

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT202-008-S2024/it202-milestone-1-2024/grade/df39>

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

Submissions:

Submission Selection

1 Submission [active] 4/24/2024 7:42:41 PM

Instructions

[^ COLLAPSE ^](#)

Prereqs:

Go through each lesson from "Project Setup and SQL" to "User Login Enhancement" and follow the branching names while gathering the code

Merge each into Milestone1 branch

Mark the related GitHub Issues items as "done"

Implement your own custom CSS (something much different than the default "ugly" CSS given as an example)

Consider styling all forms/inputs, data output, navigation, etc

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

Instructions:

Make sure you're in Milestone1 with the latest changes pulled

Ensure Milestone1 has been deployed to heroku dev

Gather the requested evidence and fill in the explanations per each prompt

Save the submission and generate the output PDF

Put the output PDF into your local repository folder

add/commit/push it to GitHub

Merge Milestone1 into dev

Locally checkout dev and pull the changes

Create and merge a pull request from dev to prod to deploy Milestone1 to prod

Upload this output PDF to Canvas

Branch name: Milestone1

Tasks: 26 Points: 10.00



User Registration (2 pts.)

[^ COLLAPSE ^](#)

● COLLAPSE ▲

Task #1 - Points: 1

Text: Screenshot of form on website page

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input checked="" type="checkbox"/> #1	1	Heroku dev url should be present in the address bar
<input checked="" type="checkbox"/> #2	1	Should have thoughtful CSS applied
<input checked="" 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 checked="" type="checkbox"/> #4	1	Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)
<input checked="" type="checkbox"/> #5	1	Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)
<input checked="" 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 browser window with a registration form. The URL in the address bar is `df39-1t202-008-dev-169ccb508606c7.herokuapp.com`. The form has several validation errors:

- Email: Please enter a valid email address (CLIENT)
- Passwords do not match (CLIENT)
- Username must be between 2 and 30 characters long and must only be alphanumeric characters (CLIENT)
- Password must be at least 8 characters long (CLIENT)

The developer tools' Elements tab shows the following JavaScript code in the console:

```


console.log('Changed type to text for element ${input.name || "(No name)"');
}

}

start("HTML Validation has been disabled until page reload");
})();

```

The console also lists several removed attributes from the input fields:

- Removed required from element email
- Changed type to text for element email
- Removed required from element username
- Removed maxlength from element username
- Removed required from element password
- Removed minlength from element password
- Changed type to text for element password
- Removed required from element confirm
- Removed minlength from element confirm
- Changed type to text for element confirm
- undefined

Javascript Validation of email format, username format, passwords matching and password being longer than 8 characters.

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 browser window with a registration form and an open developer tools console.

Browser Header:

- Personal
- Learn [Milestone 1] User Registr...
- Learn Comparing dev - Milestone...
- localhost:3000/Project/pro...
- df39-i202-008-dev - GH...
- https://df39-i202-008-de...

Form Errors:

- Please enter a valid email address (CLIENT)
- Please enter a valid email address (CLIENT)
- Please enter a valid email address (CLIENT)
- Passwords do not match (CLIENT)

Form Fields:

Email	janet@gmail.com
Username	wkjemstaken
Password	tatallyool
Confirm	

Buttons:

- Register

Developer Tools Console:

```
Qr Elements < > Evaluations Errors Warnings Logs
  input type="text";
  console.log("Changed type to text for element $[input.name || "the name"]);}
}

alert("HTML Validation has been disabled until page reload");
})([]);
Removed required from element email
Changed type to text for element email
Removed required from element username
Removed maxlength from element username
Removed required from element password
Removed minlength from element password
Changed type to text for element password
Removed required from element confirm
Removed minlength from element confirm
Changed type to text for element confirm
< undefined
```

This shows that each field must be required.

Checklist Items (1)

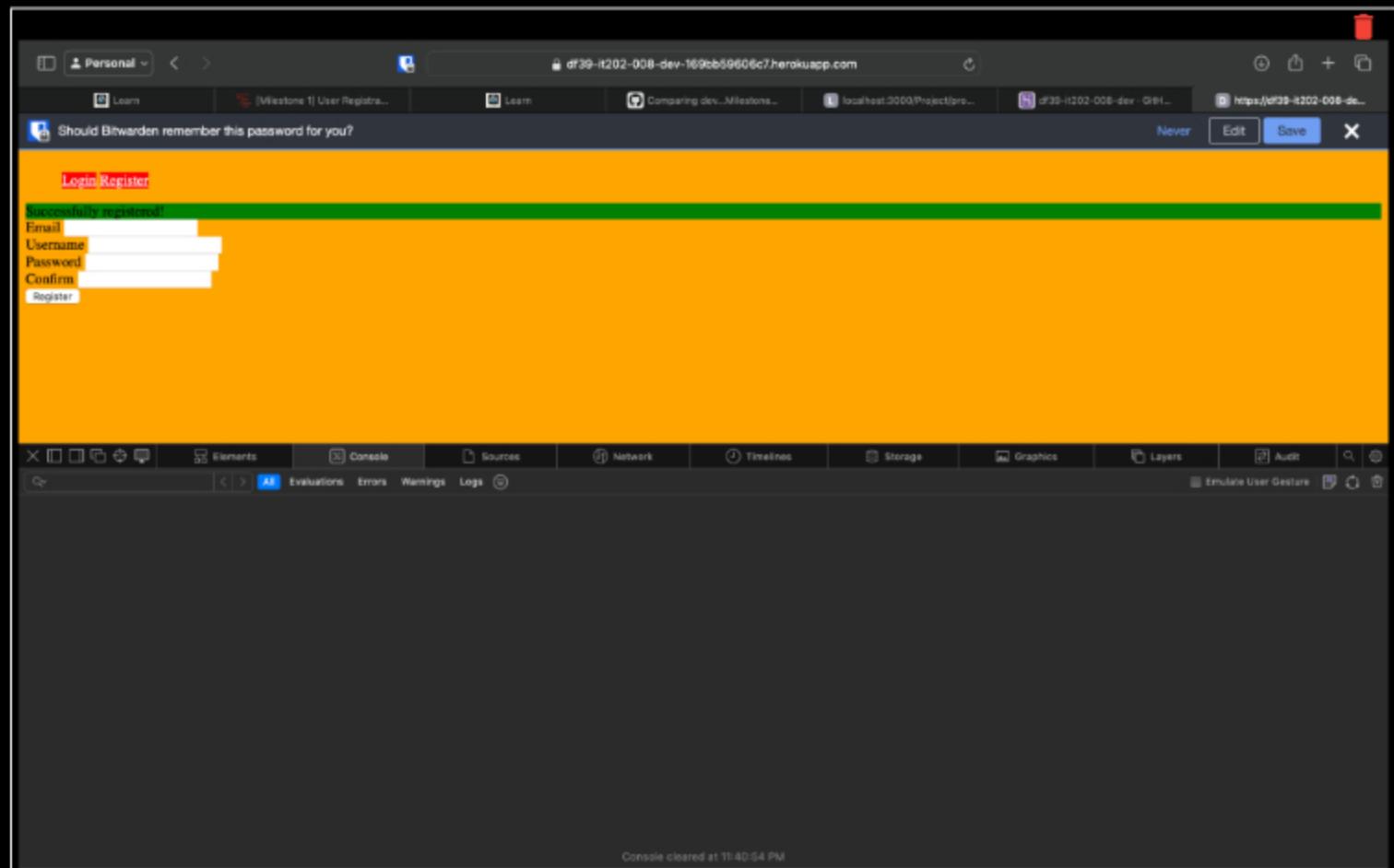
#3 Demonstrate JavaScript validation for each field [can be combined] (email validation (format), username validation (format), password validation (format), password and confirm password matching, and each field being required)

A screenshot of a web browser window. The address bar shows the URL: df39-ir202-008-dev-169bb59606c7.herokuapp.com. The main content area displays a Bitwarden registration form. A prominent yellow horizontal bar at the top contains the error message "The chosen username is not available.". Below this, there are four input fields: "Email" (placeholder "Email"), "Username" (placeholder "Username"), "Password" (placeholder "Password"), and "Confirm" (placeholder "Confirm"). The "Username" field is currently active, indicated by a red border around its input box. At the bottom of the page, the browser's developer tools are open, specifically the Network tab in the Performance panel. The status bar at the bottom of the browser window shows "Emulate User Gesture".

This demonstrates username already in use message.

Checklist Items (1)

#5 Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)



This shows user-friendly message of new account being made, also shows CSS applied to site, and heroku dev deployment link in URL bar.

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

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

Task #2 - Points: 1

Text: Screenshot of the form code

COLLAPSE ^

Details:

Should have appropriate input types for the fields

Task Screenshots:

Gallery Style: Large View

Small Medium Large

Screenshot of JavaScript validation

```

  57     }
  58     if ($this->error) {
  59         //echo $this->error;
  60         $this->password = password_hash($password, PASSWORD_BCRYPT);
  61         $hash = getHash();
  62         $stmt = $db->prepare("INSERT INTO users (email, password, username) VALUES(:email, :password, :username)");
  63         $stmt->execute();
  64         $stmt->close();
  65         $stmt = $db->prepare("SELECT * FROM users WHERE email = :email AND password = :password");
  66         $stmt->execute();
  67         $user = $stmt->fetch();
  68         $stmt->close();
  69         if ($user->username == $username) {
  70             $user->password = $hash;
  71             $user->username = $username;
  72             $user->email = $email;
  73             $user->id = $user->id;
  74             $user->date_created = date('Y-m-d H:i:s');
  75             $user->date_modified = date('Y-m-d H:i:s');
  76             $user->last_login = date('Y-m-d H:i:s');
  77             $user->status = 1;
  78             $user->is_admin = 0;
  79             $user->is_deleted = 0;
  80             $user->is_email_verified = 0;
  81             $user->is_password_reset = 0;
  82             $user->is_email_change = 0;
  83             $user->is_password_change = 0;
  84             $user->is_email_change = 0;
  85             $user->is_password_change = 0;
  86             $user->is_email_change = 0;
  87             $user->is_password_change = 0;
  88             $user->is_email_change = 0;
  89             $user->is_password_change = 0;
  90             $user->is_email_change = 0;
  91             $user->is_password_change = 0;
  92             $user->is_email_change = 0;
  93             $user->is_password_change = 0;
  94             $user->is_email_change = 0;
  95             $user->is_password_change = 0;
  96             $user->is_email_change = 0;
  97             $user->is_password_change = 0;
  98             $user->is_email_change = 0;
  99             $user->is_password_change = 0;
 100            $user->is_email_change = 0;
 101            $user->is_password_change = 0;
 102            $user->is_email_change = 0;
 103            $user->is_password_change = 0;
 104            $user->is_email_change = 0;
 105            $user->is_password_change = 0;
 106            $user->is_email_change = 0;
 107            $user->is_password_change = 0;
 108            $user->is_email_change = 0;
 109            $user->is_password_change = 0;
 110            $user->is_email_change = 0;
 111            $user->is_password_change = 0;
 112            $user->is_email_change = 0;
 113            $user->is_password_change = 0;
 114        }
 115    }
 116    </p>
 117    </div>
 118    <?php
 119    require __DIR__ . "/../../partials/footer.php";
 120  </?>

```

Screenshots of the form code of register.php

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 checked="" type="checkbox"/> #1	1	Show the JavaScript validations (include any extra files related)
<input checked="" type="checkbox"/> #2	1	Show the PHP validations (include any lib content)
<input checked="" 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

```

19      <label for="confirm">Confirm</label>
20      <input type="password" name="confirm" required minlength="8" />
21  </div>
22  <input type="submit" value="Register" />
23  </form>
24  <script>
25      function validate(form) {
26          // DF39 4/1/2024
27          var email = form.email.value;
28          var username = form.username.value;
29          var password = form.password.value;
30          var confirm = form.confirm.value;
31
32          var emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
33          var usernameRegex = /^[a-zA-Z0-9]{2,30}$/;
34
35          if (!emailRegex.test(email)) {
36              flash("Please enter a valid email address (CLIENT)", "warning");
37              return false;
38          }
39
40          if (!usernameRegex.test(username)) {
41              flash("Username must be between 2 and 30 characters long and must only be alphanumeric characters (CLIENT)");
42              return false;
43          }
44
45          if (password.length < 8) {
46              flash("Password must be at least 8 characters long (CLIENT)");
47              return false;
48          }
49
50          if (password !== confirm) {
51              flash("Passwords do not match (CLIENT)");
52              return false;
53          }
54      return true;
55  }
56 </script>
57 <?php

```

Checklist Items (2)**#1 Show the JavaScript validations (include any extra files related)****#2 Show the PHP validations (include any lib content)**

```

56 </script>
57 <?php
58 //TODO 2: add PHP Code
59 if (!isset($_POST["email"])) && !isset($_POST["password"]) && !isset($_POST["confirm"]) && !isset($_POST["username"])) {
60     $email = se($_POST, "email", "", false);
61     $password = se($_POST, "password", "", false);
62     $confirm = se($_POST, "confirm", "", false);
63     $username = se($_POST, "username", "", false);
64 //TODO 3
65     $hasError = false;
66     if (empty($email)) {
67         flash("Email must not be empty", "danger");
68         $hasError = true;
69     }
70     //sanitize
71     $email = sanitize_email($email);
72     //validate
73     if (!is_valid_email($email)) {
74         flash("Invalid email address", "danger");
75         $hasError = true;
76     }
77     if (!is_valid_username($username)) {
78         flash("Username must only contain 3-16 characters a-z, 0-9, _, or -", "danger");
79         $hasError = true;
80     }
81     if (empty($password)) {
82         flash("password must not be empty", "danger");
83         $hasError = true;
84     }
85     if (empty($confirm)) {
86         flash("Confirm password must not be empty", "danger");
87         $hasError = true;
88     }
89     if (!is_valid_password($password)) {
90         flash("Password too short", "danger");
91         $hasError = true;
92     }
93     if (
94         strlen($password) > 0 && $password !== $confirm
95     ) {
96         flash("Passwords must match", "danger");
97         $hasError = true;
98     }
99     if (!$hasError) {
100        //TODO 4

```

Validation code of the server side (php)**Checklist Items (1)****#2 Show the PHP validations (include any lib content)****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 checked="" type="checkbox"/> #1	1	Password should be hashed
<input checked="" type="checkbox"/> #2	1	Should have email, password, username (unique), created, modified, and id fields
<input checked="" type="checkbox"/> #3	1	Ensure left panel or database name is present (should contain your ucid)

Task Screenshots:

Small

Medium

Large

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

- Left Panel (Database Structure):**
 - Connected to IT202-008 (8.0.36-0ubuntu1.2)
 - Tables: Users (selected), Roles, UserRoles, Users (another instance), columns, index, partitions, Views, Procedures, Functions, information_schema.
- Top Bar:** logout.php, login.php, register.php, Users, disableHTMLValidation.js, profile.php.
- Query Editor:** SELECT * FROM `Users` LIMIT 100
- Results:** A table with 8 rows, each containing id, email, password, created, modified, and username.

	id	email	password	created	modified	username
> 1	10	danny@danny.com	\$2y\$10\$jqOyGlVuWvg:	2024-04-01 03:37:49	2024-04-02 02:49:13	danny
> 2	11	poop@poop.com	\$2y\$10\$gVR3Gg.UcKl	2024-04-01 04:23:18	2024-04-01 04:23:18	poop
> 3	13	apple@apple.com	\$2y\$10\$5IQUTC...o1sl	2024-04-01 15:59:09	2024-04-01 15:59:09	apple
> 4	14	www@wwwe.com	\$2y\$10\$awlsxQXK5eG	2024-04-02 01:44:12	2024-04-02 01:44:12	ksdmc
> 5	17	lol@danny.com	\$2y\$10\$nxccV4YiN1C	2024-04-02 03:25:52	2024-04-02 03:25:52	wekjdnwec
> 6	18	money@money.com	\$2y\$10\$Hs5X2.UVLNE	2024-04-02 03:26:09	2024-04-02 03:26:09	money
> 7	19	newemail@gmail.com	\$2y\$10\$Bu4yeYjymk0l	2024-04-02 03:34:20	2024-04-02 03:34:20	dun
> 8	21	wjncwecjncl@09u409-	\$2y\$10\$evEhf6SHEmC	2024-04-02 03:40:54	2024-04-02 03:40:54	jinveijnf

Users table shown with password hashed. Also contains, email, password, username and id fields. Left side of screenshot shows my ucid for the database

Checklist Items (3)

#1 Password should be hashed

#2 Should have email, password, username (unique), created, modified, and id fields

#3 Ensure left panel or database name is present (should contain your ucid)

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:

What are the steps involved in the registration process? How does the application handle user input and validate it before saving it to the database?

When the page loads the java script validation loads with the page but is not yet executed, when the user puts information in the fields the java script (client side) validate function is called to ensure the format of the information is correct. When the user inputs the email, user name, password and confirm password the form is then submitted. If it is incorrect a flash message will pop up telling them what is incorrect. Once the form is submitted the data is sent to the server and the PHP code will check if the email, username, password and confirmation password are set. If they are set the php code will then add these user inputs into the `$_POST` array and sanitize the input data using the `set` function. This is for an extra layer of security. The data is then validated to ensure the format of username, password, email and password confirm. If the format is correct it then hashes the password and then pulls the data base using `getDB` and inserts the data into its appropriate table (Users Table) and creates a new user.

Task #6 - Points: 1

Text: Include pull request links related to this feature

Details:

Should end in /pull/#

URL #1

<https://github.com/dunnfall/df39-it202-008/pull/19>

User Login (2 pts.)

[^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	Include correct data where username is used to login
<input type="checkbox"/> #4	1	Include correct data where email is used to login
<input type="checkbox"/> #5	1	Show success login message
<input type="checkbox"/> #6	1	Demonstrate JavaScript validation for each field [can be combined] (username format, email format, password format, and each field being required)
<input type="checkbox"/> #7	1	Demonstrate user-friendly message of when an account doesn't exist
<input type="checkbox"/> #8	1	Demonstrate user-friendly message of when password doesn't match what's in the DB
<input type="checkbox"/> #9	1	Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)
<input type="checkbox"/> #10	1	Demonstrate session data being set (captured from server logs)

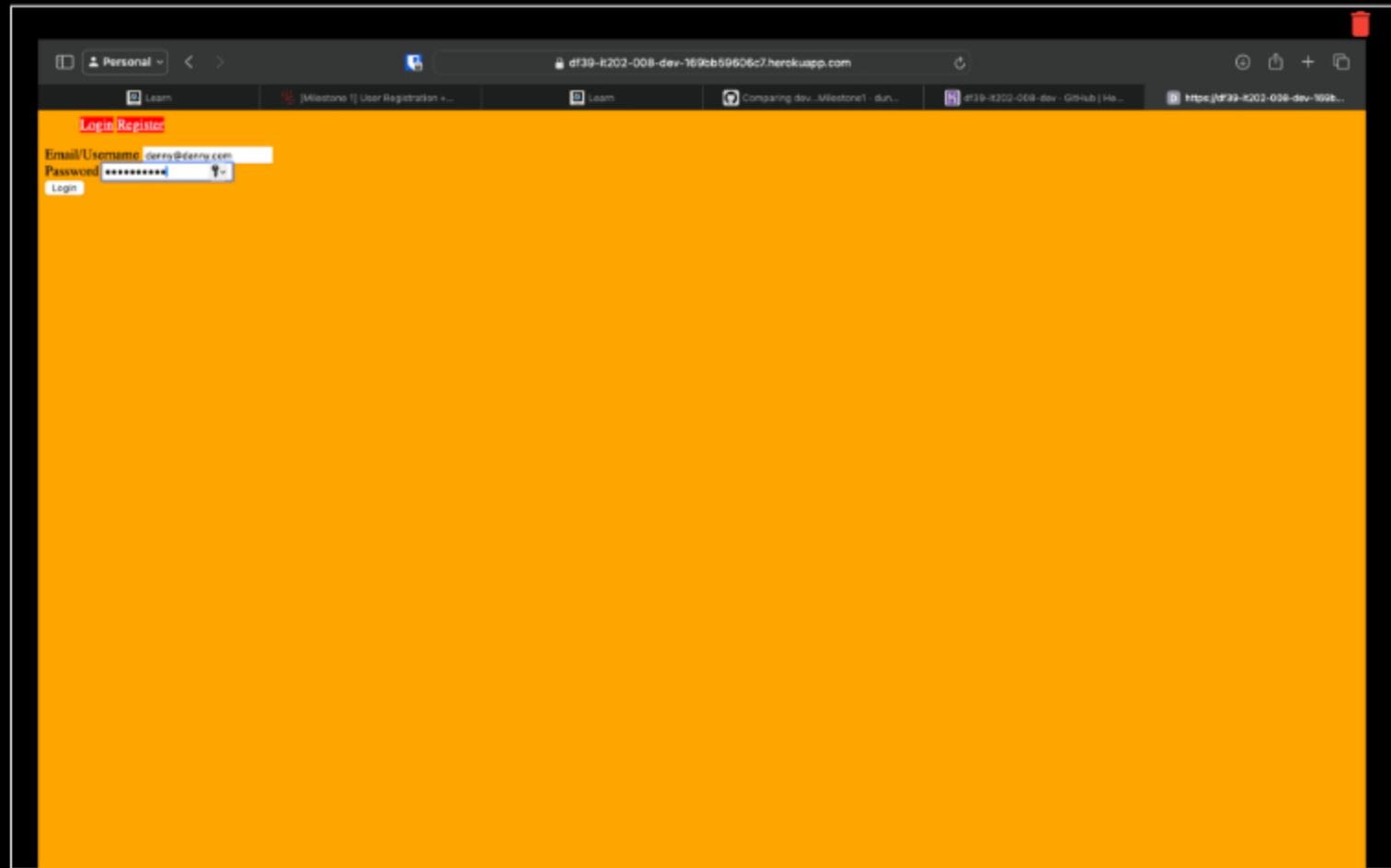
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



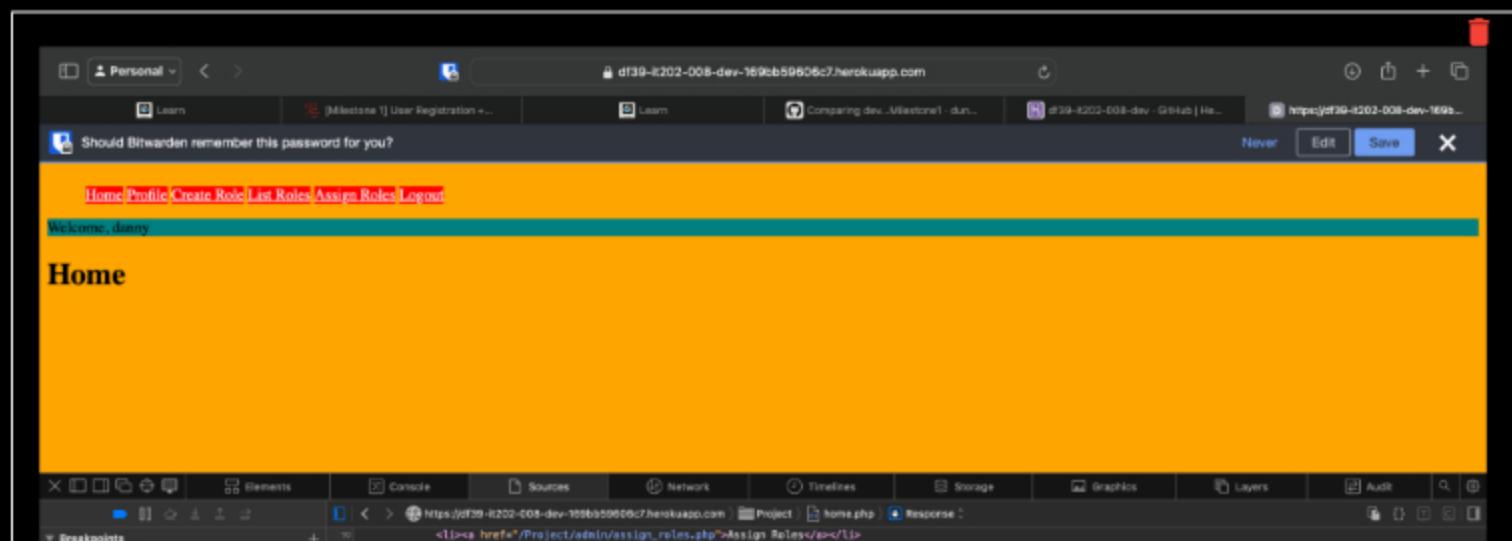
Shows heroku dev url, CSS applied on login site, correct login data is shown.

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

#3 Include correct data where username is used to login



Debugger Statements
All Exceptions
Uncaught Exceptions
Assertion Failures

By Type By Path

Project helpers.js home.php styles.css

```
</script>
  //Used to pretend the flash messages are below the first nav element
  function moveHelp(e) {
    let target = document.getElementsByTagName("nav")[0];
    if (target) {
      target.appendChild(e);
    }
  }
  moveHelp(document.getElementById("flash"));
</script>
```

Console cleared at 11:49:26 PM

Auto -- home.php

Login success

Checklist Items (1)

#5 Show success login message

Personal df39-it202-008-dev-169eb59606c7.herokuapp.com Never Edit Save X

Should Bitwarden remember this password for you?

Login Register

Please enter a valid email address (CLIENT)
 Please enter a valid username (CLIENT)
 Password must be at least 8 characters long (CLIENT)

Email/Username: danny@terry.com
 Password: t2

Login

Elements Console Sources Network Timelines Storage Graphics Layers Audit

Breakpoints Debugger Statements All Exceptions Uncaught Exceptions Assertion Failures

By Type By Path

Project helpers.js login.php styles.css

```
<!-- Include css and js files -->
<link rel="stylesheet" href="/Project/styles.css">
<script src="/Project/helpers.js"></script>
<nav>
  <a href="/Project/login.php">Login</a>
  <a href="/Project/register.php">Register</a>
</nav>
</head>
<form onsubmit="return validate(this);" method="POST">
  <div>
    <label for="email">Email/Username:</label>
    <input type="text" name="email" required />
  </div>
  <div>
    <label for="pw">Password:</label>
    <input type="password" id="pw" name="password" required minlength="8" />
  </div>
  <input type="submit" value="Login" />
</form>
<script>
```

Removed required from element email
 Removed required from element password
 Removed minlength from element password
 Changed type to text for element password
 undefined

Auto -- Page C

Javascript validation for email/username format and password format.

Checklist Items (1)

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

Personal df39-it202-008-dev-169eb59606c7.herokuapp.com Never Edit Save X

Should Bitwarden remember this password for you?

Login Register

Account not found

Email/Username:
 Password:

Login

Elements Console Sources Network Timelines Storage Graphics Layers Audit

Breakpoints Debugger Statements All Exceptions Uncaught Exceptions Assertion Failures

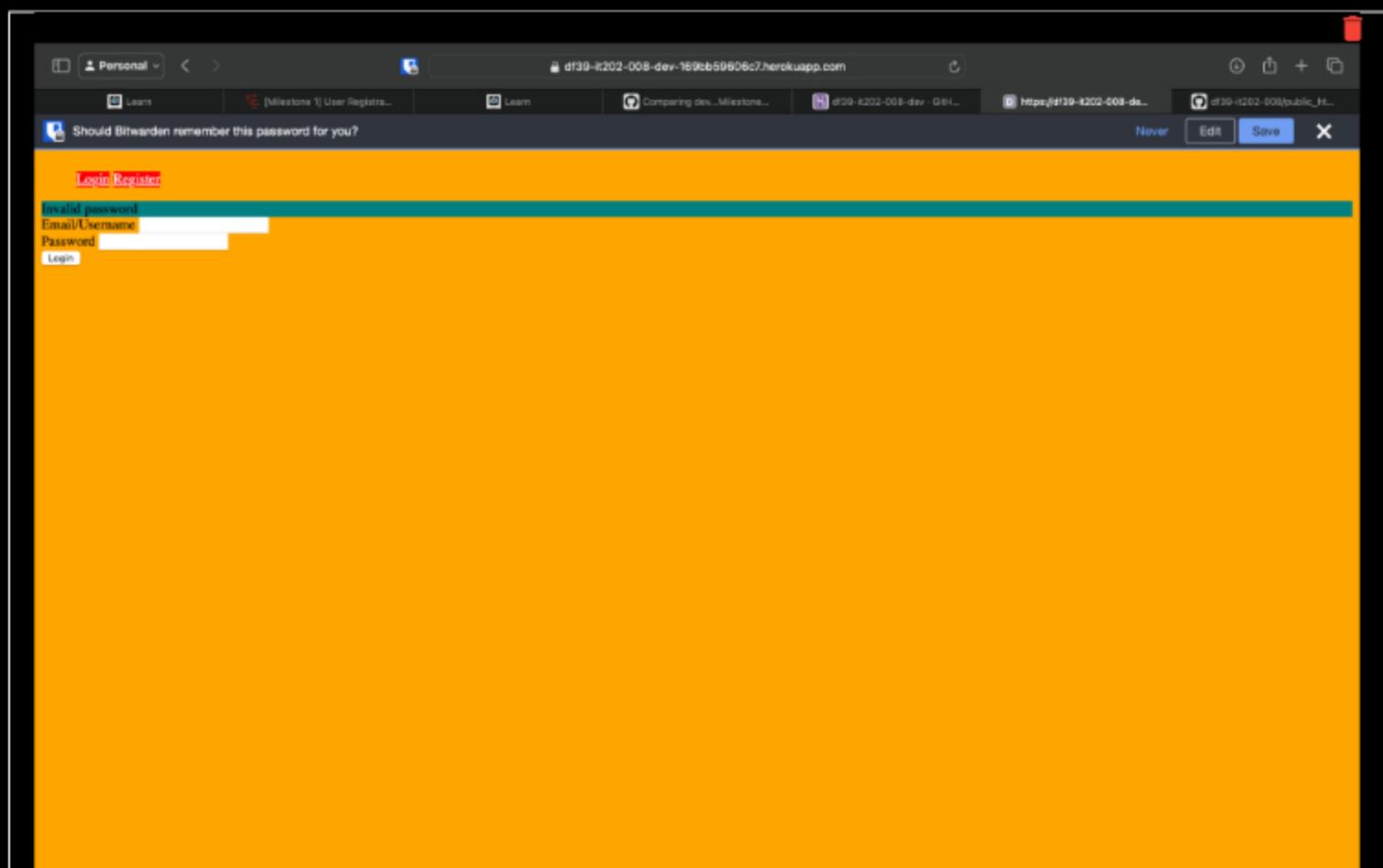
By Type By Path

Project helpers.js login.php styles.css

Account Not Found Message

Checklist Items (1)

#7 Demonstrate user-friendly message of when an account doesn't exist

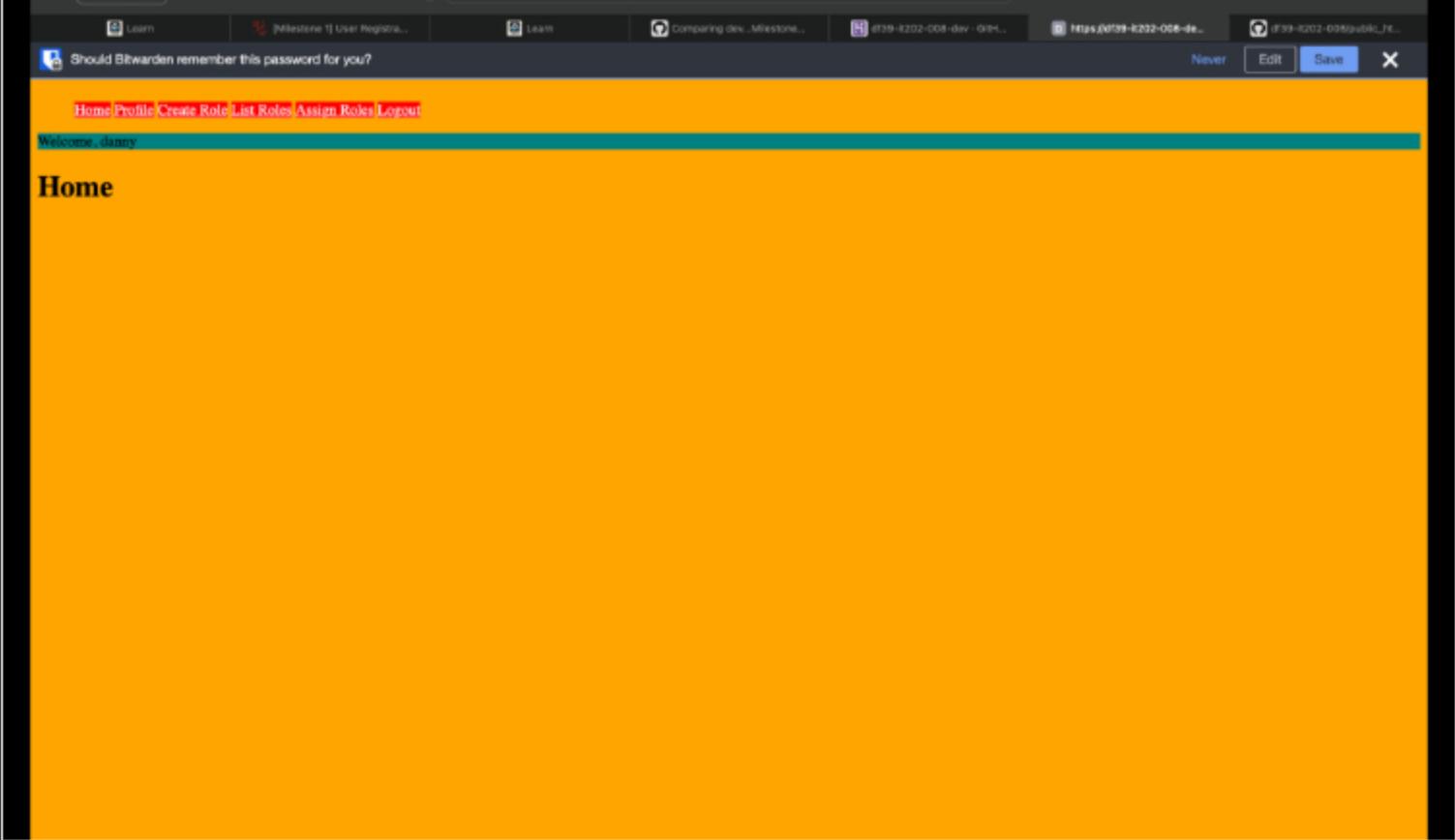


Invalid Password message

Checklist Items (1)

#8 Demonstrate user-friendly message of when password doesn't match what's in the DB





Successful login, and the destination page

Checklist Items (1)

#9 Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)

A screenshot of the Heroku dashboard. The top navigation bar shows tabs for "Personal", "Learn", "Comparing dev...Milestone...", "Log...", and "https://df39-it202-008-dev...". The main content area shows the "HEROKU" logo and a "Jump to Favorites, Apps, Pipelines, Spaces..." search bar. Below this, there's a breadcrumb trail: "Personal > df39-it202-008-dev". A "GitHub" link and a "dev" badge are also present. A "More" button is in the top right. A "Logs" section titled "Application Logs" displays server logs. The logs show several entries related to user authentication and session management. One entry includes JSON data for session data being set from server logs. At the bottom of the logs, there's a checkbox for "Autoscroll with output". The footer contains links for "heroku.com", "Blogs", "Careers", "Documentation", "Support", "Terms of Service", "Privacy", "Cookies", and "© 2024 Salesforce.com".

Session data being set from server logs.

Checklist Items (1)

#10 Demonstrate session data being set (captured from server logs)

Task #2 - Points: 1

Text: Screenshot of the form code

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input checked="" type="checkbox"/> #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)
<input checked="" type="checkbox"/> #2	1	Show JavaScript validations (include any extra files related)
<input checked="" type="checkbox"/> #3	1	Show PHP validations (include any lib content)
<input checked="" type="checkbox"/> #4	1	Include ucid/date code comments in all screenshots (one per screenshot is sufficient)

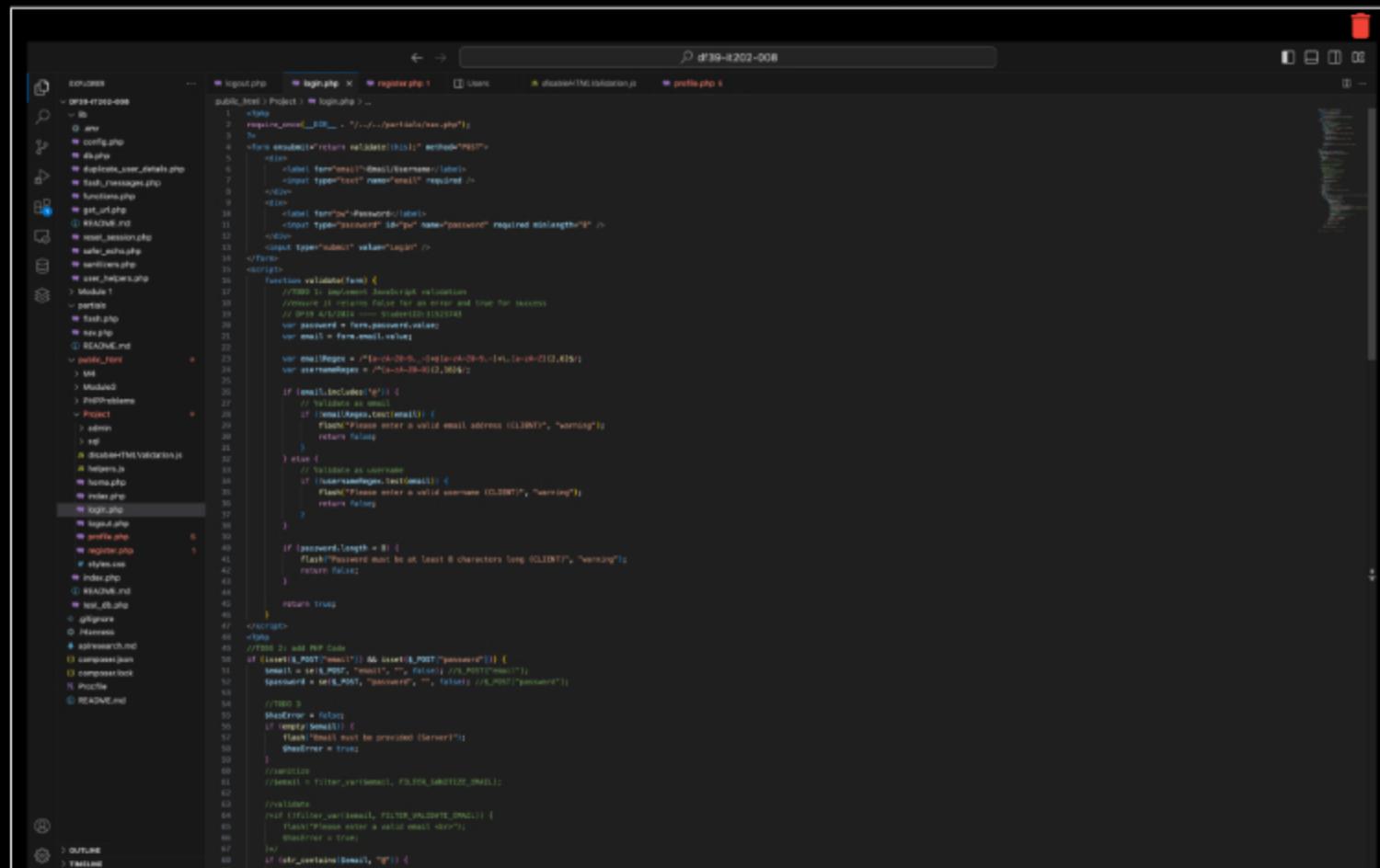
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



```
public_html> Project > login.php > register.php > Users > databaseHTMLValidation.js > profile.php >

1 <html>
2   <head>
3     require_once __DIR__ . '/../../../../partials/nav.php';
4   </head>
5   <body>
6     <form action="/> return validate(this); method='POST'>
7       <div>
8         <label type='email'>Email</label>
9         <input type='text' name='email' required />
10        <br>
11        <label type='password'>Password</label>
12        <input type='password' id='pwd' name='password' required minlength='8' />
13        <br>
14        <input type='submit' value='Login' />
15      </div>
16    </form>
17    <script>
18      function validateForm() {
19        //7000 is implemented JavaScript validation
20        //ensure it returns false for an error and true for success
21        //0918 A/3/2018 ---- StudentID-11523104
22        var password = form.password.value;
23        var email = form.email.value;
24
25        var emailRegexp = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}\$/;
26        var emailRegexp = /^[a-zA-Z0-9.-]+\.[a-zA-Z0-9]{1,3}\$/;
27
28        if (email.includes('@')) {
29          // Validate as email
30          if (!emailRegexp.test(email)) {
31            flash("Please enter a valid email address (CLIENT)", "warning");
32            return false;
33          }
34        } else {
35          // Validate as username
36          if (!usernameRegexp.test(email)) {
37            flash("Please enter a valid username (CLIENT)", "warning");
38            return false;
39          }
40        }
41
42        if (password.length < 8) {
43          flash("Password must be at least 8 characters long (CLIENT)", "warning");
44          return false;
45        }
46
47        return true;
48      }
49    </script>
50  </body>
51  </html>
52 //7000 2 add PHP Code
53 if($_POST['email']) && !isset($_POST['password']) {
54   $email = $_POST['email'];
55   $password = '';
56   $hash = '';
57   $gHasher = true;
58 }
59 //7000 3
60 $hashError = false;
61 if(empty($email)) {
62   flash("Email must be provided (Server)", "error");
63   $gHasher = true;
64 }
65 //7000 4
66 if(empty($password)) {
67   flash("Password must be provided (Server)", "error");
68   $gHasher = true;
69 }
70 //7000 5
71 if($email != filter_var($email, FILTER_SANITIZE_EMAIL)) {
72   flash("Please enter a valid email address");
73   $gHasher = true;
74 }
75 if($password != filter_var($password, FILTER_SANITIZE_STRING)) {
76   flash("Please enter a valid password");
77   $gHasher = true;
78 }
79 if($email && $password) {
80   $gHasher = true;
81 }
```

Input fields shown at the top, username and email are in a combined field.

Checklist Items (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)

Continuation of first screenshot

Checklist Items (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)

```
<script>
    function validate(form) {
        //TODO 1: implement JavaScript validation
        //ensure it returns false for an error and true for success
        // DF39 4/1/2024 ---- StudentID:31523743
        var password = form.password.value;
        var email = form.email.value;

        var emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,6}$/;
        var usernameRegex = /^[a-zA-Z0-9]{2,16}\$/;

        if (email.includes('@')) {
            // Validate as email
            if (!emailRegex.test(email)) {
                flash("Please enter a valid email address (CLIENT)", "warning");
                return false;
            }
        } else {
            // Validate as username
            if (!usernameRegex.test(email)) {
                flash("Please enter a valid username (CLIENT)", "warning");
                return false;
            }
        }

        if (password.length < 8) {
            flash("Password must be at least 8 characters long (CLIENT)", "warning");
            return false;
        }
    }

```

```

        return true;
    }

```

JavaScript validations with UCID and date

Checklist Items (1)

#2 Show JavaScript validations (include any extra files related)

```

<?php
//TODO 2: add PHP Code
//UF39 4/1/2024
if (isset($_POST["email"])) && isset($_POST["password"])) {
    $email = se($_POST, "email", "", False); //$_POST["email"];
    $password = se($_POST, "password", "", false); //$_POST["password"];

    //TODO 3
    $hasError = false;
    if (empty($email)) {
        flash("Email must be provided (Server)");
        $hasError = true;
    }
    //sanitize
    //$email = filter_var($email, FILTER_SANITIZE_EMAIL);

    //validate
    //if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    //    flash("Please enter a valid email <br>");
    //    $hasError = true;
    //}
    if (str_contains($email, "@")) {
        //sanitize
        //$email = filter_var($email, FILTER_SANITIZE_EMAIL);
        $email = sanitize_email($email);
        //validate
        //if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        //    flash("Invalid email address");
        //    $hasError = true;
        //}
        if (!is_valid_email($email)) {
            flash("Invalid email address (Server)");
            $hasError = true;
        }
    } else {
        if (!is_valid_username($email)) {
            flash("Invalid username (Server)");
            $hasError = true;
        }
    }
    if (empty($password)) {
        flash("Password must be provided (Server)");
        $hasError = true;
    }
    if (strlen($password) < 8) {
        flash("Password must be at least 8 characters long (Server)");
        $hasError = true;
    }
    if (!$hasError) {
        //TODO 4
    }
}

```

PHP validations

Checklist Items (2)

#3 Show PHP validations (include any lib content)

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

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
<input checked="" type="checkbox"/> #1	1	Don't just show code, translate things to plain English
<input checked="" type="checkbox"/> #2	1	Explain how the session works and why/how it's used

Response:

When the login page is loaded again the java-script validation is loaded with it but not called until the user attempts to submit the form. Once submitted the java script validate function is called and checks the format of the username/email and password using regex. If incorrect format is found a flash message will appear telling the user what formatting is incorrect. If the information is valid the form is submitted. The form data is added to the \$_POST array. The php code will then sanitize the data using the se function. After sanitization the server side validation occurs. The PHP code will check if the formatting of the username/email, and password field is correct and not empty. If the format is wrong or fields are empty it will send a flash message to the user with the flash function and change hasError to true. If the format is good PHP will then query the database to fetch data associated with the user email or username. If none is found the user will see an a "Account Not Found Message". After getting the user data the hashed password is matched up to the user inputed password. If it is correct the user will be logged in. Once the user is logged in their user data information is stored in the session to keep them logged in. The user data including the roles they have are also fetched and stored. The user after being authenticated is then directed to the home page.

Task #4 - Points: 1**Text:** Include pull request links related to this feature**i Details:**

Should end in /pull/#

URL #1<https://github.com/dunnfall/df39-it202-008/pull/21>**User Logout (1 pt.)**

^COLLAPSE ^

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

*The checkboxes are for your own tracking

#	Points	Details
<input checked="" type="checkbox"/> #1	1	Screenshot of the navigation when logged in (site)
<input checked="" type="checkbox"/> #2	1	Screenshot of the redirect to login with the user-friendly logged-out message (site)
<input checked="" type="checkbox"/> #3	1	Screenshot of the logout-related code showing the session is destroyed (code). Ensure uid/date comment is present.

Task Screenshots:**Gallery Style: Large View**

Small

Medium

Large



EXPLORER

- ... DF39-IT202-008
- lib
 - .env
 - config.php
 - db.php
 - duplicate_user_details.php
 - flash_messages.php
 - functions.php
 - get_url.php
 - README.md
 - reset_session.php
 - safer_echo.php
 - sanitizers.php
 - user_helpers.php
- Module 1
 - partials
 - flash.php
 - nav.php
 - README.md
 - M4
 - Module3
- PHPProblems
 - df39_it202-m2-php-problems_IT202-008-S2024.pdf
 - Problem1.php
 - Problem2.php
 - Problem3.php

```
logout.php M X
public_html > Project > logout.php
1  <?php
2  session_start();
3  require(__DIR__ . "/../../lib/functions.php");
4  reset_session();
5
6  flash("Successfully logged out", "success");
7  header("Location: login.php");
8 //DF39 4/1/2024 StudentID:31523743
```

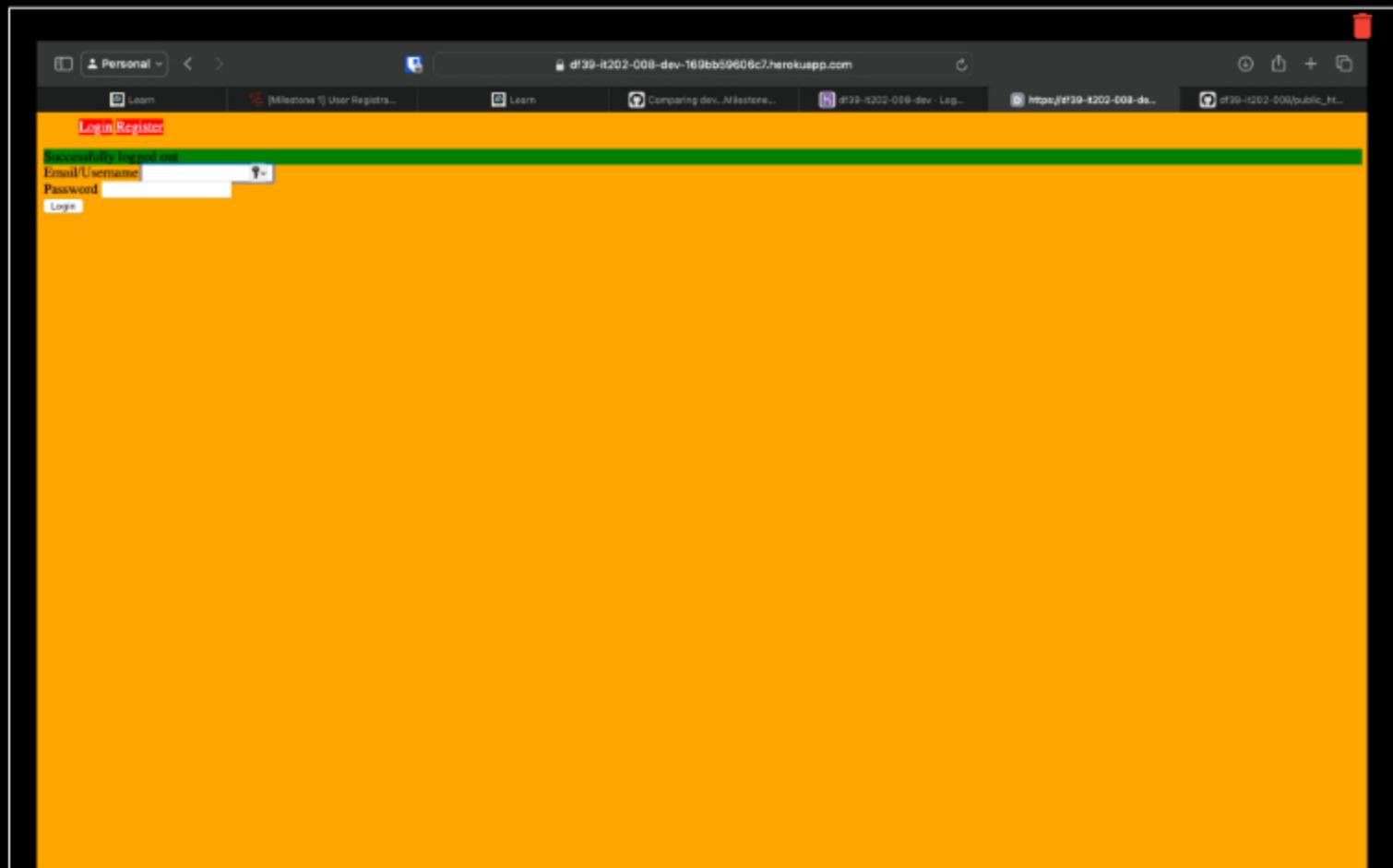
Logout related code, with session being destroyed and UCID and date.

Checklist Items (1)

#3 Screenshot of the logout-related code showing the session is destroyed (code). Ensure ucid/date comment is present.

Checklist Items (1)

#1 Screenshot of the navigation when logged in (site)



Redirect to login with logged out message

Checklist Items (1)

#2 Screenshot of the redirect to login with the user-friendly logged-out message (site)

Task #2 - Points: 1

Text: Include pull request links related to this feature

i Details:

Should end in /pull/#

URL #1

<https://github.com/dunnfall/df39-it202-008/pull/21>

i Basic Security Rules and Roles (2 pts.)

A COLLAPSE **▲**



[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

The screenshot shows a code editor interface with a search results panel on the left and the content of `user_helpers.php` on the right.

Search Results:

- 16 results in 5 files - Open in editor
- `user_helpers.php` M ↗ has_role
- `login.php` M ↗
- `register.php` I ↗
- `profile.php` S ↗

File Content (`user_helpers.php`):

```
1 <?php
2
3 /**
4 * Passing $redirect as true will auto redirect a logged out user to the $destination.
5 * The destination defaults to login.php
6 */
7 //EF39 4/1/2024
8
9 function is_logged_in($redirect = false, $destination = "login.php")
10 {
11     $isLoggedIn = isset($_SESSION["user"]);
12     if ($redirect && !$isLoggedIn) {
13         flash("You must be logged in to..."); // This triggers, the calling script won't receive a reply since die()/exit() terminates it
14         die(header("Location: $destination"));
15     }
16     return $isLoggedIn;
17 }
18
19 function has_role($role)
20 {
21     if (is_logged_in() && isset($_SESSION["user"]["roles"])) {
22         foreach ($_SESSION["user"]["roles"] as $role) {
23             if ($role === $role) {
24                 return true;
25             }
26         }
27     }
28     return false;
29 }
30
31 function get_username()
32 {
33     if (is_logged_in()) { // We need to check for login first because "user" key may not exist
34         return set($_SESSION["user"], "username", "", false);
35     }
36     return "";
37 }
38
39 function get_user_email()
40 {
41     if (is_logged_in()) { // We need to check for login first because "user" key may not exist
42         return set($_SESSION["user"], "email", "", false);
43     }
44     return "";
45 }
46
47 function get_user_id()
48 {
49     if (is_logged_in()) { // We need to check for login first because "user" key may not exist
50         return set($_SESSION["user"], "id", false, false);
51     }
52     return false;
53 }
```

Login check function in user helpers

Checklist Items (1)

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

The screenshot shows a code editor with the file `profile.php` open. The code contains logic for saving user information to a database. It includes error handling for email availability and displays error messages if the save operation fails.

```
<?php
require_once(__DIR__ . "/../../../../partials/nav.php");
//DF39 4/1/2024
is_logged_in(true);
?>
<?php
if (isset($_POST["save"])) {
    $email = se($_POST, "email", null, false);
    $username = se($_POST, "username", null, false);

    $params = ["email" => $email, "username" => $username, "id" => get_user_id()];
    $db = getDB();
    $stmt = $db->prepare("UPDATE Users set email = :email, username = :username where id = :id");
    try {
        $stmt->execute($params);
        flash("Profile saved", "success");
    } catch (Exception $e) {
        if ($e->errorInfo[1] === 1062) {
            //https://www.php.net/manual/en/function.preg-match.php
            preg_match("/Users.(\w+)", $e->errorInfo[2], $matches);
            if (!isset($matches[1])) {
                flash("The chosen " . $matches[1] . " is not available.", "warning");
            } else {
                //TODO come up with a nice error message
                echo "<pre>" . var_export($e->errorInfo, true) . "</pre>";
            }
        } else {
            //TODO come up with a nice error message
            echo "<pre>" . var_export($e->errorInfo, true) . "</pre>";
        }
    }
}
```

The logged in function is being used in nav.php home.php and profile.php

Checklist Items (2)

#2 Screenshot of the login check function being used (i.e., profile likely). Also caption what pages it's used on

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

The screenshot shows a web browser displaying a login form. The URL is `df39-it202-008-dev-169bb59606c7.herokuapp.com`. The page displays a yellow banner at the top stating "You must be logged in to view this page". Below the banner are fields for "Email/Username" and "Password", and a "Login" button.

User friendly message of accessing a page that requires to be logged in

Checklist Items (1)

#4 Demonstrate the user-friendly message of trying to manually access a login-protected page while being logged out

Task #2 - Points: 1

Text: Authorization Screenshots

Checklist

*The checkboxes are for your own tracking

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

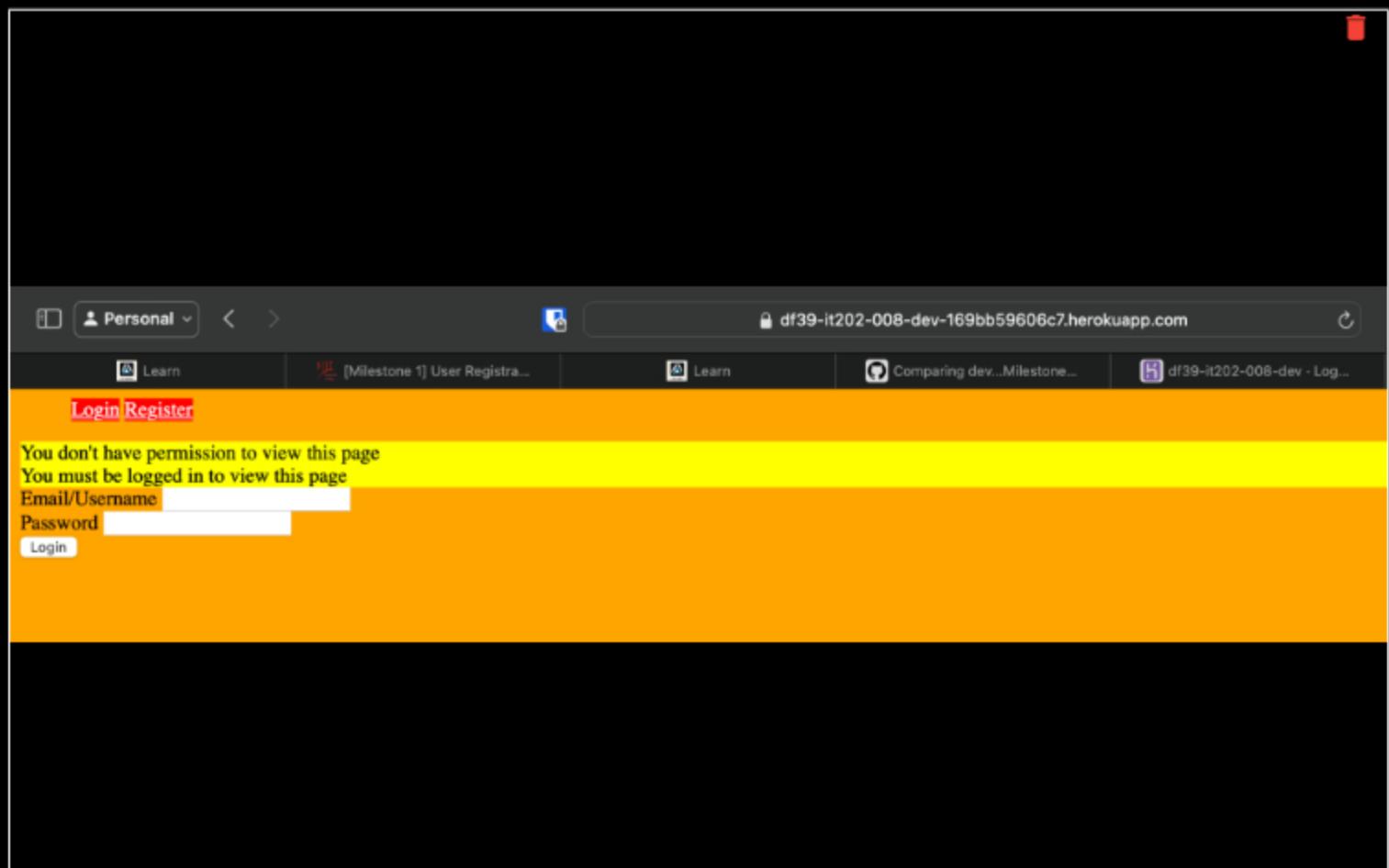
```
//DF39 4/1/2024
function has_role($role)
{
    if (is_logged_in() && isset($_SESSION["user"]["roles"])) {
        foreach ($_SESSION["user"]["roles"] as $r) {
            if ($r["name"] === $role) {
                return true;
            }
        }
    }
    return false;
}
```

Function that checks for specific roles with UCID and date

Checklist Items (2)

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

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



Message when trying to access role-restricted page

Checklist Items (1)

#4 Demonstrate the user-friendly message of trying to manually access a role-protected page while being logged out

```
27  <!-- include css and js files -->
28  <!-- DF39 4/1/2024 -->
29  <link rel="stylesheet" href="php echo get_url('styles.css'); ?&gt;"&gt;
30  &lt;script src="<?php echo get_url('helpers.js'); ?&gt;"&gt;&lt;/script&gt;
31  &lt;nav&gt;
32      &lt;ul&gt;
33          &lt;?php if (is_logged_in()) : ?&gt;
34              &lt;li&gt;&lt;a href="<?php echo get_url('home.php'); ?&gt;"&gt;Home&lt;/a&gt;&lt;/li&gt;
35              &lt;li&gt;&lt;a href="<?php echo get_url('profile.php'); ?&gt;"&gt;Profile&lt;/a&gt;&lt;/li&gt;
36          &lt;?php endif; ?&gt;
37          &lt;?php if (!is_logged_in()) : ?&gt;
38              &lt;li&gt;&lt;a href="<?php echo get_url('login.php'); ?&gt;"&gt;Login&lt;/a&gt;&lt;/li&gt;
39              &lt;li&gt;&lt;a href="<?php echo get_url('register.php'); ?&gt;"&gt;Register&lt;/a&gt;&lt;/li&gt;
40          &lt;?php endif; ?&gt;
41          &lt;?php if (has_role("Admin")) : ?&gt;
42              &lt;li&gt;&lt;a href="<?php echo get_url('admin/create_role.php'); ?&gt;"&gt;Create Roles&lt;/a&gt;&lt;/li&gt;</pre
```

```

42
43     <li><a href=<?php echo get_url('admin/create_role.php'); ?>">Create Role</a></li>
44     <li><a href=<?php echo get_url('admin/list_roles.php'); ?>">List Roles</a></li>
45     <li><a href=<?php echo get_url('admin/assign_roles.php'); ?>">Assign Roles</a></li>
46     <?php endif; ?>
47     <?php if (is_logged_in()) : ?>
48         <li><a href=<?php echo get_url('logout.php'); ?>">Logout</a></li>
49     <?php endif; ?>
50 </ul>

```

This function of has_role is used in nav.php, assign_roles.php, create_role.php and list_roles.php

Checklist Items (1)

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

Task #3 - Points: 1

Text: Screenshots of UserRoles and Roles Tables

Checklist

*The checkboxes are for your own tracking

#	Points	Details
<input checked="" type="checkbox"/> #1	1	At least one valid and enabled User->Role reference (UserRoles table)
<input checked="" type="checkbox"/> #2	1	UserRoles Table should have id, user_id, role_id, is_active, created, and modified columns
<input checked="" type="checkbox"/> #3	1	Roles Table should have id, name, description, is_active, modified, and created columns
<input checked="" type="checkbox"/> #4	1	At least one valid and enabled Role (Roles table)
<input checked="" 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 'UserRoles' table selected. The table structure is displayed with columns: id, user_id, role_id, is_active, created, and modified. Two rows of sample data are shown.

	Q	id	user_id	role_id	is_active	created	modified	
	>	1	10	10	-1	1	2024-04-01 04:15:13	2024-04-01 04:15:13
	>	2	11	10	2	1	2024-04-01 04:32:47	2024-04-01 04:32:47

email	varchar(100)	> 3	12	10	4	1	2024-04-01 04:32:47	2024-04-01 04:32:47							
password	varchar(6...	> 4	13	11	-1	1	2024-04-01 04:45:23	2024-04-01 04:45:23							
created	timestamp														
modified	timestamp														
username	varchar(3...														
> index															
> partitions															
Views															
No views found.															

UserRoles has id, user_id, role_id, is_active, created, modified columns. Also has one valid and enabled user.

Checklist Items (2)

#1 At least one valid and enabled User->Role reference (UserRoles table)

#2 UserRoles Table should have id, user_id, role_id, is_active, created, and modified columns

Properties Data Process						
SELECT * FROM `Roles` LIMIT 100						
Q	P	id	name	description	is_active	created
		int	varchar(255)	varchar(1000)	tinyint(1)	timestamp
> 1	> 1	1	Admin		1	2024-04-01 03:57:31
> 2	> 2	2	Video Role	Video	1	2024-04-01 04:24:11
> 3	> 3	4	Nice Role		1	2024-04-01 04:24:24
						2024-04-01 04:25:34

Roles table with id, name, description, is_active, created, and modified columns. Left side panel has UCID with one valid and enable role

Checklist Items (3)

#3 Roles Table should have id, name, description, is_active, modified, and created columns

#4 At least one valid and enabled Role (Roles table)

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



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

Response:

The purpose of the user roles table is to be a middle man table for other tables. It manages the relationship between the Users table and Roles table as well as assign roles to users and manage their permissions. Roles.is_active indicates if the role in the table is currently active and can be turned on and off, whereas UserRoles.is_active is a soft switch where the user can activate to deactivate their 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/dunnfall/df39-it202-008/pull/34>



^COLLAPSE ^

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 checked="" type="checkbox"/> #1	1	Heroku dev url should be present in the address bar
<input checked="" type="checkbox"/> #2	1	Should have thoughtful CSS applied
<input checked="" type="checkbox"/> #3	1	Show the profile form correctly populated on page load (username, email)
<input checked="" type="checkbox"/> #4	1	Should have the following fields: username, email, current password, new password, confirm password (or similar)

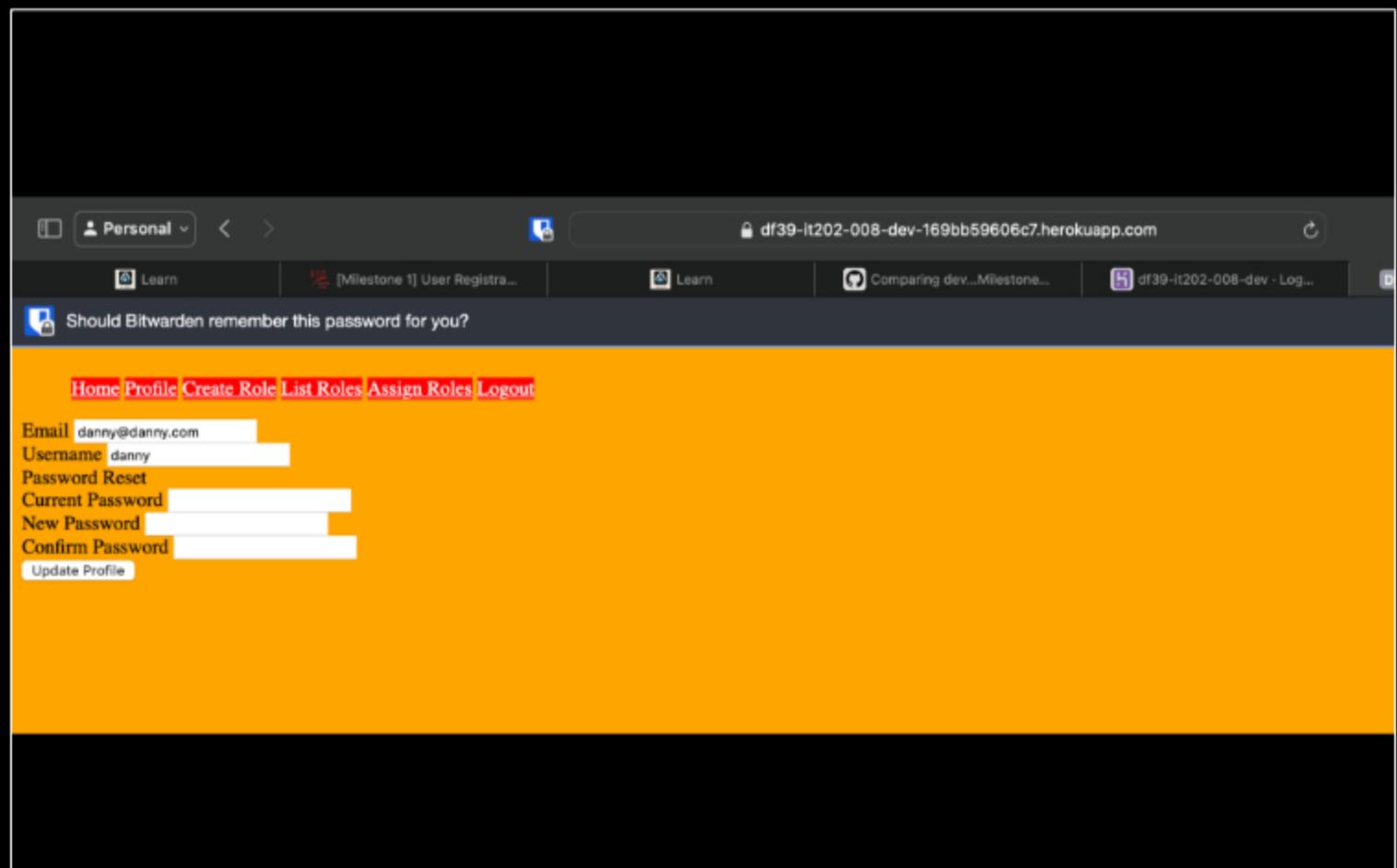
Task Screenshots:

Gallery Style: Large View

Small

Medium

Large



Heroku dev in url bar CSS applied Profile page populated with username, email, password reset, new password, confirm password

Checklist Items (4)

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

#2 Should have thoughtful CSS applied

#3 Show the profile form correctly populated on page load (username, email)

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

 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:

Response:

When the page is loaded the nav.php file is included to ensure the nav.php code is implemented at the top of the page for the user. As well as the is_logged_in(true) function is called to ensure the user is logged in. If the user is logged in it will call the nav.php file and display the nav and the rest of the profile.php file.

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

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

- Address Bar:** df39-it202-008-dev-169bb59606c7.herokuapp.com
- Page Title:** [Milestone 1] User Registration +...
- Header:** Personal
- Navigation:** Back, Forward, Stop, Refresh
- Content:**
 - Profile picture placeholder:
 - Profile information: Learn, Comparing dev...Milestone1 - dun...
 - Links: Home, Profile, Create Role, List Roles, Assign Roles, Logout
 - Form fields:
 - Email: danny@danny.com
 - Username: dunman
 - Password Reset
 - Current Password
 - New Password
 - Confirm Password
- Bottom Status Bar:** https://df39...

Update Profile

Heroku dev in URL CSS applied Before username change

Checklist Items (3)

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

#2 Should have thoughtful CSS applied

#3 Demonstrate with before and after of a username change (including success message)

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

- Address Bar:** df39-it202-008-dev-169bb59606c7.herokuapp.com
- Header:** Personal dropdown, Learn button, Comparing dev...Milestone1 - dun..., https://df39-it20...
- Navigation:** Home, Profile (highlighted in red), Create Role, List Roles, Assign Roles, Logout.
- Profile Section:** Profile saved. Fields shown: Email (danny@danny.com), Username (danman), Password Reset, Current Password, New Password, Confirm Password.
- Buttons:** Update Profile (highlighted in red).

After username change with success message

Checklist Items (1)

#3 Demonstrate with before and after of a username change (including success message)

Email

Username

Password Reset

Current Password

New Password

Confirm Password

Before email change

Checklist Items (1)

#4 Demonstrate with a before and after of an email change (including success message)

Profile saved

Email

Username

Password Reset

Current Password

New Password

Confirm Password

After email change with success message

Checklist Items (1)

#4 Demonstrate with a before and after of an email change (including success message)

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

Profile saved
Password reset

Email

Username

Password Reset

Current Password

New Password

Confirm Password

Updating password message

Checklist Items (1)

#5 Demonstrate the success message of updating password

The screenshot shows a browser window with a user registration form. The URL is <https://dt39-lt202-008-dev-169bb59606c7.herokuapp.com>. The page title is "[Milestone 1] User Registration +..". The form has several validation errors:

- Should Bitwarden remember this password for you? (Error: Please enter a valid email address (CLIENT))
- Username (Error: Username must be between 2 and 30 characters long, and be alphanumeric characters only (CLIENT))
- Email (Error: Please enter a valid email address (CLIENT))
- Password (Error: Password must be at least 8 characters long (CLIENT))
- Password and Confirm password must match (CLIENT)

The developer tools (Elements tab) are open, showing the following JavaScript code:

```
console.log('Removed $attr from element ${ins.name || "(No name)"');
}

// Change type to text if not already text
if (!["text", "submit", "reset"].includes(ins.type)) {
    ins.type = "text";
    console.log('Changed type to text for element ${ins.name || "(No name)"');
}

}
}

alert("HTML Validation has been disabled until page reload");
})();

```

The status bar at the bottom shows: Changed type to text for element email.

```
↳ Changed type to text for element currentPassword  
↳ Changed type to text for element newPassword  
↳ Changed type to text for element confirmPassword  
↳ undefined
```

Username format message shown Email format message shown Password format message shown Confirm Password message shown

Checklist Items (1)

#6 Demonstrate JavaScript user-friendly validation messages (email format, username format, password format, new password matching confirm password)

The screenshot shows a web application interface with a navigation bar at the top containing links: Home, Profile, Create Role, List Roles, Assign Roles, and Logout. Below the navigation bar, there are several input fields and error messages. The 'Username' field contains 'Danny' and has a red border, indicating it is invalid. The 'Email' field contains 'danny@danny.com' and also has a red border. Below these fields, four error messages are displayed in yellow boxes: 'The chosen username is not available.', 'Current password is invalid', 'Email danny@danny.com is already in use.', and 'Email danny@danny.com is already in use.' (repeated). At the bottom left is a 'Update Profile' button.

The chosen username is not available.
Current password is invalid
Email danny@danny.com is already in use.
Email danny@danny.com is already in use.

Email danny@danny.com
Username Danny
Password Reset
Current Password
New Password
Confirm Password

Update Profile

Username not available message shown Current password invalid message shown

Checklist Items (1)

#7 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)

The screenshot shows a web application interface with a navigation bar at the top containing links: Home, Profile, Create Role, List Roles, Assign Roles, and Logout. Below the navigation bar, there are several input fields and error messages. The 'Email' field contains 'danny@danny.com' and has a red border, indicating it is invalid. The 'Username' field contains 'Danny' and also has a red border. Below these fields, three error messages are displayed in yellow boxes: 'The chosen email is not available.', 'The chosen username is not available.', and 'The chosen email is not available.' (repeated). At the bottom left is a 'Create Role' button.

The chosen email is not available.
The chosen username is not available.
The chosen email is not available.

Email danny@danny.com
Username Danny
Password Reset
Current Password

New Password

Confirm Password

Update Profile

Email not available message

Checklist Items (1)

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

Response:

Username:

The PHP code will check if the form was submitted by checking if the save key is in the \$_POST array, if it isn't the check fails. If the form gets submitted the username and email get sanitized by the PHP code. Before the form gets submitted client side validation will be checked to see if the format for the username/email is correct and if it isn't it will not submit the form and send a friendly flash message to the user. After the form is submitted the PHP will execute an SQL statement that updates the username and email. If an error occurs it will check if the error is 1062 which means a duplicate entry has been made and will display a friendly message to the user saying the entry is already in use. If there is no duplicate entry the username and email will be updated. PHP will then fetch these changes and store the user data in the session.

Password:

Before the form is submitted the client side validation (java script) will be called after the user attempts to submit the form and ensure formatting for the password is correct and that the password and confirm password fields are matching. If it isn't the form will not be submitted. Once the format is correct the form will be submitted. Server side will also check if the passwords match. If they match, a SQL statement will fetch the users current password from the data base, if the current password entry matches the user entered current password SQL statement is ran to update the users password in the data base. If they don't match a message will appear telling the user the current password is incorrect.

[^COLLAPSE](#)

Task #5 - Points: 1

Text: Include pull request links related to this feature

i Details:

Should end in /pull/#

URL #1

<https://github.com/dunnfall/df39-it202-008/pull/31>



Misc (1 pt.)

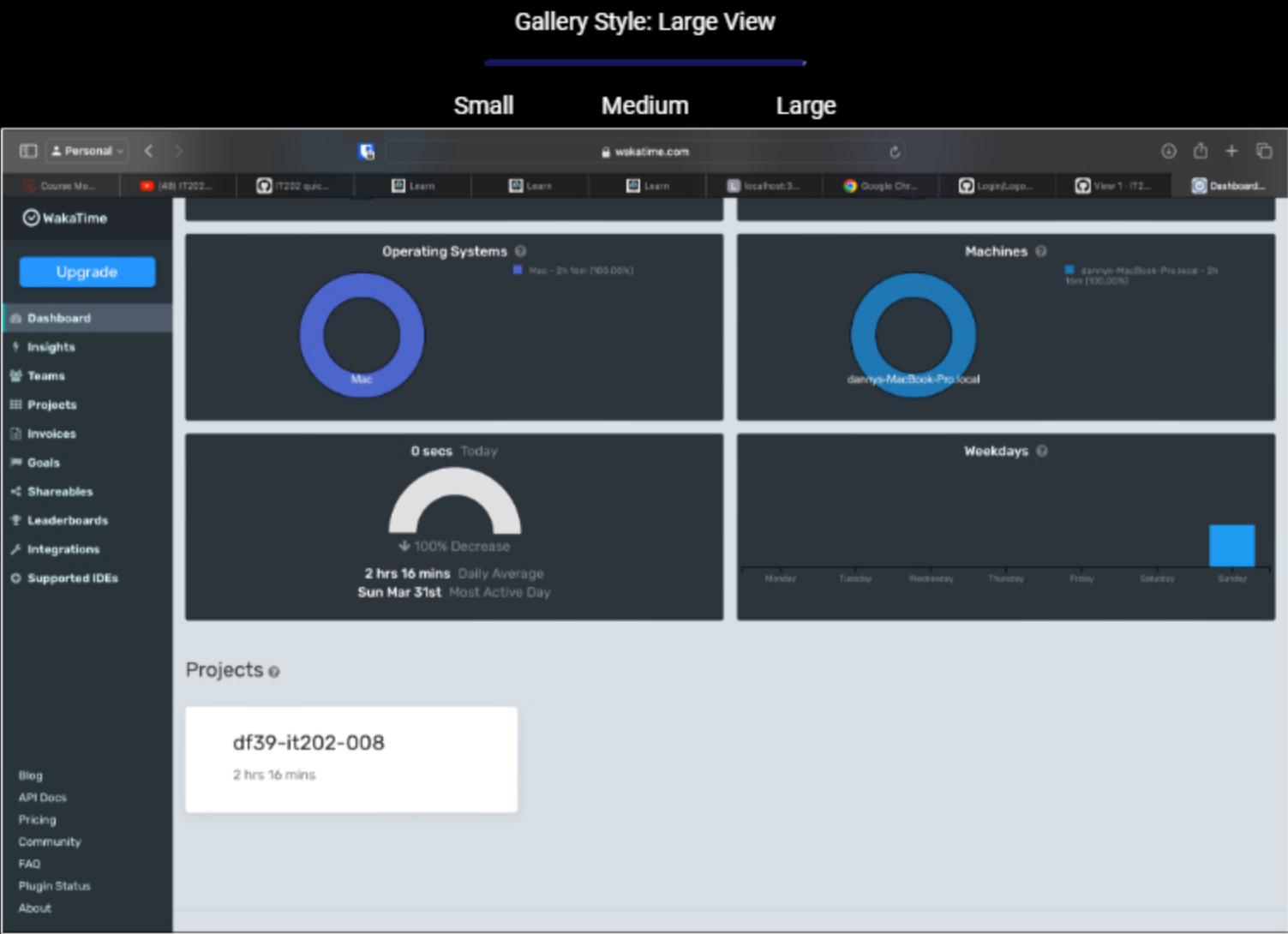
[^COLLAPSE](#)



Task #1 - Points: 1

Text: Screenshot of wakatime

Task Screenshots:



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

The screenshot shows a GitHub project board for 'IT202-008'. At the top, there's a search bar and various project management buttons. Below the header, there are three columns: 'Todo', 'In Progress', and 'Done'. Each column has a brief description: 'Todo' says 'This item hasn't been started', 'In Progress' says 'This is actively being worked on', and 'Done' says 'This has been completed'. Under the 'Done' column, there are five items, each with a small icon, a link, and a brief description:

- df39-it202-008 #18 MS1 - User Will Be Able To Register a New Account
- df39-it202-008 #20 MS1 - Users Will Be Able to Login To Their Account
- df39-it202-008 #22 MS1 - User will be able to logout
- df39-it202-008 #23 MS1 - Basic Security Rules Implemented
- df39-it202-008 #24 MS1 - Basic Roles Implemented
- df39-it202-008 #25 MS1 - Site Should Have Basic Styles/Theme Applied

At the bottom of each column, there's a '+ Add item' button.

Project board for Milestone 1

Task #3 - Points: 1

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

URL #1

<https://github.com/users/dunnfall/projects/1>

Task #4 - Points: 1

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

URL #1

<https://df39-it202-008-prod-8b39da6847aa.herokuapp.com/Project/login.php>

End of Assignment