iBar – Barcode Generator

# Introduction

A large supermarket require a UPC-A bar code generator to be developed so that it is easy for staff within the supermarket be able to print barcodes that they require instead of awaiting for them to arrive in the post.

# Basic Instructions

Please load the system using microapache with the following extensions:

SQLite – with PDO extensions is required for elements to communicate on the logic & data tier.

PHP GD – GD is the extension that provides the ability to generate the barcodes as an image.

# Functional Requirements

1. Provide a usable web enabled GUI to support all other functional requirements;

I designed the GUI with simplicity in mind to keep the website simple and really easy to use. I have used JavaScript, CSS and Ajax to assist me in flow & design of the website to give the user the best possible experience.

1. Display a complete listing of all products and associated bar codes (see Appendix);

The barcodes can be found under the Show All button on the website once the user has logged in. The images can be clicked to display a printer friendly page.

1. List a single product along with its barcode ;

The user can select a product by pressing the button “Select A Product” button once the user has logged on, then they can select the product they wish to view from the dropdown list.

1. Generate page(s) of a single product barcode suitable for printing and product mark-up;

The user can access the printer friendly page by clicking the barcode they wish to print. This page has Back buttons and print buttons which are hidden from the printer using CSS print media queries.

1. Allow for the use of pagination in an appropriate context;

Pagination has been implemented at the data tier where the data is limited by a variable to control the amount of products to display per page. This can be increased or decreased.

1. A search facility to find product and/or barcodes;

A search box has been provided in the header of the website where they can search the products in the same page as other product related functional requirements.

1. Login system to support staff of the supermarket with user levels guest & admin; admin should be able to add new staff accounts.

A login system has been provided with appropriate security to protect the system from unauthorised users and SQL injection.

# Non-Functional Requirements

# Presentation tier

The techniques I used at the presentation tier is elements from the following languages HTML, CSS, JS and Ajax to create a web 2.0 interface and seamless interaction for the user.

The HTML code was pretty simple but it incorporated elements from Javascript that work as event listeners to see when the form is submitted or changed. With these elements I was able to write client side functions that worked with ajax to post the search or selection to the server without reloading the page. This can be demonstrated with the following found in my source code:

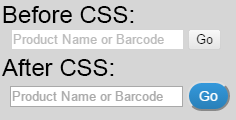
<form name="selectBarcodeForm" method="post">  
 <select name="selectBarcode" id="selectBarcode" onchange="getProduct(**this**.options[**this**.selectedIndex].value, **this**.options[**this**.selectedIndex].text);">  
  
 <option value="null" selected="selected">Please Select A Value</option>  
 **<?php** $product->selectProductsOutput();  
  
 **?>** </select>  
</form>

As you can see in Appendix A this HTML form that “onchange” it called the getProduct function with the HTML value and its title (Value = Barcode, Text = Product Name).

**function** getProduct(code, name) {  
 **if** (code!= "null") {  
 xmlHttp= **new** XMLHttpRequest();  
 **if** (xmlHttp==**null**) {  
 alert ("Browser does not support HTTP Request");  
 **return**;  
 }  
 **var** url="barcodeHandler.php";  
 xmlHttp.onreadystatechange=stateChangedSelected;  
 xmlHttp.open("POST",url,**true**);  
 xmlHttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  
 xmlHttp.send("barCode=" + code + "&name=" + name);  
 }  
}  
  
**function** stateChangedSelected() {  
 **if** (xmlHttp.readyState==4) {  
 document.getElementById('selectProductResult').innerHTML=xmlHttp.response;  
 search = **true**;  
 }  
}

Appendix B shows how the Ajax checks if the value has been passed correctly and then continues to post the values through a Post function to the barcodeHandler.php. This is all done at the presentation layer so the client’s browser does not have to refresh which reduces load time and server load.

CSS is used to enhance the users experience by making the interaction with the system much more aesthetically pleasing and is quite a simply language that modifies elements to behave in a way the developer wishes. I have created a simple diagram to demonstrate why we implement CSS and its functionality. Appendix C – CSS Comparison.



# Logic tier

The techniques I used in the logic tier was to create the system using Object Orientated Programming (OOP), Image Caching and Sessions (object Serialisation). I felt focusing on these areas would create a better logic tier that had the user and development of the system in mind.

The object orientation was a pretty strong must have because this type of system has loads of room for future developments & improvements to specific areas with additional functionality. With this in mind I have created most of the core functionality on the website as classes such as Users, Admins, Guests, Products and Searches. This allowed me to create an effective development environment that was easy to modify, extend and implement. The reason I have created users separate from admins and guests is because admins and guests extend the user class which inherits the functionality of the main class user. This allows the levels of the user accounts to have different functionality so an admin can have the ability to add new employees to the website. It also had a positive effect on the look and feel of the website for example in appendix A you can see how clean the code is where I am calling a function of the product object.

<form name="selectBarcodeForm" method="post">  
 <select name="selectBarcode" id="selectBarcode" onchange="getProduct(**this**.options[**this**.selectedIndex].value, **this**.options[**this**.selectedIndex].text);">  
  
 <option value="null" selected="selected">Please Select A Value</option>  
 **<?php** $product->selectProductsOutput();  
  
 **?>** </select>  
</form>

My image caching implementation is very basic and it was because I did not think of all the elements of caching. I focused on the development of getting the images cached and checking if they are not already created and did not think about the end user wanting to update the barcode. The image caching has been implemented rather easily within the GenerateBarcode.php file, Appendix D

//check if image exists  
$fileName = "cache/" . $barCode . ".png";  
  
**if** (!file\_exists($fileName)) {  
 //Save image for cache  
 header('Content-Type: image/png');  
 ImagePNG($im, $fileName);  
 //produce out to the screen  
 ImagePNG($im);  
}

File\_exists checks if the file does not exist and if it does exist already it will load the image from cache and if there is not file in the cache folder then it generates the image and also generates the barcode back to the user on the fly. This is so that it does not have to generate the barcode then save the image then reload the image which I thought is a really effective way of developing to minimize server load.

The objects I have implemented are in sessions which allows me to use the information throughout the website where ever I need to call upon the session information. Serialization allows for the development of inflation and deflation of the session so there could be a possibility of improving this basic implementation to allow the user to store the session just in case they are disconnected from the website. I have not serialized every element within the session as it was not necessary for the project but I have implemented the Sessions as objects. Appendix D & E demonstrate my basic use of serialization to show that I understand the conversion of storing the data as an object instead of a session so it can be reused at a later date calling upon the object information.

**Appendix D**

**function** \_\_construct($dbUsers)  
{  
 $this->db = $dbUsers;  
 $this->id = $\_SESSION['userSession'];  
 $this->name = serialize($\_SESSION['username']);  
 $this->level = $\_SESSION['userLevel'];  
}

**Appendix E**

**public function** welcomeMessage() {  
 $message = "Welcome Admin " . unserialize($this->name) . ", to print a barcode just click it or click admin at the top to add a new user.";  
 **print** $message;  
}

I also implemented into my system handlers which assisted with the handling between the presentation layer and the object orientated logic layer. For example as you can see in Appendix F the barcodeHandler.php file allows me to communicate with the data posted from the ajax command and create a new instance of the product to then call a method in product to check the if the products in the cache.

**<?php  
include\_once**('products.php');  
  
**if**(**isset**($\_POST['barCode'])){  
 $barCode = $\_POST['barCode'];  
 $name = $\_POST['name'];  
  
 $product = **new** Product($dbProducts);  
 $result = "<a href='printFriendly.php?code=" . $barCode . "&name=" . $name . "'><img src='" . $product->cacheCheck($barCode) . "' alt='" . $name . "' /></a>";  
 **print** $result;  
}  
**?>**

# Data Tier

I have created most of the data tier in a separate PHP file called db.php to handle the creation of the tables in the databases. There is two separate databases one for staff and one for products. The reason I have kept these databases separate is so that the system can be incorporated with an existing database structure of employees.

The commented out section in (Appendix G – SQLite Tables) is used to create the database files as a onetime run only use.

My queries are run throughout the document using PDO so I can access the database as an object throughout the solution. Having an object based interface makes object orientation programming run a lot smoother. For example I am able to call my queries in the user.php file that can query directly to the object. Appendix H – Login Function & Query

**public function** login($uName, $uPassword)  
{  
 **try** {  
 $stmt = $this->db->prepare("SELECT \* FROM staff WHERE username='" . $uName . "'");  
 $stmt->execute(**array**('username'=>$uName));  
 $userRow = $stmt->fetch(PDO::*FETCH\_ASSOC*);  
 **if**($userRow > 0)  
 {  
 **if**(md5($uPassword) == $userRow['password'])  
 {  
 $\_SESSION['userSession'] = $userRow['userId'];  
 $\_SESSION['username'] = $userRow['username'];  
 $\_SESSION['userLevel'] = $userRow['userLevel'];  
 **return true**;  
 }  
 **else** {  
 **return false**;  
 }  
 }  
 }  
 **catch**(PDOException $e)  
 {  
 **echo** $e->getMessage();  
 }  
}

Using PDO allows me to use the functions that are included such as prepare which allows me to prepare the SQL statements to eliminate the chance of a SQL Injection.

# Future Developments

The areas I feel like that can be improved as future developments would be a CRUD extension to allow the admins to modify the products database. This would be created through an additional page that manages the products.

The employee’s implementation I have entered with the login system is basic at the moment but can be extended quite easily to support a CRUD functionality.

My image caching could be improved by taking into consideration the CRUD of products being implemented. For example if the CRUD for products is added then image caching will need to be improved to check if the product has been updated to remove the cached image.

Serialization could be improved in the future to save the user sessions in an inflation and deflation method to keep the sessions even when the user is disconnected from the website so on return they can continue their session.

# Bibliography

PHP.net. (2016). *Serialize.* Available: http://php.net/manual/en/function.serialize.php. Last accessed 25th Feb 2016.

PHP.net. (2016). *Object Serialization.* Available: http://php.net/manual/en/language.oop5.serialization.php. Last accessed 25th Feb 2016.

W3Schools.com. (2016). *AJAX - Send a Request To a Server.* Available: http://www.w3schools.com/ajax/ajax\_xmlhttprequest\_send.asp. Last accessed 25th Feb 2016.

wiki.hashphp.org. (2016). *PDO Tutorial for MySQL Developers.*Available: http://wiki.hashphp.org/PDO\_Tutorial\_for\_MySQL\_Developers. Last accessed 25th Feb 2016.

# Appendix

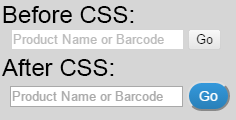
Appendices A – HTML Form with Javascript Event & Function Call – dashboard.php

<form name="selectBarcodeForm" method="post">  
 <select name="selectBarcode" id="selectBarcode" onchange="getProduct(**this**.options[**this**.selectedIndex].value, **this**.options[**this**.selectedIndex].text);">  
  
 <option value="null" selected="selected">Please Select A Value</option>  
 **<?php** $product->selectProductsOutput();  
  
 **?>** </select>  
</form>

Appendices B – Javascript/Ajax Function to post form without updateing client side – resources/js/interface.js

**function** getProduct(code, name) {  
 **if** (code!= "null") {  
 xmlHttp= **new** XMLHttpRequest();  
 **if** (xmlHttp==**null**) {  
 alert ("Browser does not support HTTP Request");  
 **return**;  
 }  
 **var** url="barcodeHandler.php";  
 xmlHttp.onreadystatechange=stateChangedSelected;  
 xmlHttp.open("POST",url,**true**);  
 xmlHttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded");  
 xmlHttp.send("barCode=" + code + "&name=" + name);  
 }  
}  
  
**function** stateChangedSelected() {  
 **if** (xmlHttp.readyState==4) {  
 document.getElementById('selectProductResult').innerHTML=xmlHttp.response;  
 search = **true**;  
 }  
}

Appendices C – CSS Comparison



Appendices D – Serialize – userAdmin.php

**function** \_\_construct($dbUsers)  
{  
 $this->db = $dbUsers;  
 $this->id = $\_SESSION['userSession'];  
 $this->name = serialize($\_SESSION['username']);  
 $this->level = $\_SESSION['userLevel'];  
}

Appendices E – Unserialize – userAdmin.php

**public function** welcomeMessage() {  
 $message = "Welcome Admin " . unserialize($this->name) . ", to print a barcode just click it or click admin at the top to add a new user.";  
 **print** $message;  
}

Appendices F – Handler Functionality – barcodeHandler.php

**<?php  
include\_once**('products.php');  
  
**if**(**isset**($\_POST['barCode'])){  
 $barCode = $\_POST['barCode'];  
 $name = $\_POST['name'];  
  
 $product = **new** Product($dbProducts);  
 $result = "<a href='printFriendly.php?code=" . $barCode . "&name=" . $name . "'><img src='" . $product->cacheCheck($barCode) . "' alt='" . $name . "' /></a>";  
 **print** $result;  
}  
**?>**

Appendices G – SQLite Tables - resources/php/db.php

**<?php**//For Creation of the Databases & Testing.  
//$dbProducts=sqlite\_open("products.db");  
//$dbUsers = sqlite\_open("staff.db");  
  
//Staff Table Creation & Populate  
//@sqlite\_query($dbUsers,"DROP TABLE staff");  
//@sqlite\_query($dbUsers,"CREATE TABLE staff (userId int(11) NOT NULL PRIMARY KEY, username varchar(15) NOT NULL, password varchar(20) NOT NULL, userLevel varchar(5) NOT NULL, UNIQUE (username))",$sqliteerror);  
//@sqlite\_query($dbUsers, "INSERT INTO staff VALUES ('1','admin','246e6a92707628bfb8df2d392c4df7b9','admin')",$sqliteerror);  
//@sqlite\_query($dbUsers, "INSERT INTO staff VALUES ('2','demouser','75580656a394292460ebb4b036ebeaf1','guest')",$sqliteerror);  
  
//Products Table Creation & Populate  
//@sqlite\_query($dbProducts,"DROP TABLE products");  
//@sqlite\_query($dbProducts,"CREATE TABLE products (name varchar(30),code varchar(12) PRIMARY KEY)",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Apples'," . checkSum('01234567890') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Oranges'," . checkSum('01234509876') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Lemons'," . checkSum('05432167890') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Bananas'," . checkSum('09456881235') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Lettuce'," . checkSum('07894563841') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Beetroot'," . checkSum('01123487645') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Carrots'," . checkSum('02234994875') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Controlled drug 1'," . checkSum('34859603004') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Box of Tissues'," . checkSum('04546938969') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Kitchen Rolls'," . checkSum('04446573995') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Toilet Rolls'," . checkSum('09866958348') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Disposable Napkins'," . checkSum('01263553425') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Tea Towels'," . checkSum('04628374758') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Washing-up Liquid'," . checkSum('06859939577') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Soap Powder'," . checkSum('08839577381') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Controlled drug 2'," . checkSum('30548999604') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Spring Water'," . checkSum('01177226637') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Champagne'," . checkSum('05969483958') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Still Water'," . checkSum('01928337773') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Orange Juice'," . checkSum('05843637378') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Tomato Juice'," . checkSum('09999948377') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Can of Beer'," . checkSum('03677748388') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Controlled drug 3'," . checkSum('36888493904') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Cheese'," . checkSum('03828475734') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Milk'," . checkSum('01727172717') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Yogurt'," . checkSum('02838264634') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Coffee'," . checkSum('07578486889') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Tea'," . checkSum('01525465634') . ")",$sqliteerror);  
//sqlite\_query($dbProducts, "INSERT INTO products VALUES ('Bread'," . checkSum('01111115253') . ")",$sqliteerror);  
  
//Close Database  
//$dbUsers = null;  
//$dbProducts = null;  
  
session\_start();  
  
**include\_once**('user.php');  
**include\_once**('products.php');  
  
**try** {  
 $dbProducts = **new** PDO('sqlite2:products.db');  
 $dbProducts->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_EXCEPTION*);  
 $dbUsers = **new** PDO('sqlite2:staff.db');  
 $dbUsers->setAttribute(PDO::*ATTR\_ERRMODE*, PDO::*ERRMODE\_EXCEPTION*);  
  
 $user = **new** user($dbUsers);  
 $product = **new** product($dbProducts);  
  
 **if**($user->checkLevel()) {  
 $admin = **new** admin($dbUsers);  
 } **else** {  
 $guest = **new** guest($dbUsers);  
 }  
  
} **catch**(PDOException $e) {  
 **echo** 'Connection failed: ' . $e->getMessage();  
}  
  
**function** checkSum($code)  
{  
 $barCode = $code;  
 $explode = str\_split($barCode);  
  
 //check Sum  
 $odd = 0;  
 $even = 0;  
  
 //1. Add together all the odd and even numbers  
 **for** ($i = 0; $i < count($explode); $i++) {  
 **if** ($i & 1) {  
 //odd  
 //echo "odd" . $explode[$i] . "<br />";  
 $odd += $explode[$i];  
 } **else** {  
 //echo "even" . $explode[$i] . "<br />";  
 $even += $explode[$i];  
 }  
 }  
  
 //2. multiply by three  
 $result = ($even \* 3) + $odd;  
  
 //3. Modulo 10 and subtract from 10  
 $result = 10 - ($result % 10);  
  
 **if** ($result == 10) {  
 $result = 0;  
 }  
 **return** $barCode .= $result;  
}  
  
**?>**

Appendices H – Login Function & Query – user.php

**public function** login($uName, $uPassword)  
{  
 **try** {  
 $stmt = $this->db->prepare("SELECT \* FROM staff WHERE username='" . $uName . "'");  
 $stmt->execute(**array**('username'=>$uName));  
 $userRow = $stmt->fetch(PDO::*FETCH\_ASSOC*);  
 **if**($userRow > 0)  
 {  
 **if**(md5($uPassword) == $userRow['password'])  
 {  
 $\_SESSION['userSession'] = $userRow['userId'];  
 $\_SESSION['username'] = $userRow['username'];  
 $\_SESSION['userLevel'] = $userRow['userLevel'];  
 **return true**;  
 }  
 **else** {  
 **return false**;  
 }  
 }  
 }  
 **catch**(PDOException $e)  
 {  
 **echo** $e->getMessage();  
 }  
}