CS2003 Practical Web1 - Items for Sale

Saleem Bhatti

17 October 2018

Deadline: 31 October 2018 (please see MMS)

Description

Your task is to build a simple online shop front that will allow purchases to be submitted, and a 'receipt' issued to the user. The user will select items to buy from a short, pre-defined list from a web browser. The relevant information about the items being purchased will be sent to the server, which will confirm the purchase by generating a 'receipt' as a web page that is displayed to the user.

Technologies you will need to use:

- HTML5 for web pages.
- CSS3 for presentation of web pages.
- Javascript for client-side processing.
- PHP for server-side processing.

---

1 Basic Requirements

Using the files given (see Section 5, below) you are required to change and extend the code as necessary to perform the following tasks:

(B1) Calculate the line-cost (quantity $\times$ item_price), and sub-total cost for the order and display it in on the web page, dynamically, as item quantities are changed.

(B2) Calculate a delivery charge, as 10% of the total order, unless the order total (before VAT is added) is £100.00 or more, in which case the delivery charge is zero. This value should be updated dynamically and displayed on the web page as item quantities are changed.

(B3) Calculate the VAT (at the rate of 20%), as the sum of (i) the sub-total cost of the items, and (ii) the delivery charge, and display it in the VAT box. This value should be updated dynamically and displayed on the web page as item quantities are changed.

(B4) Calculate the total cost of the order, as the sum of the sub-total, the delivery charge, and the VAT. This value should be updated dynamically and displayed on the web page as item quantities are changed.

(B5) At the server-side, if the order is considered correct and valid (see the checks listed below), it should be confirmed with a 'receipt page', otherwise the order should be rejected, and the user prompted to try again. The 'receipt page' produced should include all the relevant order details from the client, as well as some additional details for the transaction (the first item listed below):

- The date of the order, and a unique 'transaction ID' for the order.
- A list of all the items ordered, including quantities and unit costs, as well as line costs. (Items for which a zero quantity was requested should not appear in the order, of course.)
- The sub-total, delivery charge, VAT, and total cost of the order.
- The customer name and delivery address for the order.

- The credit card type, and only the first two digits and last two digits of the credit card number.

(Hint: except for the first item, the 'receipt page', generated by *shopback.php*, will contain similar information to the main page generated by *shopfront.php*.)

The checks that need to be performed before the order can be confirmed are:

- *MasterCard* card numbers should have 16 digits and the first digit should be 5.
- *Visa* card numbers should have 16 digits and the first digit should be 4.
- The credit card security code must have 3 digits and be a positive value.
- Order quantities should be positive, whole numbers.
- Email addresses should be valid: they should contain an ampersand ('@'), at least one dot/full-stop ('.'), no spaces, and the relative positions of ampersand and dots should be correct.
- All personal details and payment details must be completed before the order can be submitted.

(B6) Ask the user to confirm that the information is correct *before* the order is submitted to the server. Various checks should be performed at the client-side. This could be a simple verification through a basic dialogue 'pop-up', allowing the user to agree to submit the form or cancel, which would be a basic minimum. Much better is for the user to be presented with all the order information listed as a web page, and then either confirm submission, or be given the option to go back to the ordering page and modify the order.

When testing your code, please do not use actual credit card information from any real credit cards, such as your own credit card, or anyone else's credit card.

An incremental approach is recommended — make sure that each part works before going on to the next part. (Hint: Make a safe copy of whatever you have got working so far, so that you always have a working solution to submit, while you are developing the next part of your solution.)

2 Additional Requirements

Please make sure that you have a working submission fulfilling the Basic Requirements before you attempt any of the Additional Requirements listed below.

Below are some suggestions for additional functionality that you can chose to implement in your submission. (Difficulty is indicated in brackets.)

(A1) In your report, discuss the security and privacy issues with your application, especially with respect to the kind of information the order contains. *(Easy.)*

(A2) Improve the presentation of the main web-page. This should not just be a case of changing colours and fonts, but something more useful to improve usability, such as improving the layout, making it easier to see which information still needs to be entered before submission, etc. *(Easy.)*

(A3) Allow the server program to record successful orders in a file, *orders.txt*, as text (human-readable) information, stored at the server in the same location as the file *store.txt*. *(Easy.)*

(A4) Create new server-side and client-side code to allow successful orders to be viewed remotely by a web-browser. *(Easy to Medium.)*

- A simple approach will just read the *orders.txt* file (see above) and display the contents to a web-page that is generated dynamically. *(Easy.)*
- A better approach would be to list all the orders (e.g. by transaction ID, or by date), and allow the user to select an order to be viewed separately, e.g. on a separate web-page. *(Medium.)*

(A5) Add an additional 'column' to the *stock.txt* to record stock levels – the quantity of that item currently in stock, i.e. add a new field to the CSV file, which we will call `item_stock`. The `item_stock` should be displayed for the use on the main web-page (generated by *shopfront.php*) in a sensible way. Perform appropriate client-side and server-side checks for orders with respect to the `item_stock`. Update the stock quantity in *stock.txt* when a successful order is made. *(Medium to Hard.)*

(A6) Create new server-side and client-side code that lets you manage the stock for sale remotely from a web-browser. *(Easy to Hard.)*

- A simple approach would just be to allow increase and decrease of stock levels (i.e. adjust the value of `item_stock` – see above). *(Easy.)*
- A more comprehensive approach would enable such things as:
  1. Remove items completely from stock. *(Easy.)*
  2. Change any of the information for existing stock, e.g. name, description, price, stock levels. *(Easy to Medium.)*
  3. Add completely new items to the stock (including upload of photos). *(Hard.)*
  4. Allow remote search of transactions by any specific criteria (e.g. transaction ID, name of customer, etc.), or across multiple fields. *(Hard.)*

3 Submission

Your MMS submission should be a single `.zip`, or `.tar.gz` file containing:

(a) Any PHP, HTML, CSS, and Javascript files. A single directory, called `shopfront` (which may have subdirectories if for resources such as images), with all the source code needed to run your application. Your submission should have server-side and client-side code. It should be possible to take your submitted directory, copy it to School filespace, and for it to be accessible via the School web server.

Please note the following important items:

- You must not use any external / third-party libraries or source code (whether HTML, CSS3, PHP, Javascript) in your solution.
- Please do not use absolute path-names for files or URLs in your submission. You should not need to access any external resources from your code. So, all your HTML, CSS3, PHP and Javascript files (as well as any other files, such as images, or files created by your PHP code) must all be in the single directory that you submit.
- If the marker cannot run the code you have submitted on the Linux workstations in the lab, and access it via the School web-server from either Firefox or Chrome web-browsers, then you will be penalised in marking.

(b) Report. A single PDF file which is a report with the information listed below. In your report, where appropriate, try to concentrate on why you did something (design decisions) rather than just explaining what the code does. In your report, please include:

- A simple guide to running your application, including which web-browser you used for testing – Firefox or Chrome.
- A summary of the operation of your application indicating the level of completeness with respect to the *Basic Requirements* and *Additional Requirements* listed above.
- Suitable, simple, diagrams, as required, showing the operation of your program, especially if you have designed and implemented any of the *Additional Requirements*.
- An indication of how you tested your application, including screenshots of the web pages showing your application working. If you have designed and implemented any of the *Additional Requirements*, then testing and evidence of that functionality should also be provided.

4 A note on collaboration

I encourage you to discuss the assignment with other members of the class. This may be especially useful in examining the code that has been given out as your starting point, but also for discussing both the *Basic Requirements* and the *Additional Requirements*. However, the code you develop for your solution should be your own, individual work, and your report should be written by you alone.

5 Getting started

On studres, you will find a directory called `shopfront` in `studres/CS2003/Practicals/Web1/`. This contains code that you should modify and extend directly for your solution. The files you will find are:

- *shopfront.php*: a PHP file which generates the main web page.
- *shopfront.css*: a CSS file with some basic formatting for the main web page.
- *shopfront.js*: a Javascript file that is used for client-side processing in the main web page.
- *shopback.php*: a PHP file which is a skeleton for the page invoked when the form is submitted from the main page.
- *stock.txt*: a text file which contains a list of the initial items for sale, formatted as CSV records (one item per line), each line being:
  `item_id, item_name, item_info, item_price`
  - `item_id`: a unique identifier for the item.
  - `item_name`: a name for the item, to be displayed on the web page.
  - `item_info`: information / description for the item, to be displayed on the web page.
  - `item_price`: cost of an item, in UK pounds (£), excluding Value Added Tax (VAT – 20%).

You should copy the `shopfront` directory from studres into your own web filespace. This will then be accessible through your School virtual web host. Once you have copied the files, access them through a web-browser (Firefox or Chrome), to understand the current functionality. Please perform all your testing with either Firefox or Chrome web-browsers.

You should be able to complete the *Basic Requirements* by modifying and/or extending these files. The file *shopback.php* should perform the required checks on the order, and then generate a 'receipt' page for confirming the order for the user: a skeleton is provided.

In order to fulfil some of the *Additional Requirements*, you may need to create additional PHP files and Javascript files (and perhaps HTML and CSS files), as well as modify and extend the files listed above.

Marking

Having completed the *Basic Requirements*, to gain a mark above 13, you will need to complete some of the *Additional Requirements* listed above. In your report, highlight where you have attempted any of the *Additional Requirements*. However, please first build a working solution meeting all the *Basic Requirements*, before trying anything more complex.

Note that attempting one or more of the items listed as Additional Requirements does not guarantee you a mark above 13. Your mark will depend on the overall quality of your submission, both the code and the report.

The submission will be marked on the University's common reporting scale according to the mark descriptors in the Student Handbook at:

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors

Lateness

The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):

https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#lateness-penalties

Good Academic Practice

Please ensure you are following the relevant guidelines on Good Academic Practice as outlined at:

https://www.st-andrews.ac.uk/students/rules/academicpractice/