

## Assignment 2 of HIT6323/3323 – Web Programming

### **Assignment submission:**

The assignment is due at **8:30am Thursday 24 May 2007** and should be submitted as an **individual** work, using **ESP** (Electronic Submission Processor). ESP details are available at: <https://esp.it.swin.edu.au/docs/ESPGeneralInstructionsS12005.pdf>

**Note:** Make sure to test your program on the Mercury server before you submit. Please compress all your html and PHP files as a zip file to submit via the ESP tool. You can submit more than once via ESP and the current submission overwrites the previous one.

### **Note: You need to have the following files as submission:**

- All HTML files which you have produced.
- All PHP files which you have produced.

**Note:** All screen shots are simply some samples from the look-and-feel viewpoint. You can have a similar user interface for this assignment.

**Note:** All enquiries about the assignment should go to the Coordinating Tutor, Nauman Saeed (nsaeed@ict.swin.edu.au). Enjoy the assignment!

### **Assignment background description:**

For this assignment, you need to work on appropriate HTML, PHP, and MySQL database.

This is a simplified business-to-customer (B2C) e-commerce application about online shopping cart based on the 'eshop' database.

The 'eshop' database should contain two tables, 'items' and 'cart':

- for the 'items' table, you should create the following fields:
  - item\_id – the unique identifier number for each item, e.g., 1,2,3. Set this field's properties as 'integer', 'primary key', 'unsigned', 'not null' and 'auto\_increment'.
  - item\_name – the name of the item for selling, e.g. 'shorts'. Set this field's properties as 'varchar', 'size=20', and 'null'.
  - item\_desc – the description of the item, e.g. 'available in S, M, L, XL sizes'. Set this field's properties as 'varchar', 'size=50', and 'null'.
  - item\_price – the unit price of the item, e.g. '\$39.99'. Set this field's properties as 'double', 'unsigned' and 'null'.
  
- for the 'cart' table, you should create the following fields:
  - item\_id – the unique identifier number for each item in the cart, e.g. 1, 2, 3. Set this field's properties as 'integer', 'primary key', 'unsigned' and 'not null'.
  - item\_name – the name of the item for purchasing, e.g. 'shorts'. Set this field's properties as 'varchar', 'size=20', and 'null'.
  - item\_qty – the quantity of an item selected for purchasing, e.g. '2'. Set this field's properties as 'integer', and 'null'.
  - total\_price – the unit\_price of an item multiplies the item\_qty, e.g. '39.99\*2=79.98'. Set this field's properties as 'double', 'unsigned' and 'null'.

Note 1: You should define the 'eshop' database and corresponding tables exactly as described above. This is essential as your submission will be marked with such a 'standard' database that tutors have for marking.

Note 2: All figures provided are for your references whilst you can have (slightly) different look-and-feel.

### **Assignment requirements specification (part 1):**

#### **Task 1 – Home page: ‘index.htm’**

This should be a very simple Web page containing:

- Your name
- Your student ID number
- Your email address
  
- A statement similar to: “I believe that I have fully achieved Tasks [1..10], partly completed Tasks [1..10], and not tackled Tasks [1..10].”
  
- A statement: “I declare that this assignment is my individual work. I have not worked collaboratively nor have I copied from any other student’s work or from any other source.”
  
- Two buttons or links: the ‘Customers’ button/link provides a link to the front-end of the shopping cart (Task 2) and the ‘Administration’ button/link provides a link to the back-end of the shopping site (Task 7).

## Front-end (public access) of the Shopping Cart (Tasks 2-6):

### Task 2 – List the items on shopping cart - 'list\_items.php' (see Figure 1):

This task will

- check the 'items' table of the 'eshop' database to retrieve all the existing items for sale (ordered by 'item\_id') and display them with: 'item\_id', 'item\_name', 'item\_desc', and 'item\_price'.
- have a link 'Add to Cart' for every item in the list, for adding that item into the cart. Once clicked, the 'item\_id', 'item\_name', 'item\_qty' and 'total\_price' (i.e., item price \* item quantity) of that item should be inserted into the 'cart' table by invoking 'add\_to\_cart.php' (Task 3).
- use the 'cart' table to display the number of items in the cart along with the total price of the cart (see Figure 1 – close to the top).
- contain two buttons or links: 'View Cart' and 'Empty Cart' with links to 'view\_cart.php' (Task 4) and 'empty\_cart.php' (Task 6) respectively.



Figure 1: 'list\_items.php'

### **Task 3 – Add items into shopping cart - 'add\_to\_cart.php':**

This task will

- retrieve the selected record (based on Task 2) from the 'items' table and insert it into the 'cart' table, but not insert this record into the 'cart' table if it already exists, instead, it will increase the quantity by 1 and update the total price of that item.
- redirect the customer back to 'list\_items.php' page (Task 2) automatically for adding more items into the shopping cart.

**Note: Here is a JavaScript for your reference to handle redirection which is simple yet sufficient for this assignment:**

```
<HTML>
<HEAD><TITLE> Loading, please wait </TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
    location.href = 'list-items.php';
</SCRIPT>
</BODY>
</HTML>
```

#### Task 4 – View the items added in the shopping cart - ‘view\_cart.php’ (see Figure 2):

This task will

- be a follow up of Task 2 when the ‘View Cart’ button/link is pressed on the ‘list\_items.php’ page. This task is mainly for the customer to view the items s/he has added into the cart.
- retrieve the item quantity, name, unit price and total price information from the ‘cart’ and the ‘list\_items’ tables. If there are no items in the ‘Cart’ table, display message “Your Cart is Empty” with a link to go back to ‘list\_items.php’ (Task 2).
- display the above information for each of the selected items along with its total price depending on the quantity.
- display the total number of items in the cart and the total price of the cart.
- allow customers to change the quantity of any items and then press the ‘Update Quantities’ button to update (Task 5). The total price(s) of the item(s) and the corresponding total price of the cart should be adjusted accordingly.
- allow customers to remove all items from the cart by pressing on the ‘Empty Cart’ button (Task 6).
- allow customers to go back to the ‘list\_items.php’ page (Task 2) to add more items into the cart by pressing the ‘Home’ button.



The screenshot shows a web page for 'e.shop' with the title 'Your Shopping Cart Details'. It features a table with columns for Quantity, Item, Unit Price, and Total. The table lists two items: 'Shorts' and 'Shoes'. Below the table are three buttons: 'Home', 'Update Quantities', and 'Empty Cart'.

Quantity	Item	Unit Price	Total
<input type="text" value="1"/>	Shorts	\$ 39.99	\$ 39.99
<input type="text" value="1"/>	Shoes	\$ 30	\$ 30
2 items			\$ 69.99

Home   Update Quantities   Empty Cart

Figure 2: ‘view\_cart.php’

### Task 5 – Update the cart - ‘update\_cart.php’:

This task will

- be a follow up of Task 4 when the ‘Update Quantities’ button is pressed, item(s)’ total price(s) and the total price of the cart should be updated depending on the quantities.
- redirect the customer back to the ‘view\_cart.php’ page (Task 4) automatically after updating the cart.

### Task 6 – Empty the cart - ‘empty\_cart.php’ (see Figure 3):

This task will

- be a follow up of Task 4 when the ‘Empty Cart’ button is pressed. All records from the cart table should be deleted.
- redirect the customer back to the ‘list\_items.php’ page (Task 2) automatically with no items in the shopping cart (see top of Figure 3).



Figure 3: ‘empty\_cart.php’

**Assignment requirements specification (part 2):**

**Back-end (administrator part) of the Shopping Cart (Tasks 7-10):**

**Task 7 – Administrator control page - ‘main.htm’ (see Figure 4):**

This task will allow the administrator of this B2C application to insert, update and delete records in the ‘items’ table.

- have a link to insert a record into the ‘items’ table (Task 8).
- have a link to update an existing record in the ‘items’ table (Task 9).
- have a link to delete a record from the ‘items’ table (Task 10).



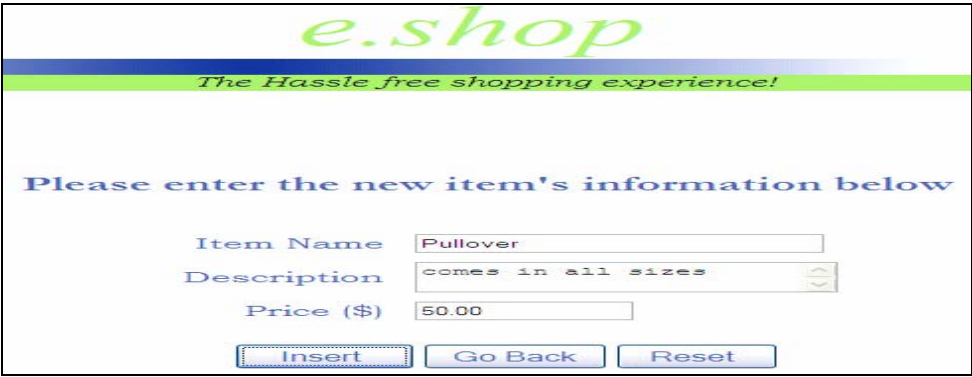
Figure 4: ‘main.htm’

### Task 8 – Insert records into ‘items’ table – ‘insert.htm’ and ‘insert.php’:

This task will allow the administrator to add a record into the ‘items’ table.

- For ‘insert.htm’ (see Figure 5), have an html form with 3 text fields for item name, item description and item price, and 3 buttons/links, namely ‘Insert’ (to insert a record into the ‘items’ table by invoking ‘insert.php’), ‘Reset’ (to reset form entries) and ‘Go Back’ (to go back to the ‘main.htm’ page - Task 7).
- Once the ‘Insert’ button is pressed, i.e. ‘insert.php’ is invoked to insert the record into the ‘items’ table, display a confirmation message for the successful entry or an error message for entry failure in the database, with a link to go back to the ‘main.htm’ page (Task 7).

Note: Since the ‘item\_id’ field is set as auto increment in the ‘items’ table, it is unnecessary to insert the value manually.



The screenshot shows a web form for adding a new item. The form is titled 'Please enter the new item's information below'. It contains three input fields: 'Item Name' with the value 'Pullover', 'Description' with the value 'comes in all sizes', and 'Price (\$)' with the value '50.00'. Below the input fields are three buttons: 'Insert', 'Go Back', and 'Reset'.

Figure 5: ‘insert.htm’

**Task 9 –Update a record in the ‘items’ table – ‘update.htm’, ‘update.php’ and ‘final-update.php’:**

This task will allow the administrator to update a record in the ‘items’ table.

- For ‘update.htm’ (see Figure 6), have an html form with 1 text field to enter the ‘id’ of the item which needs to be updated, along with 3 buttons, namely ‘Update’ (to retrieve the selected record from the ‘items’ table by invoking ‘update.php’), ‘Reset’ (to reset form entry) and ‘Go Back’ (to go back to the ‘main.htm’ page – Task 7). Use proper error handling if the provided ‘id’ does not exist in the database. Note: We assume the administrator knows the ‘id’ by looking up the table if needed.
- Once the ‘Update’ button in Figure 6 is pressed, i.e. ‘update.php’ is invoked, retrieve and display the selected record (see Figure 7) in order to modify the record.
- Once the ‘Update’ button in Figure 7 is pressed, i.e. ‘final-update.php’ is invoked, the change will be recorded in the ‘items’ table. Also display a confirmation message for successful update or an error message for update failure in the database with a link to go back to the ‘main.htm’ page (Task 7).



*e.shop*  
*The Hassle free shopping experience!*

Please enter the ID of the item you want to update

Item ID

Figure 6: ‘update.htm’



*e.shop*  
*The Hassle free shopping experience!*

Change the item details that you want to update then click on the Update button

Item Name	Description	Price (\$)
<input type="text" value="Shoes"/>	<input type="text" value="leather shoes"/>	<input type="text" value="30"/>

Figure 7: ‘update.php’

### Task 10 – Delete a record from the ‘items’ table – ‘delete.htm’, ‘delete.php’ and ‘final-delete.php’:

This task will allow the administrator to delete a record from the ‘items’ table.

- For ‘delete.htm’ (see Figure 8), have an html form with 1 text field to enter ‘id’ of the item which needs to be deleted, along with 3 buttons, namely ‘Delete’ (to retrieve the selected record from the ‘items’ table by invoking ‘delete.php’), ‘Reset’ (to reset form entry) and ‘Go Back’ (to go back to the ‘main.htm’ page – Task 7). Use proper error handling if the provided ‘id’ does not exist in the database.
- Once the ‘Delete’ button in Figure 8 is pressed, i.e. ‘delete.php’ is invoked, retrieve and display the selected record (see Figure 9) in order to delete the record.
- Once the ‘Delete’ button in Figure 9 is pressed, i.e. ‘final-delete.php’ is invoked, the selected record will be deleted permanently from the ‘items’ table. Also display a confirmation message for successful deletion or an error message for deletion failure from the database, with a link to go back to the ‘main.htm’ page (Task 7).

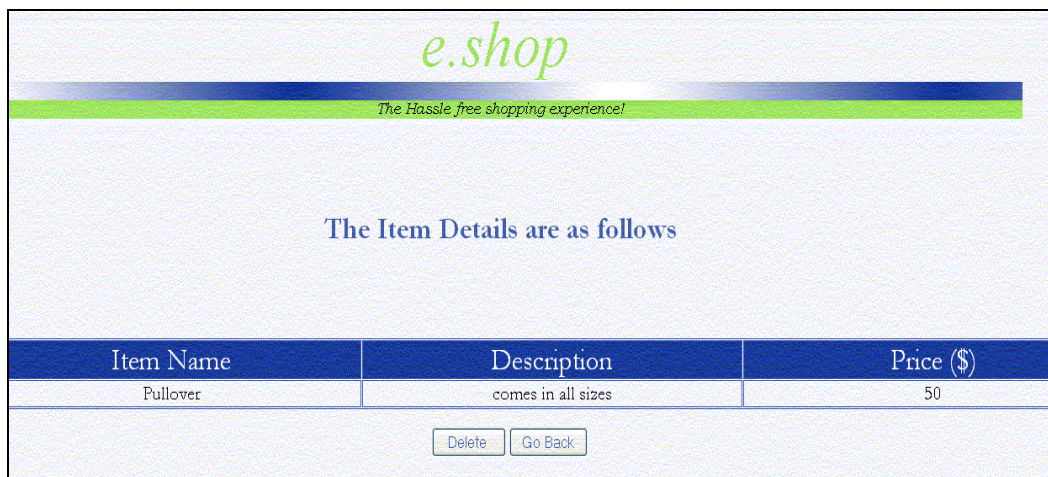


*e.shop*  
*The Hassle free shopping experience!*

Please enter the ID of the item you want to delete

Item ID

Figure 8: ‘delete.htm’



*e.shop*  
*The Hassle free shopping experience!*

The Item Details are as follows

Item Name	Description	Price (\$)
Pullover	comes in all sizes	50

Figure 9: ‘delete.php’