159.339
Assignment 3 Report

Team
Katie Dempsey - 15309679
Tony Crompton - 17288296
Chris Bishop - 93067924

Our team has developed a simple retail application under the name KTC Tools - For the purpose of providing an example showing understanding of PHP/MySQL/JavaScript, relational database design and real-world application.

Specifications

- Log-in / create user / log-out
- Separate views for admin and users
- Register / Create user page
- View a welcome page when successfully logged in
- Dynamic search page
- Top Menu
- Products with appropriate information
- Presentation - Front-end HTML must be styled using CSS
- Appropriate database

This application provides a simple example of how a retailer might implement an online website for its customers. It allows customers to create an account that can login and logout, view and search for available products.

The UI is simple and easy to use - with appropriate controls for whether you are an administrator or a customer. The App provides simple tabs that users can click on that link to appropriate pages.

The program follows the MVC framework and divides the related program logic into their respective elements - Model, View, Controller. A router provided by the lecturer is used to direct traffic to the appropriate destinations.

A SQL database stores all data made by administrators and customers. The database and tables are filled with some sample data at the first creation of the application.

Specifications set by the assignment outline

The code code used must be PHP 7.1 and MySQL 5.7 compatible.
Must use Docker - to create, deploy, and run the application using containers.
Must implement the coding style guidelines set forward by PSR-1 and PSR-12 standards – As referred by http://www.php-fig/psr/.
The database must be created (if it doesn't exist) and populated with required tables and sample data in the code.
Must use the example MVC application provided as a base for the code.
Database connection - Must use the database name a3 and access it using user root with password root.

Files must be arranged in the following way:

- Controllers in *controller* subdirectory
- Models in *model* subdirectory
- Views in *view* subdirectory

Code must demonstrate an understanding of the following concepts through their practical application:

- Data modelling
- Model-View-Controller architecture
- Object-Relational mapping
- Relational database design and normalisation
- Persistence via Sessions/Cookies
- Object-oriented programming

Design Choices

Passwords

For securing passwords - When a customer creates a user account, the password they choose will be hashed using the php password_hash() function which creates a new hash using a strong one-way hashing algorithm. It will use the BCRYPT algorithm to create the hash. Which will produce a standard crypt() compatible hash using the "$2y$" identifier. The result will always be a 60 character string, or FALSE on failure. As we are not manually entering a salt, a random salt will be generated by the password_hash() function for each password hashed.

CSS & website Layout

The CSS used is simple but effective. With clear buttons and links to different pages. And unobtrusive colourings or stylings. Most forms and data are presented to the user in a table format for easy understanding and formality. There are two separate main views that can be accessed - the admin view and the customer view. This allows for the administrator to oversee and make change. Some features include:

- Admin have full access
- Customers have limited access and only see information relevant to them
- Both can see and search products

Search Page

For the search page we originally implemented this code below which was quick, but it broke the mvc separation of concerns as it was in the product controller.

```
public function searchAction(){
     session_start();
    $output ='';

      if (isset($_SESSION['userName'])) {
    try {
        $collection = new SearchCollectionModel();
        $products = $collection->retrieveProduct($_POST["search"]);
        $output .= "<table>
    <tr>
        <th>SKU</th>
        <th>Name</th>
        <th>Price</th>
        <th>Category</th>
        <th>Stock on Hand</th>
    </tr>";

        foreach ($products as $prods){
            $output .= "
                    <tr>
                        <td>$prods[1]</td>
                        <td>$prods[2]</td>
                        <td>$prods[3]</td>
                        <td>$prods[4]</td>
                        <td>$prods[5]</td>
                        </tr>";
        }
        echo $output;
```

The final code we implemented followed the existing pattern and obeyed mvc rules but was slower to respond. We felt that for assignment purposes and mvc correctness this was the best solution. However we felt we would implement the faster code for a client.

Database Layout

The database is quite simple and self explanatory - with only two tables. A user table which holds all the different user accounts. A product table which holds all the products that have been created. The relationships between these tables are explained and shown below.