170014605 31. Oct 2018 Web1 Tutor: Dawand Sulaiman

Usage:
For my testing I used Chrome.

The **shop front** page is implemented in the shopfront.php file. The URL is written below:
https://aci2.host.cs.st-andrews.ac.uk/cs2003/shop/shopfront.php
The quantity input boxes indicate the quantity of the given item you wish to purchase. This value can not be greater than the stock of the item. For the form below, the card number must start with a 4 for visa's or 5 for mastercard's. Once the «Place Order» button is pressed, you may either either click «Confirm» to complete the transaction, or «Cancel» to return to the form.

The page which displays **previous transactions** is implemented in the displayOrders.php file. The URL is written below:
https://aci2.host.cs.st-andrews.ac.uk/cs2003/shop/displayOrders.php

The page which allows the user to **edit the items' stock, description, name and price** is implemented in the updateStock.php file. The URL is written below:
https://aci2.host.cs.st-andrews.ac.uk/cs2003/shop/updateStock.php
Each input box corresponds to the header above. Once you click the «Update Values» button, the stock.txt file will be updated with these values. The values in the boxes when the page is first opened are the current values of the items.

Design:
Basic requirements:
When implementing the basic requirements I first worked on formatting and testing the key value pairs given in the shopback.php file. This involved making sure that the security code on the car was positive and three digits, that the first number on the card number corresponded with the card type and that the total number of items was more than one. If the information was wrong a relevant error message would be printed out along with a button leading back to the shop front. As described in the extension section, these checks have been moved to a redirecting page. I also added a function which formats the key values into more presentable strings.

After completing the receipt page I worked on being able to update the sub-total, VAT, delivery-charge and total cost. This was rather straight forward, as I simply created a new method for each value, all of which are called inside the updateLine() method. This method is called when the value in an item quantity input box is updated. The more complicated part of the shop front was being able to send these values, as well as the line cost value for each item, in the POST values that are submitted with the form. In the end I settled on using hidden input tags, each corresponding with a value to be sent, which are updated each time their corresponding visible tags are updated. This was done because the form only submits information in input tags.

For the user confirmation of the form details, I decided on printing out a summary of the values and information input by the user as well as creating two buttons. One button causes the form to be submitted, the other hides the confirmation information and un-hides the form.

This works because when the «Place Order» button below the form is clicked, the form is not meant to go to the next php file at submition, instead it calls the confirmation() method in shopfront.js which opens the confirmation tag. However, when the «Confirm» button is clicked below the confirmation information, the submition value of the form changes to making the page change to the next php file. The form is also then submited. This is doe with the openReceipt() method in shopfront.js.

Although checks are done in the page redirecting to the receipt page, these are only there to fulfil the practical requirements. This is because all of the neccesary checks are already done on the client side before the form in the shop front is submitted. This means that the only way of triggering an error message in shopRedirect, shopBack or stockRedirect is to open them directly. Here are their URL's:

https://aci2.host.cs.st-andrews.ac.uk/cs2003/shop/shopRedirect.php

https://aci2.host.cs.st-andrews.ac.uk/cs2003/shop/shopback.php

https://aci2.host.cs.st-andrews.ac.uk/cs2003/shop/stockRedirect.php

Extensions:
The first extension I focused on was improving functionality and style on the shopfront.php page. This involves features like the item quantity boxes clearing if the value in them are zero, or becoming zero if the boxes are left empty. An alert is also given if the first number of the given card number is not corresponding with the card type, and the «Place Order» button is disabled. There is also some more functionality related to the stock extension which I describe later. If the user tries to select an item quantity greater than the total stock for the item, an alert is sent and the input value is ignored. Additionally, if an item is out of stock when the shop is opened, instead of displaying the quantity input box, the price and the stock, a message reading «Out of stock!» is displayed instead. This message is highlighted better using a css style. The stock tag in general has been stylised a bit more by adding lines seperating them. The value in the item quantity input boxes are also changed to zero if the input is not an integer, negative or non a number.

A central extension which I created was the stock system. Each item has a corresponding stock value (the last column in the stock.csv file), and this value is decreased each time that kind of item are bought.

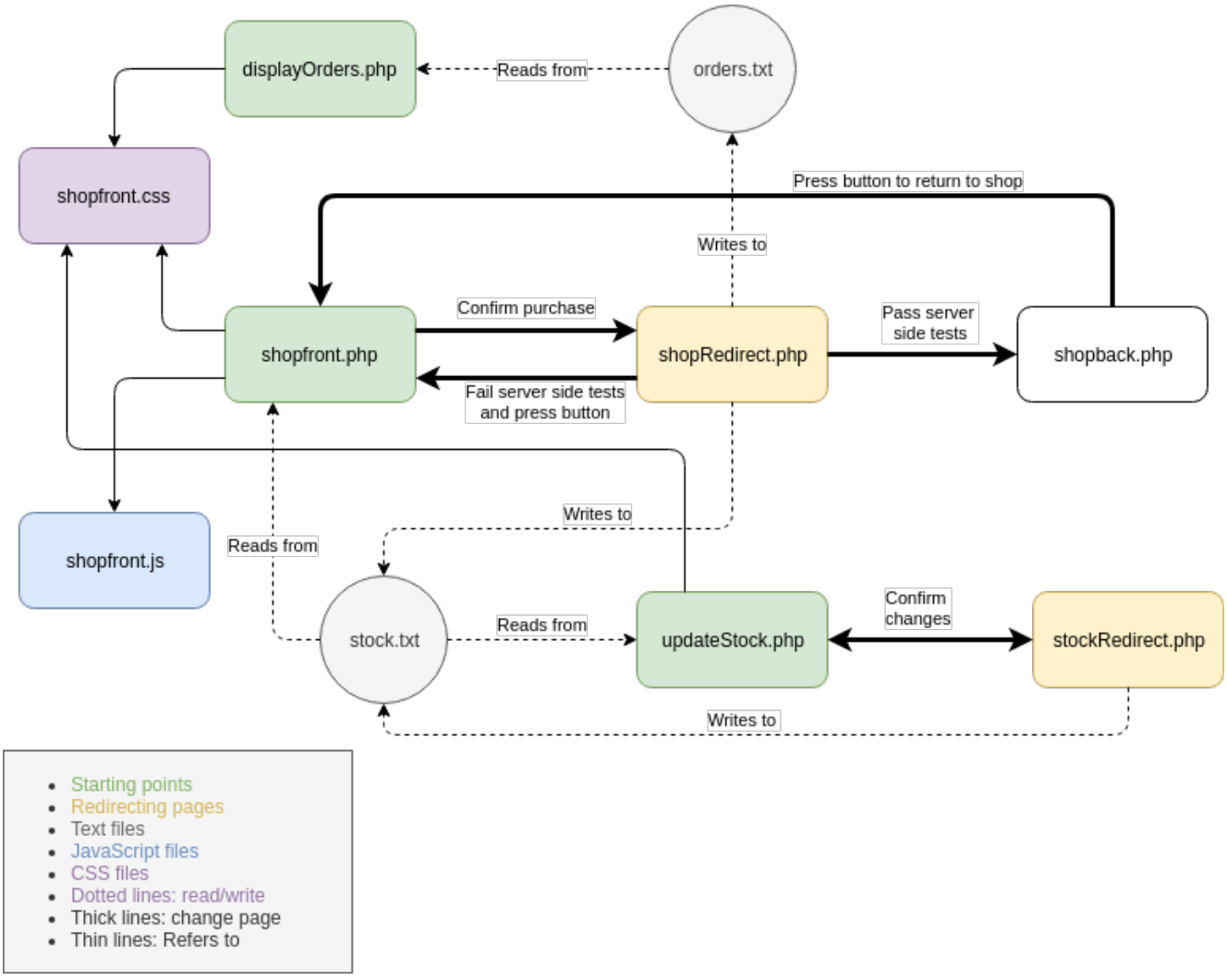
Related to the stock extension is the Item Updates extension. The updateStock.php file allows the user to edit the name, description, price and stock of any item. Once the form is submitted, the input values are passed to the stockRedirect.php page, where the actual stock.txt file is updated. There is also some measures taken to making the updateStock.php more user friendly. Before any values are changed, the default input values are the current values in the file. Additionally, when the box is empty the original values are also displayed. For the «New Stock» column javascript has also been used to clear the input box when the value is zero and the user selects it. If the box is unselected with empty input, then the value will revert to the current value in the file.

The final extension I tackled was to both record and display a record of the completed

transactions. In order to do this, a new file is edited (or created if it does not exist) in the shopRedirect.php page with some relevant information. This information includes the date, transaction ID, total cost of the purchase as well as the items bought and their quantities. These values are then stored as csv values in the orders.txt file. I then created the displayOrders.php page which reads this file and prints out the values into a table. One interesting thing to note about this table is that because the number of different types of items bought is unknown, a for loop is used to create the item quantity tags.

As I have mentioned a few times before, redirecting pages are used twice in my program. This does not only help with making the structure more tidy, but for the shopback.php page it drastically improves functionality. This is because it allows the printed information in the receipt to be static. This means that even if you refresh the page the transaction ID will not change. More importantly however, it allows the page to be refreshed without updating the orders.txt file. Because the new stock is calculated before the redirecting page is reached, the redirecting page is not neccesary to prevent the stock.txt file to be updated though.

Diagram of file structure:



Testing:

Example of confirmation page:

Are you sure that the following information is correct?

Gorilla 2.0: 1
Price: 100.60

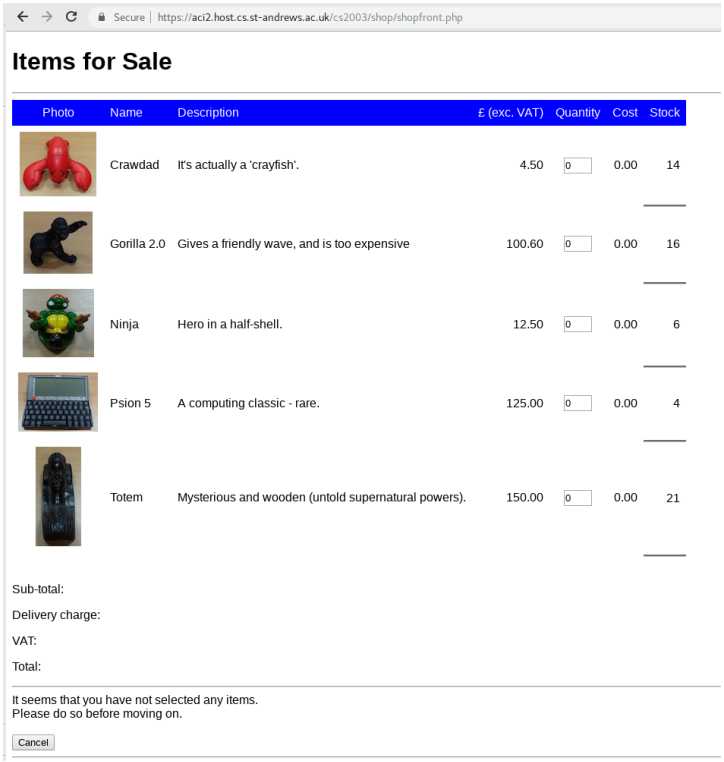
Psion 5: 1
Price: 125.00

Sub-total: 225.60
Delivery charge: 0.00
VAT: 45.12
Total Cost: 270.72

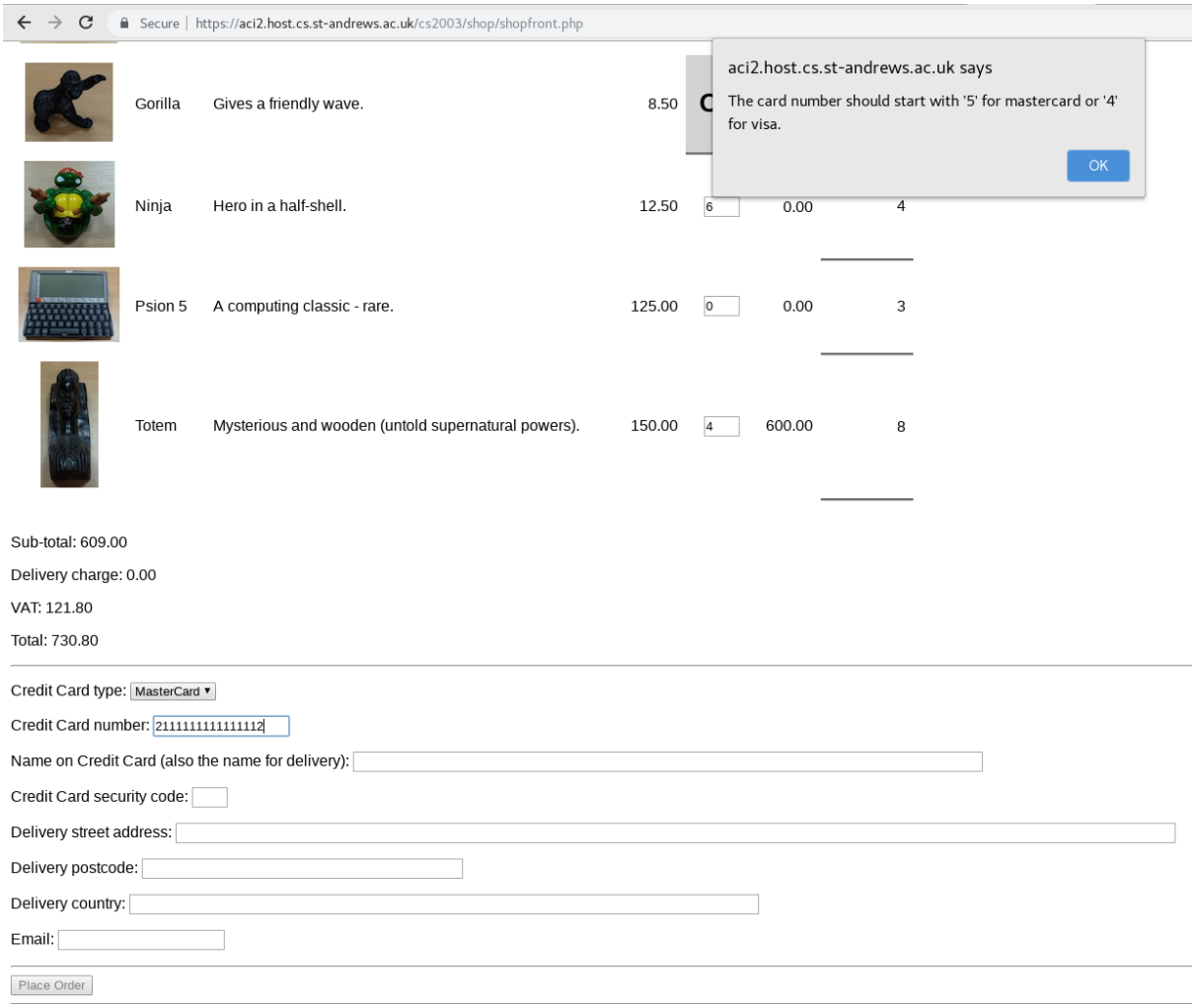
Credit Card type: mastercard
Credit Card number: 5111111111111111
Name on Credit Card (also the name for delivery): Anders
Credit Card security code: 123
Delivery street address: Anders gate
Delivery postcode: Anders tall
Delivery country: Anders land
Email: anders@anders.no

Confirm Cancel

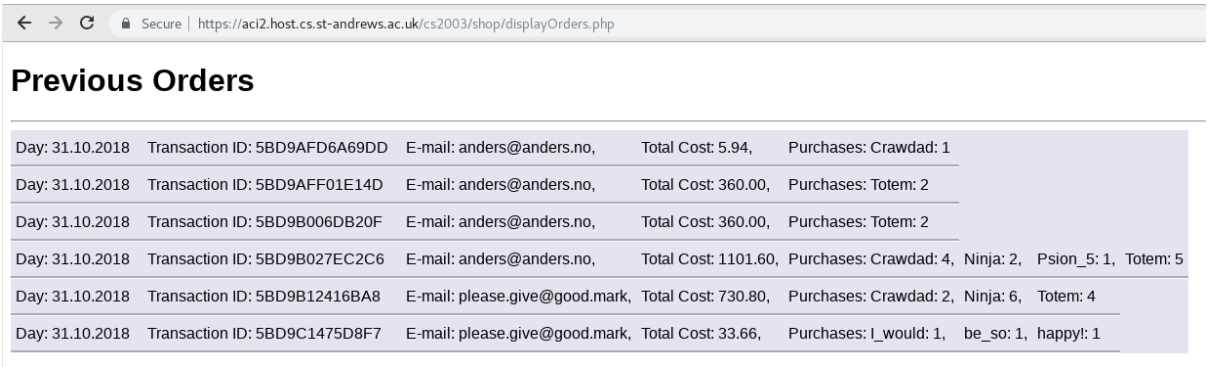
Attempting to buy nothing:



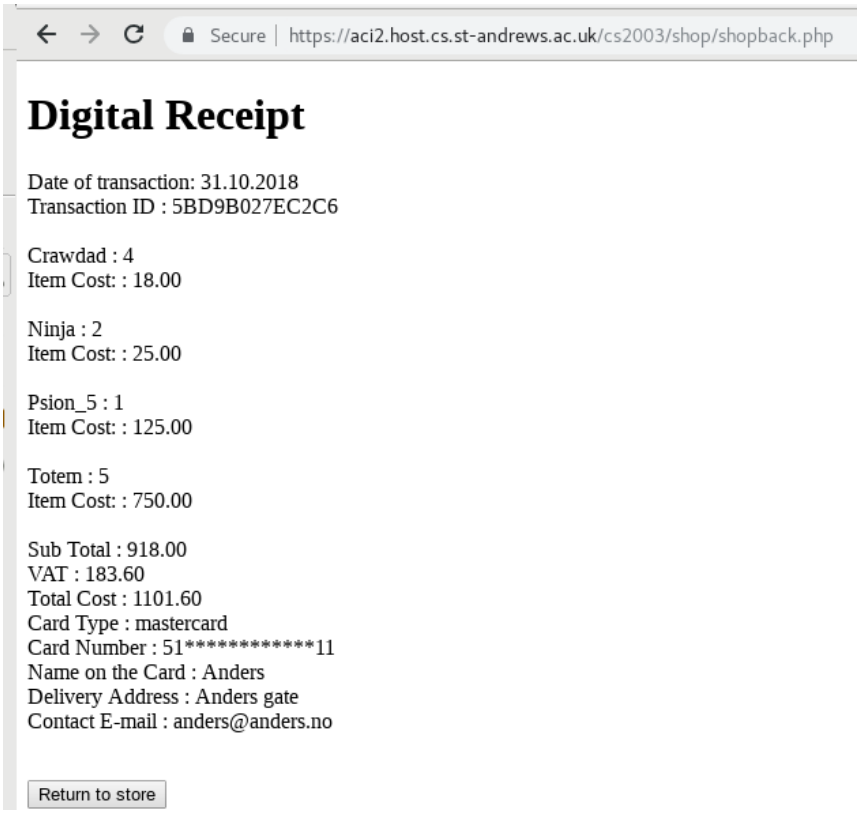
Typing a faulty card number:



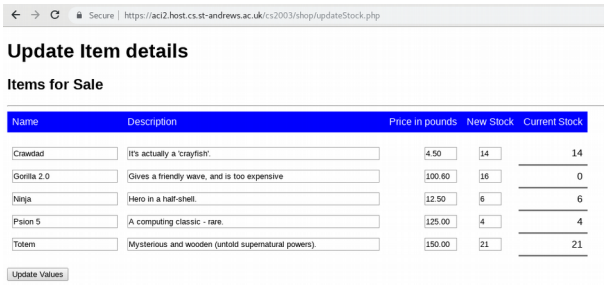
Example of orders page:

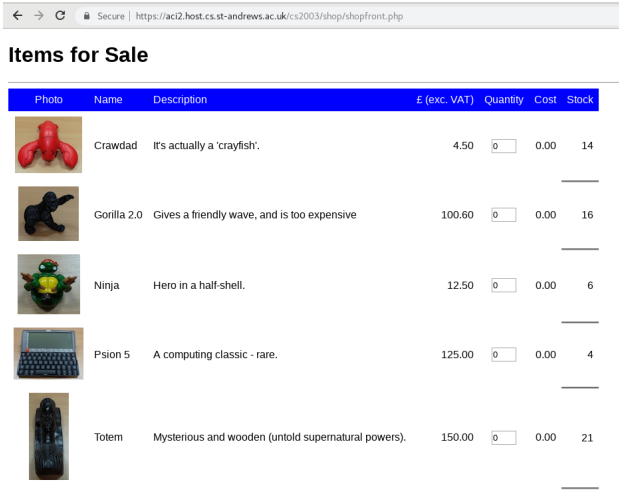


Example of receipt page:

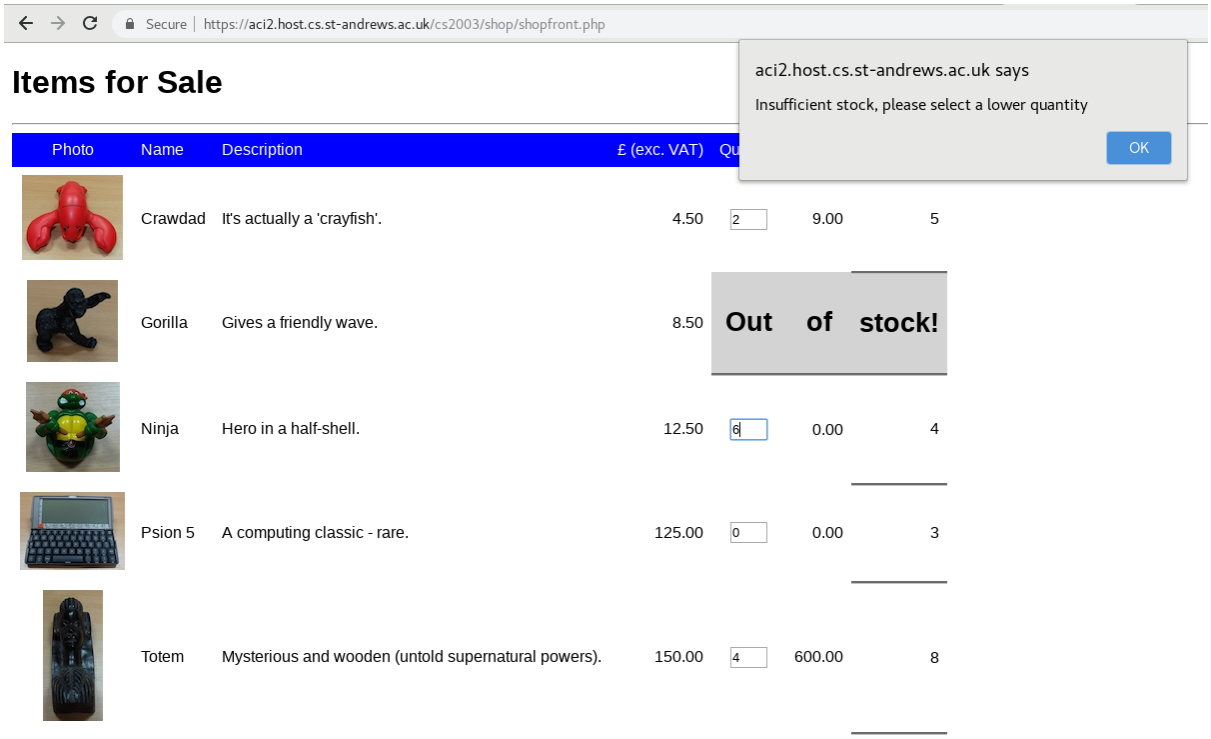


Example of editing details, name and price of at item in the Item Updates page:

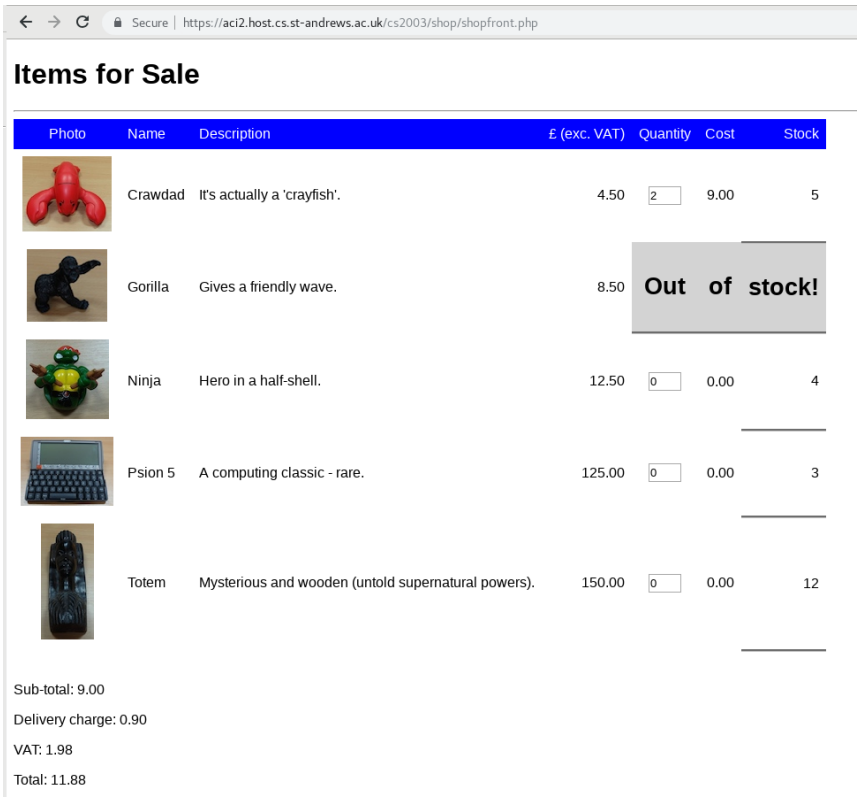




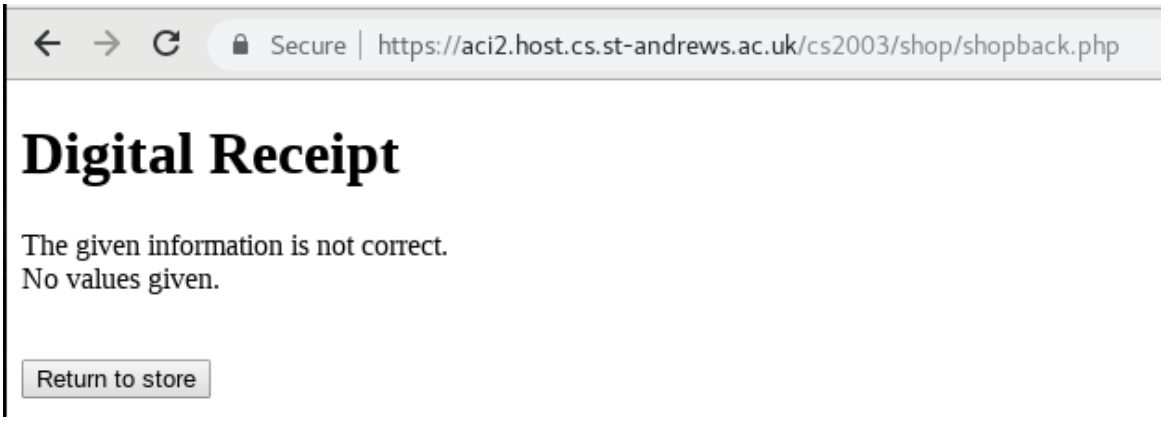
Attempting to buy more items than are in stock:



Example of selecting item quantity and also showing an out of stock item on the shop front:



Error from accessing receipt page directly:



Error from accessing stock redirecting page directly:

Security:

In general, seeing as this is a very informal store, the security is very lacking. For instance, anyone who knows about the updateStock.php page are also able to edit the stock, name, description and price of any item in the store. Additionally, by going to the displayOrders.php page any user can also view the personal e-mail of users who have previously made transactions. Originally I did only make the page print out the transaction ID, date, total cost and the items bought. Later though, I decided that the page would be designed more for the company behind the store as opposed to curious users. It is therefore rather useful to include the e-mail of users. In a real store this information would be stored in a database, and in that case much more information, such as contact address, would also be stored. I decided against that for this page since it would become a bit messy, and since it's mainly a proof of concept it would not be very relevant.

One way in which the integrity of the store has been improved is through the use of redirecting pages. This is because a user cannot update the orders.txt or stock.txt files by simply refreshing the receipt page. Through the use of checks that make sure that the POST header is not empty also ensure that opening these pages directly does also not update the files.

In general, these pages do use https, which in a real store would be essential.

Collaboration:
Most of the time I was working alone, but there were several occasions where I was working next to Sam Goldberg. Additionally, the idea to send an alert to the user if they entered an incorrect first number in the card number input box in the shop front, was his idea.