Machine Project: Grocerific Online

Case Study

In this project, we will pretend that we are being hired in as consultants for a company called Grocerific, Inc. Grocerific wishes to put up an online grocery store, where customers can buy groceries from the convenience of their own electronic devices.

You will use HTML, CSS, and asynchronous Javascript/jQuery to create a simple inventory management system for Grocerific, that performs CRUD (Create, Read, and Update, Delete) operations against a a database. The application will have the following functionality:

- Present a list of all products in the main page
- Allow for adding new products
- Allow for deleting products
- Allow for updating products

Specifications:

- Grocerific wants us to create a responsive application for them, so that users may enjoy using the same app, whether on a desktop or mobile device.

  We decide that one of the easiest ways to achieve this is by using the Bootstrap framework, which should take care of all the underlying logic that will allows a site to adapt to multiple screen sizes. This drastically improves development time.

- In addition, we are to come up with a single page application (SPA), so that the user stays *within a web single page*, leading to better user experience. To accomplish this will be combining Bootstrap concepts with jQuery. jQuery will be used to make asynchronous calls to the PHP server-side scripts (mentioned in the earlier specification) and make the necessary changes to the same page, upon server response.

Configuration and Testing

You won't need to implement backend functionality yourself, as all the server-side scripts have been implemented for you (in the form of PHP scripts). Therefore, you will only need to focus on client-side functionality.

PART 1: Set-up the backend environment:

1. Download then install WAMP Server into your computer.
2. During installation, ensure that the root of the server is set to *c:\wamp\www*
3. Copy the grocerific folder onto the document root of the wamp server (*c:\wamp\www*)
4. **Important:** Run the script that will create the MySQL 'grocerific' database complete with sample data. To do this, pull up a command line window, and enter the following commands:

```
c:\> cd c:\wamp\www\grocerific
c:\> c:\wamp\mysql\bin\mysql -u root -p < grocerific.sql
```

   Note: You will be asked for a password. If you have not supplied one during installation, the password should be empty (press Enter when prompted).

   If in case you've supplied a custom password to mysql, you will also need to open *c:\153\grocerific\products\include.inc* with a text editor and modify the `$password` variable there.

5. If set up correctly, you should be now be able to access http://localhost/products/test.php from a web browser without any errors. If you are using a port number other than port 80, do http://localhost:<portnumber>/products/test.php instead.

PART 2: Use JavaScript and jQuery to access the server-side scripts (your instructor should point out to you where to download these scripts). The PHP scripts:

- Perform manipulations and queries against the database;
- Return results in JSON format; and
- After accomplishing part 1—should now be accessible through a URL of the form http://localhost/products/<*scriptname*>.php.

Refer to the table that follows for script usage:

Script Name	Purpose	Parameters	Sample URL to Access Script
products/all.php	Gets all products. Returns product list as JSON array.	no parameters	http://localhost /products/all.php

products/product.php	Gets a single product, given an id. Returns an empty JSON array when product is not found	`id`	http://localhost /products/product.php?id=27
products/new.php	Adds a new product. The id is automatically generated by the database. Returns `success` 1 when successful, and 0 if product is not created. Also returns the id of newly-created product	`description, size, price, aisle_id`	http://localhost /products/new.php?description=TEST&size=111mL&price=11&aisle_id=5
products/delete.php	Deletes an existing product by id. Returns the number of affected rows. Returns `success` 1 when successful, and 0 if product is not deleted/non-existent.	`id`	http://localhost /products/delete.php?id=123
products/update.php	Updates an existing product, based on id. Returns the number of affected rows. Returns number of rows affected as `rows_affected`	`id, description, size, price, aisle_id`	http://localhost /products/update.php?id=123&description=UPDATED&size=222mL&price=22&aisle_id=6
aisles/aisles.php	Updates an existing product, based on id. Returns the number of affected rows. Returns number of rows affected as `rows_affected`	`id, description, size, price, aisle_id`	http://localhost /products/update.php?id=123&description=UPDATED&size=222mL&price=22&aisle_id=6

Hint: You can try out the sample URLs from the table above in a web browser to preview the results in JSON. This is also useful for debugging.

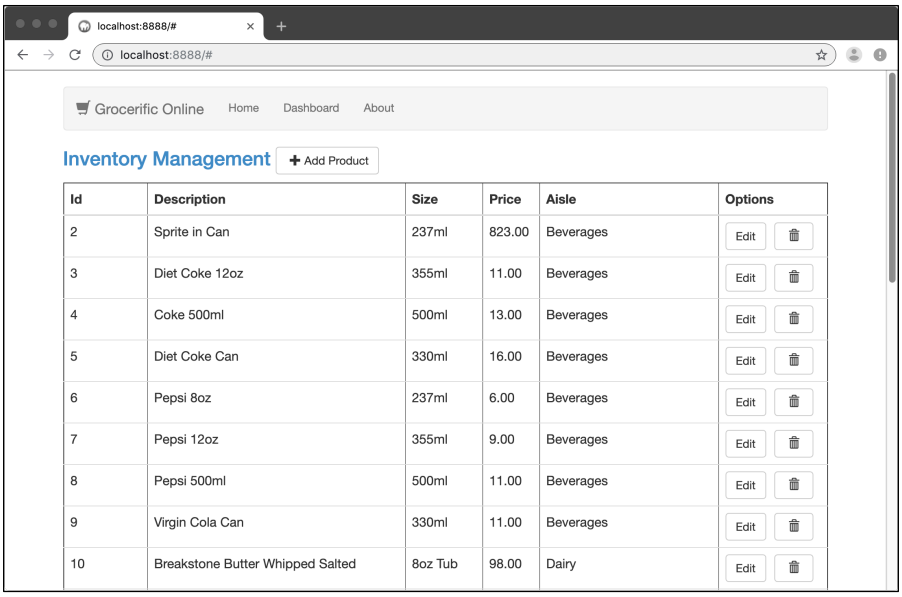
Finally, remember that you can always refresh the database by performing step 3 in part 1, if necessary.

Sample Output

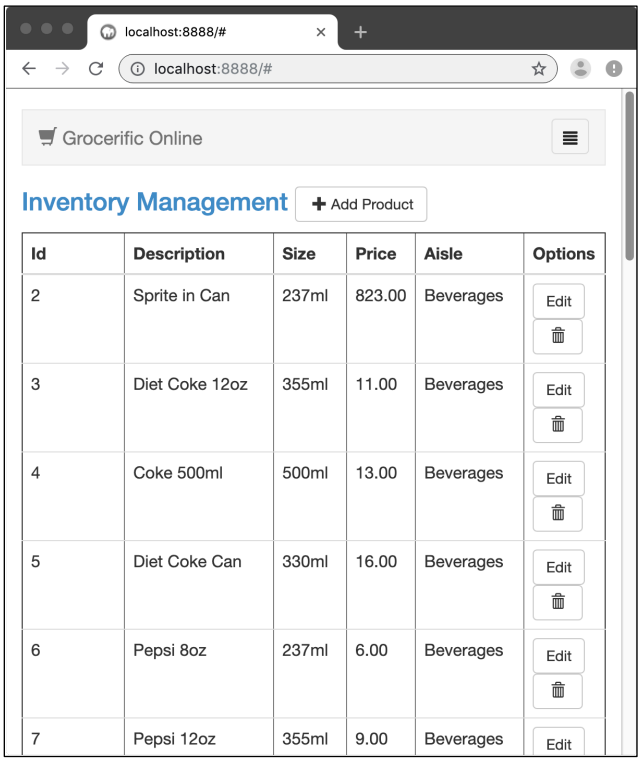
Screenshots showing parts of a sample Grocerific app follows in the next few pages. Feel free to introduce customizations to the user interface (using CSS, for example), but make sure to satisfy all functionality requirements.

1. Product Display

Main Inventory Screen:

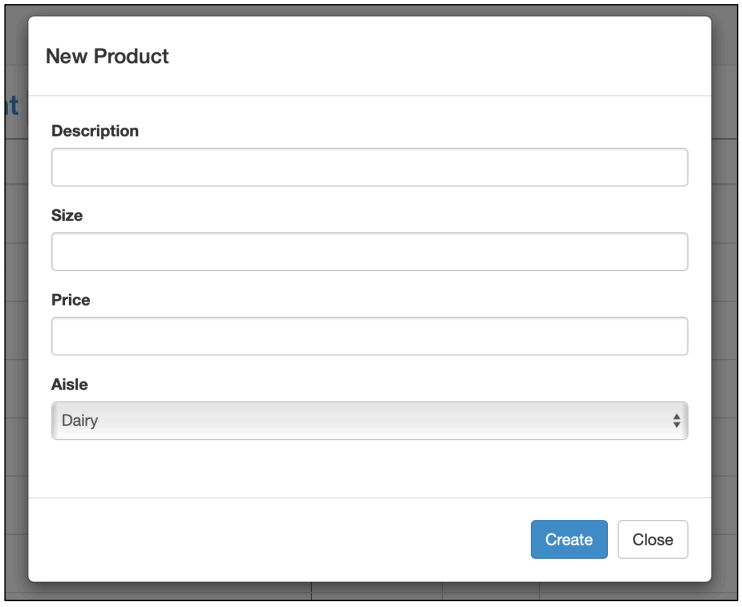


Narrow Width (Responsiveness):

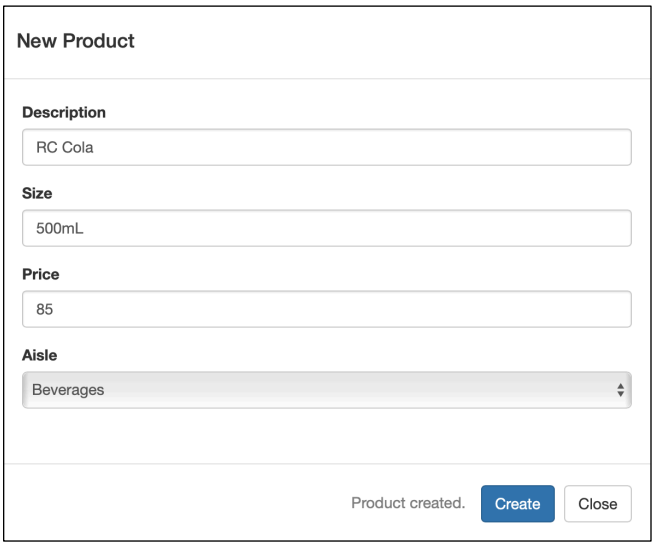


2. Product Creation

Product Creation Form:

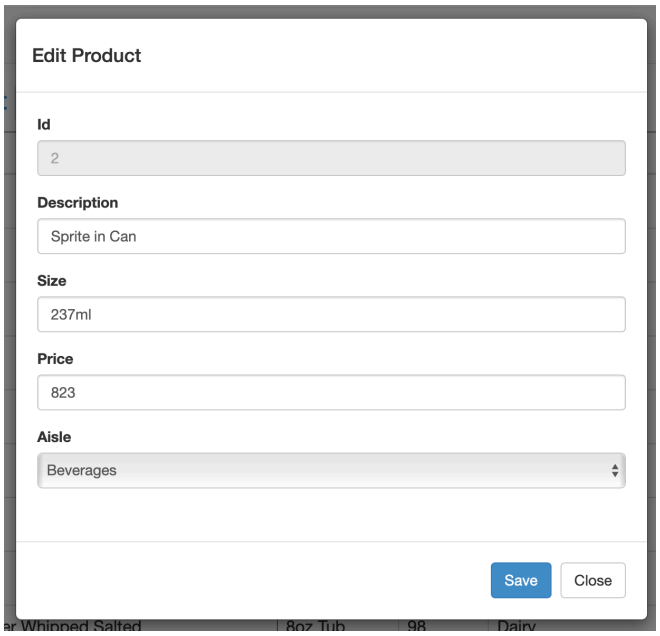


Product Creation Success

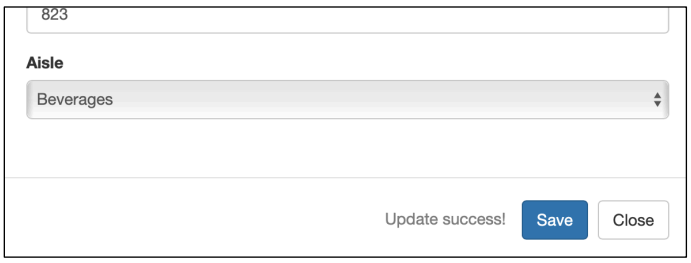


3. Product Update

Product Update Form:



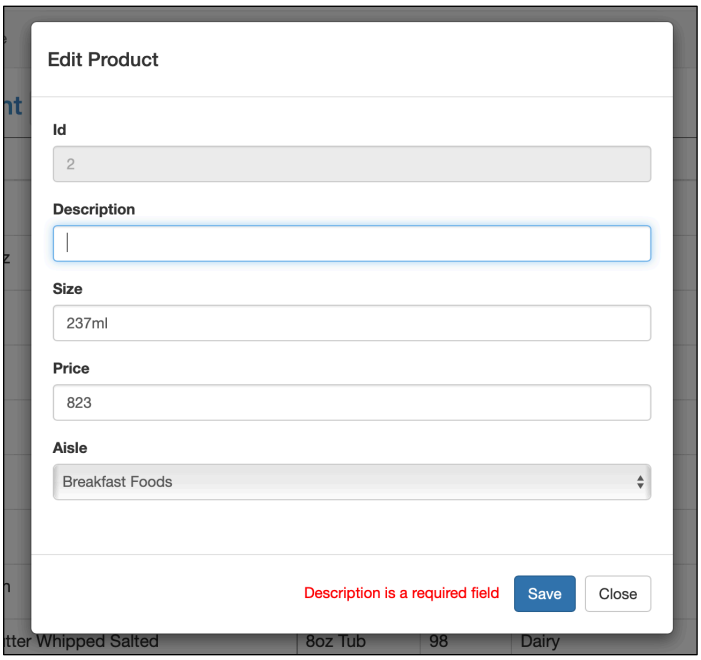
Product Edit Form(On Success)

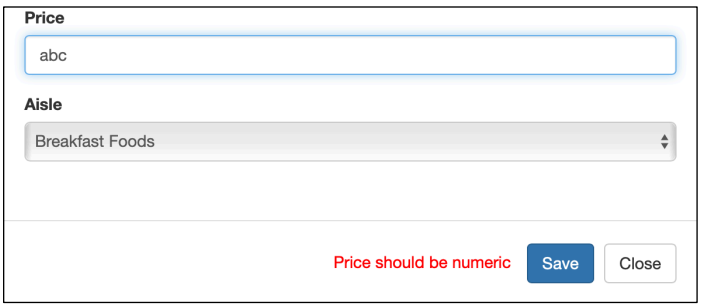


4. Product Deletion and Confirmation

5. Validation

For *both* **Edit Product** and **New Product** forms implement validation for required fields (id, description, size, and price) when Save and Create buttons are clicked. Also validate price to check if the input text is numeric. When a field value fails to validate, the focus should move to that field.




Price
abc
Aisle
Breakfast Foods
Price should be numeric
Save
Close