CS2003 Practical Web1 - Items for Sale
Deadline: 28 October 2019

1. 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 and server-side side processing.

2. 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:

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

(2) 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.

(3) 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.

(4) 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.

(5) 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', will contain similar information to the main page.)

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 'at sign' ('@'), 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.

(6) 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.

(7) 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.

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

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.)

3. Submission

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

a) **Any HTML, CSS, and Javascript files**. A single directory, called shop (which may have subdirec- tories 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 run on a school server using the node command (e.g. node shop.js)

Please note the following important items:

You must not use any external / third-party libraries or source code.

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 and Javascript files (as well as any other files, such as images, or files created by your code) must all be in the single directory that you submit.

If the marker cannot run the code you have submitted on the Linux host servers and access it from either Firefox or Chrome web-browsers, then you will be penalised in marking.

Do not change any of the names of the forms, HTML fields, attributes or variables to ease marking load. If you do not comply, 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 Requirements listed above.
- Suitable, simple, diagrams, as required, showing the operation of your program
- An indication of how you tested your application, including screenshots of the web pages showing your application working.

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 the *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 shop in studres/CS2003/Practicals/Web1/. This contains code that you should modify and extend directly for your solution. The files you will find are:

- *shop.js*: a Javascript file which generates the main web page without any functionality.

- *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.
- *piks*: a directory containing various images used by the main web 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 shop directory from studres into your own web filespace. This will then be accessible through your School virtual web host. You will also need to move the shop.js to your node.js directory so that you can execute it as your server. 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 *Requirements* by modifying and/or extending these files. The file *shop.js* should perform the required checks on the order, and then generate a 'receipt' page for confirming the order for the user.

Marking

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/