

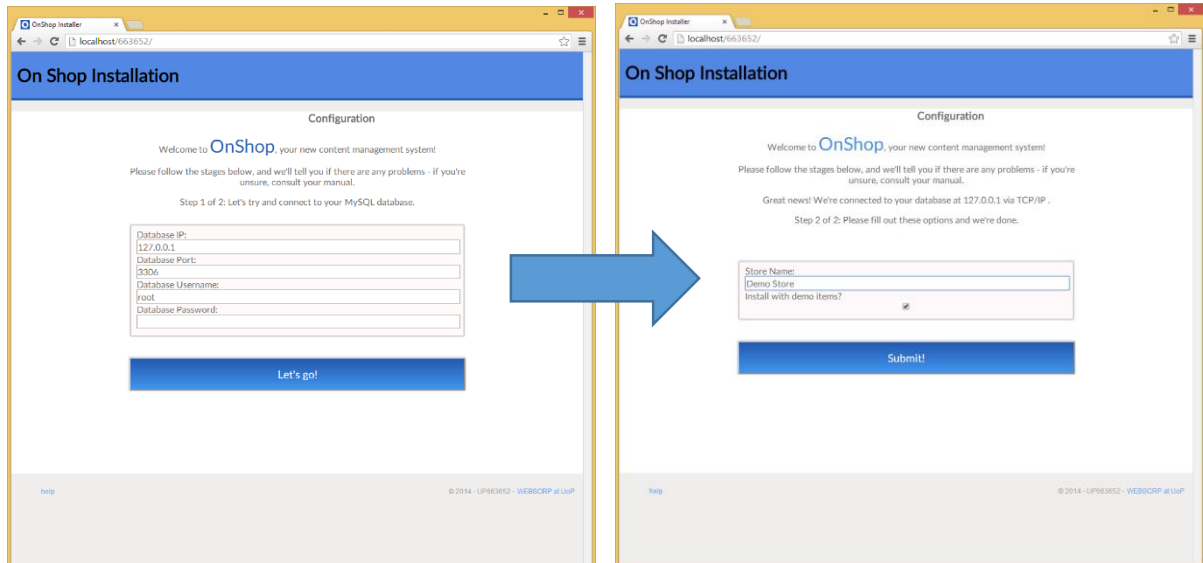
# WEBSCRP – FINAL SPEC

Specification and Rationale by 663652

## Core Functionality

### Installation

When the application is loaded for the first time, the user is shown an interactive installer. This begins by asking for database connection information, then testing those details. If successful, the user is shown customisation options, and finally sent to their new store homepage.

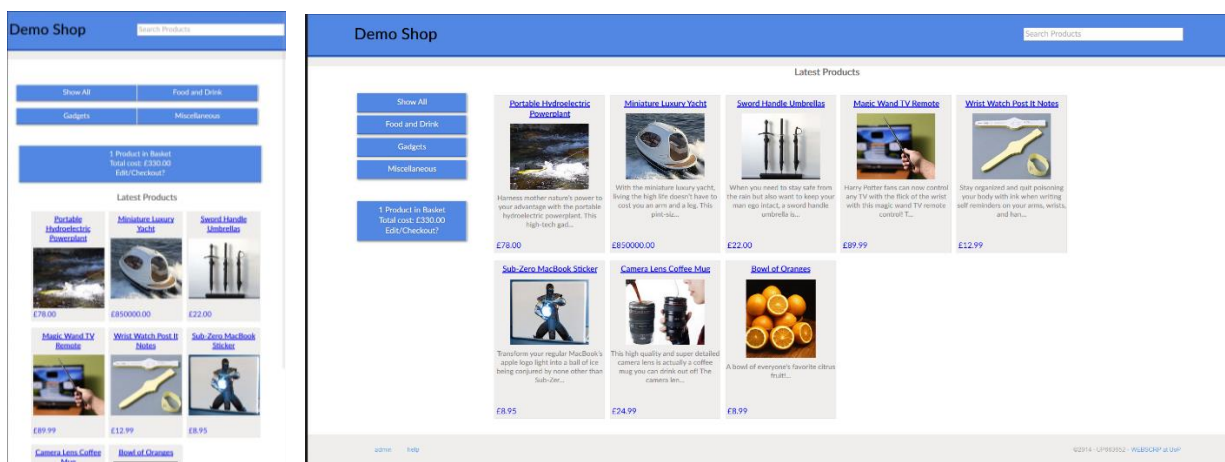


Information is shown to the user at every stage. This includes errors received by the server, to help troubleshoot any installation problems. Animations are used throughout to guide the user.

This approach was chosen in an effort to make the product as simple as possible to install, as well as create a positive first impression with the user (who at this stage, may be a webmaster analysing the product for future use).

### Design

The design was created to work on a broad range of devices. Responsive CSS allows some elements to change size and positioning when certain resolution conditions are met. Below, the view of the store on an iPhone size window and a wide desktop screen.



Changing the design on different resolutions allows for better user experiences. With limited screen space, the extract of the description is less appropriate, so large, touch-friendly product images are shown instead. Where screen space is less restrictive on the desktop, the product block can be larger and contain more information.

## Categories

Each product is assigned a category when added to the system. Categories which contain products are displayed to the client through the side-options panel. Clicking a category name filters the list of products to those matching. This filtering is done entirely by the client to save additional page requests and to lower latency.

New categories can be added through the admin control panel, and empty categories can be removed. The empty-only approach prevents the administrator causing any errors or difficult to locate products.

## Product Management

New products can be added through an admin form employing extensive validation. Only the product description and product image fields are not required. Example input is listed in placeholder elements to help the user. Feedback is shown with any errors or on success. A progress bar is displayed while the data is sent to the server.

Demo Shop

Admin Control

Manage Items  
Add an Item  
Manage Categories  
Add a Category  
View Open Orders  
View Completed Orders

Product Name:

Product Category: Food and Drink

Product Image:  No file selected.

Product Description:

Enter initial stock:

Enter price:

Submit Items:

Maximum lengths and regular expressions are employed to ensure appropriate input is entered, which means that correct and expected data is displayed to the user later on. Without max lengths for example, the strings may be silently truncated by the database, and the user confused. Feedback messages attempt to prevent any misunderstandings of the form.

After a product has been added, the stock and price of that product can be changed. This is done through the Manage Items menu option, which displays a table of existing values.

Admin Control						
Products Sorted By Stock						
Product ID	Product Thumbnail	Product Name	Product Price	Product Stock	Product Sales	Update
7		Miniature Luxury Yacht	850000.00	<input type="text" value="1"/>	0	<input type="button" value="Done"/>
6		Sword Handle Umbrellas	22.00	5	0	<input type="button" value="Edit"/> <input type="button" value="Remove"/>
8		Portable Hydroelectric Powerplant	78.00	5	0	<input type="button" value="Edit"/> <input type="button" value="Remove"/>
5		Magic Wand TV Remote	89.99	6	0	<input type="button" value="Edit"/> <input type="button" value="Remove"/>
2		Camera Lens Coffee Mug	24.99	10	0	<input type="button" value="Edit"/> <input type="button" value="Remove"/>

An edit button is shown in each row, which turns the price and stock fields into inputs, allowing the values to be updated in the database through a PATCH request when 'Done' is pressed. Feedback from the API is shown above the table, so the user is confident the changes have taken effect, or can resolve any input mistakes.

The decision was made not to reload or reorder the table after an item has been updated, so that the user does not lose track of items, or lose any other edits made at the same time.

## Basket

A basket preview is displayed to the user on every page. This preview is still shown when empty, to demonstrate the functionality is present to the user. Multiple quantities of different products may be added, and the basket will update with a product count and total cost immediately. This basket is stored in the client browser's local storage. This approach allows the customers basket to be retained across multiple visits and sessions. Local storage is an improvement over the use of cookies, as the data is no longer transmitted with each request – saving bandwidth.

When the user clicks on their basket, they are shown a screen to manage their basket. A table is displayed summarising their basket and allowing the removal of individual items.

Stock is decreased when added to a user basket, and increased again if removed from the basket. This is to ensure different customers do not add quantities of products resulting in a combined total greater than the stock present – and orders being unfulfilled.

## Orders


Underneath the basket table is a form for basic customer information. This validates input to help ensure the customer has entered correct details, and enough details to be contacted if there is a problem.

# Demo Shop

### My Basket

[Back To Products](#)

1 Product in Basket  
Total cost: £330.00  
[Edit/Checkout?](#)

Product Name	Product Thumbnail	Product Price	Product Quantity	Quantity Cost	Update
<a href="#">Sword Handle Umbrellas</a>		22.00	15	330.00	<a href="#">Remove</a>

Name:

Address:

Country:

Email address:

Payment details: An external service such as PayPal or any number of merchant services could be used here to manage transactions securely.

All done?

[admin](#) [help](#)

©2014 - UP663652 - WEBSCRP at UoP

On submission of the form, a success message is shown and the basket reset; or messages describing any errors with the data or server response.

## Extended Functionality

### Images

A single image can be attached to a product on addition, through the add product form in the admin pages. This image is restricted to 8 Mb in size, and to common supported formats (JPG, GIF, BMP, WEBP, SVG, and PNG). This is to prevent users uploading very large files which would slow page loads, or formats without widespread support which clients may not be able to view.

The images are placed in a 'Products' directory in images, and renamed to their matching product ID. If an item is removed, this directory is searched for an image matching that ID, and removed if present. This prevents unnecessary disk space being used by images no longer in use. Placeholder images are used if an image isn't uploaded, to keep the design consistent.

As users may upload different sizes of images, the display is always controlled through CSS properties. The images are never stretched beyond their original sizes however, to ensure no loss of quality.

### Search

Products are filtered as the user types in the search box. On each key press, the product names and descriptions are checked for matches. Products which do not match all of the search terms entered are hidden through an animation.

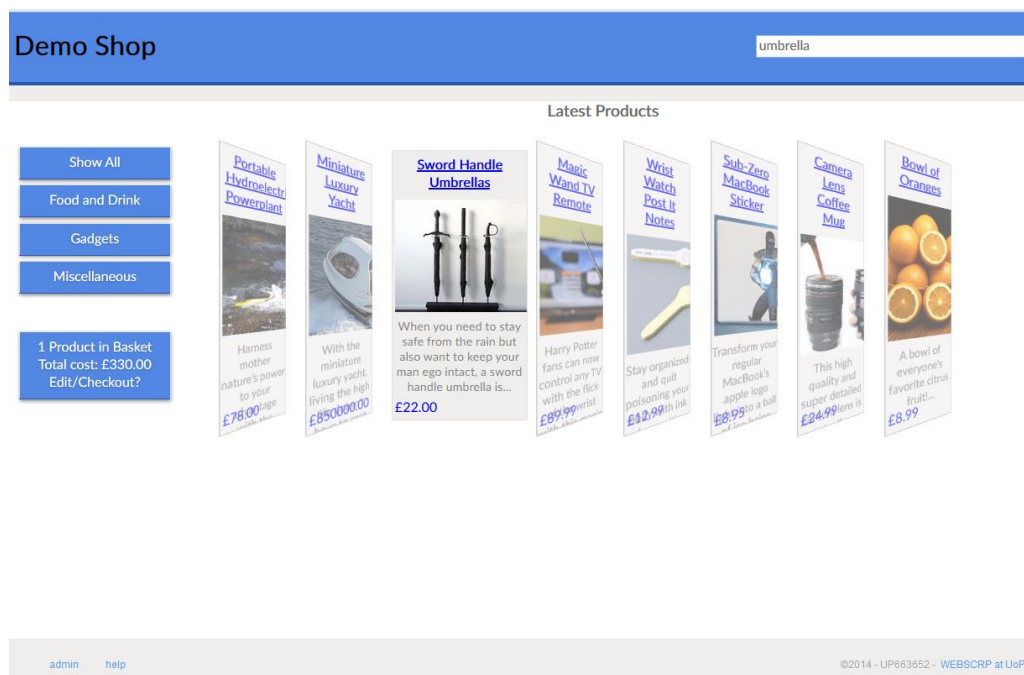


FIGURE 1: MID-TRANSITION PRODUCT FILTERING

Allowing multiple string input allows the user to be more specific with their search. Animating the results live lets the user know when they have found their product without having to submit and then modify their query afterwards.

This search is performed completely on the client, to provide instant results and reduce both bandwidth usage and database requests.

## API

The application interacts with the database entirely through an API. This means an entirely different application could also utilise the same data, or build upon the features offered.

The API attempts to be as RESTful as possible, by not creating any states or sessions with connected clients. This would allow the underlying API to be split across multiple servers without session related problems.

HTTP request types are used extensively throughout the API. For example, the product API supports GET, POST, PATCH and DELETE requests, each to perform a specific function on the relevant product. This gives improved idempotence, where multiple requests for the same method on the same element would not alter the data after the first call.

The implementation of the API (PHP) is hidden for the purposes of security. The API is placed in a wrapping directory with a version number, to allow gradual transitioning over to a new version if required in the future.

## History & Bookmarking

Despite being largely a single page application, history states and new URLs are pushed to the window so that users can navigate in a natural way (through forward and back browser actions). This also allows the user to bookmark specific products, which may help increase sales, should the user want to return for a specific item and can locate it easily.

## Code Validation & Extensibility

Efforts have been taken through validation, linting, and testing to ensure the code is both as correct and as readable as possible. This was done so future developers may understand and further modify the source as easily as possible.

## Issues

There are some issues with this implementation.

- Stock may sit idle in customers' baskets, and no actual purchases made.
- Accessibility is low due to the need for click events on many UI elements.
- PHP implementation of store-front is exposed.

## Improvements

Below are suggestions for future improvements of the product:

- Allow categories to be added simultaneously with product additions.
- Enable keyboard navigation and operation.
- Multiple product images.
- Sub-categories.
- Manage multiple currencies.
- Only load a subset of products initially. Load more as the user scrolls down the page.
- Allow a user to sort products.
- Generate additional admin information, such as graphs of orders over time.
- Promotions and coupon codes.
- Enable sorting of the various tables and their columns.
- Specification tables for products, which could be viewed in a comparison system.
- Internationalisation.