SCHOOL OF COMPUTER SCIENCE AND ELECTRONIC ENGINEERING

PRIFYSGOL
BANGOR
UNIVERSITY

ICP 3046
Web Based Applications
E-Commerce Project

Introduction

The aim of these lab sessions is to allow you to build you own basic e-commerce store.

To allow you progress at your own speed the entire semester of lab work is included in this document. Your work will be marked at regular intervals to give you feedback so you can improve in subsequent parts of the assignment.

If you encounter difficulties please ask a demonstrator for guidance. This is a complex lab, and you may require assistance. You will find the lectures and tutorials provide significant assistance in completing the lab work.

Submission guidelines

For each submission you must use **Blackboard** to submit your PHP source code files as a zip archive. Each file must:

- Contain a program header
- Include an appropriate level of comments
- Follow a consistent style of indentation
- Follow consistent naming conventions

The deadline for submission is announced on the Blackboard system and in this lab script. Late submissions will be penalised in line with School policy.

As well as the requirements for each submission being met, marks are awarded for:

- Valid html and css
- Use of good programming practices, such as code-reuse and appropriate input validation
- Comments, layout and code structure
- Conceptual understanding

When submitting work it is your responsibility to ensure that all work submitted is:

- Consistent with stated requirements
- Entirely your own work
- Submitted on time.

Please note that there are **severe penalties** for submitting work which is not your own. *If you have used code which you have found on the Internet* or *from any other source* then you **must** signal that fact with appropriate program comments.

To obtain a mark you **must** attend a laboratory session and be prepared to demonstrate your program and to answer questions about the coding. Non-attendance at labs will result in a significant number of marks being deducted.

Read the requirements carefully and attempt all questions. If you do not know how to complete a specific task, it is advisable to still try writing something. Even if it is not fully functional you may get some marks.

You must use PHP for this lab. Marks are not available for using other programming languages.

Part 1: Page templates and forms [DEADLINE: 10/02/2020]

You need to decide on a theme for your e-commerce store. This part will give you the opportunity to create html templates for a number of pages and then to develop a basic working login form.

This is assessed, and forms 25% of the overall mark for the module.

You must use the file structure that you created during lab 1, and save resources in appropriate directories. For example product images should be stores in the /images/products/ directory.

You must avoid code repetition by using the *require* and *includes* functions of PHP to include a common header and footer on each page.

Feature list

Note that you are **not** expected to get all features working during this lab! Look at the task list to see what is required during the lab.

Product listing page (homepage)

On the homepage of your e-commerce store you will show a site header, that will be on **each page** of your site. You will then show a list of all the products that are available to purchase. It would be advisable to include a small image of each product, along with its title and price. Your page will end with a standard footer that will show on **each page**.

Product details page

The product details page will include more information about the product. It may include a large image (or a number of images), the product title, a textual description of the product, and its price. It should also have some way of adding the product to the shopping basket, and specifying the quantity to add.

Basket page

This should allows the customer (sometimes referred to as the user) to see what they are going to buy, change quantities and remove items.

Registration page

This will allow a user to create an account on your site. They will usually need to supply an email address and password as a minimum. The email address may be used as the unique identifier for a user.

Login page

This will allow a user to login to your site once they have registered. Once logged in they may be able to see details of their past orders, change their password etc.

Checkout process

This will need to gather information such as the billing and shipping addresses. It will also allow the user to choose how they wish to pay, and confirm they agree to terms and conditions. This may be split over a number of pages, or it may be on one page.

Success page

Once a customer has paid they will be shown a page telling them what happens next, and showing their unique order number.

Task list

In this lab you must perform the following tasks:

1) Decide on a theme for your e-commerce store and select at least 3 products which will be available for customers to purchase. Choose a name and price for each product. Write a short description of the product. Find a suitable image to show as a thumbnail on the product listing page, and another image to show as a larger main image on the product details page. [10%]

2) Write html and CSS to produce page templates for **all** 7 of the pages listed above. Your template **must** include all the items specified in the descriptions above. [10%]

   Use sample data to show how the final page will look when it is eventually populated using a database. For example, on the product details page you could show the product price, the product description, the product's main image, a quantity box, and an add to basket button. Even though the **pages are not expected to be functional at this stage**, they should be visually correct. [10%]

   Remember to consider code re-use. You should **not** have duplicate code on each page. [10%]

3) Modify your login page so when the form is submitted it submits to itself (i.e. it is a self-processing page). [5%]

4) Define a static username and password in the file includes/configure.php. These are the credentials which must be supplied in order to log in successfully. These may be stored in plain text. [5%]

5) Create a *function* to check whether a supplied username and password match the static variables defined in configure.php. [10%]

   Your function should take two parameters. [5%]

   Your function should return a boolean to indicate whether the username and password supplied are correct. [5%]

6) Call your function when the login form is submitted, to determine whether the correct username and password are supplied. [5%]

   When the page initially loads, your function should not be called. [5%]

7) If the form is submitted with a valid username and password, a success message must be shown, and the login form must not be re-displayed. [10%]

8) If the form is submitted with an invalid username and password, an error message must be shown, and the login form re-displayed below the error message. [5%]

The username box will be pre-populated with the username that was supplied previously to prevent it needing to be re-entered. [5%]

Part 2: Databases [DEADLINE: 24/02/2020]

Your e-commerce site requires a database. In this part you will create the MySQL tables required for your e-commerce store, and populate them with sample data. You will use this data to show a list of products available in your e-commerce store, and allow a user to register and log on.

This is assessed, and forms 25% of the mark for the module.

You have two lab sessions plus your own time to complete the work.

Task List

1) Create appropriate MySQL database tables to hold information for your e-commerce store (use the feature list below). Remember what you have learned about databases in other modules. [4%]

   Ensure each table has a primary key. [4%]

   Add foreign keys where appropriate. [2%]

   Consider the following features of your e-commerce store when deciding what tables to create:

   a. The ability to list products from your store on the homepage.
   b. The ability to view details of a product on the product details page.
   c. The ability for a customer to register on your site and log in once registered.
   d. The ability to send a customer a confirmation email once payment is received.
   e. The ability to view details of products ordered by a customer, along with their shipping address, billing address and payment method. Remember an order may consist of more than one product, and you should avoid duplicate information in your database.

2) Add your three sample products to the database, along with at least three customer records. You can do this using *INSERT INTO* statements. You do not need to create an administration system for your store. [5%]

3) In your *application_top.php* file, write code to connect to your database. [4%]

   Do not show error messages from the function used to connect to your database unless debugging is enabled. [2%]

   If debugging is disabled, display a generic error message if the connection cannot be established. [4%]

4) Change the login function you created in part 1 so it checks for a valid username and password combination from your database, as opposed to using the static values in includes/configure.php. [10%]

   Ensure your code protects against SQL injection. [5%]

   Remember to test your code thoroughly with valid and invalid combinations of username and password. At this stage the password may be stored in the database using plain text.

5) Take the registration form template you developed in part 1, and ensure it is a self-processing form.

   When a user completes the form there should be a check that their email address is not already in use. Ensure this is done using a *function*. [10%]

   If the email address is in use an error message should be displayed and the user given an opportunity to recomplete the form. [5%]

   If the email address is available the user details should be added to the database. This should be done using a function. At this stage the password may be stored in the database as plain text. [10%]

6) Take the product listing page template you developed in part 1, and populate it with the sample data from your database. Ensure you write a *function* to get the details of all products from the database. [5%]

   Link each product that is listed through to the product details page by passing a parameter in the URL. This parameter identifies the product to be displayed. [5%]

7) Take the product details page template you developed in part 1, and populate it with data from your database. Ensure you write a *function* to get the product details from the database. Ensure you protect against SQL injection. [5%]

   The product to be displayed is to be identified through a parameter passed in the URL (e.g. a product ID). This means you will be able to display details for every product using the same product details page. [5%]

   If an invalid parameter is passed in the URL an error message should be displayed to indicate that the product does not exist. [5%]

8) Ensure there is a form on your product details page that allows you to submit details of what the customer wants to purchase to the basket page. This should include the quantity of the product that your visitor wishes to purchase and a unique identifier for the product. At this stage there is no requirements for the basket page to do anything with this information. [10%]

Part 3: Sessions [DEADLINE: 09/03/2020]

You have already developed a login page, and code that submits products to a basket page. In this part you will ensure that a user who has logged in remains logged in, and that any items added to the basket remain in the basket. You will then ensure that the basket page shows the items currently in the user's shopping cart and provides correct totals in terms of quantities and costs. Finally you will ensure that when a user continues to the checkout they are required to log in.

This is assessed, and forms 25% of the mark for the module.

You have two lab sessions plus your own time to complete the work.

Task List

1) In your application_top.php file, start a session. [5%]

2) Modify your login function so that if a user is successfully logged in their unique user ID is stored in a session variable. [5%]

3) If a logged in user visits login.php ensure they are redirected to the shop homepage and shown a message that they are already logged in. [5%]

4) Create a page logout.php that will log out a logged in user. [3%]

   Once they are logged out redirect the user to the homepage and show a message to confirm they have logged out. [2%]

5) Modify basket.php so that when a user adds a product to their basket, the information regarding what has been added is stored in a session variable [5%].

   Think about what happens if the same product is added for a second time, and write code to deal with the situation appropriately. [5%]

6) Ensure the basket page shows all the items that are currently in the shopping basket. The detail included should be a minimum of the product name and quantity. [10%]

   The product name must be extracted from the database using a *function*. [5%]

7) Ensure your basket shows the total cost of all items in the shopping cart. A *function* should be used to get the product prices from the database to assist with this calculation. [10%]

8) Develop a feature to allow a customer to alter the quantity of an item in their basket. When altering the quantity you can either use buttons, or a text box. Use a *function* to change the quantities. [15%]

   Ensure only positive whole numbers of products can be purchased. [5%]

9) Add a feature to allow a customer to remove an item from their basket. [10%]

10) Add a checkout button to the basket page. [5%]

    When this is clicked on, if the user is logged in they are taken to the shipping details page you developed in part 1 (this does not need to be populated at this stage). [5%]

    If they are not logged in they are automatically redirected to your login page, and on successful login they are taken back to the shipping details page. [5%]

Part 4: Security, OOP and Files [Deadline: 23/03/2020]

In this lab you will enhance the security of your e-commerce system by encrypting appropriate information in the database.

You will also develop a page for uploading new products to your e-commerce store. There will be no need to write a login system for this administration system, as you have already demonstrated an ability to do so. Note that in the real world this would be essential to protect your administration system from unauthorised access.

This is assessed, and forms 25% of the mark for the module.

Task List

1) Login security

    a. Change your registration function so that passwords are stored in the database as an appropriate hash, with a random salt. [10%]

    b. Change your login function so the password a user types on the login form is compared with the hashed value instead of the plain text value. [10%]

2) Encryption

    a. Write a PHP **class** to encrypt sensitive information that is going to be stored in your database. Develop an appropriate constructor to set the key and cipher. [5%]

    b. The class should include functions for encrypting and decrypting information. [10%]

    c. The encryption key must be stored in a configuration file so it can be easily modified. [5%]

    d. Write a PHP script to test your encryption class. [5%]

3) Adding products

a. Write a form to be completed to upload a new product. The form should have fields for the product name, description, price, a small thumbnail image, and a large image. [5%]

b. When the form is submitted the product name, description and price must be appropriately validated. Take steps to protect against cross-site scripting. [10%]

c. The images must be validated to ensure they are image files up to a maximum file size of your choosing, and that their dimensions do not exceed values of your choosing. [10%]

d. If validation passes, the images must be uploaded to your web space, and the product saved to the database. Ensure you protect against SQL injection and use functions where appropriate. [15%]

e. If validation fails an error message must be displayed. The error message should explain what went wrong. Text fields should be re-populated with the values previously entered on the form for usability reasons. [10%]

f. Ensure that the products added using this system show correctly on your frontend shop. [5%]

Part 5: Finishing the checkout [not assessed]

This is additional work to describe what you would need to do to end up with a fully functioning front-end to your e-commerce store. It is not assessed.

In this lab you will finish your checkout process by asking the customer to enter address details and choose a payment method. You will use the PayPal Sandbox to process payments. You will save the details of the products ordered and the address details. You will also ensure that your customer can see details of orders they have placed.

This work is not assessed.

Task List

1) When the checkout page is submitted it should check that a shipping address, a billing address, and a payment method have been entered. It should also ensure that the terms and conditions have been agreed to (probably using a checkbox). If this is all present redirect the user to the success page, otherwise show an error message and allow the user to correct the error.

2) Read about the PayPal Sandbox at https://developer.paypal.com/docs/classic/lifecycle/sb_overview/

3) Research PHP methods for processing payments using PayPal

4) Change your e-commerce checkout so it uses the PayPal sandbox. If payment is made successfully show your normal success page. If payment fails then return to the checkout page.

5) On your success page show a success message to the customer, including their unique order ID. Save the following details to your MySQL database, ensuring personal details are encrypted:
    a. The customer details
    b. The products the customer ordered
    c. The delivery address
    d. The payment address
    e. The payment method chosen
    f. The order date and time

6) Create a new page that shows a logged in customer the order ID and order date of all orders they have placed. When an order ID is clicked on, the details of that order including products ordered, billing address, delivery address and payment method should be shown.

7) Modify your checkout process to send an email to a customer immediately on completion of an order. This email should include the order ID, products ordered, billing address, delivery address and payment method.