**[IS113] Extra Exercises - Selected Topics**

**Resource Download**

* To download the **resource files**, go to: <https://smu.sg/hpg>
* You will see folders (**q1, q2, q3, q4,q5**).

# Download & place the folders inside <web root>/is113/extra12/ on your local computer.

**NOTE:** If you spot any mistakes/errors in the questions, please email the info to the following authors. In the email message, please be specific about what are the mistakes/errors.

|  |  |  |
| --- | --- | --- |
| **Question** | **Author** | **Contact Info** |
| 1 | David LO | davidlo@smu.edu.sg |
| 2 | Shar LK | lkshar@smu.edu.sg |
| 3 | Kyong SHIM | kjshim@smu.edu.sg |
| 4 | Kyong SHIM | kjshim@smu.edu.sg |
| 5 | Yi Meng LAU | ymlau@smu.edu.sg |
| 6 |  |  |
| 7 |  |  |

**Database Connection (from inside PHP code)**

1. **WAMP Users**
   1. Upon WAMP installation, if you have not changed your MySQL login info will be:
      1. **Username**: root
      2. **Password**: <left empty>
2. **MAMP Users**
   1. For most students we have assisted, it appears that the default MySQL login info is:
      1. **Username**: root
      2. **Password**: root
   2. Additionally, your **MySQL port** appears to be **3306** (*please verify this on your own laptop computer and* ***remember to note it down***).
      1. You will have to specify **port** in **ConnectionManager.php**.
      2. Please remember to configure **ConnectionManager.php** on your own in all Extra Exercises as well as in Lab Test 2 questions **on your own** (as we instructional staff do NOT provide a separate **ConnectionManager.php** file for non-WAMP users).

**Question 1: Find Oldest Person ( Difficulty Level: \* )**

**Given:**

* q1/
  + Person.php (**complete**)
  + input.php *(***complete***)*
  + process.php(***partial***)

**How data flow across pages:**

* **input.php -> process.php -> [input.php -> process.php]\***

**Part A: Add one person**

Complete **process\_one\_person** function of **process.php** such that it creates a new Person object based on data entered by user inputs (name and age) and stores it in the session. The session maintains an array of Person objects based on person details entered so far. Read comments in the PHP file for detailed instructions. You can assume that users always enter correct data. If Part A is completed well, the following would be the expected behavior:

|  |  |
| --- | --- |
| **(1)***New person details are entered:*    ***input.php*** | **(2)** *After “Next” button is clicked:*    ***input.php*** |
| **(3)** *New person details are entered:*    ***input.php*** | **(4)** *After “Next” button is clicked:*    ***input.php*** |

**Part B: Find Oldest**

Complete **find\_oldest\_and\_reset** function of **process.php** such that it finds and displays the oldest persons (it could be one or more) that have been entered through the form so far, and resets the array of Person objects stored in the session. Read comments in the PHP file for detailed instructions. If Part B is completed well, the following would be the expected behavior:

|  |  |
| --- | --- |
| **(1)***After multiple persons are entered:*    ***input.php*** | **(2)** *After “Find Oldest and Reset” button is clicked:*    ***process.php*** |
| (3) After “Start Again” link is clicked    ***input.php*** | |

**Question 2: Shopping Cart ( Difficulty Level: \*/\*\* )**

**Given:**

● q2/

○ shop.php *(****partial****)*

○ process.php *(***partial***)*

○ autoload.php(complete)

○ model/ConnectionManager.php (complete)

○ model/Item.php (complete)

○ model/ItemDAO.php (complete)

○ model/week12extra.sql (complete)

**How data flow across pages:**

● **shop.php -> process\_shop.php -> [shop.php -> process\_shop.php]\***

**Import week12extra.sql into MySQL database**

**Part A: Display Items from Shopping Cart (Difficulty Level: \*)**

Complete **shop.php** such that it displays, in a table, the details (item name, price, quantity) of the Item objects stored in the session. Also compute and display the total price of all the items in the shopping cart. Read comments in the PHP file for detailed instructions.

Refer to the figure below for the expected behavior.

**Part B: Add Items into Shopping Cart (Difficulty Level: \*\*)**

Complete **addToCart** function of **process\_shop.php** such that it creates a new Item object for each item quantity entered by the user in shop.php and stores it in the session. For example, if the user enters Quantity = 2 for the item **iPhoneXs** in shop.php, upon clicking “Add to Cart” button, process\_shop.php should create an Item object with properties: name=iPhoneXs, price=2088, quantity=2. Likewise for all the other item quantities the user entered.

Create an array of those Item objects and store the array in the session. That is, the session maintains an array of Item objects based on the item quantities entered so far.

Read comments in the PHP file for detailed instructions and refer to the figure below. You can assume that users always enter correct data.

**Part C: Clear Shopping Cart (Difficulty Level: \*)**

Complete **clearCart** function of **process\_shop.php** such that it clears all the item objects stored in the session.

**If Part A, Part B, and Part C are completed well, the following would be the expected behavior:**

|  |  |
| --- | --- |
| **(1)** *Item quantities are entered:*      ***shop.php*** | **(2)** *After “Add to Cart” button is clicked:*      ***process\_shop.php*** |
| **(3)** *When “Continue Shopping!” button is clicked:*      ***shop.php*** | **(4)** *User adds new items:*    ***shop.php*** |
| **(5)** *After “Add to Cart” button in shop.php and “Continue Shopping” button in process\_shop.php are clicked:*    ***shop.php*** | **(6)** *When “Clear Cart” button in shop.php is clicked:*      ***process\_shop.php***      **(7)** *When “Continue Shopping” button is clicked:*      ***shop.php*** |

**Question 3: Login & Logout (No Database) ( Difficulty Level: \*/\*\* )**

**Given:**

* q3/**NoSession**
  + css
    - signin.css *(complete)*
  + login.php *(complete)*
  + process\_login.php ***(partial)***
* q3/**Session1**
  + css
    - signin.css *(complete)*
  + login.php *(complete)*
  + process\_login.php ***(partial)***
* q3/**Session2**
  + css
    - signin.css *(complete)*
  + login.php ***(copy from q3/Session1 after completion)***
  + process\_login.php ***(copy from q3/Session1 after completion)***
  + main.php ***(partial)***
  + logout.php ***(partial)***

**Part A** **( Difficulty: \* ) - User Login & Authentication**

In Part A, we attempt to authenticate user login information (email and password). The user keys in email and password in an HTML form (**login.php**). Clicking on the SUBMIT button will forward to **process\_login.php**.

* [**Successful login**] Stay in **process\_login.php** and display welcome message & sensitive data.
* [**Unsuccessful login**] Stay in **process\_login.php** and display appropriate error message(s).

|  |
| --- |
| **Exercise Files** |
| * q3/**NoSession**   + css     - signin.css *(complete)*   + login.php *(complete)*   + process\_login.php ***(partial)*** |

**How data flow across pages**

* **login.php -> process\_login.php**

When **login.php** loads for the first time, it looks like this:

|  |
| --- |
| **login.php** |
|  |

This page uses **Twitter Bootstrap** (<https://getbootstrap.com/docs/4.3/examples/>).

|  |
| --- |
| What is **Twitter Bootstrap**? (from Wikipedia) |
| *"Bootstrap is a free and open-source front-end web framework. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions."* |

In this exercise, we use **Twitter Bootstrap**’s JavaScript and CSS (Cascading Style Sheets) libraries for styling and simple front-end interactivity.

Edit process\_login.php such that it authenticates user input (**email** and **password**) and displays appropriate messages as shown below:

|  |  |
| --- | --- |
| **login.php** | **process\_login.php** |
|  |  |
| Password: **123456** |  |
| Password: **donald123** |  |

And when the user attempts to access **process\_login.php** without form submission from **login.php**, the page must display the following message:

|  |
| --- |
| **process\_login.php**  ***(access this page directly without login.php form submission)*** |
|  |

* **NOTE**: Currently, the valid email/password pairs are hard-coded at the top of **process\_login.php**. Use this for **Part A**.
* Later on, we will store this information in a database table and authenticate against this database table.

**Part B** **( Difficulty: \*\* ) - User Login & Authentication (using Session Variable)**

In Part B, we attempt to authenticate user login information (email and password). The user keys in email and password in an HTML form (**login.php**). Clicking on the SUBMIT button will forward to **process\_login.php**.

* [**Successful login**] Stay in **process\_login.php** and display welcome message & sensitive data.
* [**Unsuccessful login**]
  + Save appropriate error message(s) as a **Session Variable** and forward to **login.php**.
  + **login.php** loads up the login screen as per normal.
  + **login.php** will **ALSO** retrieve and display error message(s).

|  |
| --- |
| **Exercise Files** |
| * q3/**Session1**   + css     - signin.css *(complete)*   + login.php *(complete)*   + process\_login.php ***(partial)*** |

**How data flow across pages**

* **login.php -> process\_login.php -> [login.php -> process\_login.php]\***

Edit **process\_login.php** such that it performs the following:

1. Authenticates user input (**email** and **password**) against the hard-coded Associative Array **$accounts**.
2. In case of any **authentication errors** (email is not found, password is incorrect):
   1. Add **authentication error messages** (String) into **$errors** Indexed Array.
   2. Create a new **Session Variable** **$\_SESSION['errors']** whose value is **$errors** Indexed Array.
   3. The user is **forwarded to** page **login.php**.

Example (Ref: <https://www.php.net/manual/en/function.header.php>)

|  |
| --- |
| <?php  header("Location: http://www.example.com/"); // Redirect web browser  // Make sure that code below doesn’t get executed when we redirect.  exit;  ?> |

1. If **authentication succeeds** (email & password combination is found in **$accounts**), the user shall remain in **process\_login.php** and the PHP page displays in **HTML** an appropriate message (as shown below).

**Test Case #1**

***Email is NOT found in $accounts in process\_login.php***

|  |  |
| --- | --- |
| **login.php** | **login.php**  **(forwarded to this page from process\_login.php)** |
|  |  |

* After the authentication fails, the user is forwarded to **login.php**.
* This time, **login.php** must **retrieve** the **Session Variable** **$\_SESSION['errors']** and display the messages as an **unordered list** below “Please sign in” header.

**Test Case #2**

***Email is found in $accounts in process\_login.php*** *but* ***the password is incorrect****.*

|  |  |
| --- | --- |
| **login.php** | **login.php**  **(forwarded to this page from process\_login.php)** |
| Password: **123456** |  |

* After the authentication fails, the user is forwarded to **login.php**.
* This time, **login.php** must **retrieve** the **Session Variable** **$\_SESSION['errors']** and display the messages as an **unordered list** below “Please sign in” header.

**Test Case #3**

***Email & password combination is valid.***

|  |  |
| --- | --- |
| **login.php** | **process\_login.php** |
| Password: **donald123** |  |

* After the authentication succeeds, the user remains in **process\_login.php**.

**Test Case #4**

***Unauthorized attempt to access process\_login.php***

And when the user attempts to access **process\_login.php** without form submission from **login.php**, the page must display the following message:

|  |
| --- |
| **process\_login.php**  ***(access this page directly without login.php form submission)*** |
|  |

**Part C** **( Difficulty: \*\* ) - User Login & Authentication (using Session Variable) & Logout**

In Part C, we attempt to authenticate user login information (email and password). The user keys in email and password in an HTML form (**login.php**). Clicking on the SUBMIT button will forward to **process\_login.php**, which will check the user input against **$accounts** for authentication. This time, **process\_login.php** will NOT display the welcome message and sensitive data. It will be done by **main.php**.

* [**Unsuccessful login**]
  + Save appropriate error message(s) as a **Session Variable** and forward to **login.php**.
  + **login.php** loads up the login screen as per normal.
  + **login.php** will **ALSO** retrieve and display error message(s).
* [**Successful login**]
  + Forward to **main.php** and display welcome message & sensitive data.
  + Make sure that **main.php** displays welcome message & sensitive data for only authenticated users.

|  |
| --- |
| **Exercise Files** |
| * q3/**Session2**   + css     - signin.css *(complete)*   + login.php ***(copy from q3/Session1 after completion)***   + process\_login.php ***(copy from q3/Session1 after completion)***   + main.php ***(partial)***   + logout.php ***(partial)*** |

**How data flow across pages**

* **login.php -> process\_login.php -> [login.php -> process\_login.php]\* -> main.php -> logout.php -> login.php**

Copy **process\_login.php** from **q3/Session1**. Further modify **process\_login.php** such that it performs the following:

1. If **authentication succeeds** (email & password combination is found in **$accounts**):
   1. Please NOTE that **process\_login.php** will no longer display welcome message & sensitive data. It will be done in **main.php**.
   2. Create a new **Session Variable** **$\_SESSION['loggedInEmail']** whose value is **user’s email address** (retrieved from **$\_POST**).
      1. **IMPORTANT**: This session variable shall be used by **main.php** to verify user login.
   3. The user is **forwarded to** page **main.php**.

Example (Ref: <https://www.php.net/manual/en/function.header.php>)

|  |
| --- |
| <?php  header("Location: http://www.example.com/"); // Redirect web browser  // Make sure that code below doesn’t get executed when we redirect.  exit;  ?> |

Edit **main.php** such that it performs the following:

1. It checks to see if **Session Variable** **$\_SESSION['loggedInEmail']** exists.
   1. If it does not exist, it means no user has logged in.
      1. Forward to **login.php**.
   2. If it exists, it means a user has logged in successfully. Then, display the following content:

|  |
| --- |
| **main.php** |
|  |

* 1. “Log Out” is a HyperLink to **logout.php** (which we implement below).

Edit **logout.php** such that it performs the following:

1. It **unsets** **Session Variable** **$\_SESSION['loggedInEmail']**.
   1. You can use **unset()** built-in function.
      1. Ref: <https://www.php.net/manual/en/function.unset.php>
   2. Alternatively, you can unset ALL session variables using **session\_unset()**.
      1. Ref: <https://www.php.net/manual/en/function.session-unset.php>
2. It **destroys** the **current session**.
   1. You can use **session\_destroy()** built-in function.
      1. Ref: <https://www.php.net/manual/en/function.session-destroy.php>
3. When done with the above, forward to **login.php**.

**Test Case #1**

***Email is NOT found in $accounts in process\_login.php***

|  |  |
| --- | --- |
| **login.php** | **login.php**  **(forwarded to this page from process\_login.php)** |
|  |  |

**Test Case #2**

***Email is found in $accounts in process\_login.php*** *but* ***the password is incorrect****.*

|  |  |
| --- | --- |
| **login.php** | **login.php**  **(forwarded to this page from process\_login.php)** |
| Password: **123456** |  |

* After the authentication fails, the user is forwarded to **login.php**.
* This time, **login.php** must **retrieve** the **Session Variable** **$\_SESSION['errors']** and display the messages as an **unordered list** below “Please sign in” header.

**Test Case #3**

***Email & password combination is valid.***

|  |  |
| --- | --- |
| **login.php** | **main.php** |
| Password: **donald123** |  |

* After the authentication succeeds, the user is forwarded to **main.php**.

**Test Case #4**

***The current Session remains active***

Once the user is **logged in**, the current **Session** remains active.

**Try This**

* Close your current web browser window where you have **main.php** open.
* Open a new browser window and go to the URL to **main.php**.
* Since the current **Session** remains active, you should be able to access **main.php** without being forwarded back to **login.php**.

|  |
| --- |
| **main.php** |
|  |

**Test Case #5**

***Log out & there is no active session***

**Try This**

* In **main.php**, click on **Log Out.**
  + If **logout.php** is implemented correctly, you will be forwarded to **login.php**.
* In the current browser window, go to the URL to **main.php**.
  + You should be forwarded back to **login.php**.
* Open a new browser window, go to the URL to **main.php**.
  + You should be forwarded back to **login.php**.

|  |
| --- |
| **login.php** |
|  |

**Test Case #6**

***Unauthorized attempt to access process\_login.php***

When the user attempts to access **process\_login.php** without form submission from **login.php**, the page must forward the user to **login.php**.

|  |  |
| --- | --- |
| **process\_login.php**  ***(access this page directly without login.php form submission)*** | **login.php** |
|  |  |

**Test Case #7**

***Unauthorized attempt to access main.php***

And when the user attempts to access **process\_login.php** without form submission from **login.php**, the page must forward the user to **login.php**.

|  |  |
| --- | --- |
| **main.php**  ***(access this page directly without login.php form submission)*** | **login.php** |
|  |  |

# Question 4: Login & Logout (Database) (Difficulty Level: \*\*)

This exercise is a continuation of **Question 3**. Previously, we had account details (pairs of **email** and **password**) hard-coded in **process\_login.php** in **$accounts** Associative Array.

In Question 4, we no longer have the account details in **process\_login.php**. Account details are saved in a MySQL database table **account**.

**How data flow across pages**

* **login.php -> process\_login.php -> [login.php -> process\_login.php]\* -> main.php -> logout.php -> login.php**

**Part A** **( Difficulty: \*\* ) - User Login & Authentication**

**Given:**

* q4/**unsafe**
  + css
    - signin.css *(complete)*
  + database
    - **create.sql** *(complete)*
  + include
    - common.php *(complete)*
    - ConnectionManager.php *(complete)*
    - Account.php *(complete)*
    - AccountDAO.php ***(partial)***
    - Task.php *(complete)*
    - TaskDAO.php ***(partial)***
  + login.php ***(copy from q3/Session2 after completion)***
  + process\_login.php ***(copy from q3/Session2 after completion)***
  + main.php ***(copy from q3/Session2 after completion)***
  + logout.php ***(copy from q3/Session2 after completion)***
  + test.php *(complete) → Contains Test Cases*

Read and use the given **create.sql** to understand and create the necessary database and tables for this question.

* Open create.sql. Take the SQL statements in this file and execute it (you may use **WorkBench** or **PHPMyAdmin**, whichever one you are comfortable with).
* It creates a schema **extra12q4unsafe**. Inside **extra12q4unsafe**, it creates two tables:
  + **account**
    - Stores login account information
    - **NOTE**: password information is **unsafely** stored in **plain text** in this table. Later on, we will encrypt account **password** data using PHP’s **password\_hash()** function.
    - Ref: <https://www.php.net/manual/en/function.password-hash.php>
  + **task**
    - Stores task information
    - Each task is linked to a particular account
      * account\_id is a foreign key

**Account**

-----------------------------------------------------------------------------------------------------------------------------------------------

Complete the method **retrieve($email)** in **AccountDAO.php**.

* The method receives one parameter **$email** (e.g. **donald@trump.com**).
* It queries database table **account** and retrieves a corresponding row (if any).
* If a matching row is found:
  + It then creates a new **Account** **object** and returns it.
* If no matching row is found in the database table, the method is to return **null**.
* **NOTE**: You may use the **test cases** in **test.php**.

-----------------------------------------------------------------------------------------------------------------------------------------------

***Copy* process\_login.php *from q3/Session2 after completion***

Modify **process\_login.php** so that it performs the following:

* Uses an **AccountDAO** object to call its method **retrieve($email)** ✔that you completed above.
* Using the returned **Account** **object**, perform login authentication.
  + **NOTE**: You can use **Account** **object**’s **Getter** methods to retrieve its attributes/properties.

**Task**

-----------------------------------------------------------------------------------------------------------------------------------------------

Complete the method **retrieveByAccountID($account\_id)** in **TaskDAO.php**.

* The method receives one parameter **$account\_id** (e.g. 1).
* It queries database table **task** and retrieves one or more rows with matching **account\_id** column value.
* For each matching row found:
  + Create a new **Task** object and add it into **$tasks** Indexed Array.
* The method returns the **$tasks** Indexed Array**.**
* **NOTE**: You may use the **test cases** in **test.php**.✔

-----------------------------------------------------------------------------------------------------------------------------------------------

***Copy* main.php *from q3/Session2 after completion***

Modify **main.php** so that it performs the following:

* Uses an **TaskDAO** object to call its method **retrieveByAccountID($account\_id)** that you completed above.✔
* Using the returned Indexed Array of **Task** **object(s)**, display the task details as an **ordered list**.
  + **NOTE**: You can use **Task object**’s **Getter** methods to retrieve its attributes/properties.

**Test Case #1**

***Email is NOT found in MySQL Database table* account**

|  |  |
| --- | --- |
| **login.php** | **login.php**  **(forwarded to this page from process\_login.php)** |
|  |  |

**Test Case #2**

***Email is found in MySQL Database table* account** *but* ***the password is incorrect****.*

|  |  |
| --- | --- |
| **login.php** | **login.php**  **(forwarded to this page from process\_login.php)** |
| Password: **123456** |  |

**Test Case #3**

***Email & password combination is valid (checked against MySQL Database table* account*)***

|  |  |
| --- | --- |
| **login.php** | **main.php** |
| Username: donald@trump.com  Password:  **donald123** |  |

* After the authentication succeeds, the user is forwarded to **main.php**.
* **NOTE**: This time, **main.php** displays all **tasks** belonging to **donald@trump.com.**

**Test Case #4**

***Email & password combination is valid (checked against MySQL Database table* account*)***

|  |  |
| --- | --- |
| **login.php** | **main.php** |
| Username: hillary@clinton.com  Password:  **hillary456** |  |

**Part B** **( Difficulty: \*\* ) - User Login & Authentication *(with Password Hashing)***

**Given:**

* q4/**safe**
  + css
    - signin.css *(complete)*
  + database
    - **create.sql** *(complete)*
  + include
    - common.php *(complete)*
    - ConnectionManager.php *(complete)*
    - Account.php *(complete)*
    - AccountDAO.php ***(copy from* Part A *after completion)***
    - Task.php *(complete)*
    - TaskDAO.php ***(copy from* Part A *after completion)***
  + login.php ***(copy from* Part A*)***
  + process\_login.php ***(copy from* Part A *after completion)***
  + main.php ***(copy from* Part A *after completion)***
  + logout.php ***(copy from* Part A*)***
  + test.php ***(copy from* Part A*)***
  + test\_password.php *(complete) → Demonstrates Password Hashing*

✔

Read and use the given **create.sql** to understand and create the necessary database and tables for this question.

* Open create.sql. Take the SQL statements in this file and execute it ✔ (you may use **WorkBench** or **PHPMyAdmin**, whichever one you are comfortable with).
* It creates a schema **extra12q4safe**. Inside **extra12q4safe**, it creates two tables:
  + **account**
    - Stores login account information
    - **NOTE**: password information is **safely** stored in **hashed format** in this table.

For example:

|  |  |
| --- | --- |
| **Plain Text** | **Hashed** |
| **donald123** | $2y$10$h.IgQD4s28VppNZSrgNXNutTT272hEIOjj0nulqrqF7w0a.KSzIUe |

* + - **Hashing** is one type of **data encryption**.
      * We will encrypt account **password** data using PHP’s **password\_hash()** function.
      * Ref: <https://www.php.net/manual/en/function.password-hash.php>
      * We will use PHP’s **password\_verify()** function to compare **plain text password** against **hashed password** (equivalent).✔
      * Ref: <https://www.php.net/manual/en/function.password-verify.php>
  + **task**
    - Stores task information
    - Each task is linked to a particular account
      * account\_id is a foreign key

***Copy* process\_login.php *from* Part A *after completion***

Modify **process\_login.php** so that it performs the following:

* Uses an **AccountDAO** object to call its method **retrieve($email)** that you completed above.
* Using the returned **Account** **object**, perform login authentication.
  + **NOTE**: You can use **Account** **object**’s **Getter** methods to retrieve its attributes/properties.
  + This time, the password (retrieved from MySQL database table **account**) is **hashed**.
  + Your code will have to compare the **password** (from HTML Form Submission) in **plain text** against the **hashed password** from MySQL database table **account**.✔
  + See below for examples.
* When done, run through **Test Cases #1 through #4** from **Part A**.

-----------------------------------------------------------------------------------------------------------------------------------------------

How to **encrypt** a **plain text** **password** for storing in database table?

|  |
| --- |
| **test\_password.php** |
| $password = 'hello123';  $passwordHash = **password\_hash**($password, PASSWORD\_DEFAULT);  echo "<h3>You entered password: $password</h3>";  echo "<h3>Hashed password: $passwordHash</h3>";  echo "<h3>This hashed password will be stored in MySQL Database table Account</h3>"; |
|  |

* Ref: <https://www.php.net/manual/en/function.password-hash.php>
* You can store **hashed password** as **VARCHAR** in MySQL database table.
* See **create.sql** for examples.

-----------------------------------------------------------------------------------------------------------------------------------------------

Okay… I retrieved a **hashed password** from database table. How do I compare it against **plain text password** (for login authentication)?

|  |
| --- |
| **test\_password.php** |
| $status = **password\_verify**( $password, $passwordHash );  if( $status ) {  echo "You keyed in correct password!";  }  else {  echo "You keyed in wrong password!";  } |
|  |

* The function **password\_verify** returns **Boolean (True/False)**.
* Ref: <https://www.php.net/manual/en/function.password-verify.php>

# Question 5: Login Pages (Difficulty Level: \*\*)

**Given:**

● q5/model

○ ConnectionManager.php, User.php (**complete**)

○ UserDAO.php *(****complete****)*

● q5/

○ common.php (**complete**)

○ register.php (**complete**)

○ process\_register.php *(****partial****)*

○ login.php (***partial***)

○ process\_login.php *(****partial****)*

○ change\_password.php (***partial***)

○ process\_change\_password.php *(****partial****)*

○ logout.php *(****partial****)*

○ welcome.php *(****complete****)*

○ database.sql *(****run this before you start)***

This exercise allows you to get familiar with typical user login and authentication functions, with the encrypted password stored in the database. It makes use of simple session management variables to manage the passing of information. It covers the following functions:

1. Register a user
2. Login as a user
3. Change password
4. Logout

Note: This exercise does not cover all possible scenarios of the login and authentication processes that typical businesses need.

The following shows one of the ways to handle session management variable in the application.

**Part A: Complete "process\_register.php"**

Complete process\_register.php to perform the following:

· Retrieve data from register.php. **Check that they are not blank**.

· Check that the Username does not already exist in the database.

· Check that the Password and Confirm Password are the same.

· If any errors, register.php will be reloaded, with username displayed in the Username field and display error messages.

· If there are no errors, hashed the Password and create a User object and add the record to the database. Redirect page to Login.php. The username should be displayed in the Username.

The following shows the flows of the web pages.

|  |  |
| --- | --- |
|  | When the page is first loaded. ✔ |
|  | Display of errors. ✔ |
|  | Check if username is taken. ✔ |
|  | Successful registered and redirected to login.php. ✔ |

**Part B: Complete "process\_login.php"**

Complete functions in process\_login.php to perform the following:

· Retrieve data from login.php. Check that data are entered.

· Check that the Username is valid.

· Verify the password against the hashed password in the database.

· If any errors, login.php will be reloaded, with username displayed in the Username field and display error messages.

· If login is successful. Redirect the page to welcome.php

The following shows the flows of the web pages.

|  |  |
| --- | --- |
|  | When the page is first loaded. |
|  | Enter usename = zack1.    Error to display username does not exist.  ✔ |
|  | Enter usename = zack, password = ‘abc’    **Error** to display error.  ✔ |
|  | Enter usename = zack, password as ‘zackp’    Redirect to welcome.php  ✔ |

**Part C: Complete "process\_change\_password.php"**

Complete functions in process\_change\_password.php to perform the following:

· Retrieve data from change\_password.php. Check that data are entered.

· Check that the Password and Confirm Password are the same.

· Check that the Username exist in the database.

· Check if the Username, Original Password pair is valid against the data in the database.

· If any errors, change\_password.php will be reloaded, with username displayed in the Username field and display error messages.

· If there are no errors, the new password is hashed and updated in the database. Redirect page to Login.php. The username should be displayed in the Username.

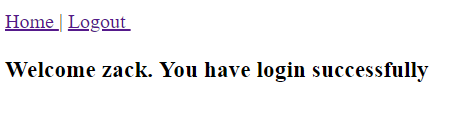
The following shows the flows of the web pages.

|  |  |
| --- | --- |
|  | When the page is first loaded.  ✔ |
|  | Enter username = zack and the rest of the fields empty.    Errors are displayed.  ✔ |
|  | Enter username = zack1, Original password = zackp, New Password = newzack and Confirm New Password = newzack.  ✔ |
|  | Enter Username = zack, Original password = zack, New Password = newzack and Confirm New Password = newzack. |
|  | Enter Username = zack, Original password = zackp, New Password = new and Confirm New Password = newzack.  ✔ |
|  | Enter Username = zack, Original password = zackp, New Password = newzack and Confirm New Password = newzack.    Successful registered and redirected to login.php  ✔ |

**Part D: Complete "logout.php"**

· Check that if no user is not authenticated and login successfully, it will redirect to login.php page.

· If there is a user that is successfully login to the application. The following message will be shown.



· Unset all session variables used.