

Submission Worksheet

CLICK TO GRADE

<https://learn.ethereallab.app/assignment/IT202-007-F2024/it202-module-6-milestone-1-2024/grade/jd755>

Course: IT202-007-F2024

Assignment: [IT202] Module 6 Milestone 1 2024

Student: Jeremy R. (jd755)

Submissions:

Submission Selection

1 Submission [submitted] 11/9/2024 6:59:45 PM

Instructions

[^ COLLAPSE ^](#)

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

Prereqs:

- Go through each lesson from "Project Setup and SQL" to "User Login Enhancement" and follow the branching names while gathering the code
- Merge each into Milestone1 branch
- Create a Project board on GitHub (if you haven't yet)
 - Add each major item from the proposal doc as an Issue item
 - Invite the grader(s) and myself as collaborators on the board (they're separate from your repository)
 - See Canvas announcements for the Usernames or check your collab list on your repo
- Mark the related GitHub Issues items as "done"
- Implement your own custom CSS (something much different than the default "ugly" CSS given as an example)
- Consider styling all forms/inputs, data output, navigation, etc
- Implement JavaScript validation on Register, Login, and Profile (include "[Client]" in the output messages to differentiate between server-side validations)

Instructions:

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

- 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

Group



Group: User Registration

Tasks: 6

Points: 2

COLLAPSE

Task



Group: User Registration

Task #1: Screenshot of form on website page

Weight: ~17%

Points: ~0.33

COLLAPSE

i Details:

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



Columns: 1

Sub-Task



Group: User Registration

Task #1: Screenshot of form on website page

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

Task Screenshots

Gallery Style: 2 Columns

4

2

1



Registration page

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

↪ Task URLs

URL #1

Missing URL

URL

Sub-Task

Group: User Registration

100%

Task #1: Screenshot of form on website page

Sub Task #2: 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)

▣ Task Screenshots

Gallery Style: 2 Columns

4 2 1

The screenshot shows a registration form with the following validation errors displayed in yellow boxes:

- [Client] Invalid email address
- [Client] Username must only contain 3-30 characters a-z, 0-9, _ or -
- [Client] Password must be 8 characters or more

The form fields include Email, Username, Password, and Confirm.

Client side warnings for invalid email, username, and password

The screenshot shows a registration form with the following validation error displayed in a yellow box:

- [Client] Passwords must match

The form fields include Email, Username, Password, Confirm, and Register.

Warning about passwords not matching

The screenshot shows a registration form with the following validation errors displayed in yellow boxes:

- [Client] Email must not be empty
- [Client] Username must not be empty
- [Client] Password must not be empty
- [Client] Confirm password must not be empty

The form fields include Email, Username, Password, and Confirm.

Empty fields

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: User Registration

100%

Task #1: Screenshot of form on website page

Sub Task #3: Demonstrate email already in use message (message text doesn't need to be exact, but should be clear)

▣ Task Screenshots

Gallery Style: 2 Columns

4 2 1

The screenshot shows a user registration form with a purple header containing the text "Login Register". Below the header, there is a yellow warning bar with the text "The chosen email is not available.". The main form area has five input fields labeled "Email", "Username", "Password", "Confirm", and "Register".

Warning when email is already used

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: User Registration

100%

Task #1: Screenshot of form on website page

Sub Task #4: Demonstrate username already in use message (message text doesn't need to be exact, but should be clear)

Task Screenshots

Gallery Style: 2 Columns

4 2 1

The screenshot shows a user registration form with a purple header containing the text "Login Register". Below the header, there is a yellow warning bar with the text "The chosen username is not available.". The main form area has five input fields labeled "Email", "Username", "Password", "Confirm", and "Register".

Warning when username is already in use

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: User Registration

100%

Task #1: Screenshot of form on website page

Sub Task #5: Demonstrate user-friendly message of new account being created

Task Screenshots

Gallery Style: 2 Columns

4 2 1

The screenshot shows a user registration form with a purple header containing the text "Login Register". Below the header, there is a yellow success message bar with the text "Your account has been created successfully!". The main form area has five input fields labeled "Email", "Username", "Password", "Confirm", and "Register".

A screenshot of a web-based user registration form. The form has a purple header with the word 'Email' and a blue footer with the word 'Response'. It contains five input fields: 'Email', 'Username', 'Password', 'Confirm', and 'Response'. Each field has a placeholder text above it.

Message when registration is successful

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 1

Task

Group: User Registration

Task #2: Screenshot of the form code

Weight: ~17%

Points: ~0.33

100%

▲ COLLAPSE ▾

ⓘ Details:

Should have the appropriate type attributes for the fields.

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



Sub-Task

Group: User Registration

Task #2: Screenshot of the form code

Sub Task #1: Show the html of the form

100%

🖼 Task Screenshots

Gallery Style: 2 Columns

4

2

1

A screenshot of the HTML code for the user registration form. The code is as follows:

```
<?php
require(__DIR__ . "/../../partials/nav.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>
```

HTML screenshot 1

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

≡, Task Response Prompt

Briefly explain the html for each field including the chosen attributes

Response:

Responses

This HTML consists of a form element that contains 4 div elements. The form element is of method post as we don't want the data of the form to easily accessible. The onsubmit attribute is set to call the javascript function 'validate'. The first div element holds a label for email and has an input of type email. The other 3 div elements follow a similar pattern but for username, password, and confirm password respectively.

End of Task 2

Task



Group: User Registration

Task #3: Screenshot of the client-side and server-side validation code

Weight: ~17%

Points: ~0.33

COLLAPSE

Details:

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



Columns: 1

Sub-Task



Group: User Registration

Task #3: Screenshot of the client-side and server-side validation code

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

Task Screenshots

Gallery Style: 2 Columns

4	2	1
<pre>// 2023-11-09T24:55:16.161Z function flash(message = "", color = "INFO") function validate_email(email) { console.log("Inside validate_email"); emailRegex = /^[a-zA-Z\d\-_\.]+@[a-zA-Z\d\-_]+\.(a-zA-Z){2,6}\$/; return (emailRegex.test(email)); } // 18-22 function validate_email(email) function validate_username(username) { usernameRegex = /^[a-zA-Z\d\-_]{3,16}\$/; return (usernameRegex.test(username)); }</pre>	<pre>// 2023-11-09T24:55:16.161Z function flash(message = "", color = "INFO") function validate_email(email) { console.log("Inside validate_email"); emailRegex = /^[a-zA-Z\d\-_\.]+@[a-zA-Z\d\-_]+\.(a-zA-Z){2,6}\$/; return (emailRegex.test(email)); } // 18-22 function validate_email(email) function validate_username(username) { usernameRegex = /^[a-zA-Z\d\-_]{3,16}\$/; return (usernameRegex.test(username)); } // 2023-11-09T24:55:16.161Z function validate_password(password) function validate_confirm_password(confirmPassword) { if (password.length < 8) { flash("Password must be at least 8 characters long.", "WARNING"); return false; } else if (password !== confirmPassword) { flash("Passwords must match.", "WARNING"); return false; } else { flash("Password must not be empty.", "WARNING"); return true; } }</pre>	<pre>// 2023-11-09T24:55:16.161Z function flash(message = "", color = "INFO") function validate_email(email) { console.log("Inside validate_email"); emailRegex = /^[a-zA-Z\d\-_\.]+@[a-zA-Z\d\-_]+\.(a-zA-Z){2,6}\$/; return (emailRegex.test(email)); } // 18-22 function validate_email(email) function validate_username(username) { usernameRegex = /^[a-zA-Z\d\-_]{3,16}\$/; return (usernameRegex.test(username)); } // 2023-11-09T24:55:16.161Z function validate_password(password) function validate_confirm_password(confirmPassword) { if (password.length < 8) { flash("Password must be at least 8 characters long.", "WARNING"); return false; } else if (password !== confirmPassword) { flash("Passwords must match.", "WARNING"); return false; } else { flash("Password must not be empty.", "WARNING"); return true; } }</pre>

Client side validation 1

Client side validation 2

```
// 2023-11-09T24:55:16.161Z function flash(message = "", color = "INFO")  
function validate_email(email)  
{  
    console.log("Inside validate_email");  
    emailRegex = /^[a-zA-Z\d\-\_\.]+@[a-zA-Z\d\-\_]+\.(a-zA-Z){2,6}$/;  
    return (emailRegex.test(email));  
} // 18-22 function validate_email(email)  
function validate_username(username)  
{  
    usernameRegex = /^[a-zA-Z\d\-\_]{3,16}$/;  
    return (usernameRegex.test(username));  
}
```

Two new functions I added in helpers.js

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

I started by getting all of the values necessary from the HTML form element by using the `.value` property. Email I first check if the length of the email is 0. If it is, I send a warning saying that the mail field cannot be blank. If it's not blank, I pass the email to the validate email function I created in `helpers.js`. If it returns false, I send a warning that the email is invalid. Username I used the same method as the email above but I used the validate username function I created in `helpers.js` and my warning message explains what a valid username must consist of. Password & Confirm Password I check if the password and confirm password is equal to 0. If it is, I sent out appropriate warnings that they cannot be empty. After, I check if password is less than 8. If it is, I send a warning saying the password can be less than 8 characters. Finally, I check if password and confirm password are equal. If they aren't, I send a warning.

Sub-Task

Group: User Registration

100%

Task #3: Screenshot of the client-side and server-side validation code

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

Task Screenshots

Gallery Style: 2 Columns

PHP validation 2

```

17    else {
18        String username = mail.getHeader("From");
19        String password = mail.getHeader("password");
20        String host = mail.getHeader("host");
21        String port = mail.getHeader("port");
22        String ssl = mail.getHeader("ssl");
23        String starttls = mail.getHeader("starttls");
24        String danger = mail.getHeader("danger");
25        String username2 = mail.getHeader("username2");
26        String password2 = mail.getHeader("password2");
27        String host2 = mail.getHeader("host2");
28        String port2 = mail.getHeader("port2");
29        String ssl2 = mail.getHeader("ssl2");
30        String starttls2 = mail.getHeader("starttls2");
31        String danger2 = mail.getHeader("danger2");
32        String username3 = mail.getHeader("username3");
33        String password3 = mail.getHeader("password3");
34        String host3 = mail.getHeader("host3");
35        String port3 = mail.getHeader("port3");
36        String ssl3 = mail.getHeader("ssl3");
37        String starttls3 = mail.getHeader("starttls3");
38        String danger3 = mail.getHeader("danger3");
39        String username4 = mail.getHeader("username4");
40        String password4 = mail.getHeader("password4");
41        String host4 = mail.getHeader("host4");
42        String port4 = mail.getHeader("port4");
43        String ssl4 = mail.getHeader("ssl4");
44        String starttls4 = mail.getHeader("starttls4");
45        String danger4 = mail.getHeader("danger4");
46    }
47
48    if (username == null) {
49        return danger;
50    }
51
52    if (password == null) {
53        return danger;
54    }
55
56    if (host == null) {
57        return danger;
58    }
59
60    if (port == null) {
61        return danger;
62    }
63
64    if (ssl == null) {
65        return danger;
66    }
67
68    if (starttls == null) {
69        return danger;
70    }
71
72    if (danger == null) {
73        return danger;
74    }
75
76    if (username2 == null) {
77        return danger;
78    }
79
80    if (password2 == null) {
81        return danger;
82    }
83
84    if (host2 == null) {
85        return danger;
86    }
87
88    if (port2 == null) {
89        return danger;
90    }
91
92    if (ssl2 == null) {
93        return danger;
94    }
95
96    if (starttls2 == null) {
97        return danger;
98    }
99
100   if (danger2 == null) {
101       return danger;
102   }
103
104   if (username3 == null) {
105       return danger;
106   }
107
108   if (password3 == null) {
109       return danger;
110   }
111
112   if (host3 == null) {
113       return danger;
114   }
115
116   if (port3 == null) {
117       return danger;
118   }
119
120   if (ssl3 == null) {
121       return danger;
122   }
123
124   if (starttls3 == null) {
125       return danger;
126   }
127
128   if (danger3 == null) {
129       return danger;
130   }
131
132   if (username4 == null) {
133       return danger;
134   }
135
136   if (password4 == null) {
137       return danger;
138   }
139
140   if (host4 == null) {
141       return danger;
142   }
143
144   if (port4 == null) {
145       return danger;
146   }
147
148   if (ssl4 == null) {
149       return danger;
150   }
151
152   if (starttls4 == null) {
153       return danger;
154   }
155
156   if (danger4 == null) {
157       return danger;
158   }
159
160   return danger;
161 }
```

PHP validation 1

```
1 <?php
2
3 function sanitize_email($email = "")
4 {
5 |     return filter_var(trim($email), FILTER_SANITIZE_EMAIL);
6 }
7 function is_valid_email($email = "")
8 {
9 |     return filter_var(trim($email), FILTER_VALIDATE_EMAIL);
10 }
11 function is_valid_username($username)
12 {
13 |     return preg_match('/^[\w\.-]{3,16}$/', $username);
14 }
15 function is_valid_password($password)
16 {
17 |     return strlen($password) >= 8;
18 }
```

sanitizers.php in the lib folder

```
<?php
function users_check_duplicate($serverInfo)
{
    if ($serverInfo[1] === 1062) {
        //https://www.php.net/manual/en/function.preg-match.php
        //NOTE: this assumes your table name is 'users', edit it accordingly
        preg_match("users:(\w+)", $serverInfo[2], $matches);
        if (isset($matches[1])) {
            flash("The chosen " . $matches[1] . " is not available.", "warning");
        } else {
            //lets come up with a nice error message
            flash("An unhandled error occurred", "danger");
            //this will log the output to the terminal/console that's running the php server
            error_log(var_export($serverInfo, true));
        }
    } <- #0-21 else
} <- #0-22 function users_check_duplicate($serverInfo)
```

duplicate_user_details.php inside the lib folder

Lesson 11 - Task 3

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

The first part of the validation works in a similar way as the JavaScript validation from before. However, this time after the validation is complete and everything is fine, an sql statement is created and sent to the database. If this brings no errors, then a success statement is sent to the user. If not, the function `duplicate_user_details.php` is called which sends to the user the error that the chosen email/username was already taken.

End of Task 3

Task

Group: User Registration

100%

Task #4: Screenshot of the Users table with a valid user entry

Weight: ~17%

Points: ~0.33

▲ COLLAPSE ▲

Checklist

*The checkboxes are for your own tracking

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

Sub-Task

Group: User Registration

Task #4: Screenshot of the Users table with a valid user entry

Sub Task #1: Show valid data per the checklist

100%

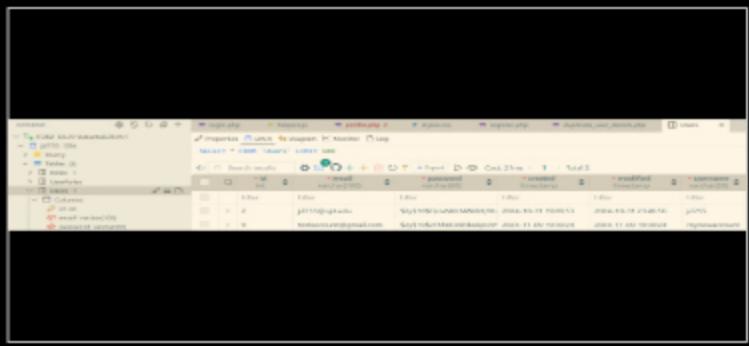
Task Screenshots

Gallery Style: 2 Columns

4

2

1



Database with user details

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 4

Task



Group: User Registration

Task #5: Explain the registration logic in a step-by-step manner from when the page loads to when the data is saved to the DB

Weight: ~17%

Points: ~0.33

[^ COLLAPSE ^](#)

① Details:

Don't just show code, translate things to plain English in concise steps.

May be worthwhile using a list.



☞ Task Response Prompt

Response:

1. User is prompted to enter an email, username, and password
2. Clientside and server side validation is used to check if the data entered into these fields are valid
3. Data is sent to database if it is valid. If the database detects a duplicate value, send user a warning message
4. If everything is clear, tell the user the registration is complete. Now the new data is inside the database.

End of Task 5

Task



Group: User Registration

Task #6: Include pull request links related to this feature

Weight: ~17%

Points: ~0.33

[^ COLLAPSE ^](#)

① Details:

Should end in /pull/#



☞ Task URLs

URL #1

<https://github.com/legitdaylight/jd755-IT202-007/pull/19>

URL

<https://github.com/legitdaylight/jd755-IT202-007>

URL #2

<https://github.com/legitdaylight/jd755-IT202-007/pull/21>

URL

<https://github.com/legitdaylight/jd755-IT202-007>

URL #3

URL

End of Task 6

End of Group: User Registration

Task Status: 5/6

Group

Group: User Login

Tasks: 4

Points: 2

100%

[▲ COLLAPSE ▲](#)

Task

Group: User Login

Task #1: Screenshot of form on website page

Weight: ~25%

Points: ~0.50

100%

[▲ COLLAPSE ▲](#)

i Details:

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



Columns: 1

Sub-Task

Group: User Login

100%

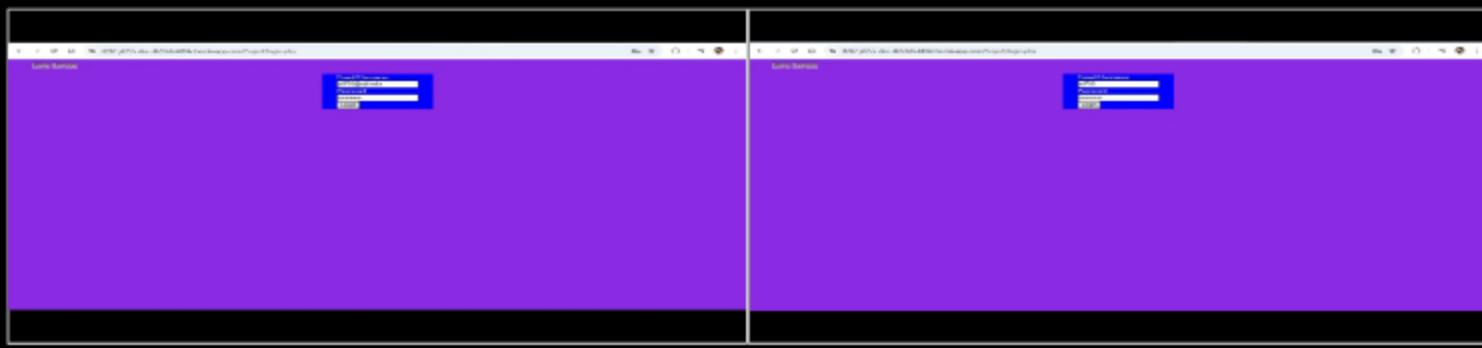
Task #1: Screenshot of form on website page

Sub Task #1: Two Screenshot of the form (one with valid email data filled and one with valid username data filled)

Task Screenshots

Gallery Style: 2 Columns

4 2 1



valid email

Valid username

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

☞ Task URLs

Sub-Task

Group: User Login

100%

Task #1: Screenshot of form on website page

Sub Task #2: Demonstrate JavaScript validation for each field [can be combined] (username format, email format, password format, and each field being required)

▣ Task Screenshots

Gallery Style: 2 Columns

4

2

1

A screenshot of a web browser displaying a login form. The form has three input fields: 'Email/Username' (empty), 'Password' (empty), and 'Login' (disabled). Above the form, there is a yellow header bar with the text 'Login Register' and some smaller, illegible text.

Empty username, email, and password

A screenshot of a web browser displaying a login form. The 'Email/Username' field contains 'ad75@gmail.com'. An error message 'Invalid email address.' is displayed above the form. The 'Password' field contains 'password1...' and the 'Login' button is visible. The browser's address bar shows the URL 'http://it202-jd755-dev-db59efc4494c.herokuapp.com/Project/login.php'.

Invalid email

A screenshot of a web browser displaying a login form. Both the 'Email/Username' field (containing 'ad75') and the 'Password' field (containing 'pass') have red error messages: 'Client] Invalid username' and 'Client] Password too short'. The 'Login' button is visible. The browser's address bar shows the URL 'http://it202-jd755-dev-db59efc4494c.herokuapp.com/Project/login.php'.

Invalid username, password too short

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: User Login

100%

Task #1: Screenshot of form on website page

Sub Task #3: Demonstrate user-friendly message of when an account doesn't exist

▣ Task Screenshots

Gallery Style: 2 Columns

4

2

1

A screenshot of a web browser displaying a login form. The 'Email/Username' field (containing 'ad75') has a red error message 'Email not found.' Above the form, there is a yellow header bar with the text 'Login Register' and some smaller, illegible text.

Password

Login

email/username not found

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: User Login

100%

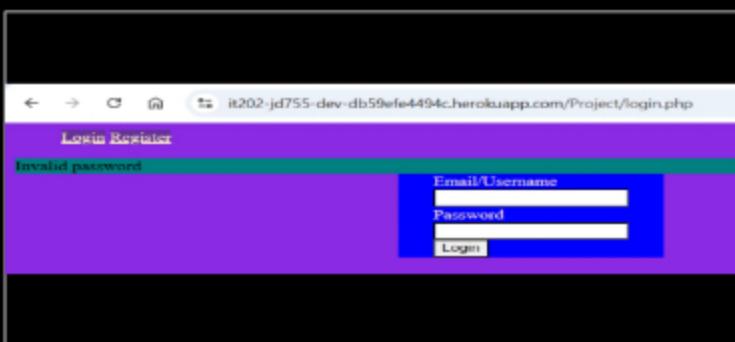
Task #1: Screenshot of form on website page

Sub Task #4: Demonstrate user-friendly message of when password doesn't match what's in the DB

■ Task Screenshots

Gallery Style: 2 Columns

4 2 1



A screenshot of a web browser displaying a login form. The URL in the address bar is `it202-jd755-dev-db59fe4494c.herokuapp.com/Project/login.php`. The page has a purple header with 'Login' and 'Register' buttons. Below the header, a green banner displays the error message 'Invalid password'. The main content area contains two input fields labeled 'Email/Username' and 'Password', and a 'Login' button.

wrong password

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: User Login

100%

Task #1: Screenshot of form on website page

Sub Task #5: Demonstrate successful login message and the destination page (i.e., home or some landing/dashboard page)

■ Task Screenshots

Gallery Style: 2 Columns

4 2 1



A screenshot of a web browser displaying a dashboard or home page. The URL in the address bar is `it202-jd755-dev-db59fe4494c.herokuapp.com/Project/home`. The page has a purple header with 'Logout' and 'Profile' buttons. Below the header, a green banner displays the message 'Welcome back!'. The main content area is titled 'Home'.

home screen after sucessful log in

Caption(s) (required) ✓


```
//jd255 11/09/24
<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>
```

HTML of the form

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Briefly explain the html for each field including the chosen attributes

Response:

The HTML consists of a form element of method post and an onsubmit attribute that calls the JavaScript validate function. The form contains two divs: one for the email/password input and the other for the password input.

Sub-Task

Group: User Login

100%

Task #2: Screenshot of the form code

Sub Task #2: Show the JavaScript validations of the form (include any extra files related if you made separate files)

Task Screenshots

Gallery Style: 2 Columns

4

2

1

```
//jd255 11/09/24
function validateForm()
{
  //CODE 3: implement message validation
  //CODE 4: implement placeholder for an error and true for success
  let email = document.querySelector('#email');
  let password = document.querySelector('#pw');
  let notError = true;

  if(email.length === 0)
  {
    flash("[client] Email/username must not be empty", "warning");
    notError = false;
  }
  else
  {
    if(email.includes('@'))
    {
      if(validate_username(email))
      {
        if(notError)
        notError = true;
      }
      else
      {
        flash("[client] Invalid username", "warning");
        notError = false;
      }
    }
    else
    {
      if(validate_email(email))
      {
        if(notError)
        notError = true;
      }
      else
      {
        flash("[client] Invalid email address", "warning");
        notError = false;
      }
    }
  }
  return notError;
}
```

JS validation 1

```
//jd255 11/09/24
if(password.length === 0)
{
  flash("[client] Password must not be empty", "warning");
  notError = false;
}
else
{
  if(password.length < 8)
  {
    flash("[client] Password too short", "warning");
    notError = false;
  }
}
else
{
  //CODE 5: update clientside validation to check if it should
  //valid email or username
  return notError;
}

//CODE 6: Function validate(form)

```

JS Validation 2

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

Email/Username I first check if the email/password field is empty. If it is, I send a warning to the user that the username/password can't be empty. If it is not empty, I check if the email contains the character '@'. If it does, I call the validate email function I created in helpers.js and check if the email is valid. If the email does not contain that character, I treat it as a username and I called the validate username function in helpers.js. Password I check if the password field is 0. If it is, I send a warning to the user that the password field cannot be empty. If it is not empty, I check if the length of the password is less than 8. If it is, I send a warning to the user that the password is too short. If it is not, I return true. If both fields are not empty and both validations pass, I return true. If either field is empty or either validation fails, I return false.

check if the length is less than 8. If it is, I send a warning to the user that the password is too short.

Sub-Task

Group: User Login



Task #2: Screenshot of the form code

Sub Task #3: Show PHP validations (include any lib content)

Task Screenshots

Gallery Style: 2 Columns

4 2

2

1

PHP Validation 1

100

PHP Validation 2

PHP Validation 3

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works.

Response:

The basic validation works the same as above in the JavaScript. When this validation is complete with 0 errors, an SQL statement is created with all the data inputted and is sent to the database. If the SQL statement doesn't return anything, then this means that username/email doesn't exist. This sends a warning to the user. If the username/email does exist but if the password is wrong, then another warning is sent. If everything goes well, the user is redirected to the home page with a welcome message containing their username.

End of Task 2

Task

Group: 1 User Login

Task #3: 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
Weight: ~25%
Points: ~0.50

[▲ COLLAPSE ▲](#)

ⓘ Details:

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



May be worthwhile using a list.

☞ Task Response Prompt

Response:

1. The user enters in their email/username or password. Basic validation is done to check if these inputs are valid.
2. If the inputs are valid, this information is sent to the database. If the email/username doesn't exist or if the password doesn't match, a warning is sent to the user.
3. If the database check was successful, the user is redirected to the home screen.

End of Task 3

Task



Group: User Login
Task #4: Include pull request links related to this feature
Weight: ~25%
Points: ~0.50

[▲ COLLAPSE ▲](#)

ⓘ Details:

Should end in /pull/#



☞ Task URLs

URL #1

<https://github.com/legitdaylight/jd755-IT202-007/pull/22>

URL

<https://github.com/legitdaylight/jd755-IT202-007>

URL #2

<https://github.com/legitdaylight/jd755-IT202-007/pull/20>

URL

<https://github.com/legitdaylight/jd755-IT202-007>

End of Task 4

End of Group: User Login

Task Status: 4/4

Group

Group: User Logout

Tasks: 2

Points: 1

100%

▲ COLLAPSE ▲

Task

Group: User Logout

Task #1: Capture the following screenshots

Weight: ~50%

Points: ~0.50

100%

▲ COLLAPSE ▲

Columns: 1

Sub-Task

Group: User Logout

Task #1: Capture the following screenshots

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

100%

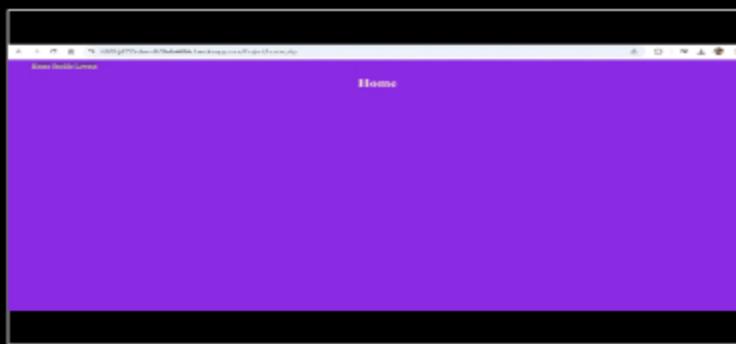
Task Screenshots

Gallery Style: 2 Columns

4

2

1



Home screen when logged in

Caption(s) (required) ✓Caption Hint: *Describe/highlight what's being shown***Sub-Task**

Group: User Logout

Task #1: Capture the following screenshots

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

100%

Task Screenshots

Gallery Style: 2 Columns

4

2

1



Logging out

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task



Group: User Logout

Task #1: Capture the following screenshots

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1

```
1 //jd755 11/09/24
2 <?php
3 session_start();
4 require(__DIR__ . "/../../../../lib/functions.php");
5 reset_session();
6
7 flash("Successfully logged out", "success");
8 header("Location: login.php");
```

Log Out code

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

The details of the session are first deleted. Then the user is redirected to the log in page and a flash message appears stating they were logged out.

End of Task 1

Task



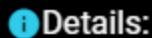
Group: User Logout

Task #2: Include pull request links related to this feature

Weight: ~50%

Points: ~0.50

COLLAPSE



Details:
Should end in /pull/#



Task URLs

URL #1

<https://github.com/legitdaylight/jd755-IT202-007/pull/20>

URL

<https://github.com/legitdaylight/jd755-IT202-007>

End of Task 2

End of Group: User Logout

Task Status: 2/2

Group

Group: Basic Security Rules and Roles

Tasks: 5

Points: 2

100%

COLLAPSE

Task

Group: Basic Security Rules and Roles

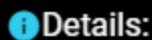
Task #1: Authentication Screenshots

Weight: ~20%

Points: ~0.40

100%

COLLAPSE



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



Columns: 1

Sub-Task

Group: Basic Security Rules and Roles

Task #1: Authentication Screenshots

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

100%

Task Screenshots

Gallery Style: 2 Columns

4 2 1

```
//jd755 11/09/24
function is_logged_in($redirect = false, $destination = "login.php")
{
    $isloggedin = isset($_SESSION["user"]);
    if ($redirect && !$isloggedin) {
        //If this triggers, the calling script won't receive a reply since die()/exit() terminates it
        flash("you must be logged in to view this page", "warning");
        die(header("location: $destination"));
    } <- #10-14 if ($redirect && !$isloggedin)
    return $isloggedin;
} <- #15 function is_logged_in($redirect = false, $destination = "login.php")
```

Function to check if user is logged in

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

The function accepts two parameters: redirect which is defaulted to false, and destination which is defaulted to login.php. The function starts by checking if the session with the current user exists. If it doesn't exist and if redirect is true, a warning is displayed to the user that they must be logged in to view that page and they are automatically redirected to login.php. If redirect is false, then the user won't be redirected and won't see the message.

Sub-Task



Group: Basic Security Rules and Roles

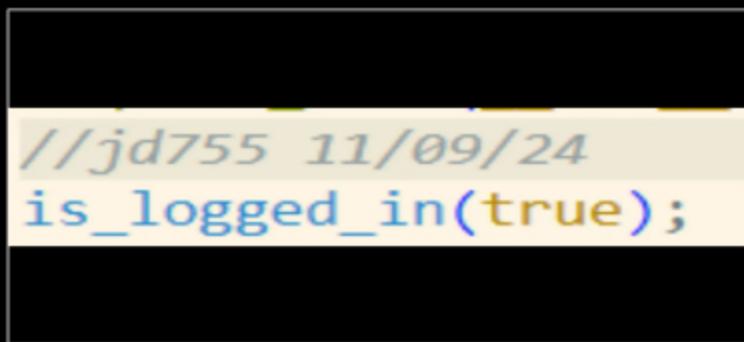
Task #1: Authentication Screenshots

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1



is_logged_in function as seen in profile.php

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

This function is called at the beginning of profile.php. True is passed in meaning the user will be redirected if they are not logged in.

Sub-Task

Group: Basic Security Rules and Roles

100%

Task #1: Authentication Screenshots

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Trying to access a login-protected page while logged out

Caption(s) (required) ✓Caption Hint: *Describe/highlight what's being shown*

End of Task 1

Task

100%

Group: Basic Security Rules and Roles

Task #2: Authorization Screenshots

Weight: ~20%

Points: ~0.40

▲ COLLAPSE ▲

i Details:

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



Columns: 1

Sub-Task

100%

Group: Basic Security Rules and Roles

Task #2: Authorization Screenshots

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1



```
//jd755 11/09/24 <- m8-16 function is_logged_in($redirect = false, $dest)
function has_role($role)
{
    if (!is_logged_in() || !in_array($role, $_SESSION['roles'])) {
        // ...
    }
}
```

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

Function that checks if the user has a role

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

The function accepts one parameter which is the name of the role. It checks if the user is logged in (without passing in true) and it checks if the user has any roles associated with them. If these are both true, it loops through each role the user has and checks if it matches the passed in role. If a match is found, return true.

Sub-Task

Group: Basic Security Rules and Roles

100%

Task #2: Authorization Screenshots

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

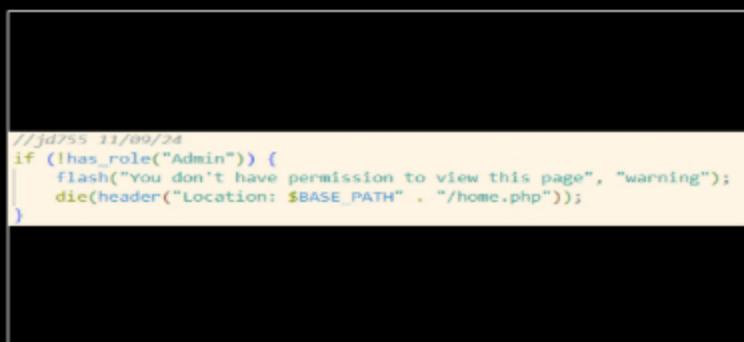
Task Screenshots

Gallery Style: 2 Columns

4

2

1



has_role function as seen in assign_roles.php

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Task Response Prompt

Explain in concise steps how this logically works

Response:

The if statement checks if the has_roles function with the role "Admin" returns false. If so, a warning is displayed to the user and they are redirected to the home page.

Sub-Task

Group: Basic Security Rules and Roles

100%

Task #2: Authorization Screenshots

Sub Task #3: Demonstrate the user-friendly message of trying to manually access a role-protected

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Trying to access a role protected page while logged out

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 2

Task



Group: Basic Security Rules and Roles

Task #3: Screenshots of UserRoles and Roles Tables

Weight: ~20%

Points: ~0.40

[▲ COLLAPSE ▾](#)

Details:

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



Columns: 1

Sub-Task



Group: Basic Security Rules and Roles

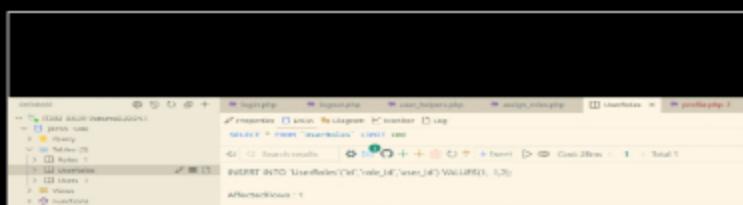
Task #3: Screenshots of UserRoles and Roles Tables

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1



User	Role	Is_Active	Created	Modified
John Doe	Administrator	True	2024-11-08 21:00:00	2024-11-08 21:00:00

UserRoles table

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task



Group: Basic Security Rules and Roles

Task #3: Screenshots of UserRoles and Roles Tables

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

Task Screenshots

Gallery Style: 2 Columns

4

2

1

Roles Table Screenshot									

Roles table

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 3

Task



Group: Basic Security Rules and Roles

Task #4: Explain how Roles and UserRoles tables work in conjunction with the Users table

Weight: ~20%

Points: ~0.40

▲ COLLAPSE ▲

Columns: 1

Sub-Task



Group: Basic Security Rules and Roles

Task #4: Explain how Roles and UserRoles tables work in conjunction with the Users table

Sub Task #1: UserRoles

Task Response Prompt

What's the purpose of the UserRoles table?

Response:

The UserRoles table combines a role id in the Roles table with a user id in the User table. This essentially "assigns" a

role to a user .

Sub-Task



Group: Basic Security Rules and Roles

Task #4: Explain how Roles and UserRoles tables work in conjunction with the Users table

Sub Task #2: Roles

Task Response Prompt

How does Roles.is_active differ from UserRoles.is_active?

Response:

Roles.is_active determines if a role itself is turned on or off. On the otherhand, UserRoles.is_active determines if a certain user can keep using the permissions of a certain role.

End of Task 4

Task



Group: Basic Security Rules and Roles

Task #5: Include pull request links related to this feature

Weight: ~20%

Points: ~0.40

 COLLAPSE 

① Details:

Should end in /pull/#



Task URLs

URL #1

<https://github.com/legitdaylight/jd755-IT202-007/pull/29>

URL

<https://github.com/legitdaylight/jd755-IT202-007>

End of Task 5

End of Group: Basic Security Rules and Roles

Task Status: 5/5

Group



Group: User Profile

Tasks: 5

Points: 2

 COLLAPSE 

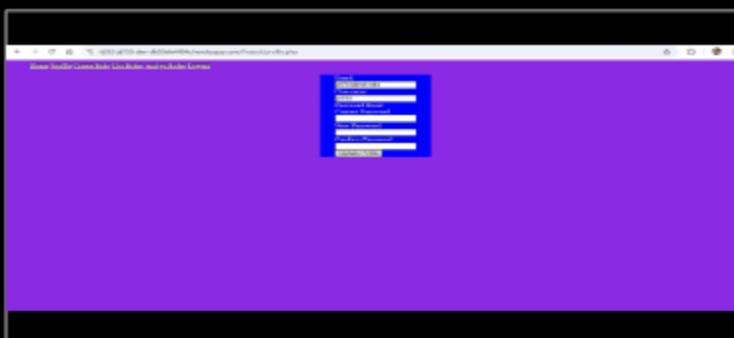
Task**Group:** User Profile**Task #1:** View Profile Website Page

Weight: ~20%

Points: ~0.40

▲ COLLAPSE ▲**i Details:**

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

**Sub-Task****Group:** User Profile**Task #1:** View Profile Website Page**Sub Task #1:** Show the profile form correctly populated on page load (username, email) Form should have the following fields: username, email, current password, new password, confirm password (or similar)**☒ Task Screenshots****Gallery Style: 2 Columns****4 2 1**

The profile page

Caption(s) (required) ✓**Caption Hint:** *Describe/highlight what's being shown***End of Task 1****Task****Group:** User Profile**Task #2:** Explain the logic step-by-step of how the data is loaded and populated when the profile page is visited

Weight: ~20%

Points: ~0.40

▲ COLLAPSE ▲**i Details:**

Don't just show code, translate things to plain English in concise steps.



Task Response Prompt

Response:

1. When the page is first loaded, the email and username is taken from the session data and assigned to their respective variables.
2. The HTML uses these variables and sets it as the default value of the email and username inputs of the form.

End of Task 2

Task

Group: User Profile



Task #3: Edit Profile Website Page

Weight: ~20%

Points: ~0.40

[▲ COLLAPSE ▲](#)

Details:

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



Columns: 1

Sub-Task

Group: User Profile



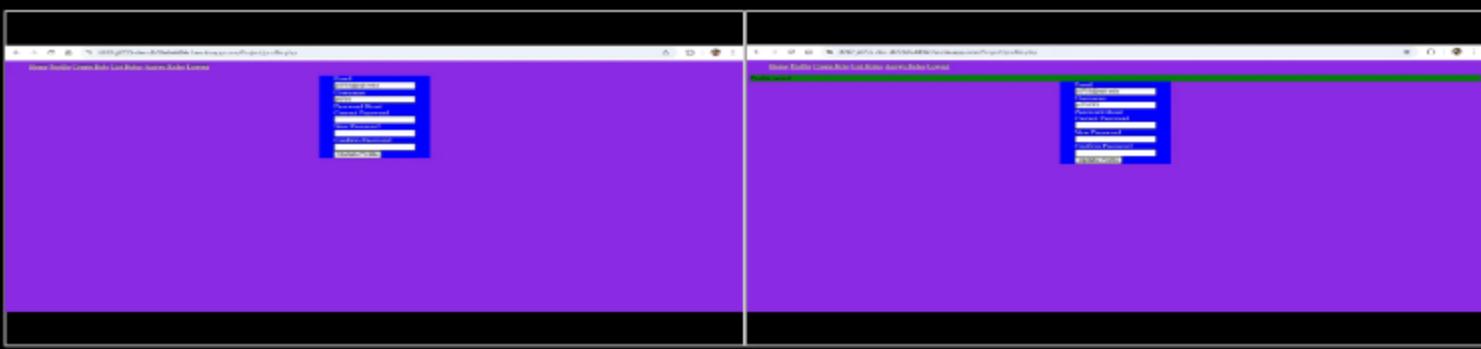
Task #3: Edit Profile Website Page

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

Task Screenshots

Gallery Style: 2 Columns

4 2 1



before username change

after username change

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: User Profile

Sub Task

Group: User Profile

Task #3: Edit Profile Website Page

Sub Task #2: Demonstrate the success message of updating password

Task Screenshots

Gallery Style: 2 Columns

4

2

1

A screenshot of a web browser displaying a user profile update page. The page features a purple header with the text 'Home Profile Create Edit List Action Update Delete Logout'. Below the header is a red navigation bar with links for 'Create', 'Edit', 'List', 'Action', 'Update', and 'Delete'. The main content area is white and contains a form with several input fields. The fields are labeled: 'Email' (with placeholder 'Email'), 'Username' (with placeholder 'Username'), 'Current Password' (with placeholder 'Current Password'), 'New Password' (with placeholder 'New Password'), and 'Confirm Password' (with placeholder 'Confirm Password'). At the bottom of the form, there is a button labeled 'Update Profile'. A green circular progress bar in the top left corner indicates '100%' completion.

password reset

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

Sub-Task

Group: User Profile

Task #3: Edit Profile Website Page

Sub Task #3: Demonstrate all JavaScript user-friendly validation messages [can be combined] (email format, username format, password format, new password matching confirm password)

Task Screenshots

Gallery Style: 2 Columns

4

2

1

A screenshot of a web browser displaying a user profile update page. The page features a purple header with the text 'Home Profile Create Edit List Action Update Delete Logout'. Below the header is a red navigation bar with links for 'Create', 'Edit', 'List', 'Action', 'Update', and 'Delete'. The main content area is white and contains a form with several input fields. The fields are labeled: 'Email' (with placeholder 'Email') and 'Username' (with placeholder 'Username'). Both the 'Email' and 'Username' fields have a red border around them, indicating they are required or invalid. The rest of the page is white with a blue sidebar containing links for 'Create', 'Edit', 'List', 'Action', 'Update', and 'Delete'.

empty email and username

A screenshot of a web browser displaying a user profile update page. The page features a purple header with the text 'Home Profile Create Edit List Action Update Delete Logout'. Below the header is a red navigation bar with links for 'Create', 'Edit', 'List', 'Action', 'Update', and 'Delete'. The main content area is white and contains a form with several input fields. The fields are labeled: 'Email' (with placeholder 'Email') and 'Username' (with placeholder 'Username'). Both the 'Email' and 'Username' fields have a red border around them, indicating they are required or invalid. The rest of the page is white with a blue sidebar containing links for 'Create', 'Edit', 'List', 'Action', 'Update', and 'Delete'.

Invalid username and email

A screenshot of a web browser displaying a user profile update page. The page features a purple header with the text 'Home Profile Create Edit List Action Update Delete Logout'. Below the header is a red navigation bar with links for 'Create', 'Edit', 'List', 'Action', 'Update', and 'Delete'. The main content area is white and contains a form with several input fields. The fields are labeled: 'Email' (with placeholder 'Email'), 'Username' (with placeholder 'Username'), 'Current Password' (with placeholder 'Current Password'), 'New Password' (with placeholder 'New Password'), and 'Confirm Password' (with placeholder 'Confirm Password'). The 'New Password' field has a red border around it, indicating it is too short. The rest of the page is white with a blue sidebar containing links for 'Create', 'Edit', 'List', 'Action', 'Update', and 'Delete'.

new password too short

A screenshot of a web browser displaying a user profile update page. The page features a purple header with the text 'Home Profile Create Edit List Action Update Delete Logout'. Below the header is a red navigation bar with links for 'Create', 'Edit', 'List', 'Action', 'Update', and 'Delete'. The main content area is white and contains a form with several input fields. The fields are labeled: 'Email' (with placeholder 'Email'), 'Username' (with placeholder 'Username'), 'Current Password' (with placeholder 'Current Password'), 'New Password' (with placeholder 'New Password'), and 'Confirm Password' (with placeholder 'Confirm Password'). Both the 'New Password' and 'Confirm Password' fields have a red border around them, indicating they do not match. The rest of the page is white with a blue sidebar containing links for 'Create', 'Edit', 'List', 'Action', 'Update', and 'Delete'.

passwords not matching

Caption(s) (required) ✓Caption Hint: *Describe/highlight what's being shown***Sub-Task**

Group: User Profile

Task #3: Edit Profile Website Page

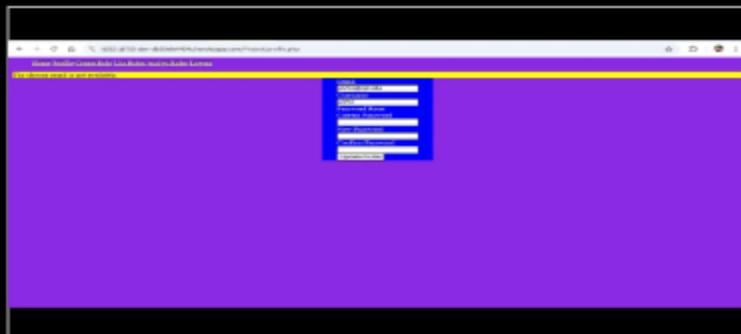
Sub Task #4: Demonstrate PHP user-friendly validation message (desired email is already in use)

100%

Task Screenshots

Gallery Style: 2 Columns

4 2 1



Email not available

Caption(s) (required) ✓Caption Hint: *Describe/highlight what's being shown***Sub-Task**

Group: User Profile

Task #3: Edit Profile Website Page

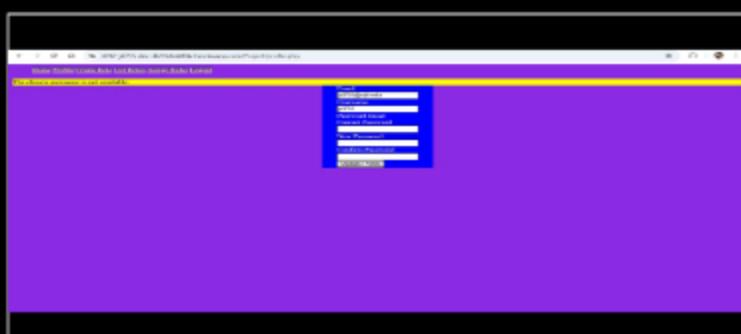
Sub Task #5: Demonstrate PHP user-friendly validation message (Desired username is already in use)

100%

Task Screenshots

Gallery Style: 2 Columns

4 2 1



username not available

Caption(s) (required) ✓Caption Hint: *Describe/highlight what's being shown***Sub-Task**

Group: User Profile

Task #3: Edit Profile Website Page

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

100%

Task Screenshots

Gallery Style: 2 Columns

4 2 1



wrong password

Caption(s) (required) ✓

Caption Hint: *Describe/highlight what's being shown*

End of Task 3

Task

100%

Group: User Profile

Task #4: Explain the logic step-by-step of how the data is checked and saved for the following scenarios

Weight: ~20%

Points: ~0.40

▲ COLLAPSE ▲

Details:

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

Columns: 1

Sub-Task

100%

Group: User Profile

Task #4: Explain the logic step-by-step of how the data is checked and saved for the following scenarios

Sub Task #1: Updating Username/Email

Task Response Prompt

Explain in concise steps how this logically works

Response:

Username and Email change After the email and username is valid, this data is sent to the database via a SQL statement. If the query returns an error, the function users_check_duplicate is called to send a message to the user that there was an issue with the process.

Sub-Task

Group: User Profile

Task #4: Explain the logic step-by-step of how the data is checked and saved for the following

100%

scenarios
Sub Task #2: Updating password

Task Response Prompt

Explain in concise steps how this logically works

Response:

It is determined if the user wants to change the password if the current, new, and confirm password fields are all filled out. If they are and if they pass validation, an SQL statement is created and sent to the database. This statement takes the ID of the user and gets the password. If the passwords do not match, a warning is sent to the user that the password is invalid. If they do match, the new password is hashed then the database is updated with the new password.

End of Task 4

Task

100%

Group: User Profile

Task #5: Include pull request links related to this feature

Weight: ~20%

Points: ~0.40

COLLAPSE

Details:

Should end in /pull/#



Task URLs

URL #1

<https://github.com/legitdaylight/jd755-IT202-007/pull/25>

URL

<https://github.com/legitdaylight/jd755-IT202-007>

End of Task 5

End of Group: User Profile

Task Status: 5/5

Group

100%

Group: Misc

Tasks: 3

Points: 1

COLLAPSE

Task

Group: Misc

100%

Task #1: Screenshot of wakatime

Weight: ~33%

Points: ~0.33

▲ COLLAPSE ▲

Details:

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

**Task Screenshots**

Gallery Style: 2 Columns

4

2

1



waka time screenshot 1

Files	Branches
26 mins	e_html/Project/register.php
20 mins	public_html/Project/login.php
11 mins	libs/update_user_details.php
8 mins	libs/user_helpers.php
8 mins	_ic_.html/Project/profile.php
4 mins	libs/helpers.php
3 mins	public_html/dynamic_form.php
2 mins	public_html/Project/logout.php
2 mins	public_html/Project/timers.js
1 min	public_html/Project/styles.css
56 secs	partials/nav.php
41 secs	public_html/Project/home.php
15 secs	libs/safer_echo.php
12 secs	_e_html/Ms/dynamic_array.php
10 secs	_ic_.html/Ms/dynamic_edit.php
4 secs	libs/functions.php
3 secs	partials/footer.php
0 secs	Project/admin/assign_roles.php

wakatime screenshot 2

End of Task 1**Task**

100%

Group: Misc

Task #2: Screenshot of your project board from GitHub (tasks should be in the proper column)

Weight: ~33%

Points: ~0.33

▲ COLLAPSE ▲

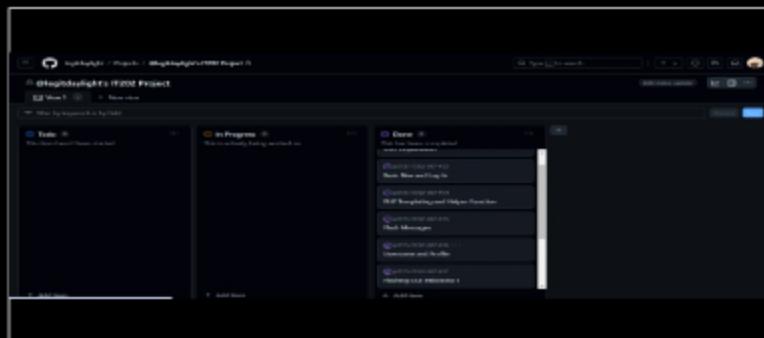
Task Screenshots

Gallery Style: 2 Columns

4

2

1



End of Task 2

Task



Group: Misc

Task #3: Provide a direct link to the project board on GitHub

Weight: ~33%

Points: ~0.33

COLLAPSE

Task URLs

URL #1

<https://github.com/users/legitdaylight/projects/2/views/1>

URL

<https://github.com/users/legitdaylight/projects/2>

End of Task 3

End of Group: Misc

Task Status: 3/3

End of Assignment