

Web 1 Report

Usage guide

The testing of my program was done in chrome at <https://sg279.host.cs.st-andrews.ac.uk/shopfront/> with the following extensions.

shopfront.php

This is the main page of the program. The user has the ability to enter quantities for how many of each item they want to purchase. Clicking on one of the boxes to change the quantity will automatically select the contents so that the user doesn't have to manually delete it. Changing this value to anything other than an integer less than or equal to the number of available stock will create an alert for the user (based on if there isn't enough stock or if their input is invalid) and reset the value to 0. If a valid integer is input the line cost will update to the quantity times the price. The subtotal, delivery charge, VAT, and total also update.

Below the list of items and totals is a form in which the user can enter their data. Regex pattern matching, the required property, and input types mean that the user must fill out the form correctly before the form will submit. Additional java script alerts the user and disables the submit button if the credit card number they enter doesn't start with the number corresponding to the selected credit card type. After the form is filled in in an acceptable way, and at least 1 product has been entered, the place order button hides the bottom form, prints out the data in a neatly formatted way, and asks the user to confirm that the information is correct. Clicking 'no' hides the formatted data and displays the original form, and clicking yes will take the user to the shopback page (via the update page).

shopback.php

This page prints out the details of the transaction as well as a transaction ID and the date. There is also a button that will take the user back to the shop front.

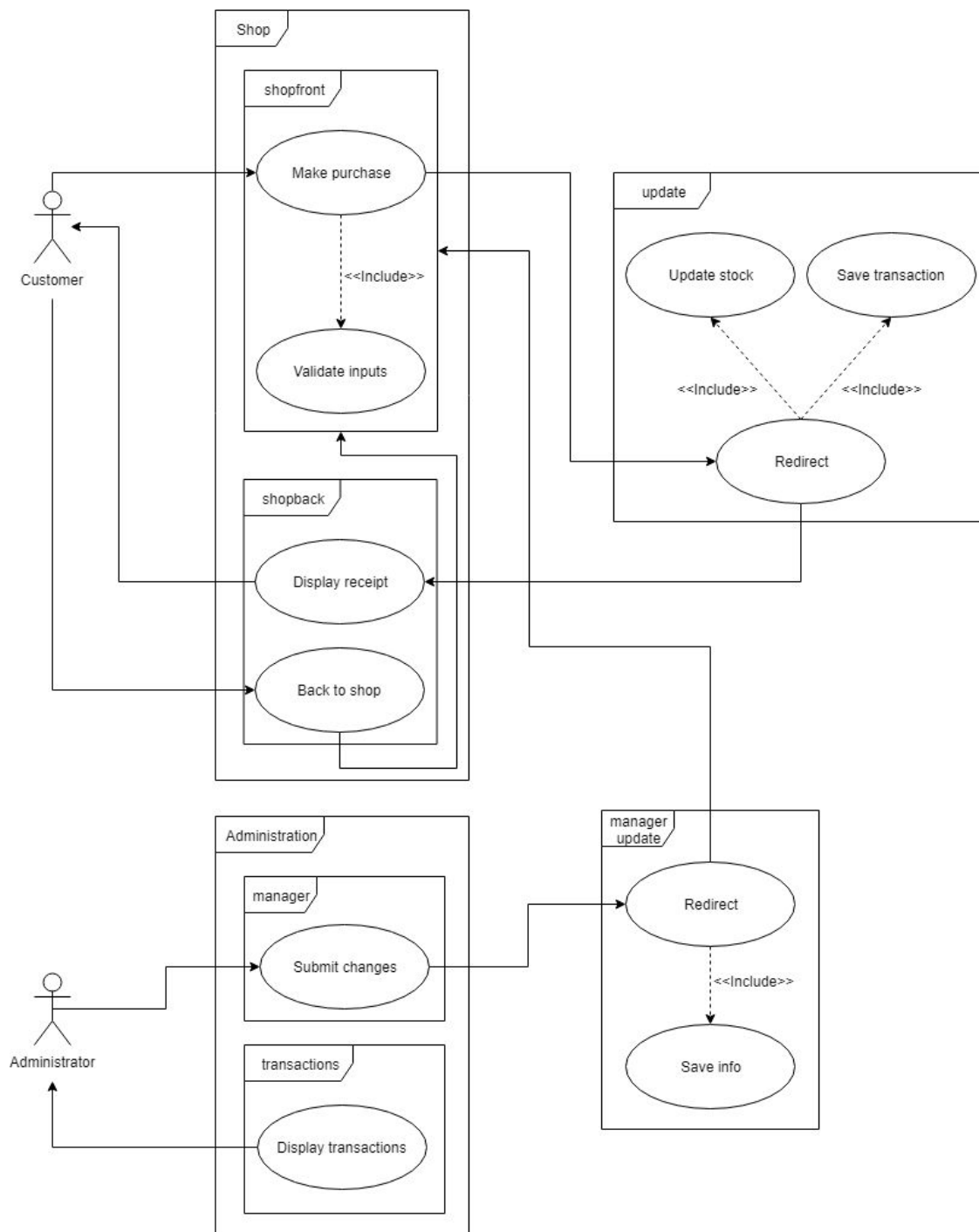
manager.php

This page allows a user to update the name, description, price, and stock of items. The format of the page is similar to the format of the item list section of the shopfront page. However, the quantity and cost columns have been removed, and the name, description, price, and stock items have been changed to input fields. Entering anything other than a number in the price field or integer in the stock field creates an alert for the user and sets the value to 0. Instead of the form after the item list is a button to update the stock. Clicking this will update the stock and take the user back to shopfront.php (via the manager update page).

transactions.php

This page contains no interactivity for the user but displays all of the receipts previously generated as comma separated lists.

Use case diagram



Operation

shopfront.php

Not many changes were made to the structure provided in the existing PHP, except for when reading the stock file, a stock value is read as a part of A5. In addition to this, if the stock value is read to be zero, the stock printed is 'out of stock'. In order to improve the usability of the web page, when the item quantity input is created an onFocus property is added so that clicking on the item quantity box automatically selects the contents. It also calls the updateLineCost java script function when it changes. Furthermore, when the line_cost property is added it is added as a read only input with the name set to the item's ID and cost. This is so that when the form is submitted, the line costs are submitted in an easily gettable way. A similar technique is done with the subtotal, delivery charge, VAT, and total lines. The only changes made to the user info form were to add required properties to the input fields so that they must be filled out, and the card number field calls the cardCheck java script function. The user input form is wrapped in a div tag, and the form action is removed and replaced with an onsubmit property that calls the confirm javascript function (and returns false to prevent the page refreshing). Finally, an empty div tag with the ID form is added to the end of the html body.

shopfront.js

The first function that gets called from the shopfront java script is updateLineCost which is called when the user changes the quantity for an item. This function relies on several other functions.

A function called checkStock gets the user's input for the item quantity and checks that the item quantity isn't out of stock or less than the user's input. If it is, an alert is sent that says there isn't enough stock, and the quantity input is set to 0 if the item stock or the maximum number available if there isn't enough stock. The function also creates an alert and sets the stock to 0 if the user's input is negative or not an integer. This is a part of the A2 requirement, as it means the user doesn't have to wait to submit the form to find out a value entered is invalid and invalid values entered are automatically set to valid options.

The set stock item value has only been changed to change the input value of the line cost element rather than the inner html of the element.

Functions called updateSubTotal, updateDeliveryCharge, updateVat, and updateTotal update the corresponding read only input values to the correct amount using the calculations provided in the specification. The updateLineCost function calls the checkStock function, and assuming that doesn't throw errors calculates a value for the line cost as the item's price times the quantity the user entered and then calls each of the previously described functions in sequence to complete B1-B4.

Another function contained in the shopfront javascript is cardCheck. I put this here instead of the server side to prevent the user having to go back in the vent an incorrect number is entered. This function is called when the credit card number field is changed and creates an alert and disables the submit button if the number entered doesn't match the card type that the user selected.

The largest function in the shopfront javascript is confirm which is implemented to achieve requirement B6. This function is called on the form being submitted. First, if the subtotal is 0 an alert is created that tells the user that no products have been selected and the function exits. Otherwise, the div tag that contains the fields in which the user can enter their data is hidden, and the empty div tag at the end of the html body is populated with the users inputs following the text that was above the input fields. Yes and no buttons are also added that ask the user if the information that they entered is correct. Each button calls a corresponding function.

The yes function edits the page's form to have no onsubmit function call and the form action set to update.php. The form is then submitted with a function call. The no function hides the confirmation form and displays the form in which the user can enter their information which will still be populated with their entries, completing B6.

update.php

The update.php page was created to add security to extensions A3 and A5. Having shopback save transactions and update stock means that if the user refreshes the page the transaction will be saved again and the stock will be reduced again. This is because the form post data still exists even when the page is refreshed. To avoid this, an intermediate page called update.php saves the transaction and updates the stock and then immediately redirects the user to shopback.php.

A function called updateStock takes two parameters, one for the item id and one for the number purchased. The stock file is then read as a csv, and for each row read (as an array), and if the first item in the array matches the id parameter, the stock parameter is subtracted from the fourth item in the array. The array is then added to another array called stock_list as a comma separated string (whether or not stock was subtracted). After all of the items have been read, each comma separated string is written to a wiped stock.txt file as a csv row.

The other functions in the update.php page are to format post keys to user friendly strings, to get values from the post array by key, and to append a string to transactions.txt.

On loading, if the post array has no items in it, the user is immediately redirected to shopback.php. Otherwise, a unique transaction id is created with the uniqid function, and a string called transaction is created that contains the transaction id and the date. Then, for each key value pair in the post array, up until the subtotal key is reached, if the value isn't 0 (meaning the key value pair is an item the user has selected at least one of and the quantity entered or an item the user has selected at least one of and the line cost) the formatted key and value are added to the transaction string and the updateStock function is called with the

key and value as parameters. After the subtotal key is reached, every formatted key value pair is added to the transaction string except for the credit card code. The credit card value string is also formatted to display only the first and last two digits and xs otherwise. The saveTransaction function is then called for the transaction string.

In the html for the page, hidden inputs are created for the transaction ID and date as well as each key value pair in the post and added to a form with the action shopback.php. Javascript then automatically submits this form taking the user to shopback.php. The reason the transaction id and date are submit with this instead of being created at the shopback page is to prevent them from changing when the user refreshes. This completes the A3 and A5 requirements.

shopback.php

As all of the tests for B5 are done client side, the only test done in shopback.php is if the size of the post array is 0, in which case an error is displayed. The reason this is all that is needed is the only way to access the shopback (or update) pages without valid data is to directly type in the url (for the update or shopback php as the update php redirects to shopback if there is no post data), in which case there will be no post data. The shopback page is almost identical to the update.php page, except instead of adding the formatted key value pairs to a string they are printed (the same technique of only displaying non 0 values up to the subtotal to only display items purchased is reused). Finally, instead of a form of hidden inputs, a form containing a button that redirects the user to shopfront.php is added to the end of the page. This completes B5.

manager.php

As a part of A6, a page called manager.php was added. This page uses the same display for items as shopfront.php, but removes the line cost and quantity columns, and replaces the name, description, price, and stock columns with inputs with ids created from the item's id and _name, _infor, _price, and _stock for each element. The stock and price inputs call checkCost and checkStock java script functions when changed. The totals and user input form are removed and the submit order button is replaced with a button that reads Update Stock. The form's action is also set to managerUpdate.php.

manager.js

Manager.js only contains two functions. The first of these is checkCost which is called when the item cost property is changed and receives the item cost input as a parameter. If the value entered can be parsed to a float and is greater than 0 the items price input value is set to the user's input parsed to a float and rounded to two decimal places. Otherwise, an error alert is created and the item cost is set to 0 rounded to two decimal places.

The other function is checkStock and receives the item's stock input as a parameter. If the value entered isn't an integer or is less than 0, an error alert is created and the stock input is set to 0.

managerUpdate.php

This php completes the A6 requirement. This page exists to process the post data submitted by manager.php. It uses the same code as update.php uses to update item stock, but instead of only changing the fourth item in the csv row for matching items, each item in the row is updated to the value of the input with that rows id plus _name, _info, _price, or _stock for the items in the row. These rows are again added to an array as comma seperated strings and written to the wiped stock.txt file as csv rows. The user is then redirected back to manager.php.

transactions.php





This is the simplest php, and simply reads through the transactions file and prints each line to complete the A4 requirement (albeit very basically).

Testing

Due to the nature of some of the functionality described (i.e automatically setting invalid inputs to valid one) it is not possible to show evidence for each use case (as the result just looks like a valid input) but destructive testing was done for each use case as well as testing successful use cases which have been shown where possible.

This shows the totals and line cost being updated:

Items for Sale

Photo	Name	Description	£ (exc. VAT)	Quantity	Stock	Cost
	Crawdad	It's actually a 'crayfish'.	12.00	<input type="text" value="1"/>	3	12.00
	Gorilla	Gives a friendly wave.	3.50	<input type="text" value="0"/>	4	0.00
	Ninja	Hero in a half-shell.	12.50	<input type="text" value="0"/>	1	0.00
	Psion 5	A computing classic - rare.	125.00	<input type="text" value="0"/>	2	0.00
	Totem	Mysterious and wooden (untold supernatural powers).	150.00	<input type="text" value="0"/>	1	0.00

Sub-total: 12.00






Delivery charge: 1.20

VAT: 2.64

Total: 15.84

This shows the totals being updated with no delivery cost:

Items for Sale

Photo	Name	Description	£ (exc. VAT)	Quantity	Stock	Cost
	Crawdad	It's actually a 'crayfish'.	12.00	<input type="text" value="1"/>	3	12.00
	Gorilla	Gives a friendly wave.	3.50	<input type="text" value="0"/>	4	0.00
	Ninja	Hero in a half-shell.	12.50	<input type="text" value="0"/>	1	0.00
	Psion 5	A computing classic - rare.	125.00	<input type="text" value="1"/>	2	125.00
	Totem	Mysterious and wooden (untold supernatural powers).	150.00	<input type="text" value="0"/>	1	0.00

Sub-total: 137.00

Delivery charge: 0.00

VAT: 27.40

Total: 164.40

Invalid quantity (decimal):

Items for Sale

Photo	Name	Description	£ (exc. VAT)	Quantity	Stock	Cost
	Crawdad	It's actually a 'crayfish'.	12.00	<input type="text" value="1.1"/>	3	0.00

sg279.host.cs.st-andrews.ac.uk says
Invalid quantity!

OK

Invalid quantity (character):

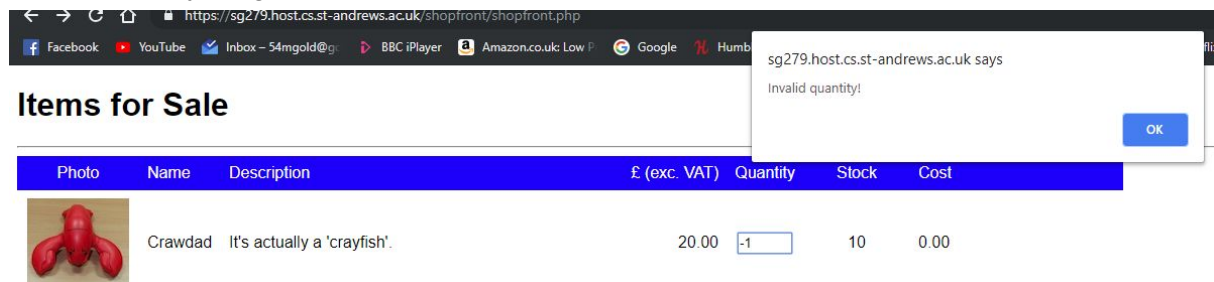
Items for Sale

Photo	Name	Description	£ (exc. VAT)	Quantity	Stock	Cost
	Crawdad	It's actually a 'crayfish'.	12.00	<input type="text" value="a"/>	3	0.00


sg279.host.cs.st-andrews.ac.uk says
Invalid quantity!

OK

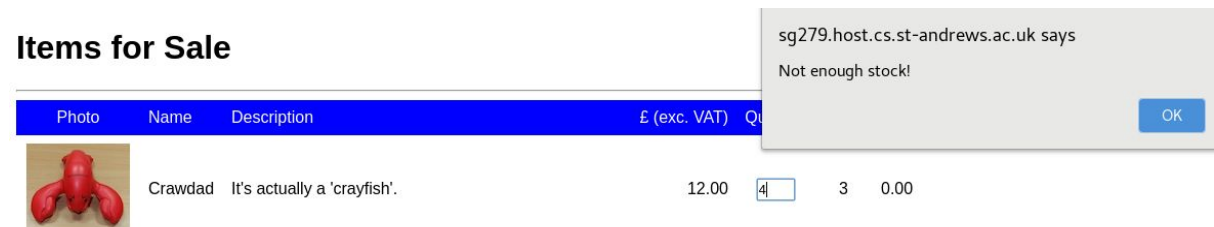
Invalid quantity (negative):




The screenshot shows a web browser window with the URL `https://sg279.host.cs.st-andrews.ac.uk/shopfront/shopfront.php`. The page title is "Items for Sale". A table lists items for sale. The first item is "Crawdad" with a price of 20.00, a stock of 10, and a cost of 0.00. The quantity input field is set to -1. A modal dialog box is displayed over the table, stating "sg279.host.cs.st-andrews.ac.uk says Invalid quantity!" with an "OK" button.

Photo	Name	Description	£ (exc. VAT)	Quantity	Stock	Cost
	Crawdad	It's actually a 'crayfish'.	20.00	-1	10	0.00

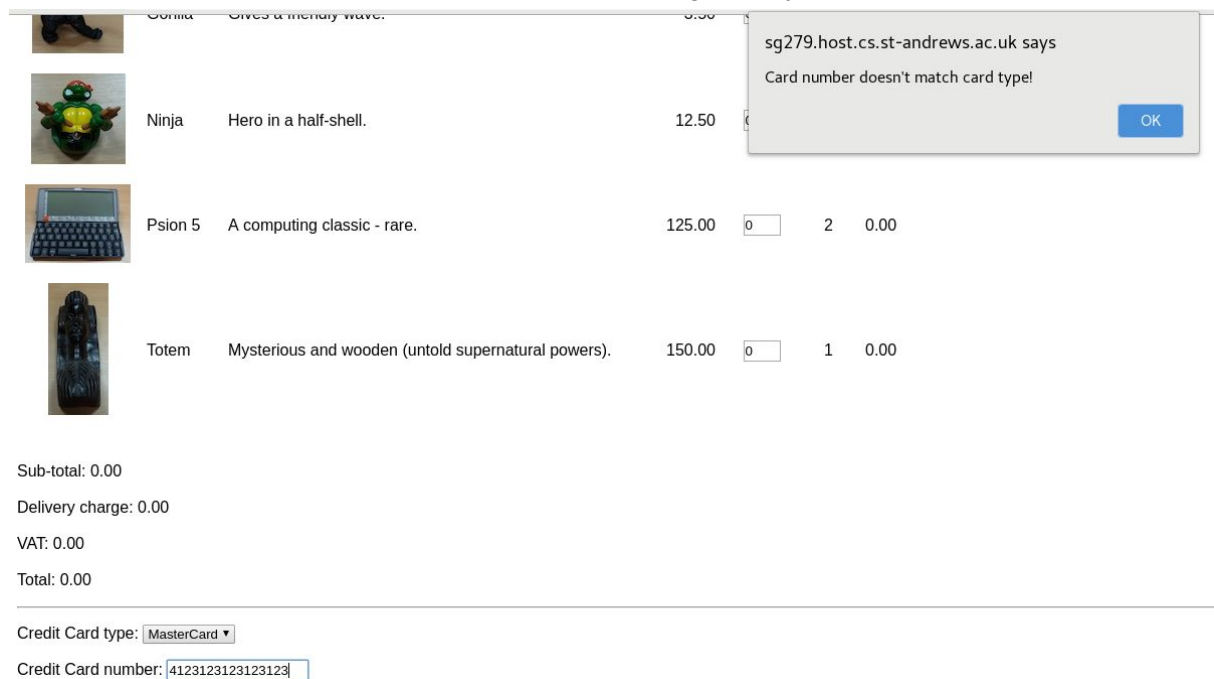
This shows the alert for there not being enough stock and the place order button disabled.







The screenshot shows the same web browser window. The quantity input field for "Crawdad" is now set to 4. A modal dialog box is displayed, stating "sg279.host.cs.st-andrews.ac.uk says Not enough stock!" with an "OK" button.

Photo	Name	Description	£ (exc. VAT)	Quantity	Stock	Cost
	Crawdad	It's actually a 'crayfish'.	12.00	4	3	0.00

These show errors for the card numbers not matching card types.



The screenshot shows the same web browser window. The quantity input field for "Crawdad" is now set to 0. A modal dialog box is displayed, stating "sg279.host.cs.st-andrews.ac.uk says Card number doesn't match card type!" with an "OK" button.

Photo	Name	Description	£ (exc. VAT)	Quantity	Stock	Cost
	Crawdad	It's actually a 'crayfish'.	0.00	0	3	0.00
	Ninja	Hero in a half-shell.	12.50	0	0	0.00
	Psion 5	A computing classic - rare.	125.00	0	2	0.00
	Totem	Mysterious and wooden (untold supernatural powers).	150.00	0	1	0.00


Sub-total: 0.00
Delivery charge: 0.00
VAT: 0.00
Total: 0.00

Credit Card type: MasterCard
Credit Card number: 4123123123123123

Items for sale

https://sg279.host.cs.st-andrews.ac.uk/shopfront/shopfront.php


Facebook YouTube Inbox - 54mgold@g BBC iPlayer Amazon.co.uk Low Google Humb



Ninja

Hero in a half-shell.


12.50



Psion 5

A computing classic - rare.

125.00



Totem

Mysterious and wooden (untold supernatural powers).

150.00

Sub-total: 0.00

Delivery charge: 0.00

VAT: 0.00

Total: 0.00

Credit Card type:

Visa

Credit Card number:

sg279.host.cs.st-andrews.ac.uk says

Card number doesn't match card type!




OK

https://sg279.host.cs.st-andrews.ac.uk/shopping/shopfront.php

Facebook YouTube Inbox - S4mgold@g BBC iPlayer Amazon.co.uk: Low P Google Humb

sg279.host.cs.st-andrews.ac.uk says
Card number doesn't match card type!

OK

	Ninja	Hero in a half-shell.	12.50	0	0.00
	Psion 5	A computing classic - rare.	125.00	0	2 0.00
	Totem	Mysterious and wooden (untold supernatural powers).	150.00	0	1 0.00

Sub-total: 0.00
Delivery charge: 0.00
VAT: 0.00
Total: 0.00

Credit Card type:

Credit Card number:

Credit Card type:

Credit Card number:

Name on Credit Card (also the name for deli

Credit Card security code:

Delivery street address:

Delivery postcode:

Delivery country:

Email:

Clicking place order with invalid entries results in a hint being shown and the form not being submitted.

Credit Card number:




Name on Cr

Credit Card security code:

Trying to submit the form with no products selected:

sg279.host.cs.st-andrews.ac.uk says
You have not selected any products!

OK

	Ninja	Hero in a half-shell.	12.50	<input type="text" value="0"/>		
	Psion 5	A computing classic - rare.	125.00	<input type="text" value="0"/>	2	0.00
	Totem	Mysterious and wooden (untold supernatural powers).	150.00	<input type="text" value="0"/>	1	0.00

Sub-total: 0.00
Delivery charge: 0.00
VAT: 0.00
Total: 0.00

Credit Card type:

Credit Card number:

Name on Credit Card (also the name for delivery):

Credit Card security code:

Delivery street address:

Delivery postcode:

Delivery country:

Email:

This shows a correctly filled out form, enabling the place order button.

Sub-total: 23.00
Delivery charge: 2.30
VAT: 5.06
Total: 30.36

Credit Card type:

Credit Card number:

Name on Credit Card (also the name for delivery):

Credit Card security code:

Delivery street address:

Delivery postcode:

Delivery country:

Email:

The confirmation form:

Is this information correct?

Credit Card type: MasterCard

Sub-total: 23.00

Delivery charge: 2.30

VAT: 5.06

Total: 30.36

Credit Card number: 5123123123123123

Name on Credit Card (also the name for delivery): ron

Credit Card security code: 111

Delivery street address: ron's house

Delivery postcode: ron

Delivery country: ronville

Email: ron@ron.com

Yes

No

The shopback receipt:

Transaction receipt

Transaction ID: 5bd9e99eb5480
Date: 31/10/2018
Gorilla: 3
Gorilla cost: 10.50
Ninja: 1
Ninja cost: 12.50
Sub-total: 23.00
Delivery charge : 2.30
VAT : 5.06
Total : 30.36
Credit card type : mastercard
Credit card number : 51xxxxxxxxxxxxx23
Name : ron
Address : ron's house
Postcode : ron
Country : ronville
Email : ron@ron.com

[Return to shop](#)

An item now out of stock:



Ninja

Hero in a half-shell.

12.50

Out of stock! 0.00

The shopback page accessed directly via the URL (via shopback or update):

Transaction receipt

Error! No transaction

[Return to shop](#)

The transactions page:

Transactions

Transaction ID: 5bd8327c4c169, Date: 30/10/2018, Crawdad: 1, Crawdad cost: 12.00, Psion: 1, Psion cost: 125.00, Sub-total: 137.00, Delivery charge : 0.00, VAT : 27.40, Total : 164.40, Credit card type : mastercard, Credit card number : 51xxxxxxxxxxxx23, Name : Jim, Address : test, Postcode : test, Country : test, Email : jim@jim.com

Transaction ID: 5bd832baaa2a5, Date: 30/10/2018, Ninja: 1, Ninja cost: 12.50, Psion: 2, Psion cost: 250.00, Sub-total: 262.50, Delivery charge : 0.00, VAT : 52.50, Total : 315.00, Credit card type : visa, Credit card number : 41xxxxxxxxxxxx23, Name : steve, Address : street, Postcode : code, Country : country, Email : steve@steve.com






Transaction ID: 5bd834d894643, Date: 30/10/2018, Crawdad: 1, Crawdad cost: 12.00, Sub-total: 12.00, Delivery charge : 1.20, VAT : 2.64, Total : 15.84, Credit card type : mastercard, Credit card number : 51xxxxxxxxxxxx89, Name : bob, Address : bob, Postcode : bob, Country : bob, Email : bob@bob.com

Transaction ID: 5bd9d1b63df81, Date: 31/10/2018, Gorilla: 1, Gorilla cost: 3.50, Sub-total: 3.50, Delivery charge : 0.35, VAT : 0.77, Total : 4.62, Credit card type : visa, Credit card number : 47xxxxxxxxxxxx23, Name : billy, Address : Billy road, Postcode : bill, Country : Bill city, Email : bill@bill.com

Transaction ID: 5bd9e99eb5480, Date: 31/10/2018, Gorilla: 3, Gorilla cost: 10.50, Ninja: 1, Ninja cost: 12.50, Sub-total: 23.00, Delivery charge : 2.30, VAT : 5.06, Total : 30.36, Credit card type : mastercard, Credit card number : 51xxxxxxxxxxxx23, Name : ron, Address : ron's house, Postcode : ron, Country : ronville, Email : ron@ron.com

The manager page:

Item manager

Photo	Name	Description	£ (exc. VAT)	Stock
	<input type="text" value="Crawdad"/>	<input type="text" value="It's actually a 'crayfish'."/>	<input type="text" value="12.00"/>	<input type="text" value="3"/>
	<input type="text" value="Gorilla"/>	<input type="text" value="Gives a friendly wave."/>	<input type="text" value="3.50"/>	<input type="text" value="1"/>
	<input type="text" value="Ninja"/>	<input type="text" value="Hero in a half-shell."/>	<input type="text" value="12.50"/>	<input type="text" value="0"/>
	<input type="text" value="Psion 5"/>	<input type="text" value="A computing classic - rare."/>	<input type="text" value="125.00"/>	<input type="text" value="2"/>
	<input type="text" value="Totem"/>	<input type="text" value="Mysterious and wooden (unt"/>	<input type="text" value="150.00"/>	<input type="text" value="1"/>

The alert for an invalid price (character):

Facebook

YouTube

Inbox - 54mgold@g

BBC iPlayer

Amazon.co.uk: Low P

Google

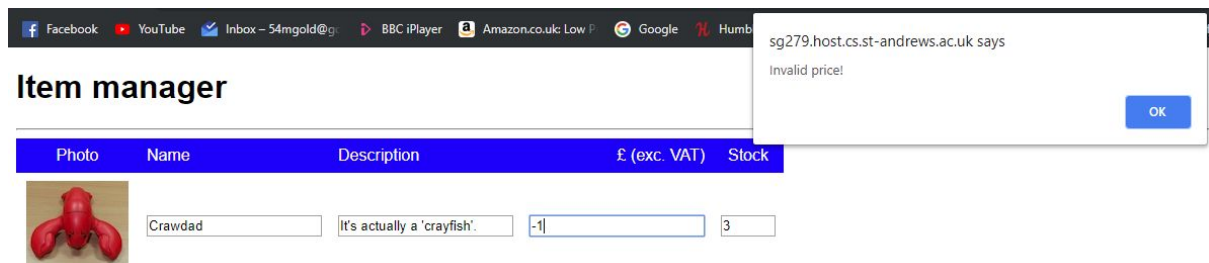
Humb

Item manager

Photo	Name	Description	£ (exc. VAT)	Stock
	<input type="text" value="Crawdad"/>	<input type="text" value="It's actually a 'crayfish'."/>	<input type="text" value="a"/>	<input type="text" value="3"/>

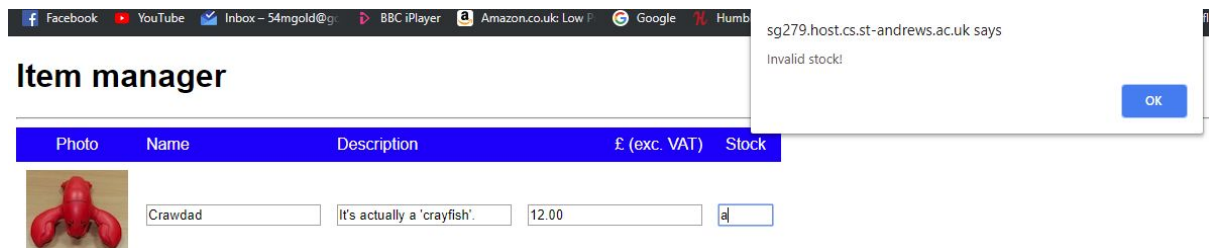
sg279.host.cs.st-andrews.ac.uk says
Invalid price!

The alert for an invalid price (negative):



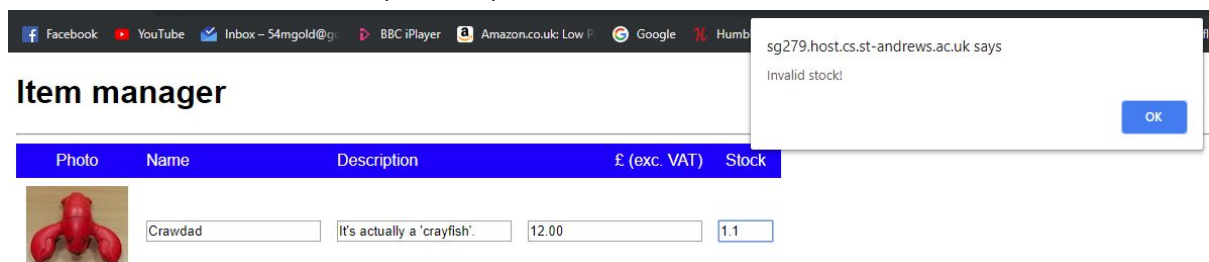
The screenshot shows a web browser window with the 'Item manager' form. The form has a blue header with the title 'Item manager'. Below the header is a table with columns: Photo, Name, Description, £ (exc. VAT), and Stock. The first row shows a red crayfish image, the name 'Crawdad', the description 'It's actually a 'crayfish'', a price of '-1', and a stock of '3'. A modal alert box is open on the right, displaying the message 'sg279.host.cs.st-andrews.ac.uk says Invalid price!' with an 'OK' button.

The alert for an invalid stock (character):



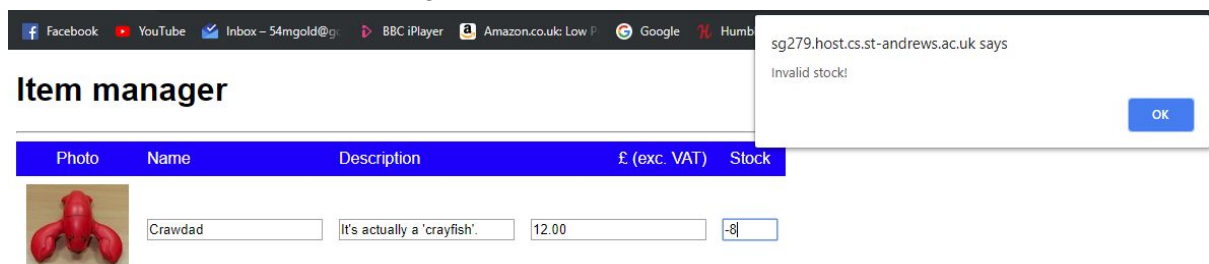
The screenshot shows the 'Item manager' form with the price field set to '12.00' and the stock field set to 'a'. A modal alert box is open on the right, displaying the message 'sg279.host.cs.st-andrews.ac.uk says Invalid stock!' with an 'OK' button.

The alert for an invalid stock (decimal):



The screenshot shows the 'Item manager' form with the price field set to '12.00' and the stock field set to '1.1'. A modal alert box is open on the right, displaying the message 'sg279.host.cs.st-andrews.ac.uk says Invalid stock!' with an 'OK' button.

The alert for an invalid stock (negative):



The screenshot shows the 'Item manager' form with the price field set to '12.00' and the stock field set to '-8'. A modal alert box is open on the right, displaying the message 'sg279.host.cs.st-andrews.ac.uk says Invalid stock!' with an 'OK' button.

Valid changes made to item being visible in the shopfront page:

Item manager



The screenshot shows the 'Item manager' form with the following values: Name 'Crawdad test', Description 'It's actually a 'crayfish'. test', Price '20.00', and Stock '10'. The form is displayed in a clean, modern layout with a blue header and a white body.

Items for Sale

Photo	Name	Description	£ (exc. VAT)	Quantity	Stock	Cost
	Crawdad test	It's actually a 'crayfish'. test	20.00	<input type="text" value="0"/>	10	0.00