Final Specification

WEBSCRP

612596

University of Portsmouth

Introduction

The purpose of this report is to outline what the product has achieved, in the sense of functionality, as well as why it was implemented the way it was. It will also mention the areas of the product that need improving and anything left out, and why.

An area of the site I am rather happy with is the way I have implemented the pages and handled events on them. When I started the project I was unaware of the capabilities of JavaScript. All of my pages loaded as PHP files, and it was a very pre-2000 way of doing things. Since the unit progressed and I researched more and more into JavaScript and new HTML5 APIs, I found the History API.

Though manipulating browser history has been around for a while, it was never as easy as the new History API implemented in the newest version of HTML. This, along with JavaScripts event listeners allows me to completely manage the way the browser handles loading pages. None of the pages are actually loaded, but instead, the site catches the requests, and handles it using AJAX. Once the page has loaded, it then calls a function which fills the page with the relevant data. All event driven.

This only really works in the newest versions of browsers, and is still only partially supported in a few browsers. I feel it was the best way for me in this project to handle page navigation.

Administration & the Content Management System

The way the **Admin** & **Content Management System** *(CMS)* was implemented was in a way that combined the both. While planning the project, it seemed logical to have both the CMS (product & category management) and the admin side intertwine.

This brought benefits to the user as everything they needed was in one place, and all the different functionality *'talked'* to each other. Such as setting the default picture and the products. However, this interweaved functionality brought problems as it took twice as much planning and manpower to handle all the different possibilities.

Basic functionality laid out early on in planning were **core** functions such as adding products and removing products. As it stands I feel the site handles these effectively, using event driven AJAX to send the request and display the results. During development it was mentioned that shop owners never fully remove information about old or outdated products, for audit purposes. Therefore, instead of implementing a complete remove for products, I chose to allow the users to set the *status* of the product. This status seemed the easiest method to completely remove it from the site, but to keep it archived to allow for effective audits later.

Also, outlined in the project plan and specification was extra functionality with adding pictures to products. Initially I wanted to implement an AJAX drag and drop uploader to allow the user to easily drop their pictures onto the site and the site handle it all. However, this functionality turned out to be a little too advanced for the time I had to achieve it in, so it was decided to leave this out. Because of the way the unit was taught, I feel like I learnt the skills to be able to achieve this a little too late, and given enough time, it is something that I would have like to implement. But, this has to be a future enhancement.

Products

The products section of the admin area displays a list of all the products. Functionality that I was keen on including was pagination. So the user can sieve through the pages to easily locate products. The way the pagination works is solely AJAX, and some APIs I have created to handle it. First off it calls the count API which counts how many products there are to be shown, from that number it then calculates how many pages are to be needed. It then calls range.php which retrieves the products for whichever page it has requested. It means that not every product is loaded, but only 10 at a time, speeding up the delivery time of the pages.

I feel that this works really well, and makes good use of APIs. However, I do feel that there is a better way to manage the pages, and if I had enough time it is something that I would have liked to research into.

The initial plan mentioned that the site would include a stock control system. I took a brief look into how to implement this, but could not decipher a relevant method to use, therefore to have some kind of reference, though admittedly not fantastic, the list of products highlights which products currently have low stock. This is highlighted by displaying red on the quantity.

Categories

I was aware from the start that it was necessary to filter products, and I needed a navigation on the public side. Therefore, it seemed logical to implement categories for the products. The current system has functionality for adding categories and removing categories (moving it to trash). However, if you want to edit the category, including its place in the menu, you cannot do so.

This is definitely an area of the site that I would like to improve upon if I had the time to. A piece of functionality that I really wanted to include was a method of dragging a category in the list to reorder it. Making use of the new APIs for dragging DOM objects.

Orders

Fundamental to online shops, staff need to view who has ordered what, so they can prepare it for delivery. After that they need to mark the order as dispatched. This product implements that very basically. The site allows staff to view current orders, including the name of the person who made that order, and which products they have ordered. It also allows staff to mark orders as dispatched.

However, I would have liked to expand on this. At present the only information stored in the database about the customer is their name, therefore the staff have **no idea** where to send the products to, or how to get in touch with the customer. This is something that is easily expanded to include email addresses and/or postal addresses.

The initial specification for the project stated that login systems were not required, therefore I did not want to implement anything too advanced here. I tried to develop it with expansion on mind. But is an area of the site that would require a lot of security and protection.

Settings

The site has some basic settings that the shop owner or staff might want to change from time to time. Initially, because the product is a template for any company, the shop name is simply ‘Shop’. This is most certainly something that the company will want to set themselves, and the settings panel allows them to do just that.

A second basic option that I am sure the owner of the shop would want to be able to set themselves was the default ‘Image no available’ picture (the picture that displays if no image is uploaded on creation). This is something that I was rather proud I got working. Though again, it would be nice if it included a drag and drop file upload, but for now this is effective. It sets the ‘default.jpg’ picture in the ‘media’ directory, which is the file read if no image is uploaded.

The final setting I thought would be nice for the shop owner to have access to was a method to clear the database. This drops all the foreign key constraints, truncates all the tables and reapplies the constraints. Once the staff have played around with the default information pre-loaded when the database is created, they can remove this information and being uploading their own products.

At the moment, there are only three settings. This has great potential for other information the user may want to be able to change, for example, something that I discussed at the beginning of the project was the ability to set the layout and/or colours of the site. This is something that would be a good improvement, allowing shop owners to make sure their website matches the company culture and style.

Public or Client System

So the client side is split into 4 main pages, which collectively handle the entire content of the site. These are **Category**, **Product**, **Search**, and **Basket**, and are easily known by the address bar, for example http://localhost/up612596/**product**/2 displays product with *sku* number ‘2’.

Category handles all the lists of products. Displaying all products under a specific category. Product is the template for all the product pages, and can viewed whenever the user loads up an individual product. Search is self-explanatory and displays all the search results. Finally basket, which again is very self-explanatory, and displays the basket information.

I split the core areas of the public system up into very simple areas to make development as easy as possible, and to make it as readable and understandable as I could for future developers. Trying to get into the habit of documenting code and modularizing.

Modularizing development led to creating APIs to handle server side functions for the site. They dealt with things such as adding data to the database, to wiping the database clean. These APIs made AJAX calling **very** simple, and understanding the code.

Searching

The site makes use of a simple search feature. This takes information supplied by the user and, using AJAX, returns and displays the information. Because the code was split up into APIs it made it surprisingly easy to handle this request. The search box accept both searches by title and by *sku* number, this was to maximise of results or accuracy of results.

Basket & Stock

The basket makes use of session storage. When the customer adds a product to the basket, the basket is a group of objects and each object represents a product. When the basket page is loaded, it takes the information in from the session and displays it.

Whenever a product is added to the basket, it holds the stock from other users (takes the quantity from the total stock). Making sure other customers cannot order the same product as well. The opposite occurs when the customer removes it from the basket, it adds the stock back to the product.

However, there is a few bug with this system. There are 2 ways of clearing the session data, by using the built in methods, which the site handles, and also by closing the browser. This the product does not handle, and if the customer has added products to the basket and closes the browser before they confirm the order, the stock is lost forever. It was something that would need to be looked into before the site was made commercially available.

Statistics - Or lack of?

An area of the site that I wanted to include from a very early stage was order statistics. The total revenue, revenue over a period of time, units sold etc. However I did not get around to the implementation. It would be the first future enhancement I would add to the site, as I feel it would make my interpreted solution a better and more full one.

Conclusion

As the project progressed, the technical specification changed tenfold. The initial plan was very complex and ambitious, and as time moved swiftly on functionality was striped back to the bare minimum. I do feel that, though the sites functionality was minimum, it carries it out effectively.

The efficiency of the code is not particularly great. I feel it is okay, and does the job well. However, while coding I noticed early on that I was repeating a lot of code. As a good example of this, is the AJAX calls. Every time an AJAX call is required, it creates a new XMLHttpRequest object, opens and sends it, checks the return status and executes the code if successful. This repetitive code occurs too many times, and I am aware that there is a function that can be written to deal with the requests, to reduce code. But, by the time I learnt about this, it was too late to implement, and is something that I would definitely change in the future.

I feel that my interpretation of the solution is fit for purpose and has solid functionality. It contained a few little bugs that were beyond my ability within the time I had to fix them, but all in all I feel that the product was a success. Something that I would definitely take away from this project is that I should make it a regular occurrence to test development on other operating systems, mainly Windows (as an OS X user). But otherwise I have learnt a great deal throughout the course of the unit and will be applying it to new and future projects.