Hardware store

2

Scenario

The manager of a hardware store requires a web site to advertise products and services on-line, in order to attract customers to the store. The business stocks a wide range of tools and appliances for use in the home and garden, and also provides a service to design and install bathrooms and kitchens in customers' homes. There is no requirement at the present time to sell products or services directly from the web site.

Design

An outline flowchart for the project is shown below. The website will be constructed in two sections: one section will be freely available for viewing by the public, whilst the other section will be password-protected and available only to staff of the hardware store.

The public website will display a set of pages giving general information about the store and the products and services provided. A catalogue page will then display details of products available in the store, including description, photograph and price. The catalogue will obtain its content by accessing an on-line database. Customers may search by category of product, such as: power tools or bathroom fittings.

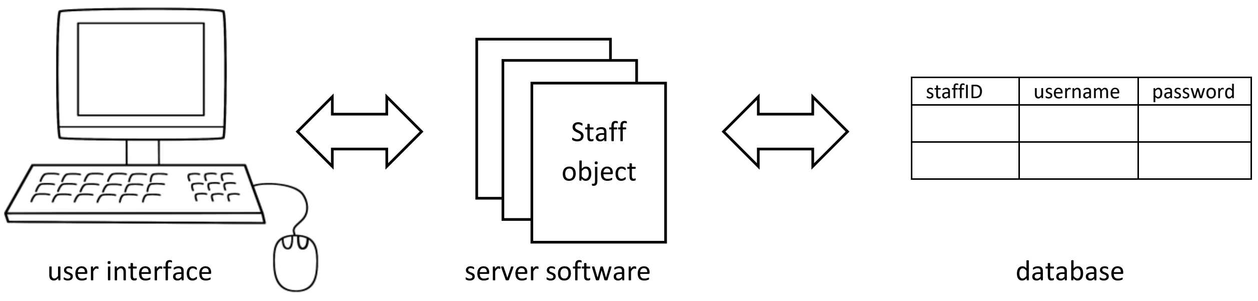
The staff section will be accessed through a log-in page. User name and password will be verified with the on-line database. Staff may view product records, edit or delete current products or add new products as required.

Programming

Basic web pages will be produced with HTML code, with CSS styles used to improve the formatting and appearance of the pages.

Each page will display a standard header and menu system. To avoid unnecessary duplication, these components will be stored as a CSS style sheet and a PHP code file which can be accessed by each separate page as necessary.

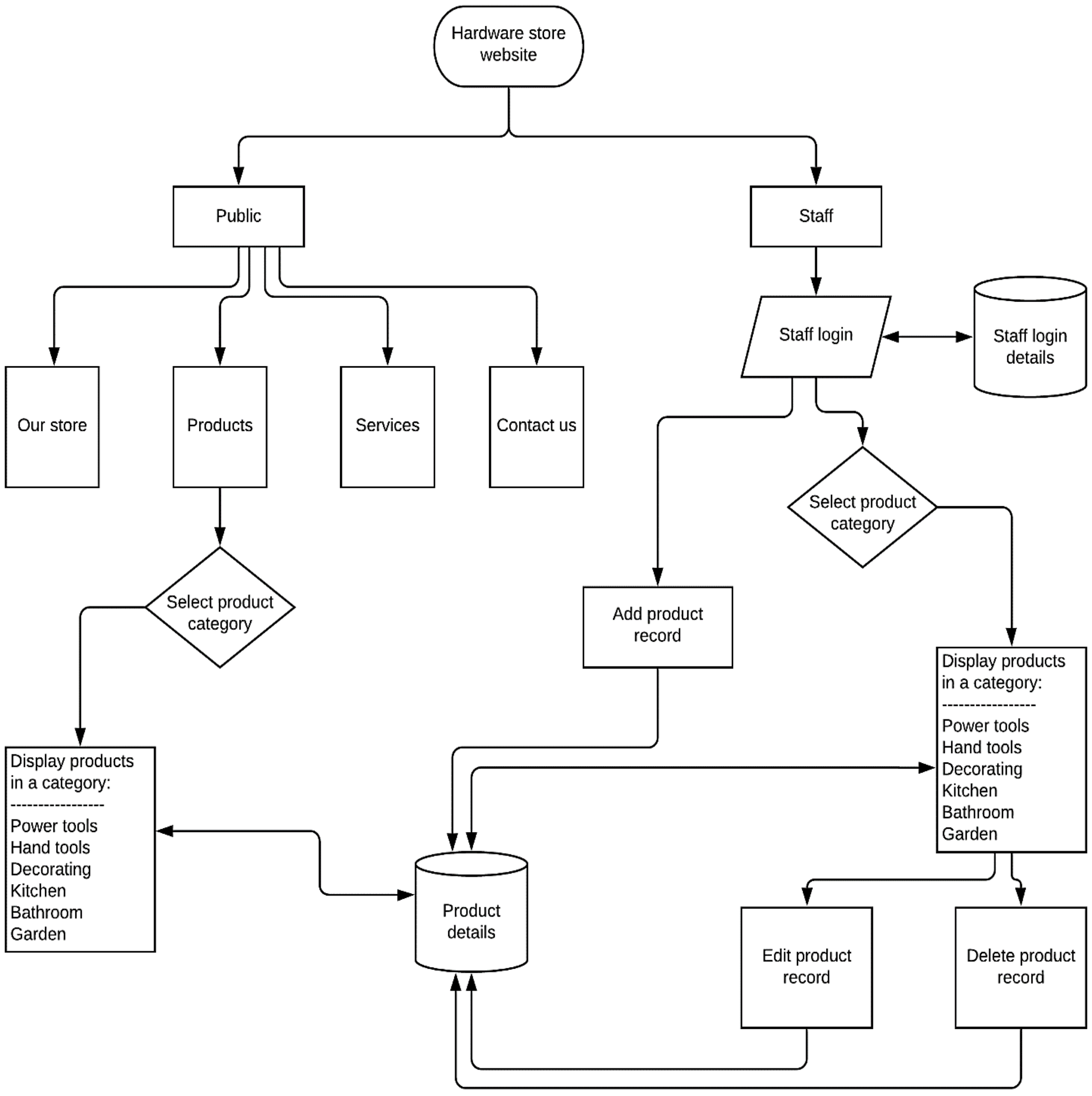
We will use an object oriented approach in PHP for handling staff log-in and product records. Records will be transferred to the server memory as sets of objects:



The **staff** class will contain a function to check the input user name and password, returning a boolean value to indicate whether a valid log-in has been made.

The **product** class will contain functions to carry out each of the database operations: creating new records, and retrieving, updating and deleting existing records.

The classes can be stored in PHP code files, and their functions accessed by any web page as necessary. This approach again reduces the amount of code duplication, and can increase the program reliability by allowing the objects to be thoroughly tested before inclusion in the web site.



Method

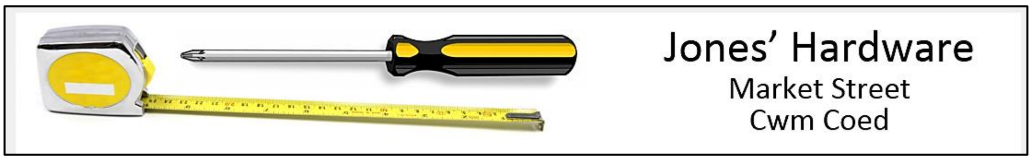
Start by creating a folder to store files for the project; this may be called **hardware**. Open a blank text document and save this into the folder as **index.php**. Add the lines of code shown below.

```
     <html>
     <head>
        <title> Hardware store </title>
        <link rel="Stylesheet" type="text/css" href="styleSheet.css" />
     </head>

     <body>
        <?
            include('menu.php');
        ?>
    </body>
    </html>
```

The program includes links to a style sheet and a separate file which will contain a menu system.

Re-save the **index.php** file. Create a **hardware** folder on the server and upload the file into this folder. All files for the project will be placed within this folder.

Use a desktop publishing or graphics application such as Microsoft Word or Photoshop to produce a title bar for the head of each page. This should be approximately 1100 pixels wide by 140 pixels high. Transfer this to the server as **header.jpg**



Begin a style sheet by creating an empty text file in a text editor and adding the code:

```
body
{
        background-color: #F0F0F0;
        font-family: Arial, Helvetica, sans-serif;
        color: black;
}

#header
{
        height: 155px;
        background-image : url(header.jpg);
        background-repeat: no-repeat;
        background-color: #FFFFFF;
}
```

Save this file into the **hardware** folder with the name **styleSheet.css**, then copy it to the server.

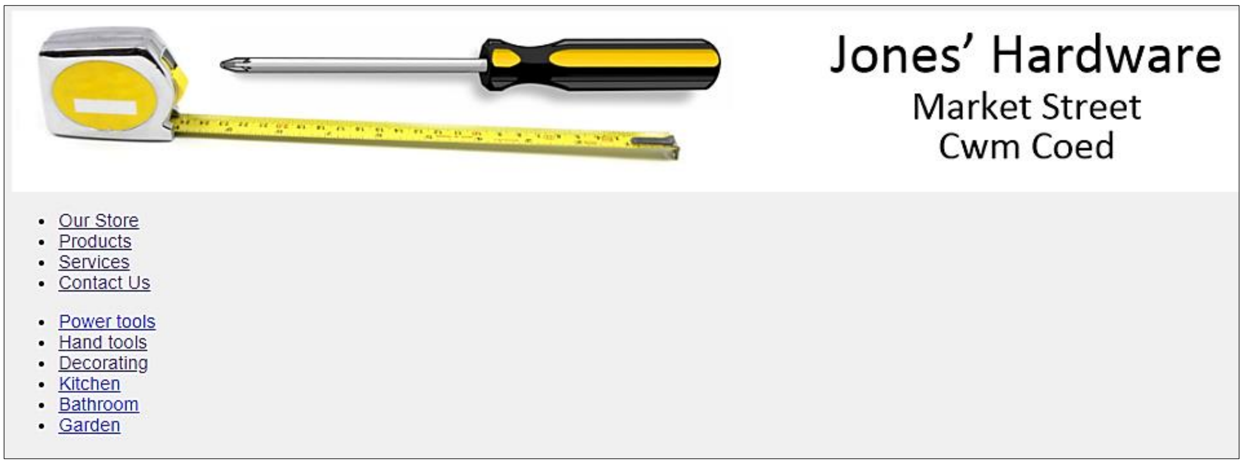
We need one final file before testing the page. Create a new text file and add the code below to set up the menus. Save this file as **menu.php** and copy it to the server.

```
<?

  echo"<div id='header'>";
  echo"</div>";

  echo"<div id='navigation'>";
  echo"<ul id='nav'>";
     echo"<li><a href='index.php'>Our Store</a></li>";
     echo"<li><a href='products.php'>Products</a></li>";
     echo"<li><a href='services.php'>Services</a></li>";
     echo"<li><a href='contactUs.php'>Contact Us</a></li>";
  echo"</ul>";
  echo"</div>";

  echo"<div id='products'>";
  echo"<ul id='cat'>";
     echo"<li><a href='displayItems.php?category=power'>Power tools</a></li>";
     echo"<li><a href='displayItems.php?category=hand'>Hand tools</a></li>";
     echo"<li><a href='displayItems.php?category=decorating'>Decorating</a></li>";
     echo"<li><a href='displayItems.php?category=kitchen'>Kitchen</a></li>";
     echo"<li><a href='displayItems.php?category=bathroom'>Bathroom</a></li>";
     echo"<li><a href='displayItems.php?category=garden'>Garden</a></li>";
  echo"</ul>";
  echo"</div>";
 ?>
```

Test the **index.php** web page in a browser. To do this, enter the domain name for your site, followed by the directory name **hardware**, e.g:

www.website.com/hardware

The page **index.php** will be load automatically as the default homepage for the site. The header and two sets of menu items should be displayed.



- Our Store
- Products
- Services
- Contact Us

- Power tools
- Hand tools
- Decorating
- Kitchen
- Bathroom
- Garden

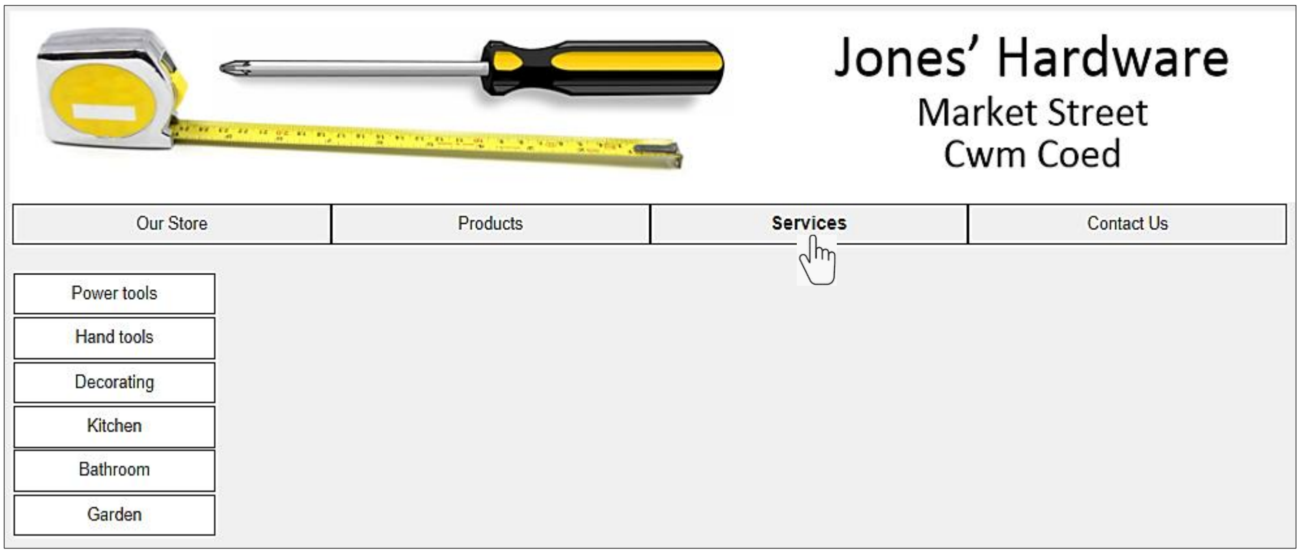
Reopen the style sheet file **styleSheet.css**, then add the additional code shown in the rounded rectangles on the two pages below. Save the amended style sheet file, and copy it to the server.

```
#header
{
        height: 155px;
        background-image : url(Header.png);
        background-repeat: no-repeat;
        background-color: #FFFFFF;
}
#navigation
{
       width: 1150px;
       height: 40px;
}
#nav
{
        list-style: none;
}
#nav ul
{
        list-style: none;
        display: none;
}
#nav li
{
        font-size: 14px;
        float: left;
        position: relative;
        width: 270px;
        height: 32px;
        left: -38px;
        top: -15px;
        border: 1px solid #000000;
}
```

```
#nav a:link, #nav a:active, #nav a:visited
{
       display:block;
       color: #000000;
       text-decoration: none;
       padding: 8px;
       text-align: center;
}
#nav a:hover
{
       font-weight:bold;
       padding: 6px;
       font-size: 16px;
}
#products
{
       background-color: #F0F0F0;
       float: left;
       width: 200px;
       list-style: none;
}
#cat
{
       list-style: none;
}
#cat ul
{
       list-style: none;
       display: none;
}
#cat li
{
       background-color: #FFFFFF;
       margin: 1px;
       font-size: 14px;
       float: left;
       position: relative;
       width: 170px;
       height: 32px;
       left: -38px;
       top: -15px;
       border: 1px solid #000000;
}
#cat a:link, #nav a:active, #nav a:visited
{
       display:block;
       color: #000000;
       text-decoration: none;
       padding: 8px;
       text-align: center;
}
#cat a:hover
{
       font-weight:bold;
       padding: 6px;
       font-size: 16px;
}
```

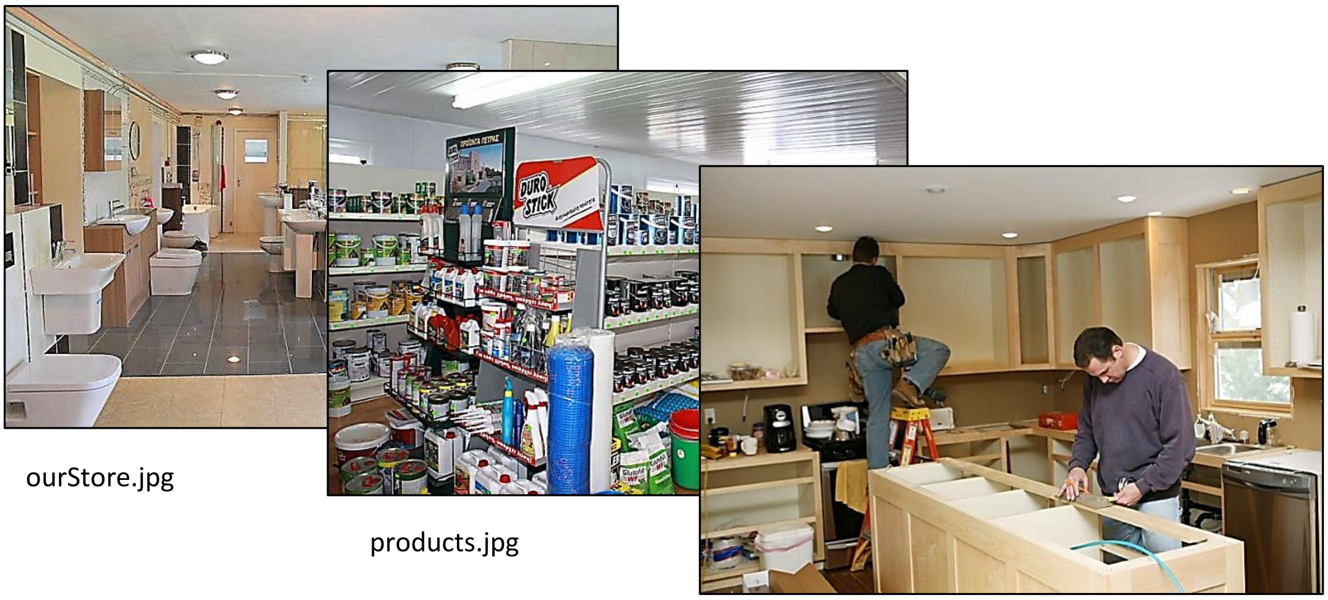
Re-run the **index.php** page by clicking the refresh icon in the browser whilst holding down the CTRL key. This ensures that the style sheet is re-loaded from the server.

The menu items now appear in boxes, and should be highlighted as the mouse pointer passes over them. Notice that the two lists of menu items in the **index.php** file were given the id values '**nav**' and '**cat**' . This allowed the style sheet to apply different formatting styles to the boxes making up the two menu blocks. These are identified in the style sheet by the tags **#nav** and **#cat**.



A template has been created, which can be used to produce each page of the website.

The pages for ***Our Store***, ***Products*** and ***Services*** will display images. Obtain suitable .JPG photographs and upload them to the server. Give the images the names ***ourStore.jpg***, ***products.jpg*** and ***services.jpg***.



ourStore.jpg

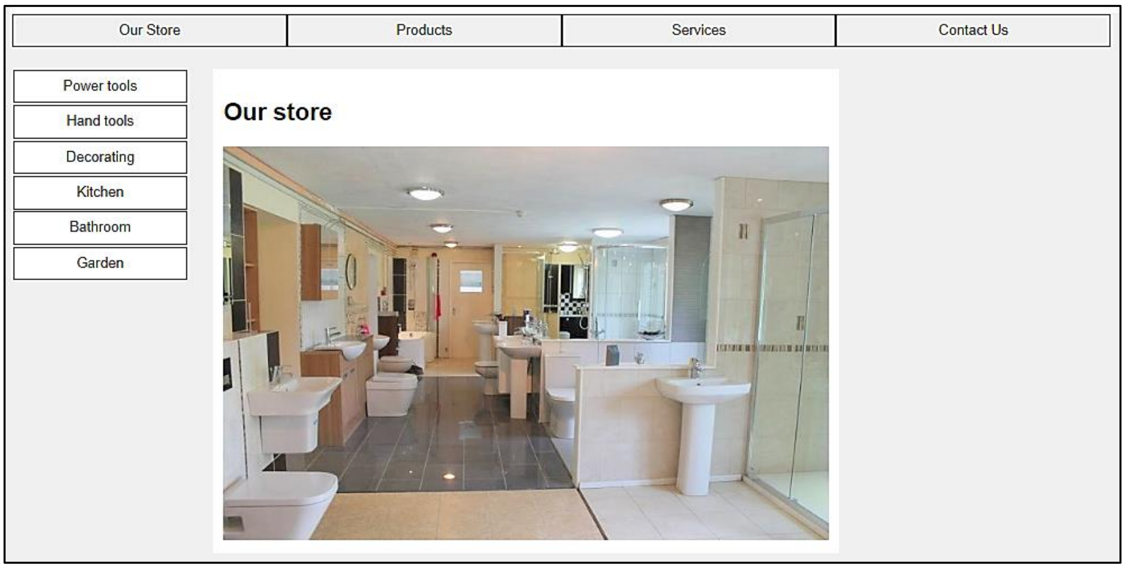
products.jpg

services.jpg

Return to the **index.php** script and add a table to display a heading 'Our store' and the corresponding image, as shown in the program code below.

```
        <body>
         <?
           include('menu.php');
         ?>
          <table cellpadding = 10>
                 <tr><td bgcolor="white">
                    <h2>Our store</h2>
                    <img src="ourStore.jpg" width="600" />
                 </td></tr>
          </table>
      </body>
      </html>
```

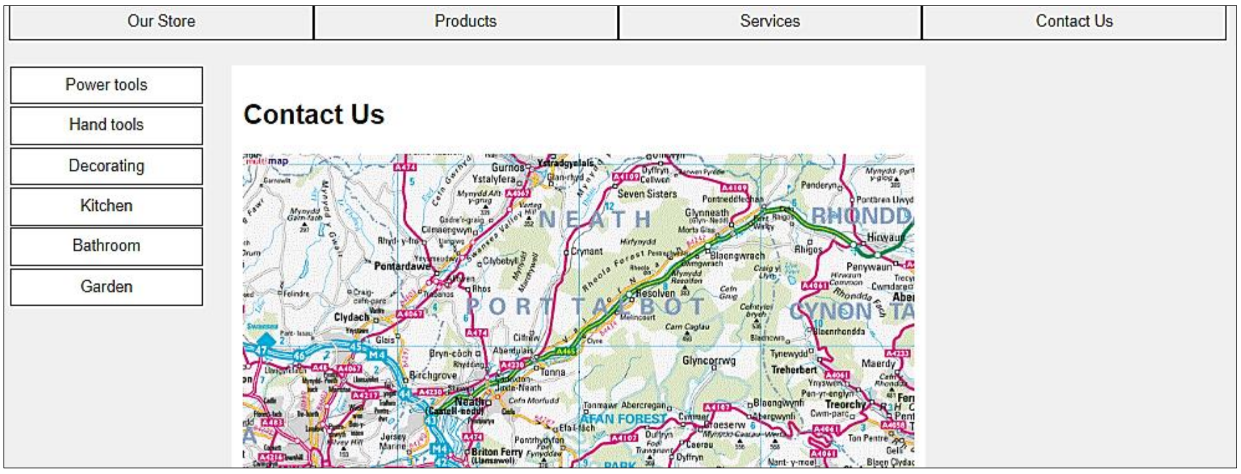
Save the amended file **index.php** and copy it to the server.



Additional text or images could be added to this page as required by the manager of the shop.

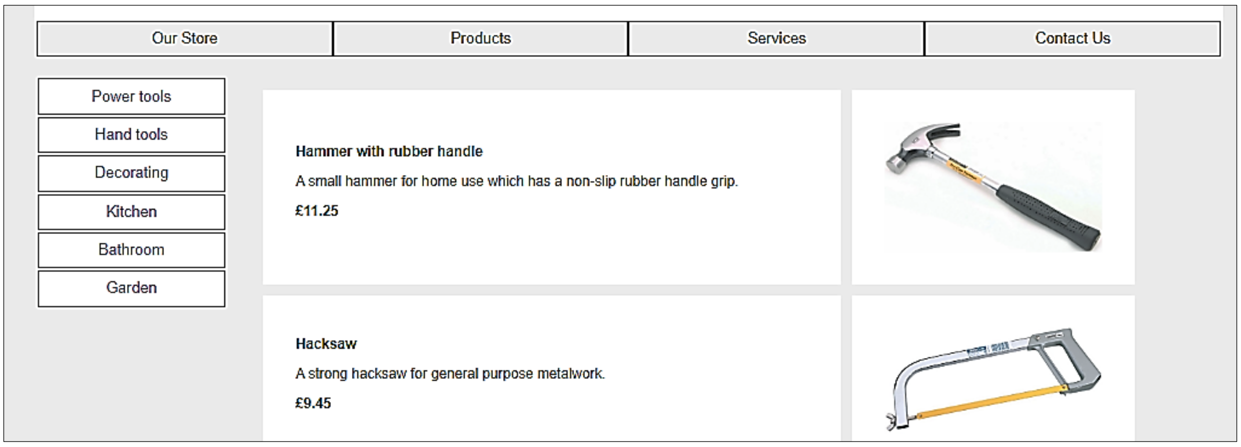
Copy the **index.php** file and save this three times, using the file names **products.php**, **services.php** and **contactUs.php**. Open each file in turn and amend the heading and image name in the table:

- In **products.php**, insert the heading '**Products**' and the image file name '**products.jpg**'.
- In **services.php**, insert the heading '**Services**' and the image file name '**services.jpg**'.
- In **contactUs.php**, insert the heading '**Contact Us**'. A map can be displayed to show the location of the store. Obtain a suitable map image and copy this to the server as **map.jpg.** Insert this file name in the **contactUs.php** file.



Upload the amended **products.php**, **services.php** and **contactUs.php** to the server. Test that the menu system allows the user to move between each of these pages.

We can now turn our attention to the main function of the web site, which is to display information about the hardware items for sale. When the user selects a product category from the left side menu, the available items should be listed, along with picture images:



Hammer with rubber handle

A small hammer for home use which has a non-slip rubber handle grip.

£11.25

Hacksaw

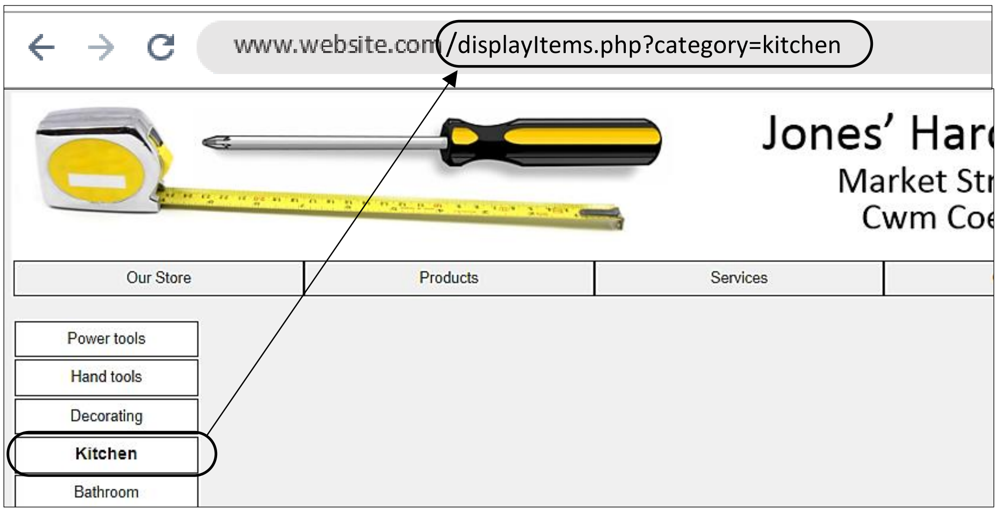
A strong hacksaw for general purpose metalwork.

£9.45

When a category is selected, a new page will open to display the available items. Create this by copying the **index.php** file and saving it with the name **displayItems.php**. Delete the **<table> … </table>** block to leave the code shown below:

```
<html>
 <head>
    <title> Hardware store </title>
    <link rel="Stylesheet" type="text/css" href="styleSheet.css" />
 </head>

 <body>
    <?
        include('menu.php');
    ?>
</body>
</html>
```

Save the changes and copy the file **displayItems.php** to the server and re-run the web site. Click any button on the category menu at the left of the page. The blank page should open. Notice that the category selected is identified by a variable attached to the page URL:



We will return later to complete this page when details of products stocked by the hardware store have been uploaded.

The input of product details will be handled by a content management system which will allow the staff to easily update the range of products. This must be password protected to prevent unauthorised access.

Begin a staff log-in screen by opening a new blank text file and entering the following code:

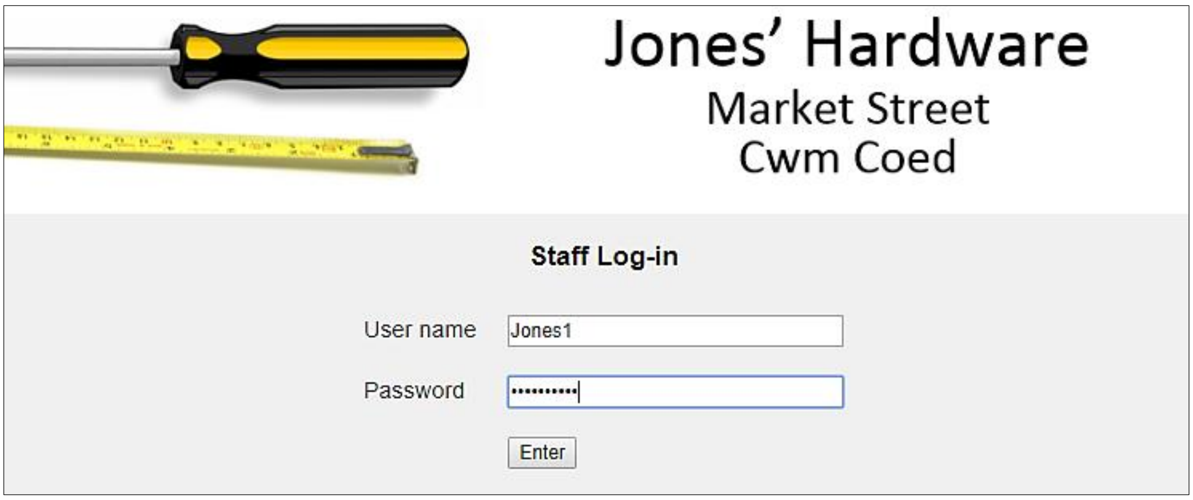
```
<html>
<head>
   <link rel="Stylesheet" type="text/css" href="styleSheet.css" />
   <title>Jones' Hardware</title>
</head>

<body>
   <div id="header">
   </div>
   <form action="editStockItem.php" method="POST">
   <center>
       <h3>Staff Log-in</h3>
       <table border="0" cellpadding="10" >
       <tr>
         <td>User name</td>
         <td>
            <?
              echo "<input type=text size=30 name=user >";
            ?>
          </td>
       </tr>
       <tr>
          <td>Password</td>
          <td>
             <?
              echo "<input type=password size=30 name=pass >";
             ?>
          </td>
        </tr>
        <tr>
          <td></td>
          <td>
              <input type=submit value="Enter">
          </td>
        </tr>
        </table>
   </center>
   </form>
 </body>
</html>
```

Save the file as **staffLogin.php** and copy it to the server.

Run the staffLogin page by entering a browser URL which includes the file name, e.g.

www.website.com/hardware/staffLogin.php

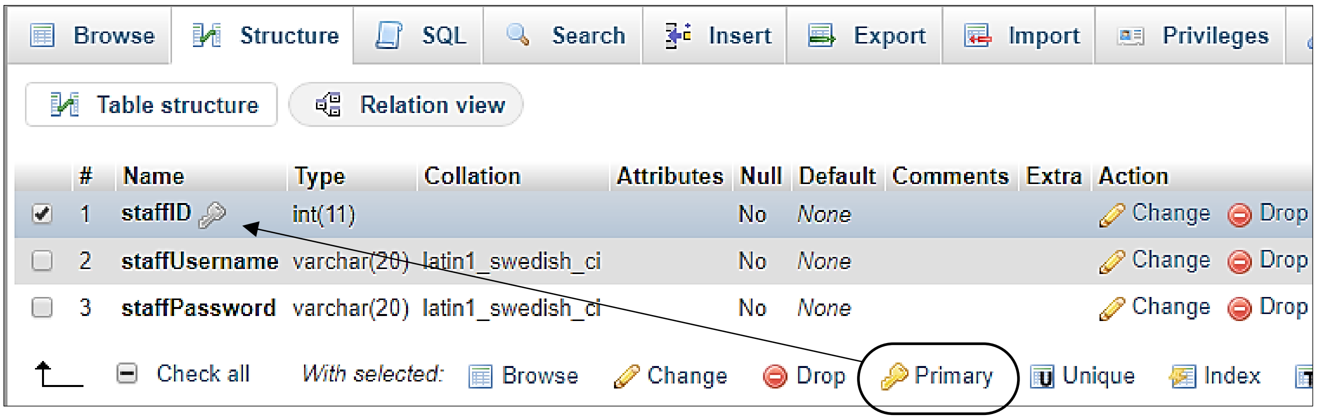
Entry boxes should appear. Notice that the password box is set to conceal the text which is being entered.


Jones' Hardware
Market Street
Cwm Coed

Staff Log-in

User name	Jones1
Password	••••••••••

We must now set up a database table for staff usernames and passwords. Log-in to the PHP MyAdmin web site for your database account and display the list of tables in the database. Select the **New** option from the list of tables. Set up three fields: **staffID** as integer, **staffUsername** and **staffPassword** both of type varchar with a length of 20 characters. Name the table as '**staff**' and save the table design.

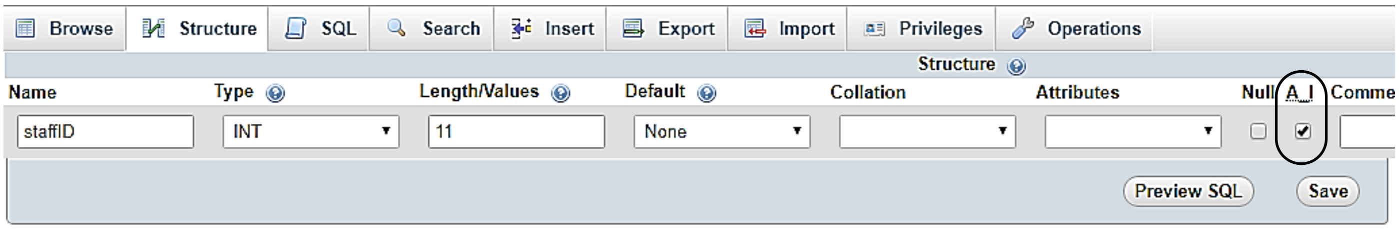


	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
✓	1	staffID	int(11)			No	None			Change Drop
	2	staffUsername	varchar(20)	latin1_swedish_ci		No	None			Change Drop
	3	staffPassword	varchar(20)	latin1_swedish_ci		No	None			Change Drop

Click the checkbox alongside the **staffID** field, then click the **Primary** icon to set this as the primary key field of the table. Notice that we have used the name of the table (staff) followed by the letters 'ID' as the identifier for the primary key field **staffID**. We will use this naming convention for all tables, to make the primary key easy to identify.

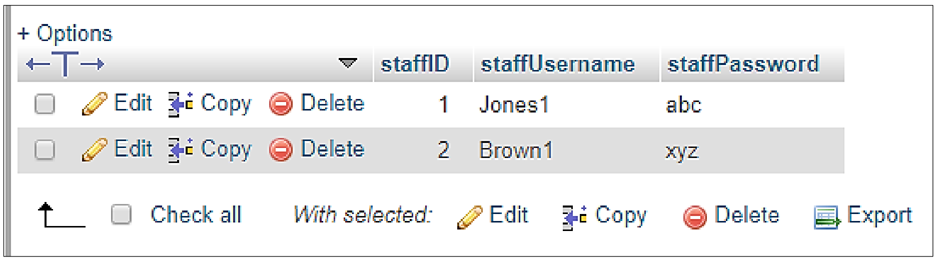
We will set our own values for **staffID**. It is important that members of staff are not accidentally assigned the same ID number. This can be avoided by allowing the database to automatically insert ID numbers which will be incremented sequentially as each new record is added.

Click the Change option on the **staffID** line, then tick the auto increment (**A_I**) box:



Name	Type	Length/Values	Default	Collation	Attributes	Null	A_I	Comme
staffID	INT	11	None			☐	✓	

Use the **Insert** option to add several members of staff as test data, as in the examples below:



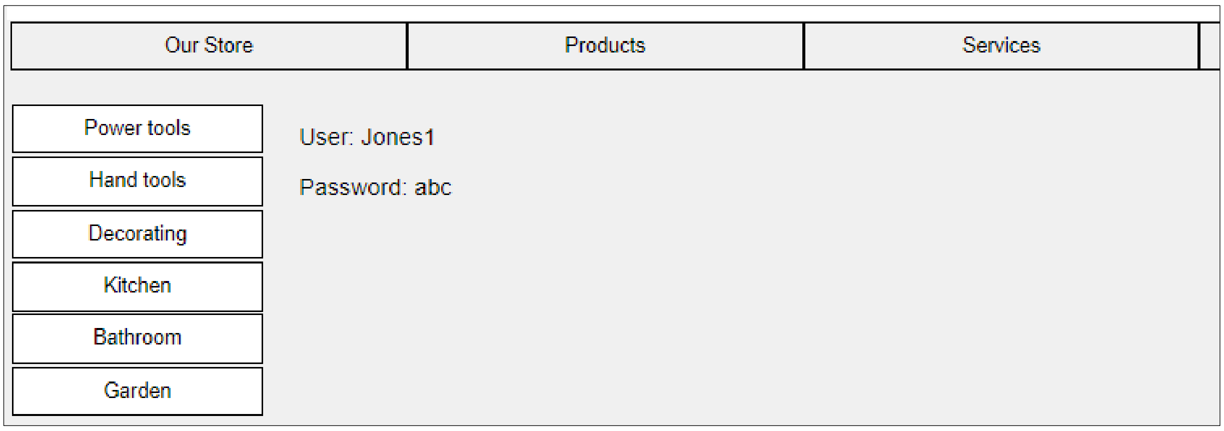
	staffID	staffUsername	staffPassword
Edit Copy Delete	1	Jones1	abc
Edit Copy Delete	2	Brown1	xyz

The first web page that a member of staff will access after logging-in will display menu options to edit or delete stock records. Create this web page, save it as **editStockItem.php** and copy the file to the server.

```
<?
   $user=$_REQUEST['user'];
   $pass=$_REQUEST['pass'];
?>

<html>
 <head>
    <title> Hardware store </title>
    <link rel="Stylesheet" type="text/css" href="styleSheet.css" />
 </head>

 <body>
    <?
        include('menu.php');
        echo"<p>User: ".$user;
        echo"<p>Password: ".$pass;
    ?>
</body>
</html>
```

Before the page runs, the PHP $_REQUEST lines will obtain the values entered in the text boxes on the log-in page, using the variable names **user** and **pass** which we allocated to these text boxes. The remainder of the page is simply our blank page template, with the addition of lines of code to output the user name and password for testing purposes. These can be removed once the program is working correctly.

Run the **staffLogin.php** page, enter a user name and password and click the 'Enter' button. Check that **editStockItem** page loads correctly with the user name and password displayed**.**



User: Jones1

Password: abc

The next stage is to access the staff database table to verify the data entered. We will begin by setting up a **user.inc** file to authorise access to the on-line database. This has the format:

```
<?
   $username="YOUR USER NAME";
   $password="YOUR PASSWORD";
   $database="YOUR DATABASE NAME";
?>
```

Create a blank text file and copy the lines above. Replace "YOUR USER NAME" and "YOUR PASSWORD" with the username and password which give you access to the PHP MyAdmin website. The entry for "YOUR DATABASE NAME" is normally the same as the username entered on the first line. Save this small file as **user.inc** and copy it to the server.

An object oriented approach will be used when working with the database. A **Staff** class will provide a link between the staff database table and the web pages, allowing log-in details to be verified.

Open a blank text file and save this as **Staff.php**. By convention, class flies have names beginning with an upper case letter. Add the lines of code below. The attributes **$user** and **$pass** are defined for a Staff object. A constructor method is then added, which can accept a user name and password and create a new Staff object with these attributes.

```
<?
class Staff
{
       private $user;
       private $pass;

       function __construct($user,$pass)
       {
           $this->user = $user;
           $this->pass = $pass;
       }
}
?>
```

Add a function which checks a Staff object against the log-in user name and password. A **true** value is returned if the log-in matches the current Staff object, or a **false** value is returned if they differ.

```
         function __construct($user,$pass)
         {
             $this->user = $user;
             $this->pass = $pass;
         }
         private function checkUser($userWanted,$passWanted)
         {
              if (($userWanted==$this->user)&&($passWanted==$this->pass))
              {
                 return true;
              }
              else
              {
                 return false;
              }
          }
    }
    ?>
```

Save the **Staff.php** file.

We have dealt so far with an individual staff record in the database, being able to convert the record to an object and then check whether the attributes of that object match the user name and password entered on the log-in screen. To produce a fully functional log-in system, we must add an overall method which will run a loop to create objects for every staff record, then check whether any of them matches the entered log-in values.

Add the method **checkPassword( )** shown below to the **Staff.php** file. This is designated as **static**, to indicate that it operates on the whole Staff class rather than on only one particular Staff object.

```
        else
        {
            return false;
        }
    }

    public static function checkPassword($userWanted,$passWanted)
    {
        include('user.inc');
        $conn = new mysqli(localhost, $username, $password, $database);
        if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
        $query="SELECT * FROM staff";
        $result=mysqli_query($conn, $query);
        $num=mysqli_num_rows($result);
        mysqli_close($conn);
        $i=1;
        while ($i <= $num)
        {
            $row=mysqli_fetch_assoc($result);
            $user= $row["staffUsername"];
            $pass=$row["staffPassword"];
            $staff[$i] = new Staff($user,$pass);
            $i++;
        }
        $found=false;
        for ($i=1;$i<=$num;$i++)
        {
            $answer= $staff[$i]->checkUser($userWanted,$passWanted);
            if ($answer==true)
            {
               $found=true;
            }
        }
        return $found;
     }
  }
  ?>
```

The **checkPassword( )** method begins by loading the **user.inc** file which will authorise access to the on-line database. All the records in the **staff** table are then copied to the server using the SQL SELECT command. The database is closed when the record transfer is completed. The variable **$num** records the number of records in the staff table.

The next section of the method uses a loop to access each record, creating a new Staff object with the corresponding user name and password attributes.

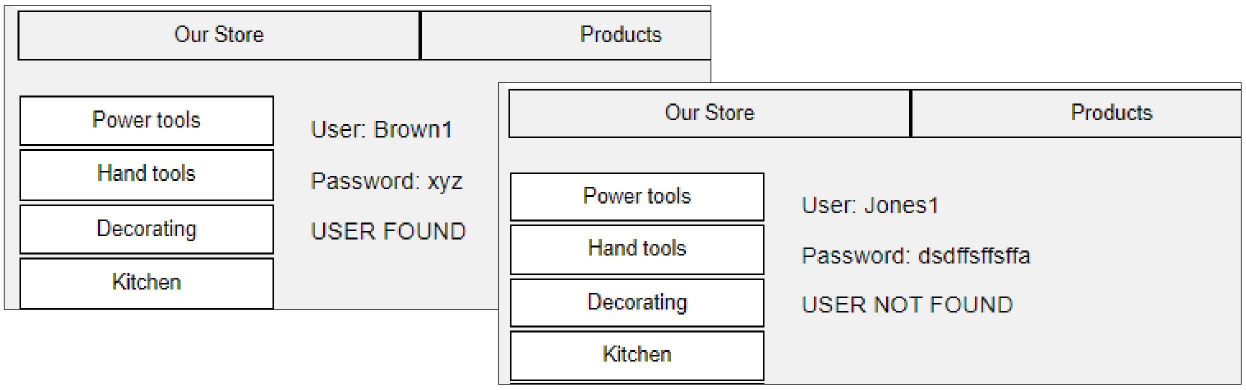
In a final loop, each Staff object is checked against the entered log-in details by means of the **checkUser( )** function which we created earlier for individual objects. A result of **true** is returned if a matching staff record is found, whilst an overall result of **false** is returned if none of the staff records match the log-in entries. Save the completed **Staff.php** class file and copy this to the server**.**

We will now test the log-in system. Add lines of code to the **editStockItem.php** file which will link to the Staff class file and run the **checkPassword( )** method. Save the file and copy it to the server.

```
<?
    include('menu.php');
    echo"<p>User: ".$user;
    echo"<p>Password: ".$pass;
    include('Staff.php');
    if (Staff::checkPassword($user,$pass)==true)
    {
        echo"<p>USER FOUND";
    }
    else
    {
        echo"<p>USER NOT FOUND";
    }
  ?>
```

Notice that a **double colon (::)** is used when a static function or variable is accessed within an object class.

Run the **staffLogin.php** page. Enter both correct and incorrect log-in details and check that these are identified correctly by the program.



```
User: Brown1

Password: xyz

USER FOUND
```

```
User: Jones1

Password: dsdffsffsffa

USER NOT FOUND
```

When the program is working correctly, re-open the **editStockItem.php** file. The lines displaying the user name, password and test message should be deleted and replaced by a **header** command, as shown below. This will immediately return the user to the log-in screen if an invalid user name or password is detected. Save the updated file and copy it to the server. Check that an incorrect log-in does not allow the user to access the stock items page.

```
<body>
  <?
      include('menu.php');
      include('Staff.php');
      if (Staff::checkPassword($user,$pass)==false)
      {
           header('Location: staffLogin.php');
      }
  ?>
</body>
```

We will now assume that a member of staff has reached the **editStockItem.php** page after entering valid log-in details. Staff will require different menu options to members of the public, so the next step is to produce a modified menu file. Open the **menu.php** file, then re-save the file with the name **staffMenu.php**.

Edit the first set of options.

```
echo"<div id='navigation'>";
echo"<ul id='nav'>";
  echo"<li><a href='addStockItem.php'>Add stock item</a></li>";
  echo"<li><a href='editStockItem.php'>Edit stock item</a></li>";
  echo"<li><a href='deleteStockItem.php'>Delete stock item</a></li>";
  echo"<li><a href='index.php'>Return to homepage</a></li>";
echo"</ul>";
echo"</div>";
echo"<div id='products'>";
```

Edit the second set of options to direct users back to the **editStockItem.php** page,

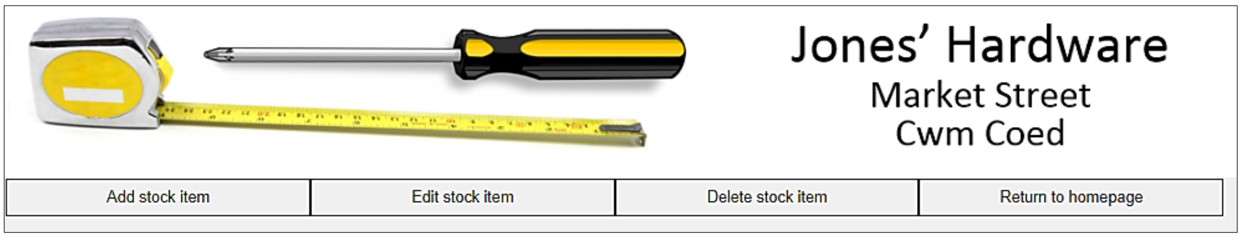
```
echo"<div id='products'>";
echo"<ul id='cat'>";
  echo"<li><a href='editStockItem.php?category=power'>Power tools</a></li>";
  echo"<li><a href='editStockItem.php?category=hand'>Hand tools</a></li>";
  echo"<li><a href='editStockItem.php?category=decorating'>Decorating</a></li>";
  echo"<li><a href='editStockItem.php?category=kitchen'>Kitchen</a></li>";
  echo"<li><a href='editStockItem.php?category=bathroom'>Bathroom</a></li>";
  echo"<li><a href='editStockItem.php?category=garden'>Garden</a></li>";
echo"</ul>";
echo"</div>";
```

Re-save the modified **staffMenu.php** file, then copy it to the server.

Finally, edit the line of code near the start of <body> in **editStockItem.php** so that it will load the modified **staffMenu.php** file. Save the amended **editStockItem.php** file and copy it to the server.

```
<body>
    <?
      include('staffMenu.php');
      include('Staff.php');
```

Run the page and log-in as a member of staff. Check that the staff menu is now displayed:



Before continuing, we need to attend to one further security issue. If an unauthorised user guessed the URL for a page within the staff system, such as **addStockItem.php**, they would be able to by-pass the log-in procedure and load the page directly. It is simple to close this loophole using a **session variable** which we can call **login**. After successful password entry, **login** will be set to '**YES**'. The variable can then be checked as each staff page is loaded, and access denied if **login** does not have a '**YES**' value.

To use a session variable, the command **session_start( )** must be inserted as the very first line of the **editStockItem.php** page. We will then read the current value of the **login** variable.

```
<?
   session_start();
   $login=$_SESSION['login'];

   $user=$_REQUEST['user'];
   $pass=$_REQUEST['pass'];
?>

<html>
 <head>
    <title> Hardware store </title>
    <link rel="Stylesheet" type="text/css" href="styleSheet.css" />
 </head>
```

Before accessing the staff database, we check whether the **login** variable already has a 'YES' value because an authorised staff member is returning to this page. If not, the password check is carried out and the **login** variable set to a 'YES' value if successful.

```
 <body>
    <?
       include('staffMenu.php');
       include('Staff.php');

       if (!($login=='YES'))
       {

              if (Staff::checkPassword($user,$pass)==false)
              {
                     header('Location: staffLogin.php');
              }

              else
              {
                     $_SESSION['login']='YES';
              }
       }
    ?>
</body>
```

Save the modified **editStockItem.php** page, and copy this to the server.

We just need to add lines of code at the start of **staffLogin.php** to activate the session and set the **login** variable back to a blank string. Open the **staffLogin.php** file and add lines of program code before the HTML page begins.

```
  <?
     session_start();
     $_SESSION['login']='';
  ?>

  <html>
     <head>
```

Save the updated file and copy it to the server.

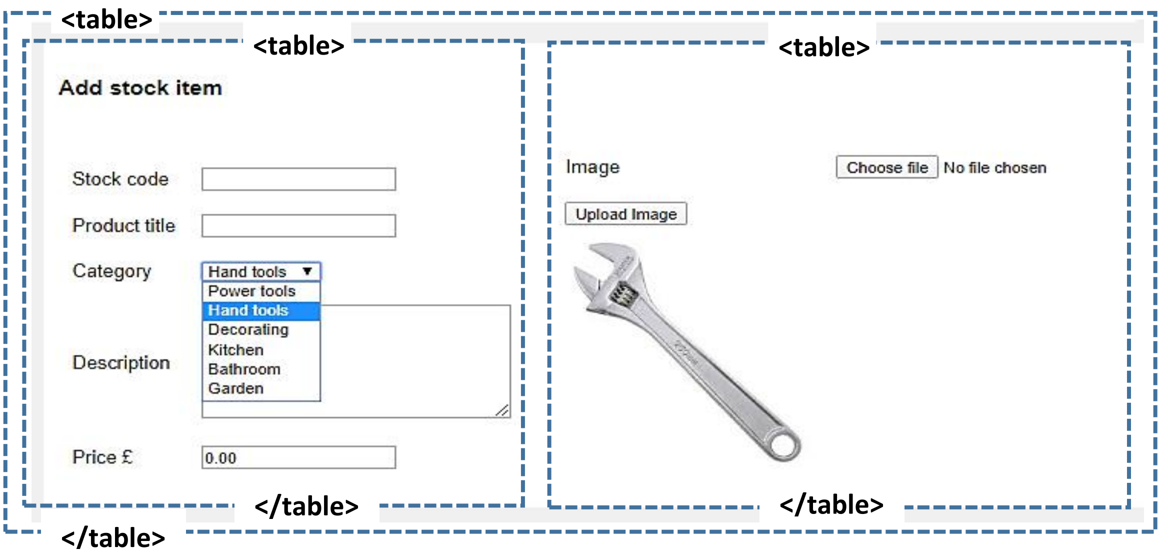
We can test the security system by creating a new **addStockItem.php** page. The page uses our standard template, but begins by accessing the login session variable. If this is not set to 'YES', the user is immediately redirected to the staff log-in page.

Open a blank file and add the program code shown below. Save the file as **addStockItem.php**, then copy it to the server.

```
<?

   session_start();
   $login=$_SESSION['login'];
   if (!($login=='YES'))
   {
         header('Location: staffLogin.php');
   }
?>
<html>
 <head>
    <title> Hardware store </title>
    <link rel="Stylesheet" type="text/css" href="styleSheet.css" />
 </head>
 <body>
    <?
        include('staffMenu.php');
    ?>
</body>
</html>
```

Run the staff login page and enter an incorrect user name or password. You should be returned to the staff log-in. Now attempt to by-pass the password system by entering the page address **addStockItem.php** in the browser. You should again be returned to the staff log-in page. Now enter a correct log-in, and it should be possible to navigate to the **addStockItem.php** page by means of the staff menu. You can check this by looking at the address bar of the browser.

We can now return to work on the **addStockItem.php** page. The page will be laid out using **table** blocks.



An outer table organises the overall structure, and contains two inner tables along side one another. The left hand table handles text input, whilst the right hand table will allow the selection, up load and display of an image of the product.

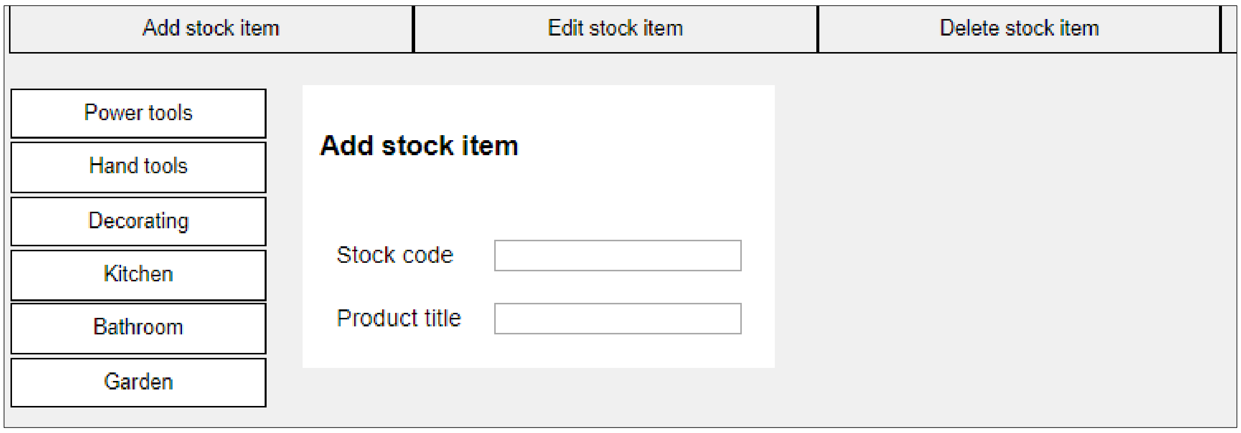
Go to the **addStockItem.php** page and add the block of program code shown below. Notice that two tables have been nested inside one another, as shown by the dotted outlines drawn on the program listing.

```
<body>
  <?
     include('staffMenu.php');
  ?>
  <table bgcolor='white' border='0' cellpadding='10'>
     <tr><td><h3>Add stock item</h3></td>
     </tr>
     <tr><td>

          <table bgcolor='white' border='0' cellpadding='10'>

          </table>

        </td>
       </tr>
    </table>
</body>
```

Within the inner **<table>** block, add lines of HTML and PHP code to input the Stock code and Product title. Please note that the **<input>** commands should be entered as a single line of code without a line break. **Name** and **id** values have been allocated to the input boxes, to identify them so that data can be copied into or collected from the boxes.

```
<table bgcolor='white' border='0' cellpadding=10>
<tr>
    <td>Stock code</td>
    <?
       echo"<td>";
       echo"<input type='text' name='txtStockcode' id='stockcode' width='300px'
                                          value='".$txtStockcode."'></td>";
    ?>
</tr>
<tr>
    <td>Product title</td>
    <?
       echo"<td>";
       echo"<input type='text' name='txtTitle' id='title' width='300px'
                                             value='".$txtTitle."'></td>";
    ?>
</tr>
</table>
</td>
</tr>
</table>
```

Re-save the updated **addStockItem.php** file and copy it to the server. Run the page and check that input boxes for Stock code and Product title are displayed.



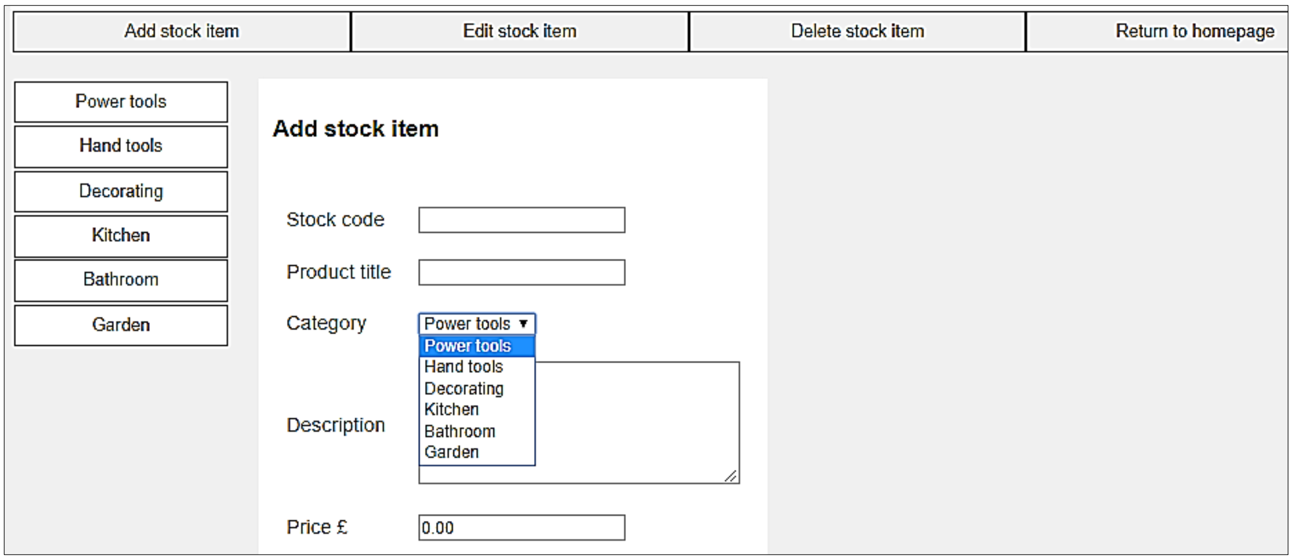
Re-open **addStockItem.php.** Insert the following PHP code to produce the drop down list of product categories below the stock code and product rows in the table. This code begins by setting up arrays for the category codes and caption text. A loop then displays the categories in the drop down list.

```
     </tr>
     <tr>
          <td>Category</td>
          <td>
             <select name="lstCategory" id="category">
             <?
                $cat[1]='power';      $text[1]='Power tools';
                $cat[2]='hand';       $text[2]='Hand tools';
                $cat[3]='decorating'; $text[3]='Decorating';
                $cat[4]='kitchen';    $text[4]='Kitchen';
                $cat[5]='bathroom';   $text[5]='Bathroom';
                $cat[6]='garden';     $text[6]='Garden';
                for ($i=1; $i<=6; $i++)
                {
                  if ($lstCategory==$cat[$i])
                     echo"<option value='".$cat[$i]."' selected>".$text[$i];
                  else
                     echo"<option value='".$cat[$i]."'>".$text[$i];
                  echo"</option>";
               }
             ?>
             </select>
          </td>
      </tr>
  </table>
  </td>
  </tr>
</table>
```

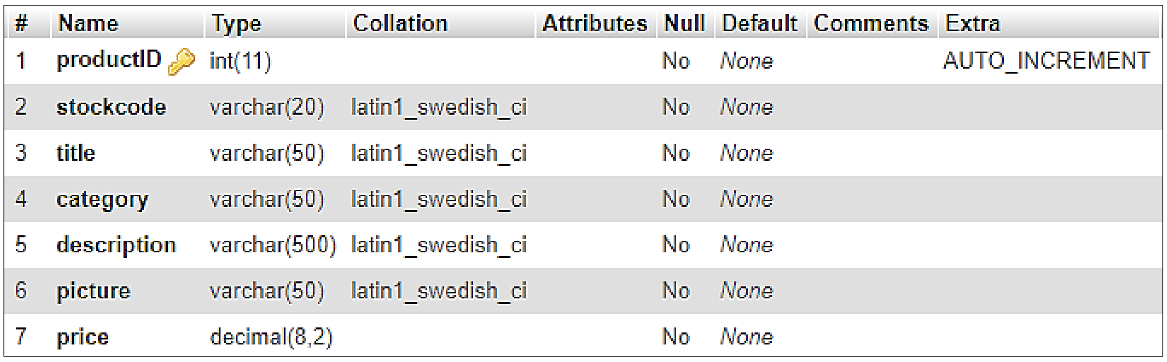
The final step in producing this section of the **addStockItem.php** page is to add input boxes for Description and Price. The **<input>** commands should be entered as a single line of code without a line break.

```
            </select>
         </td>
       </tr>
       <tr>
         <td>Description</td>
         <?
            echo"<td>";
            echo"<textarea rows = '6' cols = '30' name = 'txtDescription'
                                    id='description'>$txtDescription</textarea>";
            echo"</td>";
         ?>
       </tr>
       <tr>
         <td>Price £</td>
         <?
             echo"<td><input type='text' name='txtPrice' id='price' width='100px'
                                     value='".number_format($txtPrice,2)."'></td>";
         ?>
       </tr>
   </table>
   </td>
  </tr>
 </table>
```

Save the **addStockItem.php** file and copy it to the server. Check that the input boxes appear correctly on the page.



Go now to the PHP MyAdmin web site. Set up a new database table named **product** to store product records. As for the **staff** table, insert an integer **product ID** field and set this as the key field with auto-incrementing values. The price has the number format **decimal** with length/values set to **8,2**. The remaining fields are of type **varchar: Stockcode** has a size of 20 characters; **title**, **category** and **picture** have sizes of 50 characters; and **description** is 500 characters.



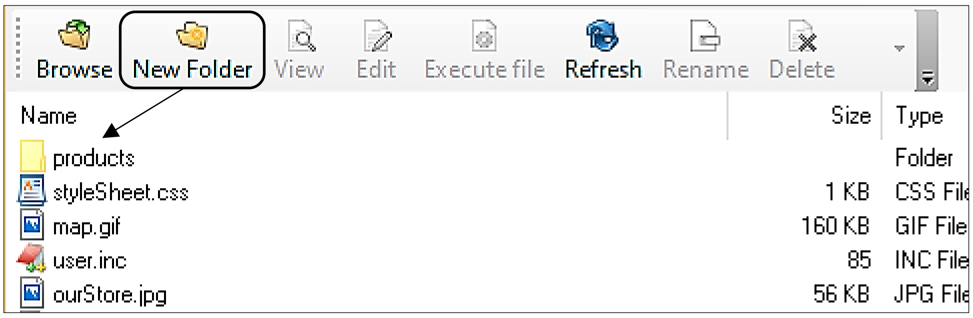
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	productID	int(11)			No	None		AUTO_INCREMENT
2	stockcode	varchar(20)	latin1_swedish_ci		No	None		
3	title	varchar(50)	latin1_swedish_ci		No	None		
4	category	varchar(50)	latin1_swedish_ci		No	None		
5	description	varchar(500)	latin1_swedish_ci		No	None		
6	picture	varchar(50)	latin1_swedish_ci		No	None		
7	price	decimal(8,2)			No	None		

The input of a picture of the stock item will involve two steps:

The user will first select the file name for an image on their client computer (e.g. hammer.jpg) and this filename will be stored in the database.

The picture file itself will be uploaded to the server and stored in a folder which we will call '**products**'.

Go to your FTP program and set up a **products** folder on the server within the **hardware** folder:



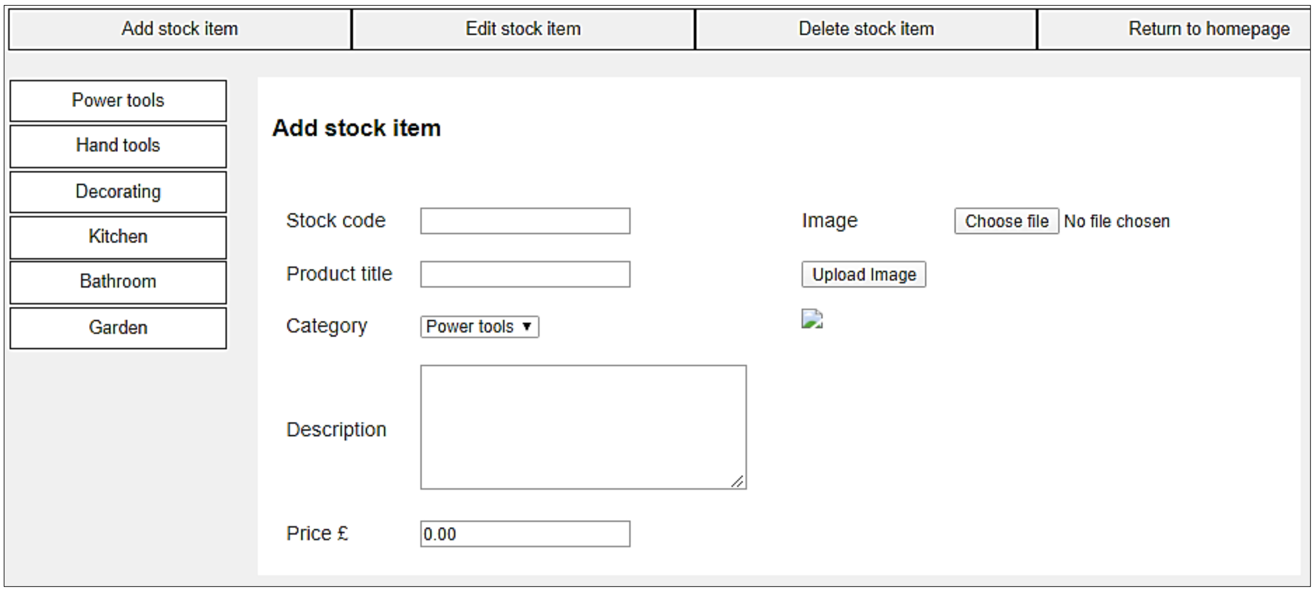
Name	Size	Type
products		Folder
styleSheet.css	1 KB	CSS File
map.gif	160 KB	GIF File
user.inc	85	INC File
ourStore.jpg	56 KB	JPG File

Return to the **addStockItem.php** file. Add the lines of code shown below to create the picture input section. The components added will be displayed in another table:

- A **file input** component will allow the picture image to be selected on the user's computer.
- An 'upload image' **button** will initiate the upload of the image file to the server.
- After the upload is completed, the **image** will be displayed on the web page and the filename will be stored in a **hidden input** for use later when we transfer the record to the database.

```
             echo"<td><input type='text' name='txtPrice' id='price' width='100px'
                                         value='".number_format($txtPrice,2)."'></td>";
         ?>
       </tr>
     </table>
     </td>
     <td valign='top'>
         <table border="0" cellpadding="10">
            <tr>
              <td>Image</td>
              <td><input type="file" name="fileToUpload" id="fileToUpload">
            </tr>
            <tr>
              <td><input type="submit" value="Upload Image" name="submit">
              <p>
              <?
                  echo"<img src='products/$imageFile' width='200'>";
                  echo"<input type='hidden'value='".$imageFile."' id='imageFile'>";
              ?>
              </td>
            </tr>
         </table>
        </td>
       </tr>
     </table>
```

Save the updated **addStockItem.php** file, then copy it to the server. Run the staff web site, navigate to the **addStockItem.php** page and check that components are displayed correctly:



The image upload will be carried out by PHP program code in a new file **upload.php**. Create a blank document and add the program code shown below. Save the file as **upload.php** and copy it to the server.

```
<?
   session_start();
   $txtStockcode = $_REQUEST["txtStockcode"];
   $txtTitle=$_REQUEST['txtTitle'];
   $lstCategory=$_REQUEST['lstCategory'];
   $txtDescription=$_REQUEST['txtDescription'];
   $txtPrice=$_REQUEST['txtPrice'];
   $imageFile=basename($_FILES["fileToUpload"]["name"]);

   $_SESSION["imageFile"]= $imageFile;
   $_SESSION["txtStockcode"] = $txtStockcode;
   $_SESSION["txtTitle"]= $txtTitle;
   $_SESSION["lstCategory"]= $lstCategory;
   $_SESSION["txtDescription"]= $txtDescription;
   $_SESSION["txtPrice"]= $txtPrice;

   $target_dir = "products/";
   $target_file = $target_dir.$imageFile;
   move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file);
   header('Location: addStockItem.php?imageUploaded=YES');
?>
```

This program begins by collecting the values input to the various text boxes and drop-down list on the **addStockItem.php** page. These values are stored stored as **session variables** for future use**.** The selected image file for the product is then uploaded to the server and stored in the products folder. Finally, the program reloads the **addStockItem.php** page.

Re-open the **addStockItem.php** file. To activate the image upload, we must add a **<form>** tag. Do this immediately before the first **<table>** tag. Please note that the **<form>** command should be entered as a single line of code without a line break.

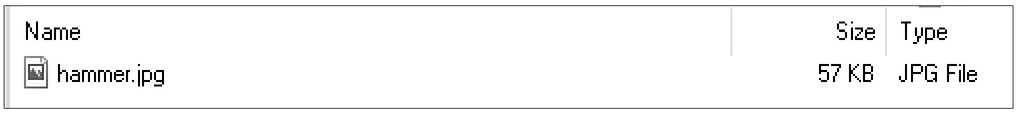
```
<body>
  <?
      include('staffMenu.php');
  ?>
   <form action="upload.php" name="itemEntryForm" method="post" enctype=
                                                  "multipart/form-data">
     <table bgcolor='white' border='0' cellpadding=10>
       <tr>
         <td><h3>Add stock item</h3></td>
       </tr>
```

Close the form with **</form>** immediately after the final **</table>** tag.

```
           </tr>
          </table>
          </form>
      </body>
```

Save the updated **addStockItem.php** file and copy it to the server.

Run the staff website. Log-in and go to the **addStockItem.php** page. Click the **Choose file** button and select an image file (**.jpg**, **.png** or **.gif** format) from any directory on your computer. Click the **Upload Image** button. Go to the server and check that the image file is now present in the **products** directory.



Name	Size	Type
hammer.jpg	57 KB	JPG File

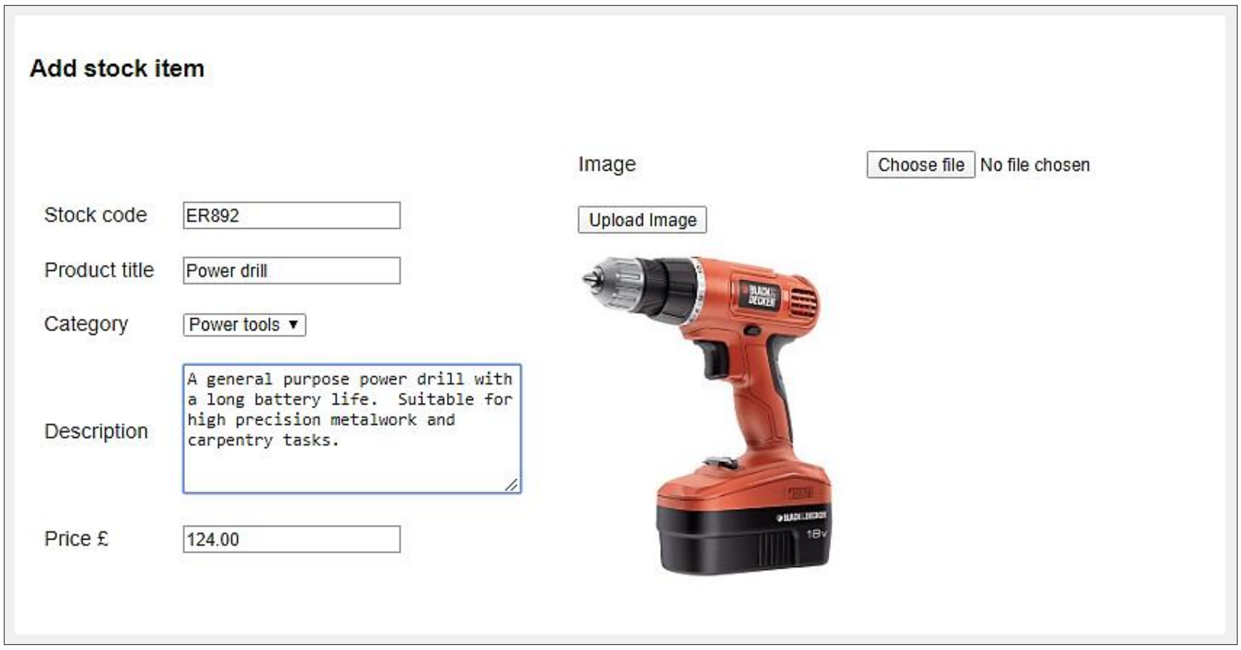
The image is not yet displayed on the **addStockItem.php** page. This is done in two stages. Firstly, we need to retrieve the file name from the session variable. This file name is then used to display the picture on the page.

Reopen the **addStockItem.php** file and add the following block of code near the start of the file.

```
<?
   session_start();
   $login=$_SESSION['login'];
   if (!($login=='YES'))
   {
        header('Location: staffLogin.php');
   }
   $imageUploaded=$_REQUEST['imageUploaded'];
   if ($imageUploaded=='YES')
   {
      $imageFile=$_SESSION['imageFile'];
      $txtStockcode=$_SESSION["txtStockcode"];
      $txtTitle=$_SESSION["txtTitle"];
      $lstCategory=$_SESSION["lstCategory"];
      $txtDescription=$_SESSION["txtDescription"];
      $txtPrice=$_SESSION["txtPrice"];
   }
?>
```

Save the updated **addStockItem.php** file and copy it to the hardware folder on the server.

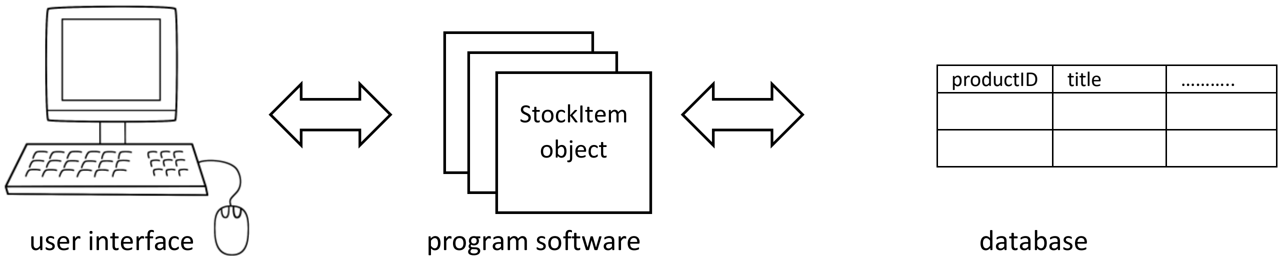
Re-run the **addStockItem.php** page. Enter a complete set of product data, making entries in each of the text boxes and selecting an image file. Click the '**Upload image**' button. The picture should now be displayed, along with the text entered in the other input boxes.



Add stock item

Stock code	ER892
Product title	Power drill
Category	Power tools
Description	A general purpose power drill with a long battery life. Suitable for high precision metalwork and carpentry tasks.
Price £	124.00
Image	Choose file No file chosen

We are now ready to save the stock record to the server. To facilitate file operations involving the stock records, we will create a **StockItem *object class*** to link the user interface and database:



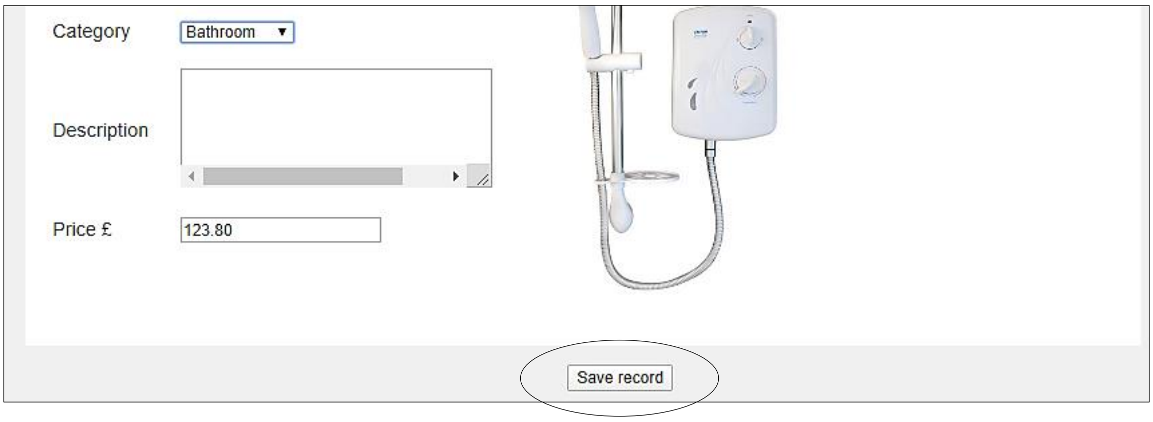
The class file contains three blocks of code:

A list of the **attributes** belonging to a StockItem object: title, description, price, etc.
A **constructor** method to create StockItem objects.
A **method** to save a product record into the database table.

Create a new blank file and add the class definition shown below. Save the file as **StockItem.php**, then copy it to the server.

```
<?
 class StockItem
 {
    private $productID;
    private $stockCode;
    private $category;
    private $title;
    private $description;
    private $image;
    private $price;

    function __construct($productID, $stockCode, $category,
                                            $title, $description, $image, $price)
    {
       $this->productID = $productID;
       $this->stockCode = $stockCode;
       $this->category = $category;
       $this->title = $title;
       $this->description = $description;
       $this->image = $image;
       $this->price = $price;
    }

    public static function saveRecord($productID, $stockCode, $category,
                                            $title, $description, $image, $price)
    {
        include('user.inc');
        $conn = new mysqli(localhost, $username, $password, $database);
        if (!$conn) {die("Connection failed: " . mysqli_connect_error()); }
        $query="INSERT INTO product VALUES ('','$stockCode', '$title',
                          '$category','$description','$image','$price') ";
        echo $query;
        $result=mysqli_query($conn, $query);
        mysqli_close($conn);
     }
   }
?>
```

Re-open the **addStockItem.php** file. We will add a button for uploading the stock item to the database.



Category Bathroom

Description

Price £ 123.80

Insert the button after the outer **</table>** and **</form>** close tags. The button will activate a Javascript function **button_click( )**.

```
      </tr>
    </table>
    </form>
    <center>
    <p>
    <button type="button" onclick="button_click()">Save record</button>

    <script>
      function button_click()
      {

      }
    </script>
    </center>
    </body>
```

Add the following program code to the **button_click( )** function. The first group of lines collects the necessary data values from the input boxes. Any required validation can then be carried out. For this example, **presence checks** are shown for the **stock code** and **title** fields.

```
    function button_click()
    {
        stockcode = document.getElementById("stockcode").value;
        title = document.getElementById("title").value;
        category = document.getElementById("category").value;
        description = document.getElementById("description").value;
        description = description.trim();
        imageFile = document.getElementById("imageFile").value;
        price = document.getElementById("price").value;
        var error=false;

        var n = stockcode.length;
        if (n<1)
        {
           alert("A stockcode must be entered");
           error=true;
        }

        n = title.length;
        if (n<1)
        {
           alert("A product title must be entered");
           error=true;
        }
     }
```

Any validation error will result in an error message being displayed, and an **error** variable will be set to **true**. Only if error remains **false** will the program continue. The data values for the product record are transferred to this page in the URL. Please note that the **destination** line should be entered without line breaks.

```
            alert("A product title must be entered");
            error=true;
        }
        if (error==false)
        {
               destination = 'saveStockItem.php?stockcode='+stockcode+'&title='
                       +title+'&category=' + category +'&description='+ description
                       +'&price='+ price +'&imagefile=' + imageFile;
               window.location.href= destination;
        }
     }
```

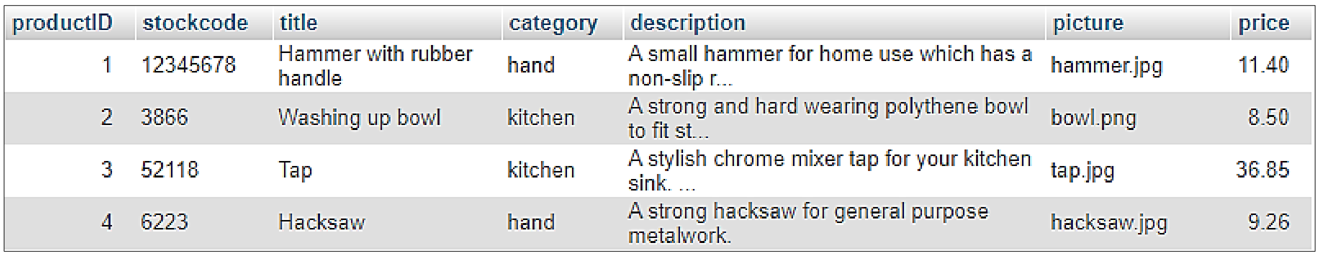
Save the completed **addStockItem.php** file and copy this to the server. Run the staff website, log-in and go to the addStockItem page. Click the 'Save record' button and check that warnings appear that the **stockcode** and **title** are missing. Validation checks could also be added for the description, price and image fields if required.

Open a new file and add the code below. Save the file as **saveStockItem.php** and copy it to the server.

```
        <html>
        <head>
          <title> Hardware store </title>
          <?
             $stockcode=$_REQUEST["stockcode"];
             $title=$_REQUEST["title"];
             $category=$_REQUEST["category"];
             $description=$_REQUEST["description"];
             $imagefile=$_REQUEST['imagefile'];
             $price=$_REQUEST["price"];
          ?>
         </head>
         <body>
         <?
             include('StockItem.php');
             StockItem::saveRecord('', $stockcode, $category, $title,
                                                 $description, $imagefile, $price);
             header('Location: editStockItem.php?categoryWanted='.$category);
         ?>
        </body>
     </html>
```

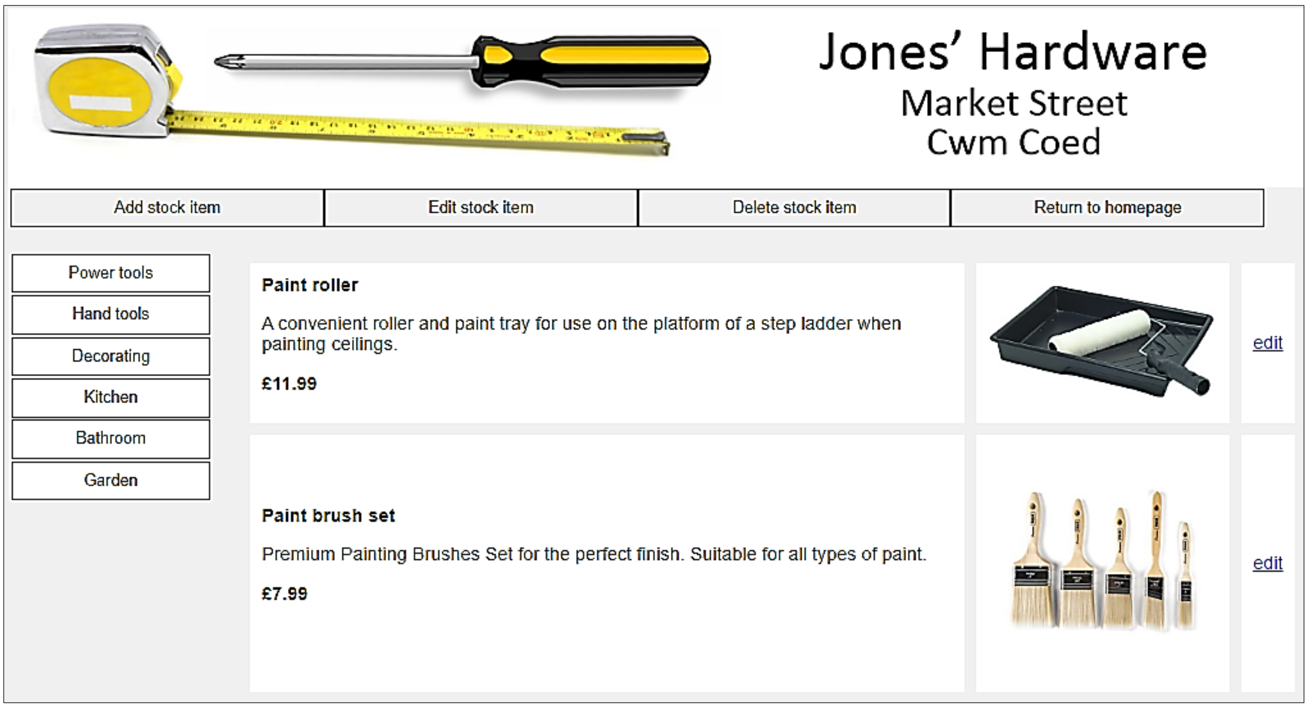
This block of code begins by collecting the field values from the page URL. The **saveRecord( )** method in the **StockItem class** is then called to transfer the record into the database. When saving is completed, the program loads the **editStockItem.php** page where the category of products containing the new record will be displayed.

Log-in to the web site as a member of staff. Go to the addStockItem page, insert details of a product including a picture, and save the record. Check that the image has been uploaded to the **products** folder on the server, and that the product record appears in the **product** table of the database.



productID	stockcode	title	category	description	picture	price
1	12345678	Hammer with rubber handle	hand	A small hammer for home use which has a non-slip r...	hammer.jpg	11.40
2	3866	Washing up bowl	kitchen	A strong and hard wearing polythene bowl to fit st...	bowl.png	8.50
3	52118	Tap	kitchen	A stylish chrome mixer tap for your kitchen sink. ...	tap.jpg	36.85
4	6223	Hacksaw	hand	A strong hacksaw for general purpose metalwork.	hacksaw.jpg	9.26

Re-open the **editStockItem.php** file. We will now complete the task which we postponed earlier - to create displays of products within the different categories. This page will provide links to allow product records to be edited.



Paint roller

A convenient roller and paint tray for use on the platform of a step ladder when painting ceilings.

£11.99

Paint brush set

Premium Painting Brushes Set for the perfect finish. Suitable for all types of paint.

£7.99

Also open the **StockItem.php** class file, so that we can add further attributes and methods which will be needed when handling the product records.

Begin by adding two **static** variables to the **StockItem.php** class file:

$itemCount is an integer recording the number of products loaded in the required category.

$product is an array identifying each of the product objects we create from the database records. The first object will be **$product[1],** the second will be **$product[2],** etc.

The variables are designated as **static** as only one instance occurs for the whole class. This is in contrast to attributes such as **$stockCode** and **$category** which have separate instances for each object: the tap has a price of £36.85, the hacksaw has a price of £9.26, etc.

```
<?
class StockItem
{
      public static $itemCount = 0;
      public static $product= array();

      private $productID;
      private $stockCode;
      private $category;
```

Go to the end of the **StockItem** class file and insert a **loadStockItems( )** method. This takes the required product category as an input parameter, then uses this parameter in an SQL command to load only the records for products in the required category. The number of product records loaded is given by the variable **$num**.

```
    public static function loadStockItems($categoryWanted)
    {
           include ('user.inc');
           $conn = new mysqli(localhost, $username, $password, $database);
           if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
           $query="SELECT * FROM product WHERE category='".$categoryWanted."'";
           $result=mysqli_query($conn, $query);
           $num=mysqli_num_rows($result);
           mysqli_close($conn);
    }
 }
?>
```

Once the required records have been loaded from the database, the **loadStockItems( )** method must use these to create a set of StockItem objects which can be displayed on the web page.

Return to the **loadStockItems( )** method at the end of the **StockItem** class file and add the block of code shown below. Save the updated **StockItem.php** file and copy it to the hardware folder on the server.

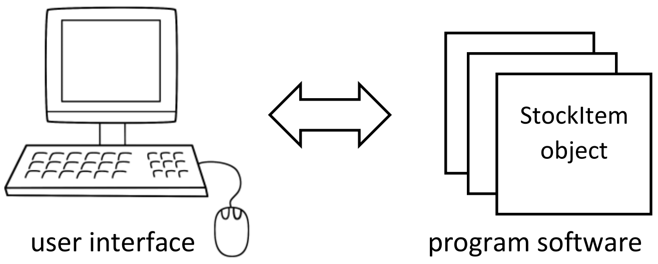
```
        $num=mysqli_num_rows($result);
        mysqli_close($conn);
        $i=1;
        while ($i <= $num)
        {
           $row=mysqli_fetch_assoc($result);
           $productID=$row["productID"];
           $stockCode=$row["stockcode"];
           $title=$row["title"];
           $category=$row["category"];
           $description=$row["description"];
           $image=$row["picture"];
           $price=$row["price"];
           $obj = new StockItem($productID, $stockCode, $category, $title,
                                       $description, $image, $price);
           StockItem::$product[$i] = $obj;
           $i++;
      }
      StockItem::$itemCount=$num;
      return $num;
    }
```

Within this block of code, a loop operates for each of the downloaded records, collecting the attribute values necessary to create a StockItem object. Once the set of attributes is complete, a new object is created using the temporary name **$obj**. The object is then allocated to an element of the **$product** array. Notice the use of the double colon symbol (::) to assign a value to a **static** variable. Finally, the function returns a count of the number of StockItem objects which have been created.

We now have the necessary file handling function, and can return to the **editStockItem.php f**ile to add program code to display the products.

In the **<body>** section of **editStockItem.php**, locate the block which checks the user name and password. Immediately after this, insert PHP code to find the category of product selected. We then call the **loadStockItems( )** method from the **StockItem** class. This will create a **$product[ ]** array of objects for the required category, count the number of objects created and return this number.

```
            $_SESSION['login']='YES';
         }
      }
      $categoryWanted=$_REQUEST['category'];
      if ($categoryWanted=="")
      {
         $categoryWanted='hand';
      }
      include('StockItem.php');
      $itemCount = StockItem::loadStockItems($categoryWanted);
   ?>
```



We now have a set of objects ready to display on the web page, but a slight problem exists. The attributes of the objects are **private**, so cannot be accessed directly by the outside program. This is a deliberate feature of object oriented programming, to prevent the accidental corruption of object data. To access the attributes we must add **get( )** methods to the **StockItem** class file.

Return to **StockItem.php** and add the group of methods shown below. These small methods act as gateways to release object attributes to the outside program.

```
            StockItem::$product[$i] = $obj;
            $i++;
         }
         StockItem::$itemCount=$num;
         return $num;
      }

         public function getProductID(){return $this->productID;}
         public function getStockCode(){return $this->stockCode;}
         public function getTitle(){return $this->title;}
         public function getCategory(){return $this->category;}
         public function getDescription(){return $this->description;}
         public function getImage(){return $this->image;}
         public function getPrice(){return $this->price;}

    }
    ?>
```

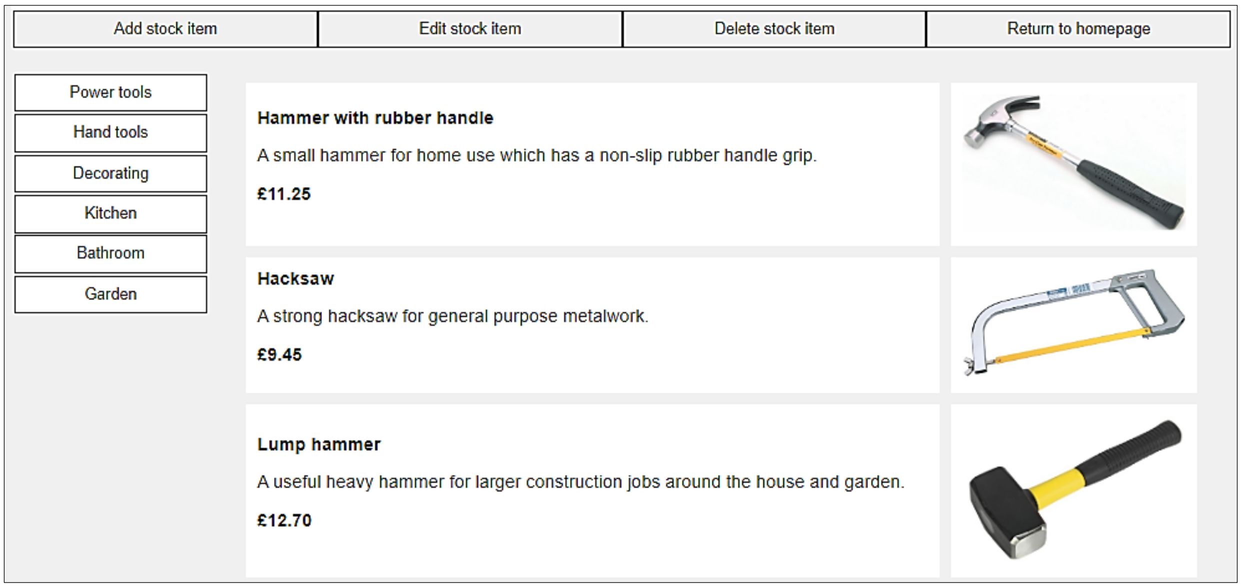
Save the updated **StockItem.php** class file, and copy it to the server.

Return to the **editStockItem.php** file and locate the block of program code which uploads the required set of StockItem objects. Immediately below this, set up a **<table>** . . . **</table>** block for display of the objects as shown below.

The program code uses a loop to access and display the attributes of each object, adding a new row to the table for each product. Notice the use of the **$product[ ]** array to identify each object, and the **double colon** (**::**) and **arrow** (**->**) operators to obtain the attributes by means of **get( )** methods.

```
      include('StockItem.php');
      $itemCount = StockItem::loadStockItems($categoryWanted);
   ?>
  <table cellspacing=10px cellpadding =10>
  <?
     for ($i=1;$i<= $itemCount;$i++)
     {
       echo"<tr><td bgcolor='white' width=600px>";
       echo"<b>".StockItem::$product[$i]-> getTitle()."</b><p>";
       echo StockItem::$product[$i]-> getDescription()."<p>";
       echo"<b>£".number_format(StockItem::$product[$i]-> getPrice(),2).
                                                            "</b><p></td>";

       echo"<td bgcolor='white'>
       <image src='products/".StockItem::$product[$i]-> getImage().
                                                      "'width=200px></td>";
     }
  ?>
  </table>
</body>
```

Save the updated **editStockItem.php** file and copy it to the server. Run the **editStockItem** page and check that product details are displayed correctly for each category.



Hammer with rubber handle

A small hammer for home use which has a non-slip rubber handle grip.

£11.25

Hacksaw

A strong hacksaw for general purpose metalwork.

£9.45

Lump hammer

A useful heavy hammer for larger construction jobs around the house and garden.

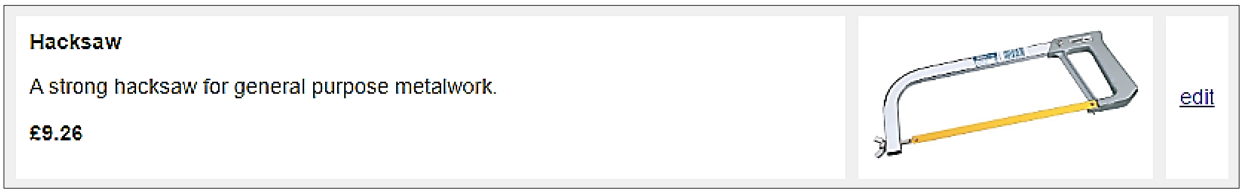
£12.70

The final step required to complete this page is to allow editing of the product records.

Insert code which will display an '**edit**' link alongside each product, and load a new page **editStockItem2.php** if this is clicked.

```
        echo StockItem::$product[$i]->description."<p>";
        echo"<b>£".number_format(StockItem::$product[$i]-> getPrice(),2)."</b><p></td>";
        echo"<td bgcolor='white'>
        <image src='products/".StockItem::$product[$i]-> getImage()."' width=200px></td>";
        echo"<td bgcolor='white'><a href='editStockItem2.php?productID="
                         .StockItem::$product[$i]-> getProductID()."'> edit</a></td>";
      }
```

Save the updated **editStockItem.php** file and copy it to the server. Run the **editStockItem.php** page and check that the 'edit' option is displayed alongside each product entry:



Hacksaw

A strong hacksaw for general purpose metalwork.

£9.26

edit

Return to the **StockItem.php** class file. Insert the method **loadByProductID( ).**

```
   public function getImage(){return $this->image;}
   public function getPrice(){return $this->price;}
   public static function loadByProductID($productIDwanted)
   {
       include ('user.inc');
       $conn = new mysqli(localhost, $username, $password, $database);
       if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
       $query="SELECT * FROM product WHERE productID='".$productIDwanted."'";
       $result=mysqli_query($conn, $query);
       $row=mysqli_fetch_assoc($result);
       $productID=$row["productID"];
       $stockCode=$row["stockcode"];
       $title=$row["title"];
       $category=$row["category"];
       $description=$row["description"];
       $image=$row["picture"];
       $price=$row["price"];
       $obj = new StockItem($productIDwanted, $stockCode, $category, $title,
                                           $description, $image, $price);
       StockItem::$product[0] = $obj;
  }
}
?>
```

The **loadByProductID( )** method works in a similar way to the loadStockItems( ) function, but only the one record to be edited will be retrieved.

Save the updated **StockItem.php** class file and copy it to the server.

The required record is transferred to an object identified as **$product[0]**. The attributes of this object can now be accessed and edited. The page for editing this product will be almost identical to the page for entering a new product record, so we can save time by copying this page file. Open the existing file **addStockItem.php**, then re-save it as **editStockItem2.php**. Copy this to the server.

Insert lines of code into **editStockItem2.php** as shown below. These call the **loadByProductID()** function in the class file, then access the properties of the object. This will allow values to be displayed in the various text boxes on the form, and the picture image will be loaded.

```
    $imageUploaded=$_REQUEST['imageUploaded'];
    $productID=0;
    if ($imageUploaded=='YES')
    {
       $productID=$_SESSION['productID'];
       $imageFile=$_SESSION['imageFile'];
       $txtStockcode=$_SESSION["txtStockcode"];
       $txtTitle=$_SESSION["txtTitle"];
       $lstCategory=$_SESSION["lstCategory"];
       $txtDescription=$_SESSION["txtDescription"];
       $txtPrice=$_SESSION["txtPrice"];
    }
    else
    {
       $productID=$_REQUEST['productID'];
       echo"<input type='hidden' id='productID' value=$productID>";
       include('StockItem.php');
       StockItem::loadByProductID($productID);
       $txtStockcode=StockItem::$product[0]->getStockCode();
       $txtTitle=StockItem::$product[0]->getTitle();
       $lstCategory=StockItem::$product[0]->getCategory();
       $txtDescription=StockItem::$product[0]->getDescription();
       $txtPrice=StockItem::$product[0]->getPrice();
       $imageFile=StockItem::$product[0]->getImage();
    }
?>
<html>
```

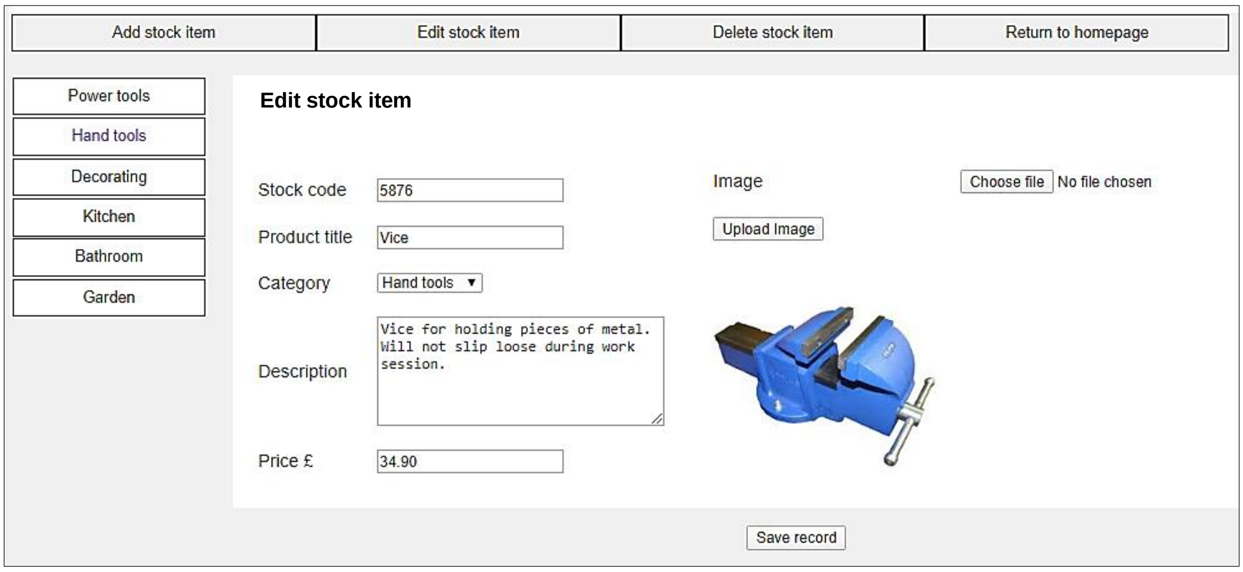
Within the <body> section of **editStockItem2.php,** make changes to the lines of code shown below:

- Between the **<form> .. </form>** tags, replace the name of the program file **upload.php** with **upload2.php.** This will alter the page which will be loaded next if the option to change the picture image is selected.
- Add a hidden **<input>** line to store the productID.
- Amend the heading to read 'Edit stock item'.

```
        include('staffMenu.php');
    ?>
    <form action="upload2.php" name="itemEntryForm" method="post"
                                             enctype="multipart/form-data">
     <?
        echo"<input type='hidden' name='productID' id='productID' value=$productID>";
     ?>
     <table bgcolor='white' border='0' cellpadding='10'>
     <tr><td><h3>Edit stock item</h3></td>
     </tr>
     <tr><td>
     <table bgcolor='white' border='0' cellpadding='10'>
```

Save the updated **editStockItem2.php** file and copy it to the server.

Run the website and log-in as a member of staff. Select a product and click the 'edit' link. Check that the product details are displayed.



Edit stock item

Stock code 5876

Product title Vice

Category Hand tools

Description Vice for holding pieces of metal. Will not slip loose during work session.

Price £ 34.90

Image No file chosen

A program file **upload2.php** will now be needed to handle the uploading of a new image to the server if the picture is changed by the user. This will be very similar to the **upload.php** file produced earlier. We will save time by copying this file.

Open the existing file **upload.php** and re-save it as **upload2.php.** Add the lines of code shown below. These commands will access **productID** along with the other fields and store it as a session variable.

```
<?
   session_start();
   $productID = $_REQUEST["productID"];
   $txtStockcode = $_REQUEST["txtStockcode"];
   $txtTitle=$_REQUEST['txtTitle'];
   $lstCategory=$_REQUEST['lstCategory'];
   $txtDescription=$_REQUEST['txtDescription'];
   $txtPrice=$_REQUEST['txtPrice'];
   $imageFile=basename($_FILES["fileToUpload"]["name"]);
   $_SESSION["productID"]= $productID;
   $_SESSION["imageFile"]= $imageFile;
   $_SESSION["txtStockcode"] = $txtStockcode;
```

Change the return address in the header line to **editStockItem2.php**.

```
   $target_dir = "products/";
   $target_file = $target_dir.$imageFile;
   move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file);
   header('Location: editStockItem2.php?imageUploaded=YES');
?>
```

Save the updated **upload2.php** file and copy it to the server.

Run the staff web site. Select a product record to edit. When the editing page appears, choose an alternative **picture file** and check that this is uploaded to the server and displayed correctly. Do not click the 'save record' button, as we must first modify the program to update the existing record rather than save a new record.

The final step in editing the product is to re-save the updated product record to the database. Open the **editStockItem2.php** file to do this.

Make a change to the URL address which is used by the '**Save record**' button at the bottom of the page. The URL will include the value of the **productID** key field, so that the correct record will be updated.

```
        if (error==false)

        {
           productID = document.getElementById("productID").value;
           destination = 'updateStockItem.php?productID='+productID+'&stockcode='
                  +stockcode+'&title='+title+'&category='+category+'&description='
                         +description +'&price='+ price +'&imagefile=' + imageFile;
           window.location.href= destination;
        }
     }
     </script>
```

Save the updated **editStockItem2.php** file and copy it to the server.

Re-open the **StockItem.php** class file and add a function to update the required record. The long line setting up the variable **$query** should be entered without line breaks.

```
     public static function updateRecord($productID, $stockCode, $category, $title,
                                                  $description, $image, $price)
     {
        include('user.inc');
        $conn = new mysqli(localhost, $username, $password, $database);
        if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
        $query="UPDATE product SET stockcode='".$stockCode."', title='".$title."',
         category='".$category."',description='".$description."', picture='".$image."',
                                 price='".$price."' WHERE productID='".$productID."'";
        echo $query;
        $result=mysqli_query($conn, $query);
        mysqli_close($conn);
     }
   }
?>
```

Save the updated **StockItem.php** class file and copy it to the server.

We will now produce an **updateStockItem.php** page. This page will not be displayed to the user, but will contain program code to update the record in the database when changes have been made to the product details.

The **updateStockItem.php** page will be very similar to the page used to save new records, so to save time we will copy this existing page file as a starting point. Load **saveStockItem.php** and resave it as **updateStockItem.php**.

Within the **<head>** block, add a line of code to obtain the **productID** for the record to be updated.

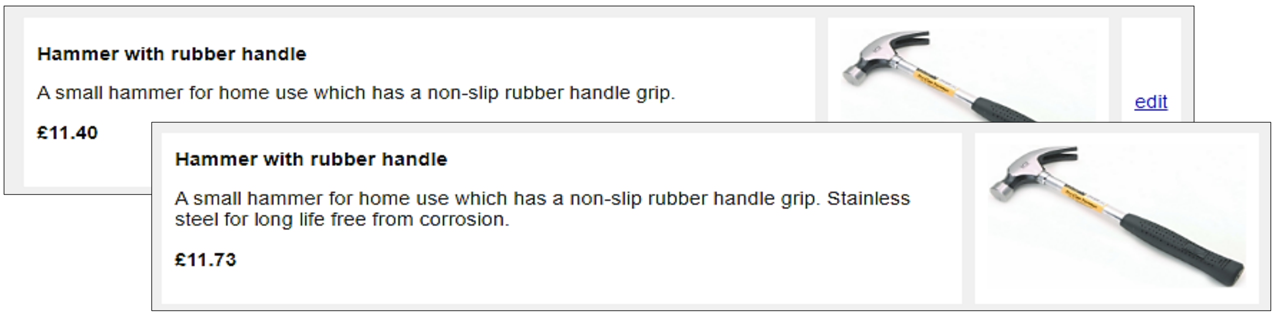
```
<html>
 <head>
  <title> Hardware store </title>
  <?
     $productID=$_REQUEST["productID"];
     $stockcode=$_REQUEST["stockcode"];
     $title=$_REQUEST["title"];
     $category=$_REQUEST["category"];
     $description=$_REQUEST["description"];
     $imagefile=$_REQUEST['imagefile'];
     $price=$_REQUEST["price"];
 ?>
 </head>
```

In the **<body>** block, edit the line of code to call the **updateRecord( )** function in the StockItem class.

```
 <body>
 <?
     include('StockItem.php');
     StockItem::updateRecord($productID, $stockcode, $category, $title,
                                          $description, $imagefile, $price);
     header('Location: editStockItem.php?category='.$category);
 ?>
 </body>
</html>
```

Save the **updateStockItem.php** file and copy it to the server.

Run the staff web site and select a product to edit. Make changes to the fields, then save the updated record. Check that the changes are shown correctly when the product is now displayed.



Hammer with rubber handle

A small hammer for home use which has a non-slip rubber handle grip.

£11.40

edit

Hammer with rubber handle

A small hammer for home use which has a non-slip rubber handle grip. Stainless steel for long life free from corrosion.

£11.73

The final menu option to complete is '**Delete stock item**'. The screen display for this option will be very similar to the **editStockItem.php** page, so we can again save time by using this file.

Open **editStockItem.php.** Make a change to the **include** line shown below, then re-save the file as **deleteStockItem.php**.

```
<body>
      <?
        include('staffMenu2.php');
         include('Staff.php');
```

Open the **staffMenu.php** file. Change the lines of code in the **products** division so that the program re-loads **deleteStockItem.php**. Save the file as **staffMenu2.php**, and copy it to the server.

```
echo"<div id='products'>";

   echo"<ul id='cat'>";
     echo"<li><a href='deleteStockItem.php?category=power'>Power tools</a></li>";
     echo"<li><a href='deleteStockItem.php?category=hand'>Hand tools</a></li>";
     echo"<li><a href='deleteStockItem.php?category=decorating'>Decorating</a></li>";
     echo"<li><a href='deleteStockItem.php?category=kitchen'>Kitchen</a></li>";
     echo"<li><a href='deleteStockItem.php?category=bathroom'>Bathroom</a></li>";
     echo"<li><a href='deleteStockItem.php?category=garden'>Garden</a></li>";
   echo"</ul>";
  echo"</div>";
 ?>
```

Re-open the **deleteStockItem.php** file and make the changes shown below. The 'edit' links alongside the products are replaced by 'delete' buttons.

```
   <table cellspacing=10px cellpadding =10>
    <?
      for ($i=1;$i<= $itemCount;$i++)
      {
       echo"<tr><td bgcolor='white' width=600px>";
        $productID=StockItem::$product[$i]-> getProductID();
        $title=StockItem::$product[$i]-> getTitle();
        echo"<b>".$title."</b><p>";
```

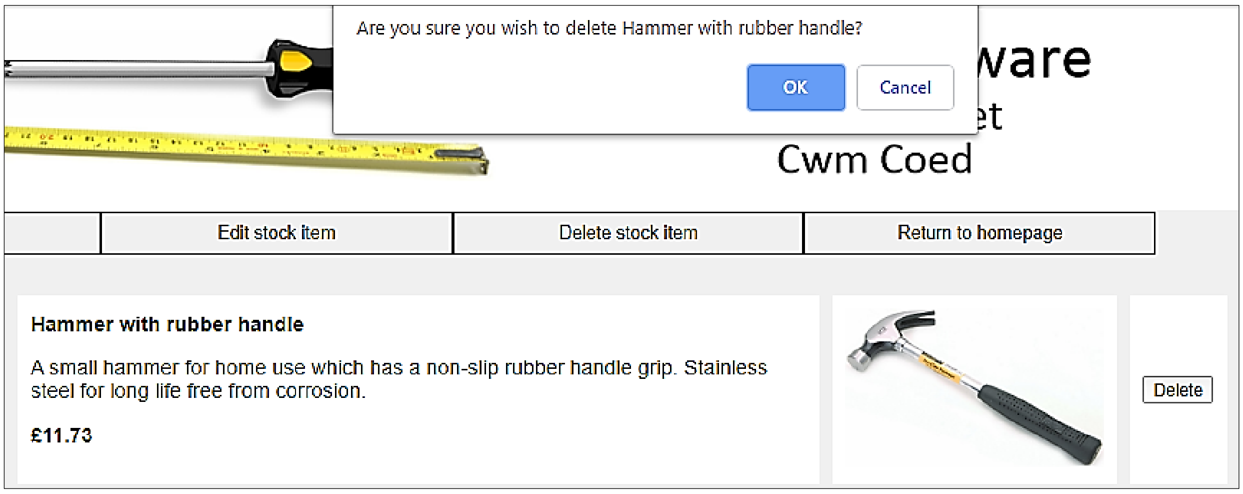
Replaces the line:
```
echo"<b>".StockItem::$product[$i]...
```

```
        echo StockItem::$product[$i]-> getDescription()."<p>";
        echo"<b>£".number_format(StockItem::$product[$i]-> getPrice(),2)."</b><p></td>";
        echo"<td bgcolor='white'>
       <image src='products/".StockItem::$product[$i]->getImage()."' width=200px></td>";
       $text=$productID."**".$title;
        echo"<td bgcolor='white'><input type='button' id='".$text."' value='Delete'
                                    onclick='confirmDelete(this)'></button></td>";
```

Replaces the line:
```
echo"<td bgcolor='white'><a href='editStockItem2.php...
```

```
      }
    ?>
    </table>
      <script>
          function confirmDelete(element)
          {
             productText = document.getElementById(element.id).id;
             words = productText.split('**');
             productID=words[0];
             title=words[1];
             choice = confirm('Are you sure you wish to delete '+words[1]+'?');
             if (choice == true)
             {
               window.location.href= "deleteStockItem2.php?productIDwanted="+productID;
             }
           }
      </script>
 </body>
```

When a button is clicked, a message window will ask for confirmation that the record is to be deleted.



Are you sure you wish to delete Hammer with rubber handle?

Hammer with rubber handle

A small hammer for home use which has a non-slip rubber handle grip. Stainless steel for long life free from corrosion.

£11.73

Save the updated **deleteStockItem.php** file and copy it to the server.

Run the staff web site. Select the **'Delete stock item'** option from the menu. Click the '**Delete**' button alongside a product, and check that the confirm message is displayed. Click '**cancel**' and return to the product display.

To complete the delete operation, we must add a method to the StockItem class file. Open the **StockItem.php** file and add the deleteProduct( ) function shown below.

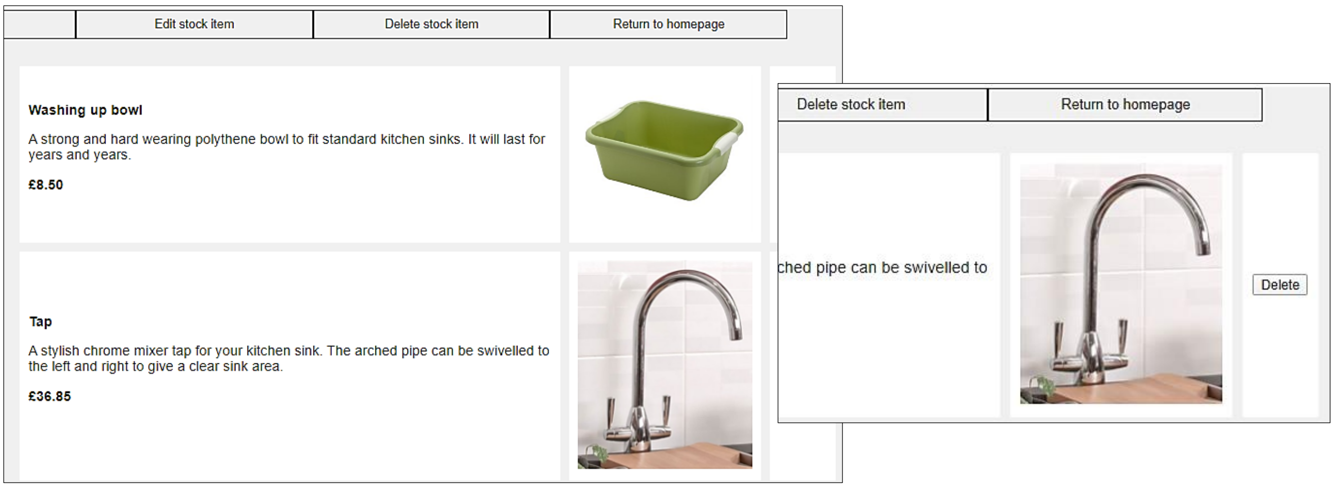
```
       public static function deleteProduct($productIDwanted)
       {
          include('user.inc');
          $conn = new mysqli(localhost, $username, $password, $database);
          if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
          $query = "DELETE FROM product WHERE productID='".$productIDwanted."'";
          $result=mysqli_query($conn, $query);
          mysqli_close($conn);
       }
    }
?>
```

The fuction inputs the **productID**, which is used in an SQL command to identify the record to be deleted from the database. Save the amended **StockItem.php** file and copy it to the server.

Finally, open a new text file and add the code shown below. Save this file as **deleteStockItem2.php** and copy it to the server. This small program file obtains the **productID**, then calls the **deleteProduct( )** method in the StockItem class file. This in turn runs the SQL query to delete the specified record from the database table.

```
<?
   $productIDwanted = $_REQUEST['productIDwanted'];
   include('StockItem.php');
   StockItem::deleteProduct($productIDwanted);
   header('Location: deleteStockItem.php');
?>
```

Run the staff web site. Enter a test record, then check that this can be deleted successfully.



Washing up bowl

A strong and hard wearing polythene bowl to fit standard kitchen sinks. It will last for years and years.

£8.50

Tap

A stylish chrome mixer tap for your kitchen sink. The arched pipe can be swivelled to the left and right to give a clear sink area.

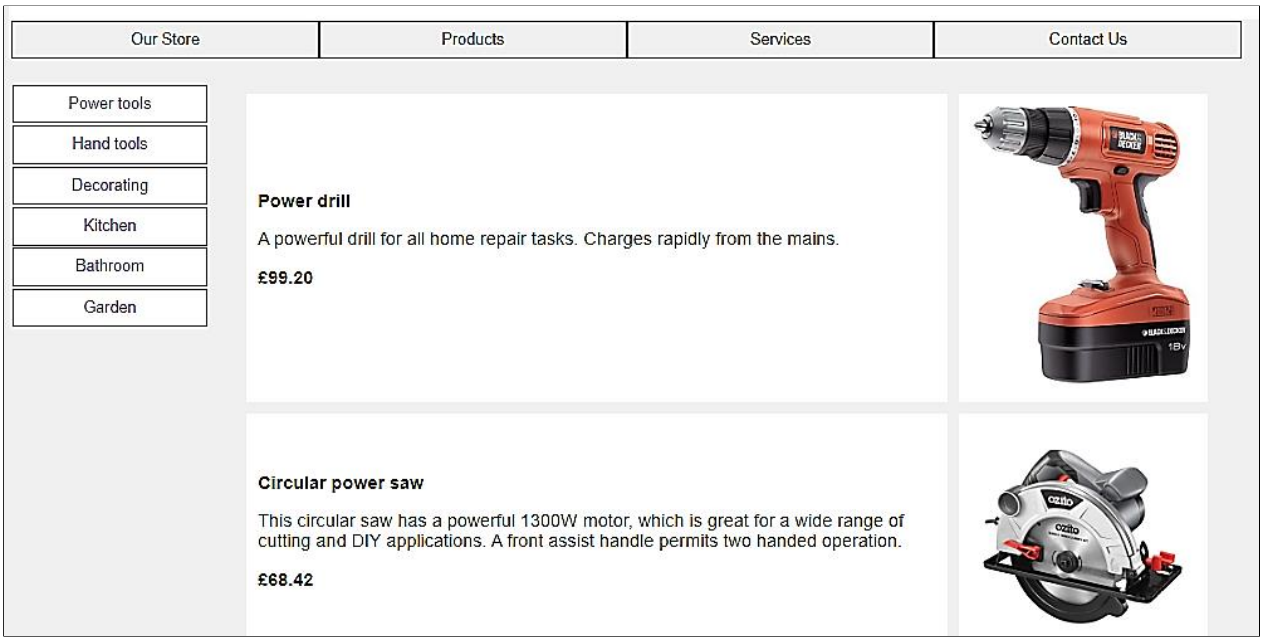
£36.85

ched pipe can be swivelled to

This completes the staff section of the website, but we need to return to the public section to set up the display of product records. Open the file **displayItems.php** and add the code shown below.

```
 <body>
   <?
     include('menu.php');
     $categoryWanted=$_REQUEST['category'];
     if ($categoryWanted=="")
     {
        $categoryWanted='hand';
     }
     include('StockItem.php');
     $itemCount = StockItem::loadStockItems($categoryWanted);
     echo"<table cellspacing=10px cellpadding =10>";
     for ($i=1;$i<= $itemCount;$i++)
     {
        echo"<tr><td bgcolor='white' width=600px>";
        echo"<b>".StockItem::$product[$i]->getTitle()."</b><p>";
        echo StockItem::$product[$i]->getDescription()."<p>";
        echo"<b>£".number_format(StockItem::$product[$i]->getPrice(),2)."</b><p></td>";
        echo"<td bgcolor='white'>
        <image src='products/".StockItem::$product[$i]->getImage()."' width=200px></td>";
     }
    ?>
</body>
</html>
```

Save the updated **displayItems.php** file and copy it to the server. Run the public section of the web site. Select categories of product from the left-hand menu, and check that product details are displayed correctly.



Power drill

A powerful drill for all home repair tasks. Charges rapidly from the mains.

£99.20

Circular power saw

This circular saw has a powerful 1300W motor, which is great for a wide range of cutting and DIY applications. A front assist handle permits two handed operation.

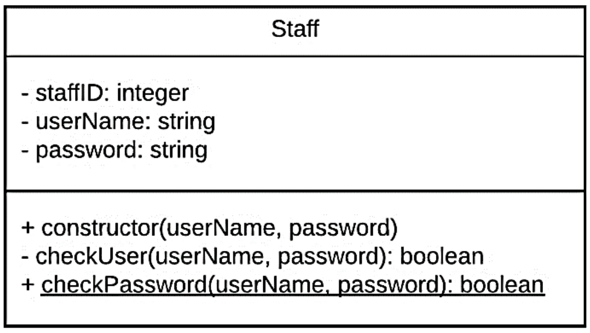
£68.42

Further development

This project could form the basis for developing a larger on-line sales and stock control system, which might also allow customers to upload reviews of the products or send on-line enquiries to the shop. Similar programs to this web site could be developed for a wide range of shops, for example: selling computer equipment, books or music CD's.

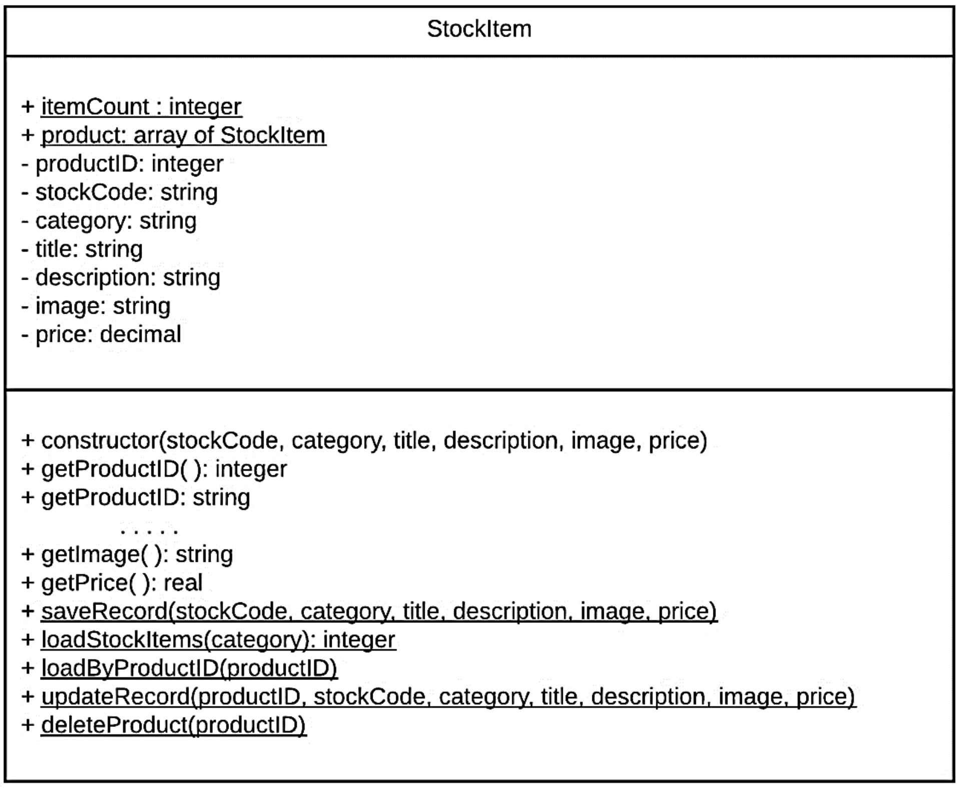
The hardware store project could form the basis for a website accepting orders and payment on-line for products, or managing bookings on-line for services. Techniques for handling on-line bookings will be explored in the *caravan park* and *airline bookings* projects which follow.

Summary of the object structures



```
Staff

- staffID: integer
- userName: string
- password: string

+ constructor(userName, password)
- checkUser(userName, password): boolean
+ checkPassword(userName, password): boolean
```

-	private
+	public
underlined	static



```
StockItem

+ itemCount : integer
+ product: array of StockItem
- productID: integer
- stockCode: string
- category: string
- title: string
- description: string
- image: string
- price: decimal

+ constructor(stockCode, category, title, description, image, price)
+ getProductID( ): integer
+ getProductID: string
          . . . . .
+ getImage( ): string
+ getPrice( ): real
+ saveRecord(stockCode, category, title, description, image, price)
+ loadStockItems(category): integer
+ loadByProductID(productID)
+ updateRecord(productID, stockCode, category, title, description, image, price)
+ deleteProduct(productID)
```