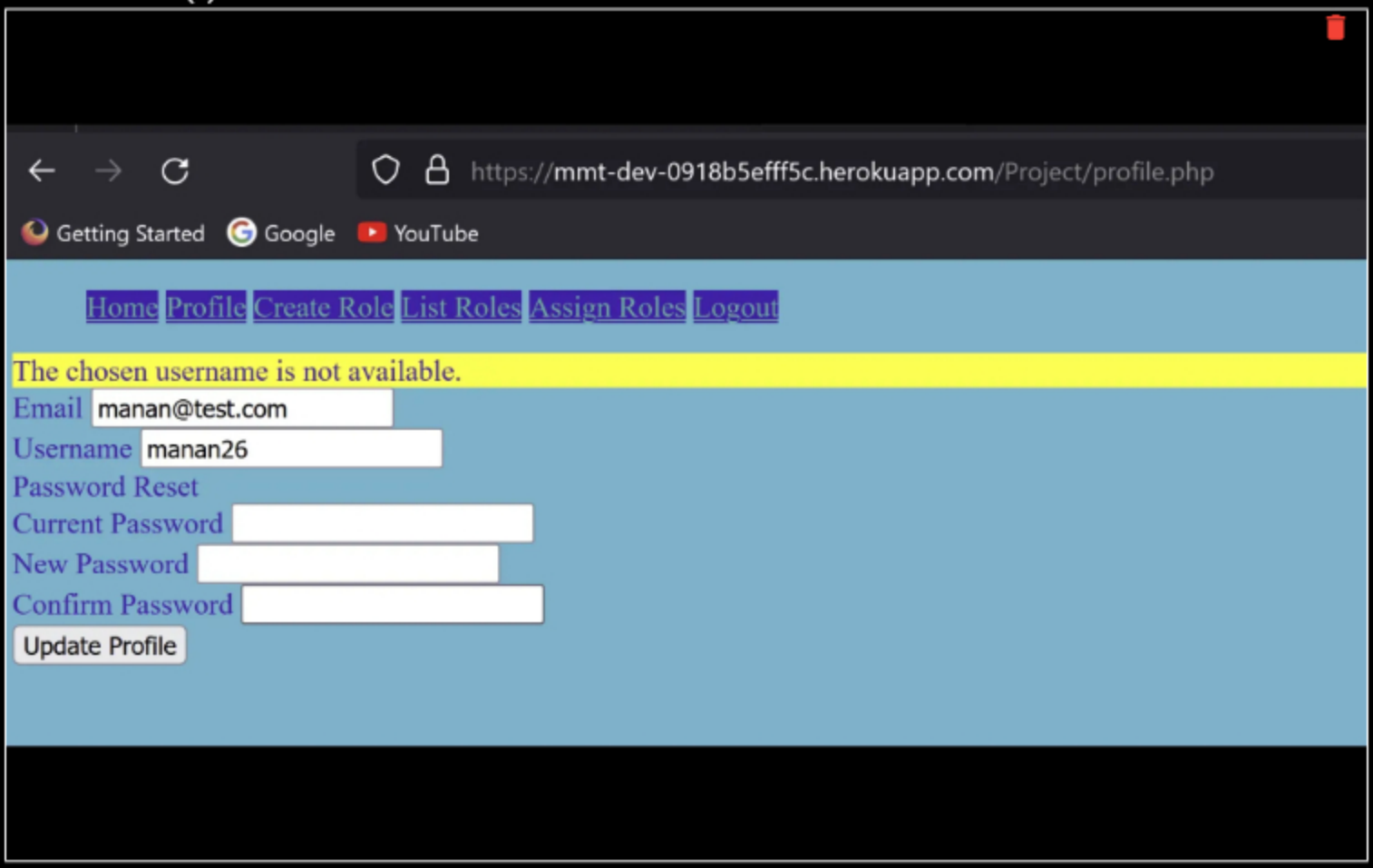
Checklist Items (0)



Current Password
New Password
Confirm Password
Update Profile

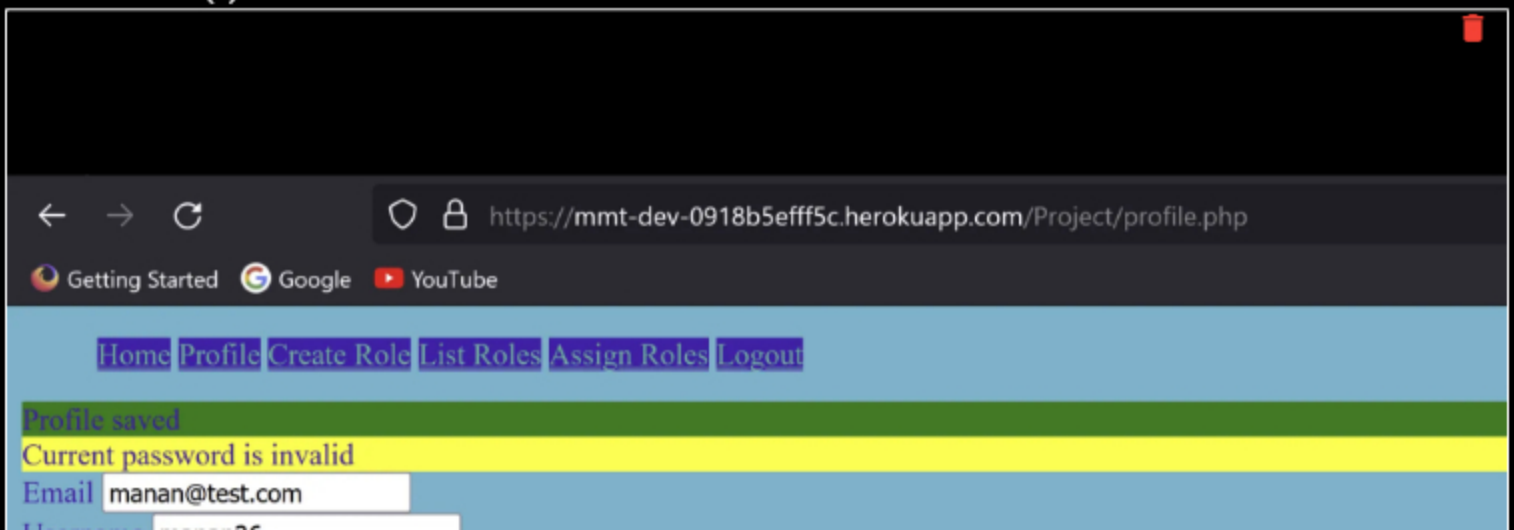
chosen email is unavailable

Checklist Items (0)



chosen username is unavailable

Checklist Items (0)



Username manan26
Password Reset
Current Password
New Password
Confirm Password
Update Profile

current pass is invalid

Checklist Items (0)

COLLAPSE

Task #4 - Points: 1

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

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Updating Username/Email
#2	1	Updating password
#3	1	Don't just show code, translate things to plain English

Response:

After verifying the email address and username entered by the user, PHP gets the form data from the $_POST variable when the form is submitted. After ensuring that the email and username are formatted correctly through server-side validation, SQL is used to update the Users table with the updated data, effectively replacing the previous information. After it is updated update, SQL queries are again to retrieve the new data from the table. This data is then used to add the email and username form fields when the profile page is reloaded.

COLLAPSE

Task #5 - Points: 1

Text: Include pull request links related to this feature

Details:
Should end in /pull/#

URL #1
https://github.com/mananthakore/mmt-it202-008/pull/33

Misc (1 pt.)

COLLAPSE

COLLAPSE

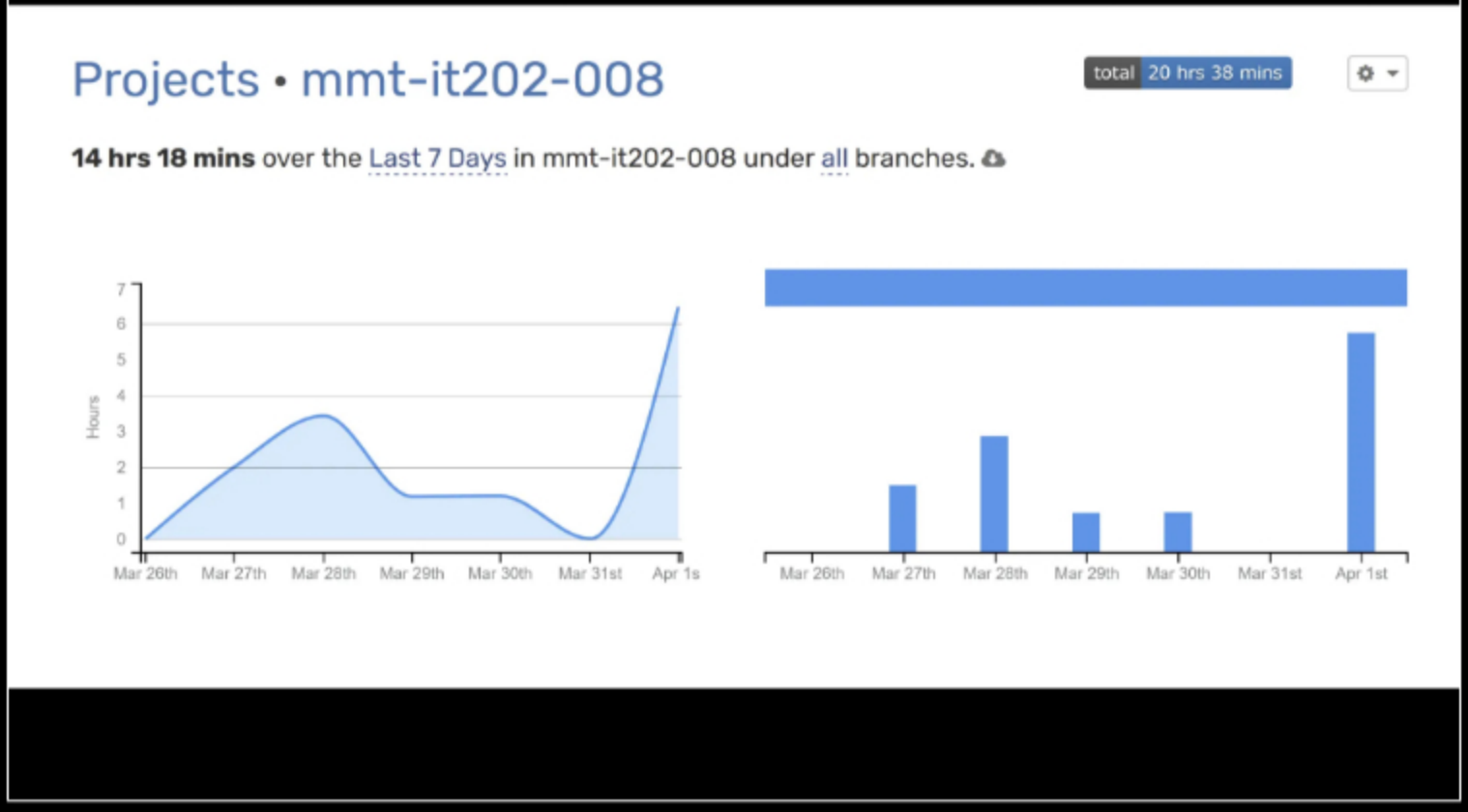
Task #1 - Points: 1

Text: Screenshot of wakatime

Task Screenshots:

Gallery Style: Large View

Small Medium Large



wakatime screenshot

COLLAPSE

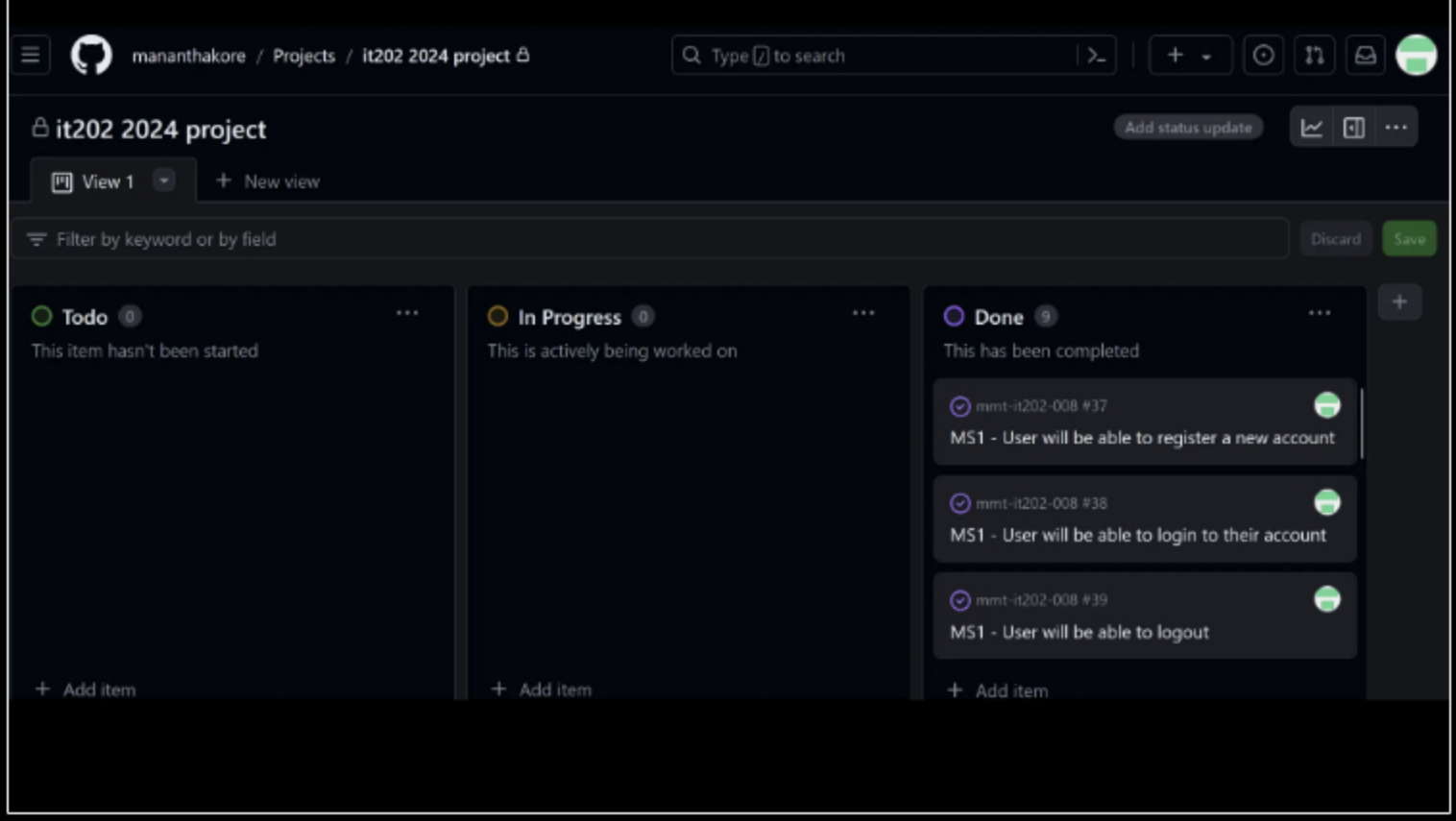
Task #2 - Points: 1

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

Task Screenshots:

Gallery Style: Large View

Small Medium Large



github board

COLLAPSE

Task #3 - Points: 1

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

URL #1

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

COLLAPSE

Task #4 - Points: 1

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

URL #1

https://mmt-prod-6beda717bbfa.herokuapp.com/Project/login.php

End of Assignment