Caravan park

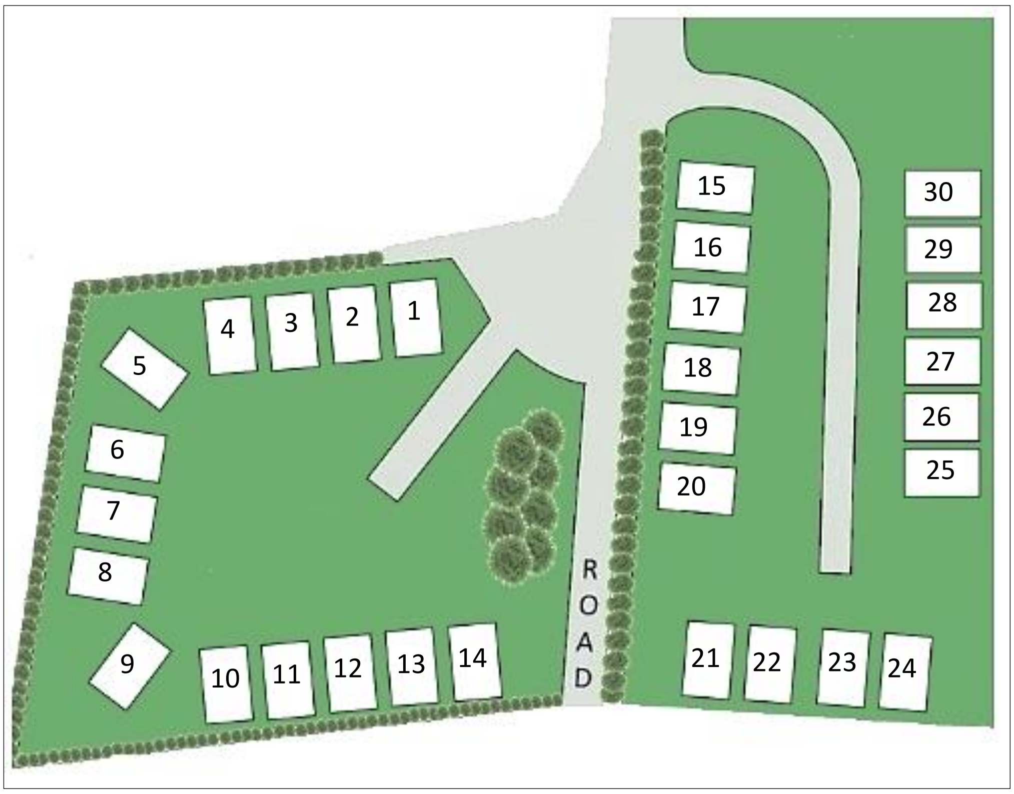
4

Scenario

A caravan park on the Welsh coast provides static caravans which visitors may rent for holidays. The park is open throughout the year, with weekly bookings running from Saturday afternoon to the following Saturday morning. The park has groups of caravans, with each group having a different design and offering different facilities. Caravans have different hire charges.

A web site is required to provide customers with information about the caravans, hire charges and available weeks, and to handle on-line bookings and payments. The web site should also provide password-protected staff pages where bookings can be viewed and changes can be made to the prices or availability of particular caravans.

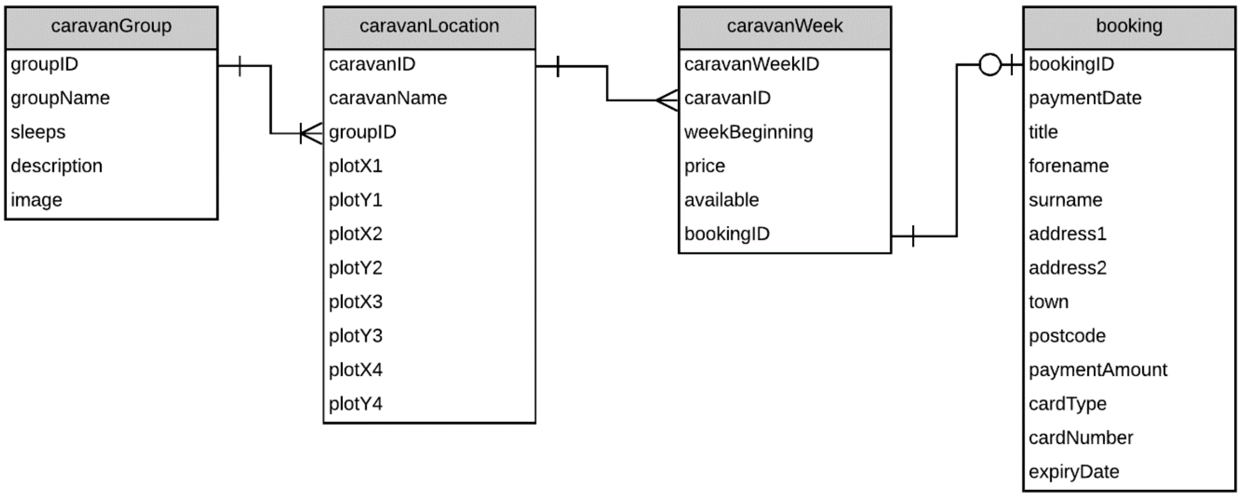
Within each group, the caravans have names relating to an aspect of the Welsh countryside: flowers, trees, rivers, mountains and castles. Caravan reference numbers are shown on the site plan, with the caravan names listed in the table below.



Flower class	Tree class	River class	Mountain class	Castle class
1: Daffodil	10: Oak	15: Wye	21: Snowdon	25: Harlech
2: Buttercup	11: Beech	16: Severn	22: Cader Idris	26: Caernarfon
3: Snowdrop	12: Hazel	17: Taff	23: Plynlimon	27: Caerphilly
4: Daisy	13: Holly	18: Dovey	24: Pen y fan	28: Criccieth
5: Bluebell	14: Birch	19: Conwy		29: Beaumaris
6: Cowslip		20: Mawddach		30: Dolwyddelan
7: Foxglove				
8: Lily				
9: Celandine				

Design

The program will obtain information from an on-line database, arranged in a series of tables as shown in the entity-relationship diagram below:



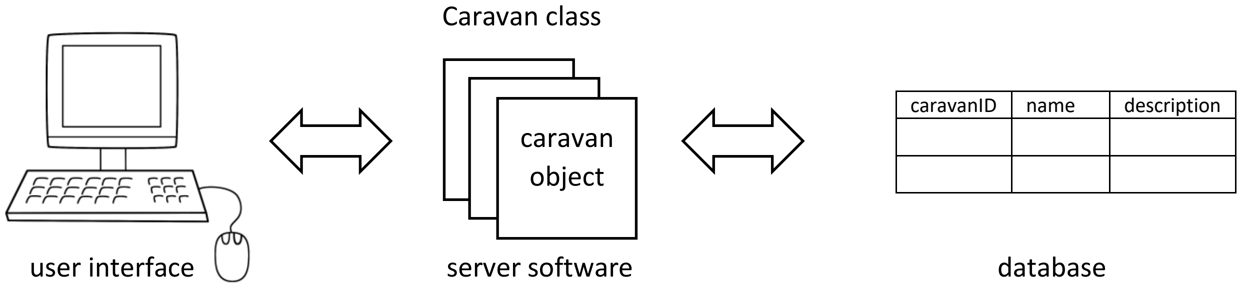
The ***caravanGroup*** table contains a description and link to a photograph for each group of caravans. Caravans within a group have the same design and facilities.

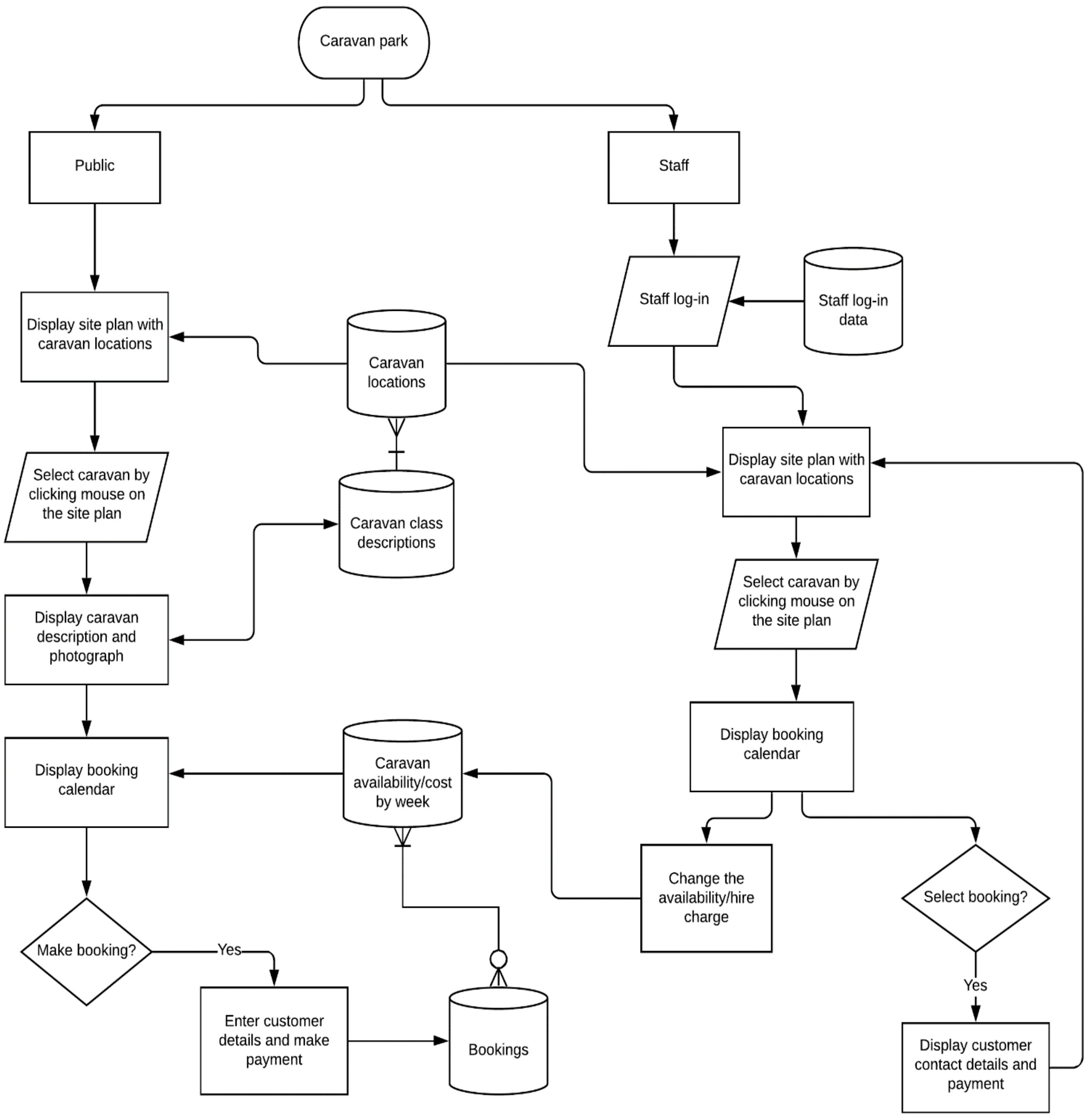
The ***caravanLocation*** table contains an entry for each caravan, giving its name and a link to the group description. Four sets of (x,y) coordinates are given for the corners of a rectangle defining the position of the caravan on the map of the site.

The ***caravanWeek*** table contains 52 week records for each caravan covering the current year. The ***weekBeginning*** field of each record specifies the Saturday start date of the hire week. The **hire charge** for the week is also listed. Hire charges may differ between quiet and busy times of the year. The **available** field specifies whether the caravan is already booked (code 0), available during the specified week (code 1), or unavailable for booking (code 2) - for example, if maintenance is being carried out.

Programming techniques

The web site will introduce the use of JavaScript graphics to produce a more realistic map selection function than the use of simple buttons. As in the previous projects, classes of objects will form an interface between the database and the local computer.




Caravan park
Public
Staff
Staff log-in
Staff log-in data
Display site plan with caravan locations
Caravan locations
Display site plan with caravan locations
Select caravan by clicking mouse on the site plan
Caravan class descriptions
Select caravan by clicking mouse on the site plan
Display caravan description and photograph
Display booking calendar
Display booking calendar
Caravan availability/cost by week
Change the availability/hire charge
Select booking?
Make booking?
Yes
Yes
Enter customer details and make payment
Bookings
Display customer contact details and payment

The web site will make use of both PHP processing which takes place on the server, and JavaScript processing which takes place on the local computer. We will need to examine ways of transferring variables between these two programming languages.

The name and map coordinates for each caravan are held as the attributes of a series of PHP **Caravan** objects. This information will also be needed by JavaScript in order to produce the caravan site map display. The PHP objects are converted into an equivalent set of JavaScript **Caravan** objects by creating JSON (JavaScript Object Notation) data. This is a block of text listing the names and values of the attributes for each object, as in this extract:

```
{"1":"locationID":"1","caravanNo":"1","caravanName":"Daffodil","x1":"215","y1":"138",
"x2":"218","y2":"174","x3":"195","y3":"176","x4":"192","y4":"140"},"2":
{"locationID":"3","caravanNo":"2","caravanName":"Buttercup","x1":"185","y1":"140","x2
":"187","y2":"177","x3":"164","y3":"179","x4":"161","y4":"142"},"3":
{"locationID":"4","caravanNo":"3","caravanName":"Snowdrop","x1":"153","y1":"144","x2"
:"157","y2":"180","x3":"133","y3":"183","x4":"131","y4":"145"},"4":
```

The JSON data is then used to construct an equivalent set of JavaScript **Caravan** objects.

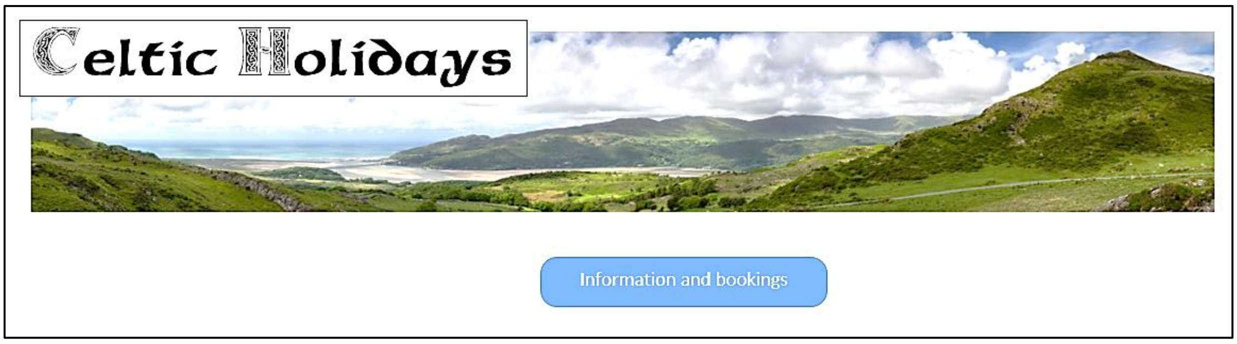
A central function of the web site will be the display of an interactive calendar showing bookings. PHP date functions will be used to produce this display by providing the day name for any date and the number of days in any month.

Method

Begin by setting up new folders with the name '**caravan**' on the local computer and on the server.

Create a header image for the home page using a desk-top publishing or graphics application such as Microsoft Word or Photoshop. This should consist of a landscape photograph with approximate dimensions of 1150 pixels by 200 pixels. The company name 'Celtic Holidays' can be added as in the example below.

Save the file as **title.jpg** and copy it to the server. All files for the project should be stored in the **caravan** folders on the local computer and on the server.



Create an image for a button with the caption 'Information and bookings'. Save the image as **button.png** and copy it to the server.

Open a new blank file. Add the HTML code below to create an introductory page.

```
 <html>
 <head>
    <title>Celtic Holidays</title>
 </head>
 <body>
   <img src ='title.jpg' >
   <br><br>
   <center>
    <a href='bookings.php?caravanNoWanted=1&monthWanted=6'>
    <img src='button.png'>
    </a>
 </body>
</html>
```

Save the file as **index.php** and copy it to the caravan folder on the server.

Run the website by entering the domain name for your site, followed by the directory **caravan**, e.g:

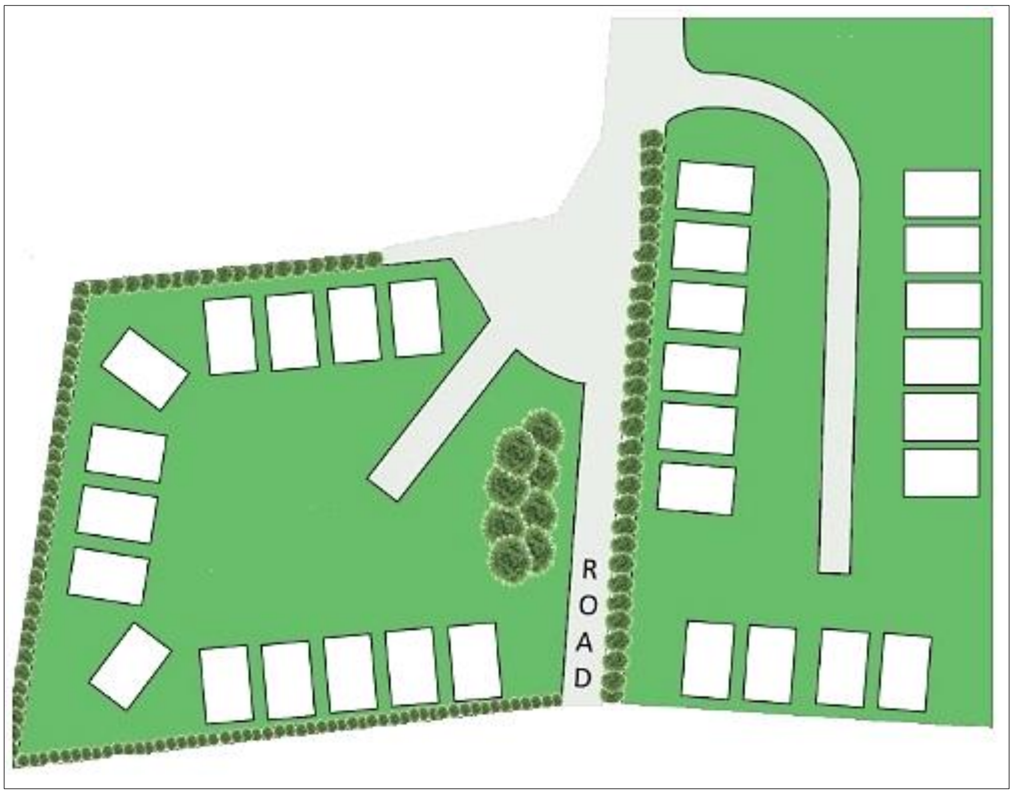
www.website.com/caravan

The page **index.php** will be load automatically as the default homepage for the site. Check that this is similar to the web page illustrated above.

Additional content can be added to the home page if required. This might include descriptions of holiday attractions in the area, and photographs of the caravan park and its facilities.

The button on the home page will lead to a booking page which we will develop next. This page is focussed around a plan of the caravan park with the individual caravans marked.

Begin by using a desk-top publishing or graphics application to create a site base map image similar to the one shown below. This should have a size of approximately 500 pixels by 400 pixels. Save the image as **map.jpg** and copy it to the server.



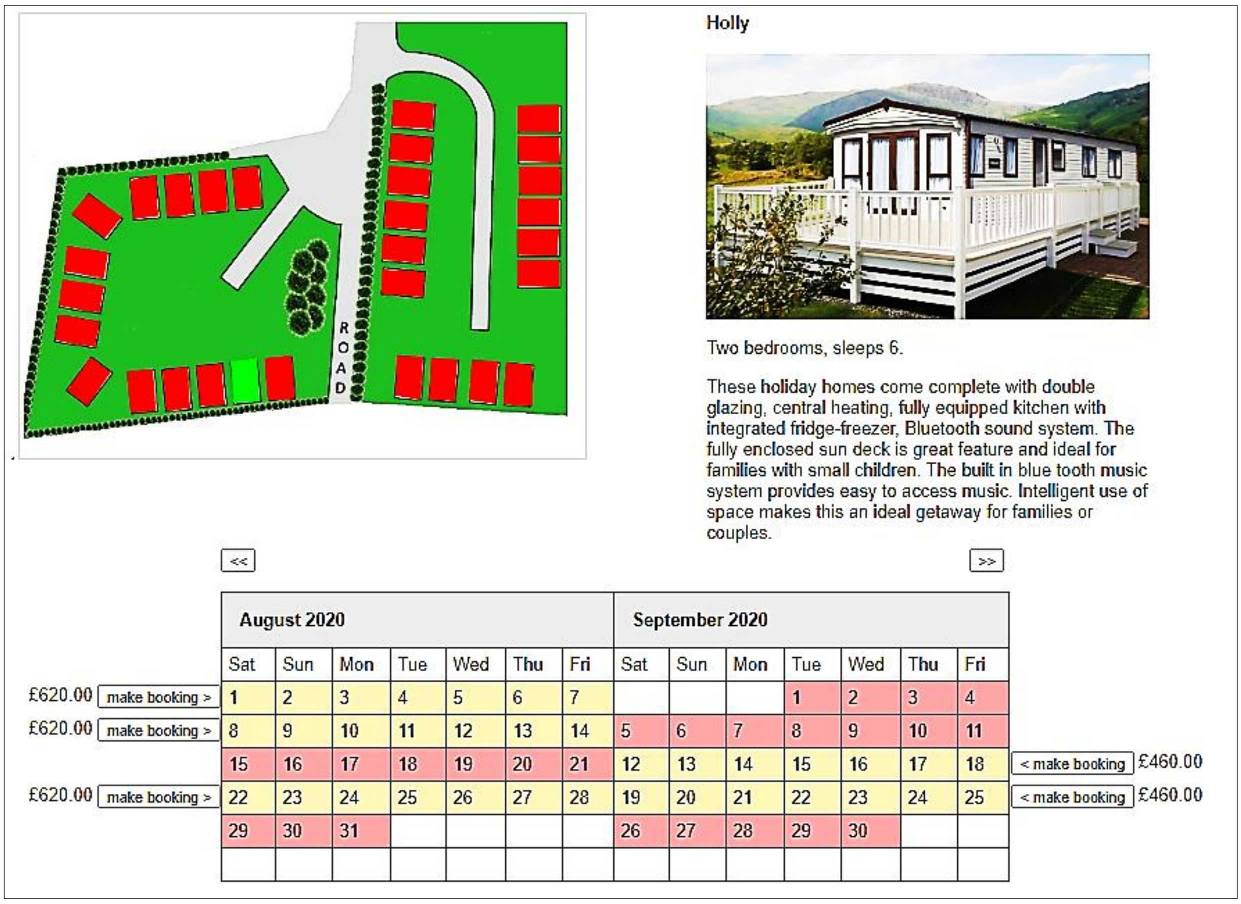
Open a blank file and add the lines of code below to produce a page to display the plan of the site.

```
<html>
 <head>
  <title> Celtic Holidays </title>
  <style>
    body
    {
           font-family: Arial, Helvetica, sans-serif;
           color: black;
    }
    p { font-size: 12pt;}
  </style>
 </head>
 <body>
  <table>
    <tr>
       <td>
           <img src="map.jpg" >
       </td>
    </tr>
   </table>
 </body>
</html>
```

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

Run the web site and click the 'Information and bookings' button on the home page. Check that the bookings.php page opens and the site plan is displayed.

The design and operation of the bookings page will be quite complex. The layout is illustrated in the screen print below.



Holly

Two bedrooms, sleeps 6.

These holiday homes come complete with double glazing, central heating, fully equipped kitchen with integrated fridge-freezer, Bluetooth sound system. The fully enclosed sun deck is great feature and ideal for families with small children. The built in blue tooth music system provides easy to access music. Intelligent use of space makes this an ideal getaway for families or couples.

	August 2020							September 2020							
	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	
£620.00 make booking >	1	2	3	4	5	6	7				1	2	3	4	
£620.00 make booking >	8	9	10	11	12	13	14	5	6	7	8	9	10	11	
	15	16	17	18	19	20	21	12	13	14	15	16	17	18	< make booking £460.00
£620.00 make booking >	22	23	24	25	26	27	28	19	20	21	22	23	24	25	< make booking £460.00
	29	30	31					26	27	28	29	30			

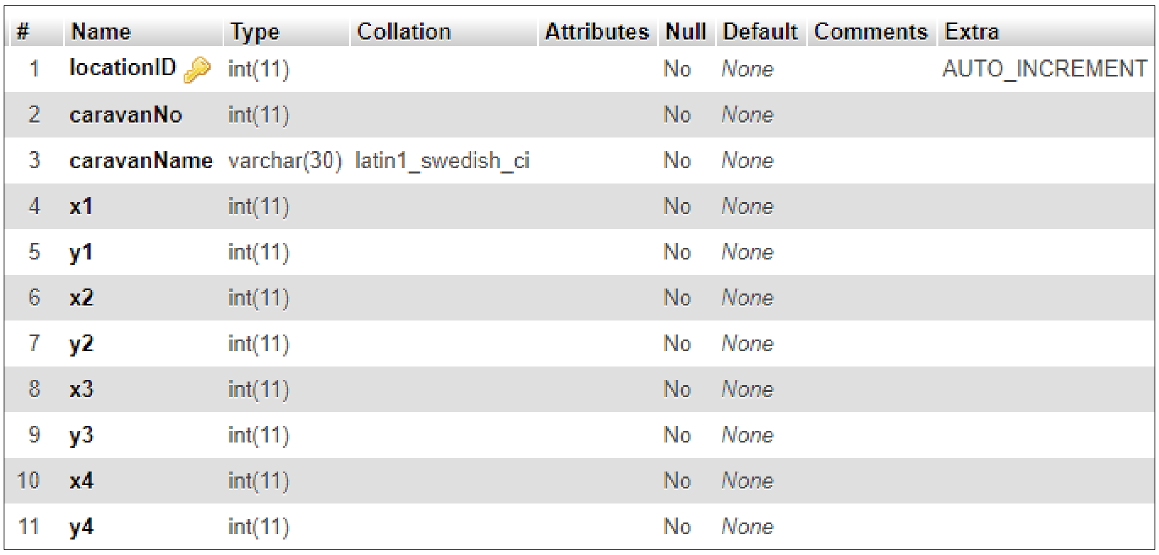
The user will be able to click the mouse on the site plan to select a particular caravan, which will be highlighted in green. The name of the caravan, along with a photograph and description will then be displayed. On the lower part of the page, a booking calendar will indicate the available weeks when the selected caravan may be booked, along with the hire charge. The calendar shows two consecutive months alongside each other. The month display may be changed by clicking the forwards '>>' and backwards '<<' buttons.

On clicking a 'make booking' button, the customer will be taken to another page where they will enter their contact details and make payment.

Log-in to the PHP MyAdmin web page for your database. Display the list of tables in the database and select the 'new' option. Further information about creating database tables is given in the **Hardware Store project** in Chapter 2.

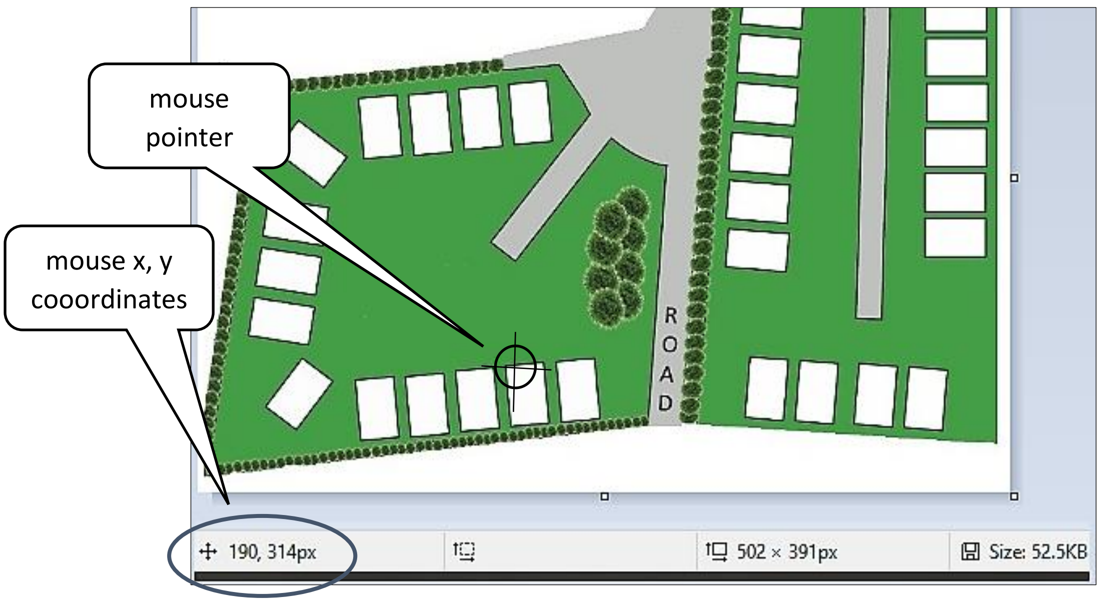
Set up a table with the name '**caravanLocation**'. This will contain the name and reference number of each caravan, along with the (x,y) pixel coordinates of its four corners on the site plan.

Add fields to the table as shown below. The field **caravanName** is of data type **vachar** with a length of 30 characters. All other fields are of **integer** data type. Select **locationID** to be the primary key, and set this to auto-increment as records are added to the table.

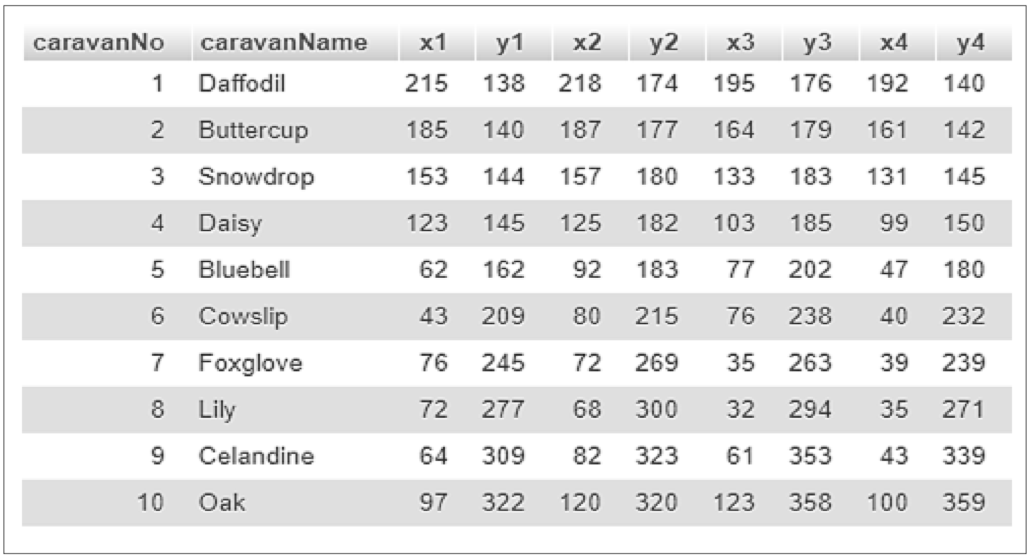


#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**locationID**	int(11)			No	*None*		AUTO_INCREMENT
2	**caravanNo**	int(11)			No	*None*		
3	**caravanName**	varchar(30)	latin1_swedish_ci		No	*None*		
4	**x1**	int(11)			No	*None*		
5	**y1**	int(11)			No	*None*		
6	**x2**	int(11)			No	*None*		
7	**y2**	int(11)			No	*None*		
8	**x3**	int(11)			No	*None*		
9	**y3**	int(11)			No	*None*		
10	**x4**	int(11)			No	*None*		
11	**y4**	int(11)			No	*None*		

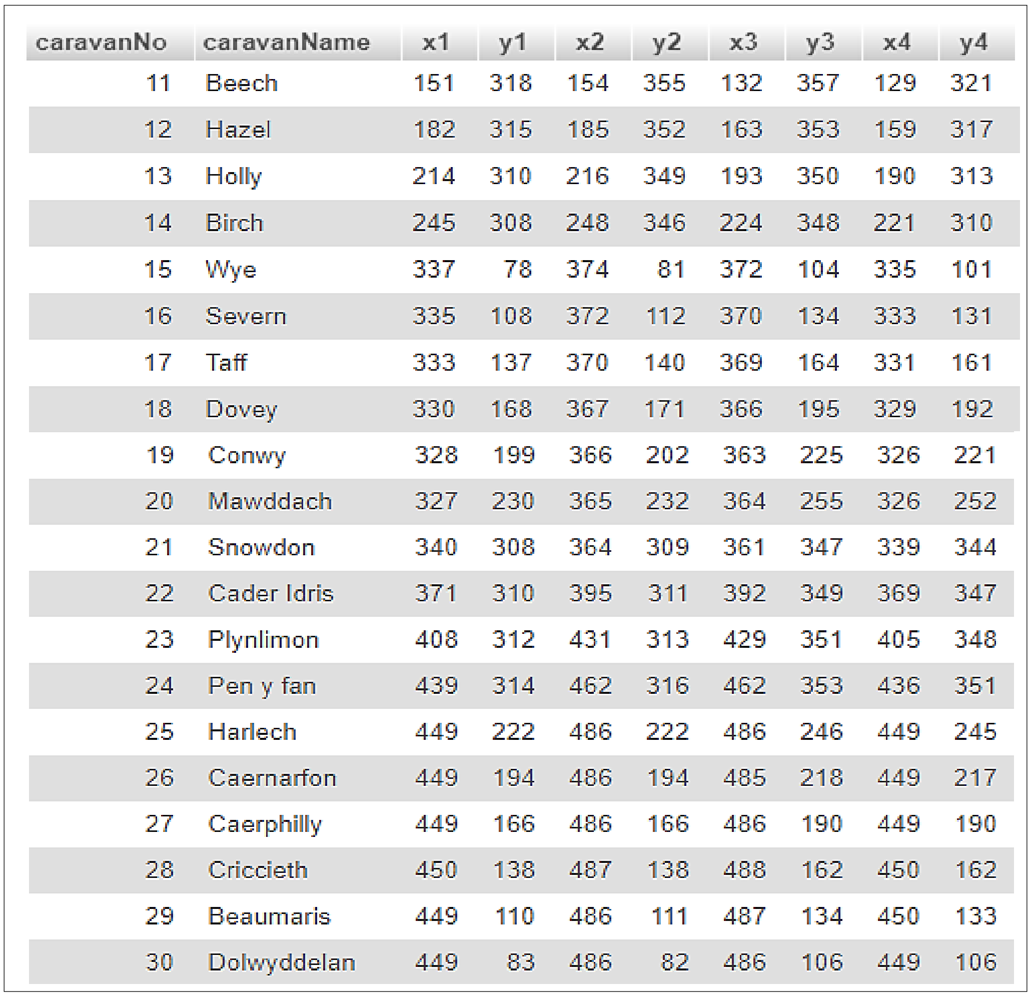
The pixel coordinates for each caravan can be found by loading the site plan image into a graphics program such as Microsoft Paint. As the mouse pointer is moved, the coordinates are displayed:



For the purpose of this example, the coordinates shown below will be used. This data should be entered into the **caravanLocation** table.



caravanNo	caravanName	x1	y1	x2	y2	x3	y3	x4	y4
1	Daffodil	215	138	218	174	195	176	192	140
2	Buttercup	185	140	187	177	164	179	161	142
3	Snowdrop	153	144	157	180	133	183	131	145
4	Daisy	123	145	125	182	103	185	99	150
5	Bluebell	62	162	92	183	77	202	47	180
6	Cowslip	43	209	80	215	76	238	40	232
7	Foxglove	76	245	72	269	35	263	39	239
8	Lily	72	277	68	300	32	294	35	271
9	Celandine	64	309	82	323	61	353	43	339
10	Oak	97	322	120	320	123	358	100	359



caravanNo	caravanName	x1	y1	x2	y2	x3	y3	x4	y4
11	Beech	151	318	154	355	132	357	129	321
12	Hazel	182	315	185	352	163	353	159	317
13	Holly	214	310	216	349	193	350	190	313
14	Birch	245	308	248	346	224	348	221	310
15	Wye	337	78	374	81	372	104	335	101
16	Severn	335	108	372	112	370	134	333	131
17	Taff	333	137	370	140	369	164	331	161
18	Dovey	330	168	367	171	366	195	329	192
19	Conwy	328	199	366	202	363	225	326	221
20	Mawddach	327	230	365	232	364	255	326	252
21	Snowdon	340	308	364	309	361	347	339	344
22	Cader Idris	371	310	395	311	392	349	369	347
23	Plynlimon	408	312	431	313	429	351	405	348
24	Pen y fan	439	314	462	316	462	353	436	351
25	Harlech	449	222	486	222	486	246	449	245
26	Caernarfon	449	194	486	194	485	218	449	217
27	Caerphilly	449	166	486	166	486	190	449	190
28	Criccieth	450	138	487	138	488	162	450	162
29	Beaumaris	449	110	486	111	487	134	450	133
30	Dolwyddelan	449	83	486	82	486	106	449	106

We will create a **Location class** to access data from the database table and form a link to the web page.

Open a blank text file. Add the code shown below which specifies the attributes for a **Location object.** Save the file as **Location.php** and copy it to the server.

```
<?
class Location
{
    public static $caravanCount = 0;
    public static $caravan = array();
    public $locationID;
    public $caravanNo;
    public $caravanName;
    public $x1; public $y1;
    public $x2; public $y2;
    public $x3; public $y3;
    public $x4; public $y4;

}
?>
```

Add a **constructor** method. As the attributes are private to each object, a set of **get()** methods will also be needed to allow access to these attributes from the main program. Please note that long lines of code marked by curved arrows should be entered by continuous typing without any line breaks.

```
    function __construct($locationID, $caravanNo, $caravanName, $x1, $y1,
                                                 $x2, $y2, $x3, $y3, $x4, $y4)
    {
        $this->locationID = $locationID;
        $this->caravanNo = $caravanNo;
        $this->caravanName = $caravanName;
        $this->x1 = $x1;    $this->y1 = $y1;
        $this->x2 = $x2;    $this->y2 = $y2;
        $this->x3 = $x3;    $this->y3 = $y3;
        $this->x4 = $x4;    $this->y4 = $y4;
    }

    public function getLocationID(){return $this->locationID;}
    public function getCaravanNo(){return $this->caravanNo;}
    public function getCaravanName(){return $this->caravanName;}

  }
?>
```

The next step is to add a method to the **Location.php** class file which will access the location table in the database and create a corresponding set of objects. Insert the **loadLocations( )** function at the end of the class file, as shown on the next page.

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

```
    public static function loadLocations()
    {
       include ('user.inc');
       $conn = new mysqli(localhost, $username, $password, $database);
       if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
       $query="SELECT * FROM caravanLocation";
       $result=mysqli_query($conn, $query);
       $num=mysqli_num_rows($result);
       mysqli_close($conn);
       $i=1;
       while ($i <= $num)
       {
              $row=mysqli_fetch_assoc($result);
              $locationID=$row["locationID"];
              $caravanNo=$row["caravanNo"];
              $caravanName=$row["caravanName"];
              $x1=$row["x1"];
              $y1=$row["y1"];
              $x2=$row["x2"];
              $y2=$row["y2"];
              $x3=$row["x3"];
              $y3=$row["y3"];
              $x4=$row["x4"];
              $y4=$row["y4"];
              $obj = new Location($locationID, $caravanNo, $caravanName,
                                 $x1, $y1,$x2, $y2, $x3, $y3, $x4, $y4);
              Location::$caravan[$i] = $obj;
              $i++;
       }
       Location::$caravanCount=$num;
       return $num;
    }
}
?>
```

Return now to the **bookings.php** web page file. Change the lines of code inside the <table>…</table> block.

```
      <body>
        <table>
         <tr>
            <td valign='top' height=480>
            <img id="map" width="2" height="2" src="map.jpg">
            <canvas id="myCanvas" width="510" height="400"
                                  style="border:1px solid #d3d3d3;">
            </canvas>
            </td>
         </tr>
         </table>
```

Add lines of code near the start of **bookings.php** as shown below. These will collect the number of the caravan and the number of the month to be displayed. To initialise the display, we previously specified **caravan 1** and **month 6** as part of the URL linked to the button on the home page.

```
      <html>
       <head>
        <title> Celtic Holidays </title>
        <?
          $caravanNoWanted=$_REQUEST['caravanNoWanted'];
          $monthWanted=$_REQUEST['monthWanted'];
        ?>
        <style>
         body
         {
            font-family: Arial, Helvetica, sans-serif;
            color: black;
```

Two blocks of code should now be added to **bookings.php**. The first block of PHP code accesses the Location class file and obtains a set of objects containing the names and locations of the caravans. These are then stored in JSON format. A <script> block then converts the JSON data to Javascript objects, and collects various other information needed to create the graphics display.

```
            </canvas>
            </td>
         </tr>
         </table>
         <?
            include ('Location.php');
            $count=Location::loadLocations();
            $caravanJSON = json_encode(Location::$caravan);
         ?>
         <script type="text/javascript">
            var caravan = <? echo $caravanJSON ?>;
            var count = <? echo $count ?>;
            var c = document.getElementById("myCanvas");
            var c2 = c.getContext("2d");
            var img = document.getElementById("map");
            var caravanNoWanted= <? echo $caravanNoWanted ?>;
         </script>
       </body>
```

Within the <script> block, add Javascript functions.

The **markCaravan( )** function will plot each caravan on the site plan as a red filled rectangle, with one specified caravan coloured instead in green. The rectangles are drawn using the x and y coordinates obtained from the database table.

The **onload( )** function will operate automatically when the page loads, and will in turn call the **markCaravan( )** function. The number of the caravan to be highlighted is specified as a parameter.

```
        var c2 = c.getContext("2d");
        var img = document.getElementById("map");
        var caravanNoWanted= <? echo $caravanNoWanted ?>;

        window.onload = function()
        {
             c2.drawImage(img, 0, 1);
             markCaravan(caravanNoWanted);
         }

         function markCaravan(caravanNoWanted)
         {
             var arrayLength = count;
             for (var i = 1; i <= arrayLength; i++)
             {
                 c2.fillStyle = '#FF0000';
                 if (caravan[i].caravanNo == caravanNoWanted)
                 {
                      c2.fillStyle = '#00FF00';
                      nameWanted=caravan[i].caravanName;
                 }
                 c2.beginPath();
                 c2.moveTo(caravan[i].x1, caravan[i].y1);
                 c2.lineTo(caravan[i].x2, caravan[i].y2);
                 c2.lineTo(caravan[i].x3, caravan[i].y3);
                 c2.lineTo(caravan[i].x4, caravan[i].y4);
                 c2.closePath();
                 c2.fill();
             }
         }

      </script>
      </body>
    </html>
```

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

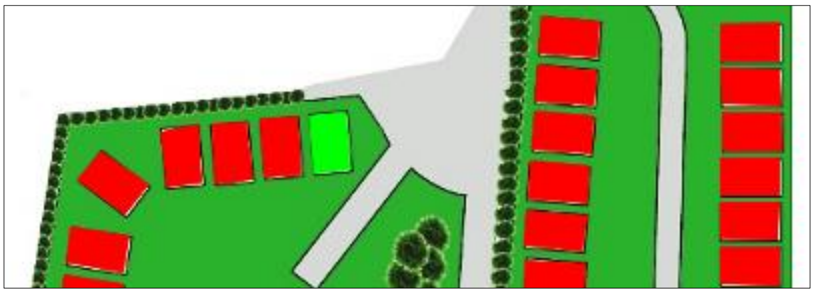
Before running the bookings page, a security file will be needed to authorise access to the on-line database. This has the format:

```
<?
   $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 the small file as **user.inc** and copy it to the server.

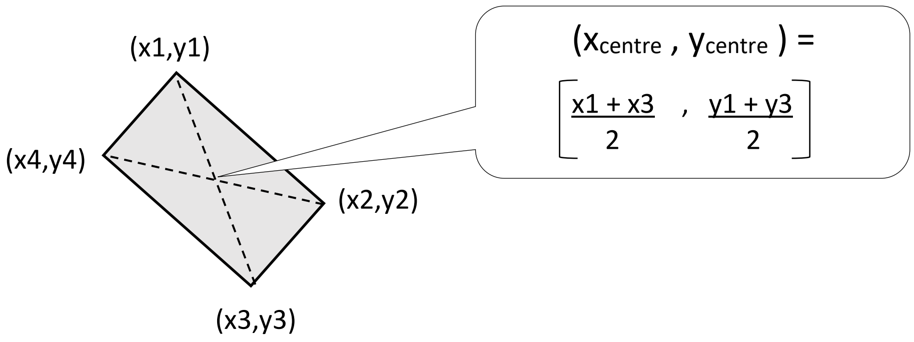
Run the website home page and click the 'bookings' button. Check that the site plan is displayed with most caravans coloured red and caravan 1 highlighted in green, as shown below.



The next step is to make the map display interactive, so that the user can select and highlight any caravan by clicking the mouse. Re-open the **bookings.php** file and modify the code within the **<canvas>** tag.

```
        <table>
           <tr>
              <td valign='top' height=480>
              <img id="map" width="2" height="2" src="map.jpg"  >
              <canvas id="myCanvas" width="510" height="400"
                style="border:1px solid #d3d3d3;" onmousedown="mouseDown()">
              </canvas>
              </td>
            </tr>
         </table>
```

When the mouse is pressed within the graphics area, a function **mouseDown( )** will be called. The function obtains the position of the mouse pointer relative to the top left corner of the map image. A loop then checks each of the JavaScript caravan objects in turn. The corner coordinates of the caravan are obtained, and the position of the rectangle centre is calculated at the intersection of the diagonals:


(x1,y1)
(x4,y4)
(x2,y2)
(x3,y3)
$(x_{centre}, y_{centre}) = \left[\frac{x1 + x3}{2}, \frac{y1 + y3}{2}\right]$

If the mouse pointer is within 20 pixels of the mid point, the caravan is selected for highlighting and its identification number is given as the parameter for the markCaravan( ) function.

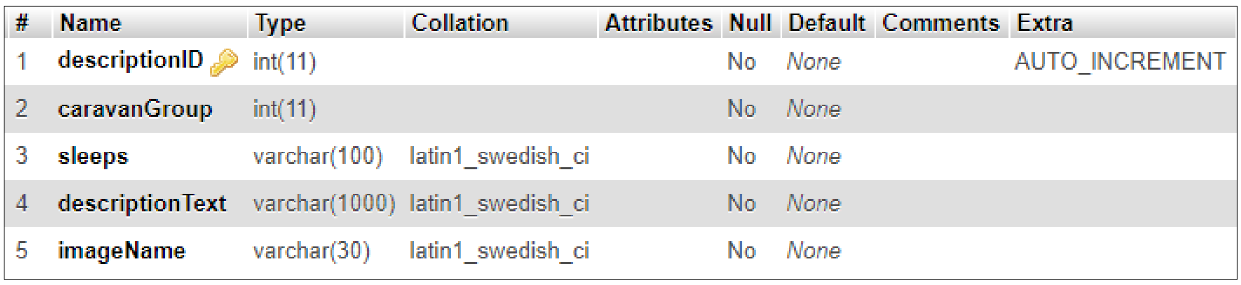
Add the **mouseDown( )** function shown below to the JavaScript code block within the **<script>** tags.

```
function mouseDown()
{
   var x = event.clientX;
   var y = event.clientY;
   var div = document.getElementById("myCanvas");
   var rect = div.getBoundingClientRect();
   var xoff = rect.left;
   var yoff = rect.top;
   x = (x - xoff);
   y = (y - yoff);
   var arrayLength = count;
   for (var i = 1; i <= arrayLength; i++)
   {
      x1=parseInt(caravan[i].x1);   y1=parseInt(caravan[i].y1);
      x2=parseInt(caravan[i].x2);   y2=parseInt(caravan[i].y2);
      x3=parseInt(caravan[i].x3);   y3=parseInt(caravan[i].y3);
      x4=parseInt(caravan[i].x4);   y4=parseInt(caravan[i].y4);
      mx=(x1+x3)/2;
      my=(y1+y3)/2;
      xdiff = Math.abs(x-mx);
      ydiff = Math.abs(y-my);
      if ((xdiff<20)&&(ydiff<20))
      {
          caravanNoWanted=caravan[i].caravanNo;
          markCaravan(caravanNoWanted);
          window.location = "bookings.php?caravanNoWanted="+
                      caravanNoWanted +"&monthWanted="+monthWanted;
       }
    }
}

</script>
</body>
```

Save the **bookings.php** file and copy it to the server. Run the program and check that each caravan can now be selected and appears highlighted in green.

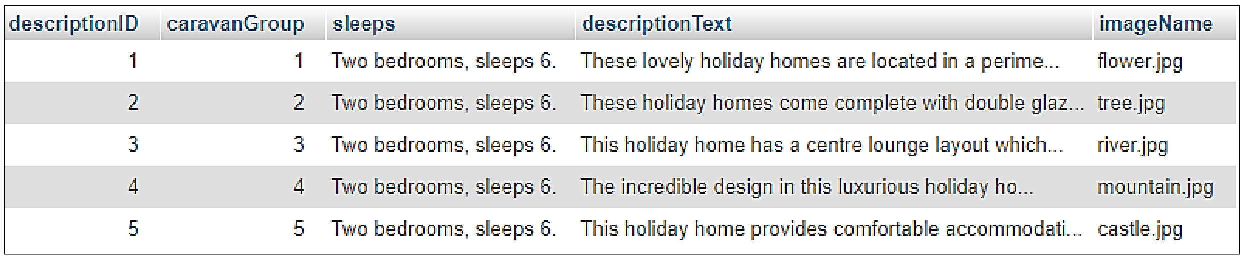
The next step is to add the photograph and description of the caravan selected. Data for this will be stored in a table in the database.

Go to the PHP MyAdmin web page for your database, list the tables, and select the 'new' option. Set up a table with the name **caravanDescription** and add fields as shown below. The integer field **descriptionID** should be specified as the primary key and set to automatically increment as records are added. The **caravanGroup** field is also of integer data type, whilst the remaining fields are of type **varchar**. The **sleeps** field has a size of 100 characters, **descriptionText** is 1,000 characters, and **imageName** is 30 characters.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**descriptionID** 🔑	int(11)			No	*None*		AUTO_INCREMENT
2	**caravanGroup**	int(11)			No	*None*		
3	**sleeps**	varchar(100)	latin1_swedish_ci		No	*None*		
4	**descriptionText**	varchar(1000)	latin1_swedish_ci		No	*None*		
5	**imageName**	varchar(30)	latin1_swedish_ci		No	*None*		

Obtain suitable picture images for each of the five caravan groups, and write paragraphs of descriptive text. Upload the image files to the server. Add the description and image file name to the database table for each of the caravan groups.



descriptionID	caravanGroup	sleeps	descriptionText	imageName
1	1	Two bedrooms, sleeps 6.	These lovely holiday homes are located in a perime...	flower.jpg
2	2	Two bedrooms, sleeps 6.	These holiday homes come complete with double glaz...	tree.jpg
3	3	Two bedrooms, sleeps 6.	This holiday home has a centre lounge layout which...	river.jpg
4	4	Two bedrooms, sleeps 6.	The incredible design in this luxurious holiday ho...	mountain.jpg
5	5	Two bedrooms, sleeps 6.	This holiday home provides comfortable accommodati...	castle.jpg

A Description class can now be created. This will allow the database records to be transferred to objects in PHP. These objects can in turn be converted to JavaScript objects for display on the web bookings page.

Open a blank file and add the **Description class** shown in the two boxes below. This begins by defining attributes which correspond with the fields of the **caravanDescription** database table. Methods are then included to download records from the database and create **Description** objects.

```
<?
class Description
{
    public static $groupCount = 0;
    public static $group = array();
    public $descriptionID;
    public $caravanGroup;
    public $sleeps;
    public $descriptionText;
    public $imageName;

    function __construct($descriptionID, $caravanGroup, $sleeps,
                                             $descriptionText, $imageName)
    {
        $this->descriptionID = $descriptionID;
        $this->caravanGroup = $caravanGroup;
        $this->sleeps = $sleeps;
        $this->descriptionText = $descriptionText;
        $this->imageName = $imageName;
    }

    public static function loadDescriptions()
    {
        include ('user.inc');
        $conn = new mysqli(localhost, $username, $password, $database);
        if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
        $query="SELECT * FROM caravanDescription";
        $result=mysqli_query($conn, $query);
        $num=mysqli_num_rows($result);
        mysqli_close($conn);
        $i=1;
        while ($i <= $num)
        {
            $row=mysqli_fetch_assoc($result);
            $descriptionID=$row["descriptionID"];
            $caravanGroup=$row["caravanGroup"];
            $sleeps=$row["sleeps"];
            $descriptionText=$row["descriptionText"];
```

```
                $imageName=$row["imageName"];
                $obj = new Description($descriptionID, $caravanGroup, $sleeps,
                                                   $descriptionText, $imageName);
               Description::$group[$i] = $obj;
                $i++;
            }
           Description::$groupCount=$num;
           return $num;
        }
     }
     ?>
```

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

As in the case of the caravan location data, we will load the **Description** objects in PHP and then convert them to JavaScript objects by means of JSON encoding. Re-open the **bookings.php** file and add the lines of code shown below.

```
   <?
      include ('Location.php');
      $count=Location::loadLocations();
      $caravanJSON = json_encode(Location::$caravan);
      include ('Description.php');
      $groupCount=Description::loadDescriptions();
      $descriptionJSON = json_encode(Description::$group);
   ?>
   <script type="text/javascript">
      var caravan = <? echo $caravanJSON ?>;
      var count = <? echo $count ?>;
      var description = <? echo $descriptionJSON ?>;
      var groupCount = <? echo $groupCount ?>;
      var monthWanted = <? echo $monthWanted ?>;
           var c = document.getElementById("myCanvas");
           var c2 = c.getContext("2d");
```

Go now to the beginning of the table and locate the <td> ...</td> block holding the map image. After this, insert program code to display the name of the caravan selected, its image and description.

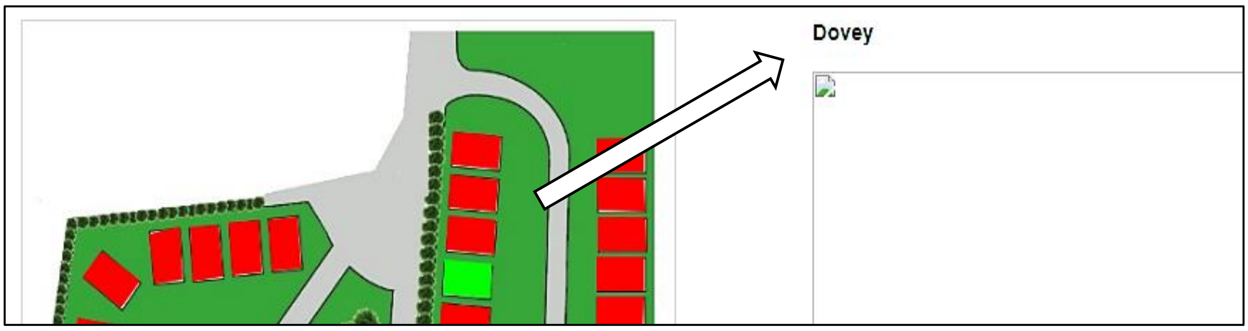
```
        <td valign='top' height=480>
         <img id="map" width="2" height="2" src="map.jpg"  >
         <canvas id="myCanvas" width="510" height="400"
               style="border:1px solid #d3d3d3;" onmousedown="mouseDown()">
         </canvas>
        </td>
        <td width=100></td>
        <td width = 400 style="vertical-align:top">
        <h3><p id="caravanName" ></p></h3>
         <img id="caravanImage" src=""  width = 400>
         <p id="sleeps" ></p>
         <p id="descriptionText" ></p></td>
     </tr>
```

```
        </table>
```

Finally, locate the JavaScript **markCaravan( )** function. At the end, add a line of code to set the caravan name which will be displayed.

```
         function markCaravan(caravanNoWanted)
         {
                .  .  .  .  .
             c2.lineTo(caravan[i].x4, caravan[i].y4);
             c2.closePath();
             c2.fill();
         }
         document.getElementById("caravanName").textContent = nameWanted;
      }
```

Save the **bookings.php** file and copy it to the server. Run the website and go to the bookings page. Check that clicking on any caravan causes the correct name to be displayed on the right of the screen.



Re-open the **bookings.php** file. Immediately after the 'caravan name' line, insert a block of code which will locate and display the appropriate photograph and description for the caravan group containing the selected caravan.

```
      c2.fill();
    }
    document.getElementById("caravanName").textContent = nameWanted;
    c=getCaravanGroup(caravanNoWanted);
    for (j=1;j<=groupCount;j++)
    {
       if (description[j].caravanGroup == c)
       {
         document.getElementById("caravanImage").src = description[j].imageName;
          document.getElementById("sleeps").textContent = description[j].sleeps;
         document.getElementById("descriptionText").textContent =
                                             description[j].descriptionText;
       }
    }
}
```

A final step is to add a JavaScript function which will use the caravan reference number to determine the reference number of the group to which it belongs.

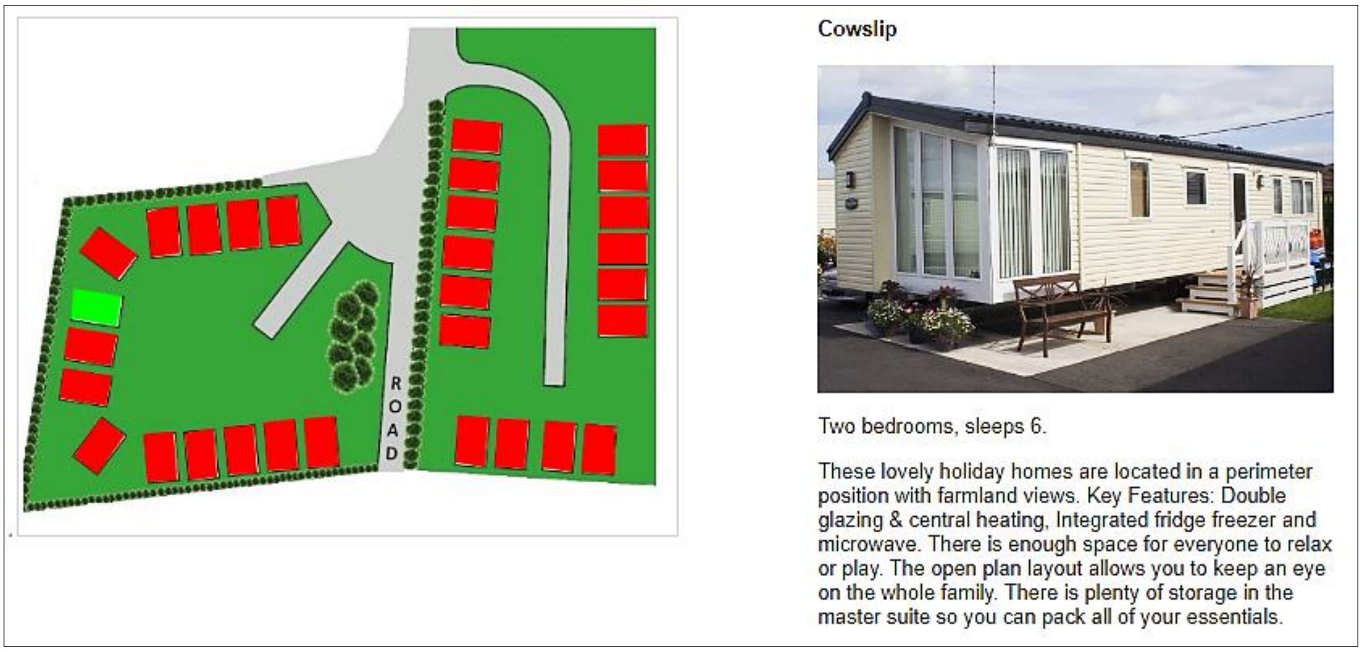
Immediately after the **markCaravan( )** function, add a **getCaravanGroup( )** function as shown below.

```
            document.getElementById("sleeps").textContent = description[j].sleeps;
            document.getElementById("descriptionText").textContent =
                                                   description[j].descriptionText;
         }
      }
   }

   function getCaravanGroup(caravanNoWanted)
   {
      var group=1;
      if (caravanNoWanted>9)
          group=2;
      if (caravanNoWanted>14)
          group=3;
      if (caravanNoWanted>20)
          group=4;
      if (caravanNoWanted>24)
          group=5;
      return group;
   }

   function mouseDown()
   {
      var x = event.clientX;
      var y = event.clientY;
```

Save the **bookings.php** file and copy it to the server. Run the website and go to the bookings page. Select caravans from each of the groups and check that the picture image and description text is displayed correctly, as in the illustration below.

If the text for any group fails to appear, a common problem is that an apostrophe symbol (') has been used in the description. This symbol is a control character and can affect the loading of data. A simple solution is to replace apostrophes in the database text with an alternative but very similar symbol (`) which is found on the top left hand key of most keyboards.

Before leaving this section, it is useful to note that the two object classes **Location** and **Description** have their attributes set as **public**, rather than private as in most object classes. This was necessary in order to allow access to the attributes when creating JSON data.



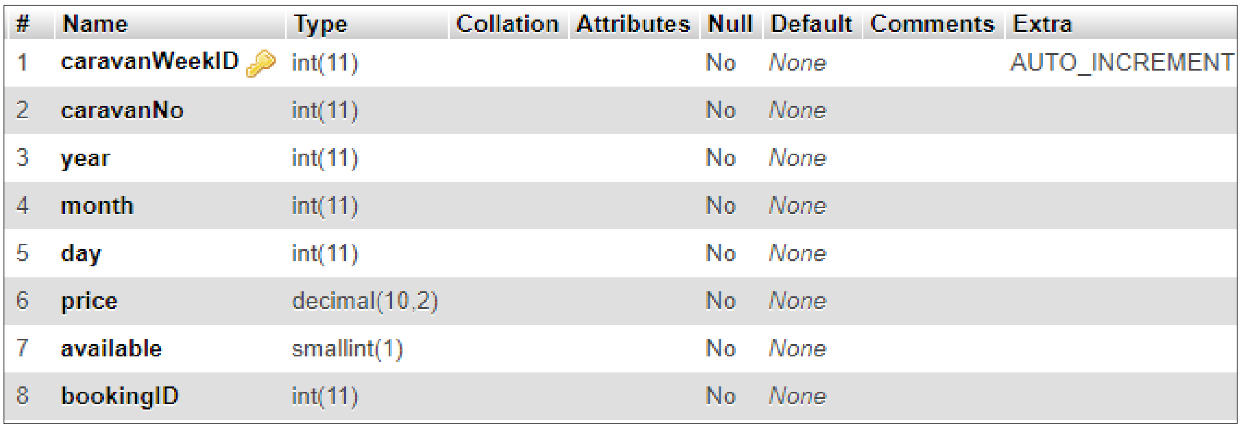
Cowslip

Two bedrooms, sleeps 6.

These lovely holiday homes are located in a perimeter position with farmland views. Key Features: Double glazing & central heating, Integrated fridge freezer and microwave. There is enough space for everyone to relax or play. The open plan layout allows you to keep an eye on the whole family. There is plenty of storage in the master suite so you can pack all of your essentials.

The next step is to work on the bookings calendar. Go to the PHP MyAdmin web site and create a new table with the name **caravanWeeks**. This will contain a record for each caravan for each week of the year. The **available** field will use a code number to indicate whether the caravan is already booked (0), available for hire (1), or unavailable for hire (2) during the specified week.

Add fields to the table as shown below. The caravanWeekID field should be specified as the primary key, and set to auto-increment as records are added. Other fields are of **integer** data type, with the exception of **available**, which can be set as a **small integer**, and **price** which is set as a **decimal** number displaying two decimal places.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	caravanWeekID	int(11)			No	*None*		AUTO_INCREMENT
2	caravanNo	int(11)			No	*None*		
3	year	int(11)			No	*None*		
4	month	int(11)			No	*None*		
5	day	int(11)			No	*None*		
6	price	decimal(10,2)			No	*None*		
7	available	smallint(1)			No	*None*		
8	bookingID	int(11)			No	*None*		

We will now produce a **CaravanWeek class** file to load records from the database and create a set of **CaravanWeek objects** which can be accessed by the booking calendar display.

Open a blank file and add the block of code below, being careful to avoid line breaks in long lines of code. This begins by defining the attributes for a **CaravanWeek** object, then provides a constructor method for the objects. Save the file as **CaravanWeek.php**.

```
<?
class CaravanWeek
{
   public static $weekCount;
   public static $week= array();
   private $caravanWeekID;
   private $caravanNo;
   private $year;
   private $month;
   private $day;
   private $price;
   private $available;
   private $bookingID;

   function __construct($caravanWeekID, $caravanNo, $year,
                       $month, $day, $price, $available, $bookingID)
   {
      $this->caravanWeekID = $caravanWeekID;
      $this->caravanNo = $caravanNo;
      $this->yearBeginning = $year;
      $this->monthBeginning = $month;
      $this->dayBeginning = $day;
      $this->price = $price;
      $this->available = $available;
      $this->bookingID = $bookingID;
   }
}
?>
```

Add a series of **get( )** methods which will allow access to the private object attributes from the main program.

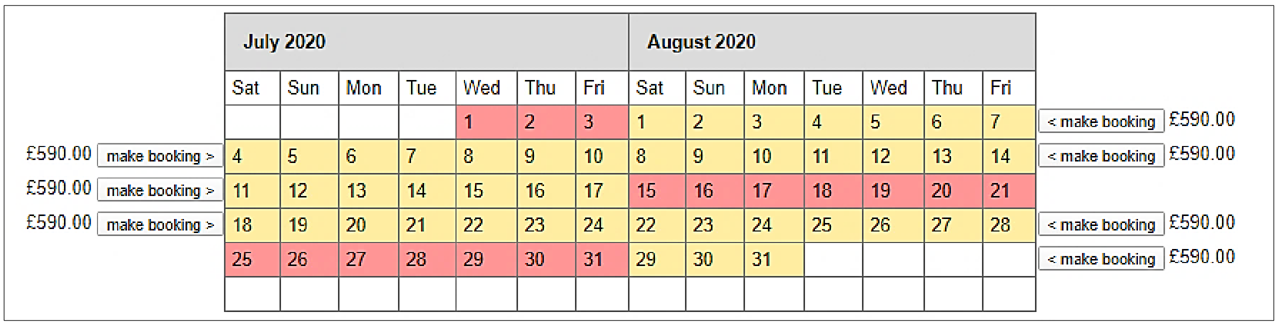
```
        public function getCaravanWeekID(){return $this->caravanWeekID;}
        public function getCaravanNo(){return $this->caravanNo;}
        public function getYearBeginning(){return $this->yearBeginning;}
        public function getMonthBeginning(){return $this->monthBeginning;}
        public function getDayBeginning(){return $this->dayBeginning;}
        public function getPrice(){return $this->price;}
        public function getAvailable(){return $this->available;}
        public function getBookingID(){return $this->bookingID;}

    }
    ?>
```

Add a **loadCaravanWeeks( )** method as show below. This will access the database and obtain the weekly booking records for a specified caravan.

```
public static function loadCaravanWeeks($caravanNoWanted)
{
       include ('user.inc');
       $conn = new mysqli(localhost, $username, $password, $database);
       if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
       $query="SELECT * FROM caravanWeeks WHERE caravanNo=".$caravanNoWanted;
       $result=mysqli_query($conn, $query);
       $num=mysqli_num_rows($result);
       mysqli_close($conn);
       $i=1;
       while ($i <= $num)
       {
           $row=mysqli_fetch_assoc($result);
           $caravanWeekID=$row["caravanWeekID"];
           $caravanNo=$row["caravanNo"];
           $yearBeginning=$row["year"];
           $monthBeginning=$row["month"];
           $dayBeginning=$row["day"];
           $price=$row["price"];
           $available=$row["available"];
           $bookingID=$row["bookingID"];
           $obj = new CaravanWeek($caravanWeekID, $caravanNo,
                      $yearBeginning,$monthBeginning, $dayBeginning,
                                   $price, $available, $bookingID);
           CaravanWeek::$week[$i] = $obj;
           $i++;
       }
       CaravanWeek::$weekCount=$num;
       return $num;
  }

}
?>
```

Save the **CaravanWeek.php** file.

It is intended that the booking system should display a calendar with available and unavailable weeks shown in different colours, as in the example below:



		July 2020							August 2020								
		Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri		
						1	2	3	1	2	3	4	5	6	7	< make booking	£590.00
£590.00	make booking >	4	5	6	7	8	9	10	8	9	10	11	12	13	14	< make booking	£590.00
£590.00	make booking >	11	12	13	14	15	16	17	15	16	17	18	19	20	21		
£590.00	make booking >	18	19	20	21	22	23	24	22	23	24	25	26	27	28	< make booking	£590.00
		25	26	27	28	29	30	31	29	30	31					< make booking	£590.00

Before setting up the calendar, we will create a set of test data. As this would be a large job to carry out manually, we will automate the process.

Add the **initialiseBookings( )** method to the **CaravanWeek** class file after the loadCaravanWeeks( ) method, as shown on the two pages below.

The program uses loops which repeat for each week of the year and for each caravan. The hire cost for the caravan is set according to the caravan group and the month of the year, with lower prices outside the main holiday period. The program then generates a random value between 0 and 2 for the **available** field, creating a random pattern of bookings through the year for each caravan. Finally, each weekly record is saved into the database table.

```
public static function initialiseBookings($year)
{
    include('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    for ($month=1; $month<=12; $month++)
    {
      $dt = $year."-".$month."-01";
      $firstDay= date("D", strtotime($dt));
      $lastDay= cal_days_in_month(CAL_GREGORIAN,$month,$year);
      for ($i=1; $i<=$lastDay; $i++)
      {
        $day = date("D", strtotime($year."-".$month."-".$i));
        if ($day == "Sat")
        {
           $weekBeginning = $year."-".$month."-".$i;
           for ($van=1; $van<=30; $van++)
           {
             if ($van<=9)
             {
                if ($month>=4 && $month<=8)
                   $price = 580.00;
                else
                   $price = 420.00;
             }
             else
             {
                if ($van<=14)
                {
                   if ($month>=4 && $month<=8)
                      $price = 620.00;
                   else
                      $price = 460.00;
                }
                else
                if ($van<=14)
                {
                   if ($month>=4 && $month<=8)
                      $price = 620.00;
                   else
                      $price = 460.00;
                }
                else
                {
                   if ($van<=20)
                   {
                      if ($month>=4 && $month<=8)
                         $price = 550.00;
                      else
                         $price = 430.00;
                   }
                   else
                   {
                      if ($van<=24)
                      {
                         if ($month>=4 && $month<=8)
                             $price = 780.00;
                         else
                             $price = 620.00;
                      }
                      else
                      {
                         if ($month>=4 && $month<=8)
                             $price = 590.00;
                         else
                             $price = 440.00;
                      }
                    }
                  }
                }
                $bookingCode=rand(0,2);
                $query="INSERT INTO caravanWeeks VALUES ('','$van','$year',
                                   '$month', '$i','$price','$bookingCode','0')";
                echo"<br>".$query;
                $result=mysqli_query($conn, $query);
              }
            }
         }
      }
      mysqli_close($conn);
   }
 }
?>
```

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

Re-open the **bookings.php** file. Add temporary lines of code at the start of the file which will run the **initialiseBookings( )** method when the page is loaded.

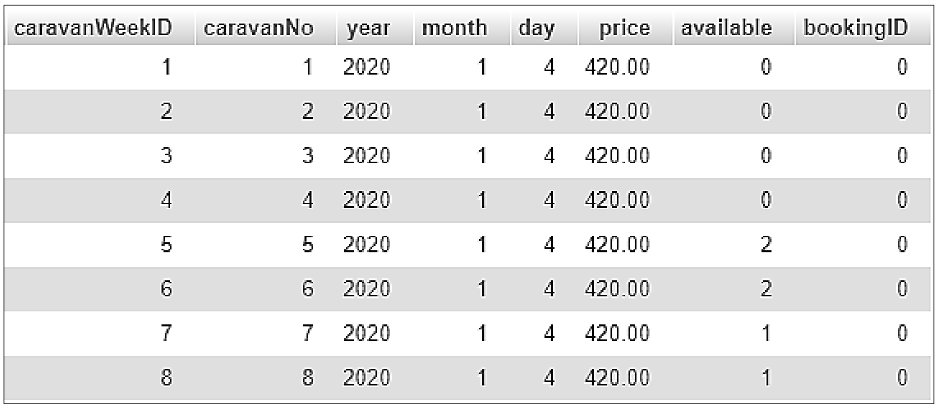
The method includes the booking calendar year number as a parameter. Set this to the year that you want the calendar to show, e.g. **initialiseBookings(2022)**

```
    <?
        include('CaravanWeek.php');
        CaravanWeek::initialiseBookings(2020);
    ?>

    <html>
     <head>
      <title> Celtic Holidays </title>
```

Set to the required calendar year, e.g. 2022

Save **bookings.php** and copy it to the server. Load the website homepage and click the 'bookings' button. Weekly records for each caravan and week of the year will be listed for test purposes, followed by the caravan site map. Close the page by clicking the cross symbol on the browser tab. It is important that the **initialiseBookings( )** method is run only once, to avoid creating duplicate records in the database.

Go to the PHP MyAdmin web page and open the **caravanWeeks** table. If all has gone well, this should now contain a full set of 52 weekly records from the beginning of January to the end of December for each of the 30 caravans, making 1,560 records in total. The **available** field should show a random pattern of 0, 1 and 2 code values.



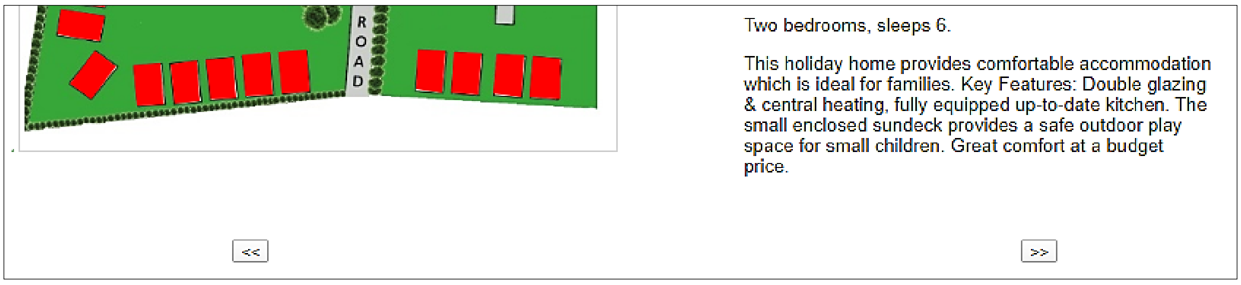
caravanWeekID	caravanNo	year	month	day	price	available	bookingID
1	1	2020	1	4	420.00	0	0
2	2	2020	1	4	420.00	0	0
3	3	2020	1	4	420.00	0	0
4	4	2020	1	4	420.00	0	0
5	5	2020	1	4	420.00	2	0
6	6	2020	1	4	420.00	2	0
7	7	2020	1	4	420.00	1	0
8	8	2020	1	4	420.00	1	0

Re-open the **bookings.php** file and remove the lines of PHP code shown in the rounded box at the top of this page. At this point it will be useful to add some further formatting to the **<style> ... </style>** block as shown in the two boxes below. This will be needed to produce the calendar display. The **background-color** commands specify different colour shading for booked, available and unavailable weeks.

```
   p { font-size: 12pt;}
   table.cal {
       border-collapse: collapse;
   }
   th.cal, td.cal, td.av, td.bk {
       border: 1px solid gray;
   }
   th.cal {
       background-color: #cfcfcf;
       color: black;
       padding: 15px;
       text-align: left;
   }
  </style>
```

```
      th.cal {
          background-color: #cfcfcf;
          color: black;
          padding: 15px;
          text-align: left;
      }

      td.cal {
         padding: 5px;
         text-align: left;
         background-color: #ffffff;
      }
      td.av {
         padding: 5px;
         text-align: left;
         background-color: #ffe699;
      }
      td.bk {
         padding: 5px;
         text-align: left;
         background-color: #ff9090;
      }
    </style>
```

We will begin the calendar display by creating **forward >>** and **back <<** buttons below the map and caravan description, to allow the calendar months to be selected.



Two bedrooms, sleeps 6.

This holiday home provides comfortable accommodation which is ideal for families. Key Features: Double glazing & central heating, fully equipped up-to-date kitchen. The small enclosed sundeck provides a safe outdoor play space for small children. Great comfort at a budget price.

Add lines of program code to **bookings.php** as shown below.

```
                window.location = "bookings.php?caravanNoWanted="+caravanNoWanted
                                                     +"&monthWanted="+monthWanted;
             }
          }
      }
      </script>

      <?
         echo"<form method=post action='bookings.php?caravanNoWanted=".
                          $caravanNoWanted."&monthWanted=".$monthWanted."'>";
      ?>
      <table width=900>
      <tr>
        <td width=320>
        <td width=560><input type=submit name='changeMonth' value='<<'>
        <td width=560>
        <td><input type=submit name='changeMonth' value='>>'>
     </table>
     </form>

   </body>
  </html>
```

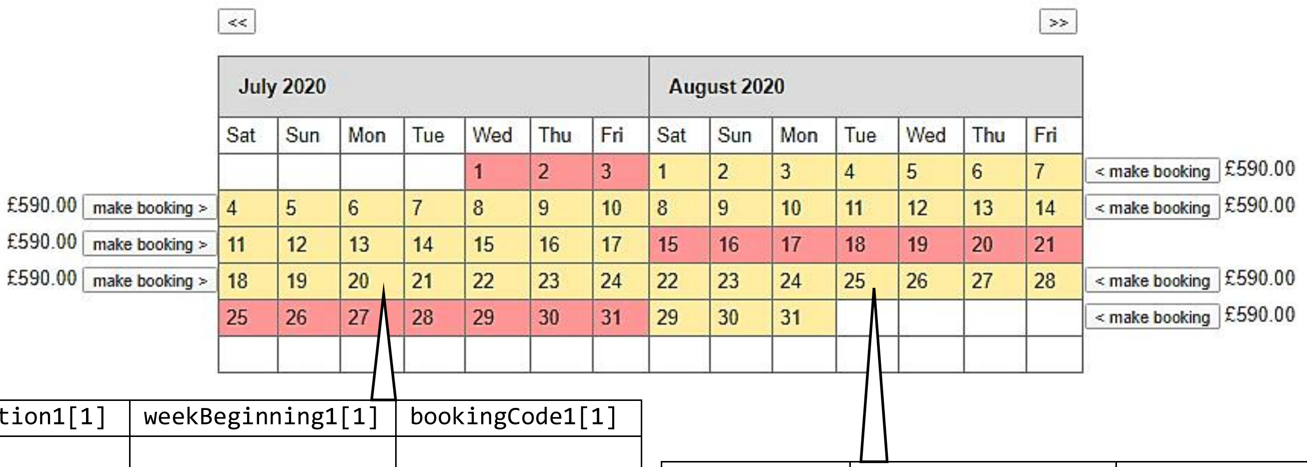
Save the **bookings.php** file and copy it to the server. Run the website and go to the bookings page. Check that the **forward** and **back** buttons are displayed.

Re-open the **bookings.php** file. Add lines of code after the **</table>** tag as shown below.

```
        <input type=submit name='changeMonth' value='<<'>
        <td width=560>
          <td><input type=submit name='changeMonth' value='>>'>
      </table>
      <?
        include('CaravanWeek.php');
        $weekCount=CaravanWeek::loadCaravanWeeks($caravanNoWanted);
        $p=0;
        $q=0;
        for($i=1; $i<=$weekCount; $i++)
        {
           $yearFound = CaravanWeek::$week[$i]->getYearBeginning();
           $monthFound = CaravanWeek::$week[$i]->getMonthBeginning();
           $dayFound = CaravanWeek::$week[$i]->getDayBeginning();
           $available = CaravanWeek::$week[$i]->getAvailable();
           $price = CaravanWeek::$week[$i]->getPrice();
           $weekBeginning=$dayFound."-".$monthFound."-".$yearFound;
           $nextMonth=intval($monthWanted+1);
        }
      ?>
     </form>
```

This block of code opens the **CaravanWeek** class file and runs the **loadCaravanWeeks( )** method to load the 52 weekly booking objects for the selected caravan for the current year. A loop then operates for each week, using **get( )** methods to obtain the attribute values from the objects.

The next step is to create arrays which hold the hire charge **caption[ ]**, start date **weekBeginning[ ]**, and availability **bookingCode[ ]** for each of the calendar rows:



<< >>

		July 2020							August 2020								
		Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri		
						1	2	3	1	2	3	4	5	6	7	< make booking	£590.00
£590.00	make booking >	4	5	6	7	8	9	10	8	9	10	11	12	13	14	< make booking	£590.00
£590.00	make booking >	11	12	13	14	15	16	17	15	16	17	18	19	20	21		
£590.00	make booking >	18	19	20	21	22	23	24	22	23	24	25	26	27	28	< make booking	£590.00
		25	26	27	28	29	30	31	29	30	31					< make booking	£590.00

caption1[1]	weekBeginning1[1]	bookingCode1[1]
	. . . . . .	
caption1[6]	weekBeginning1[6]	bookingCode1[6]

caption2[1]	weekBeginning2[1]	bookingCode2[1]
	. . . . . .	
caption2[6]	weekBeginning2[6]	bookingCode2[6]

The program reads the booking records in weekly order from the beginning of January to the end of December. If the week belongs entirely in the first selected month, as in the case of 4 July in the calendar above, the values are entered into the first set of arrays. If the week belongs entirely in the second selected month, as in the case of 8 August, the values are entered into the second set of arrays.

The situation is a little more complex in the case of weeks which are only partly in a selected month, as in the case of week 1 on the July calendar which actually began on 27 June. To allow for this, the program remembers the previous week's data, so that the booking code can be correctly recorded if a partial week is encountered at the start of the month. Add lines of code to process the bookings, as shown below.

```
        $price = CaravanWeek::$week[$i]->getPrice();
        $weekBeginning=$dayFound."-".$monthFound."-".$yearFound;
        $nextMonth=intval($monthWanted+1);
        if($monthFound==$monthWanted)
        {
           if (($dayFound>1)&&($dayFound<8))
           {
              $p++;
              $weekBeginning1[$p]=$previousWeekBeginning;
              $caption1[$p]=$previousPrice;
              $bookingCode1[$p]=$previousAvailable;
           }
         }
         if ($monthFound==$monthWanted)
         {
              $p++;
              $weekBeginning1[$p]=$weekBeginning;
              $caption1[$p]=$price;
              $bookingCode1[$p]=$available;
         }
         if($monthFound==$nextMonth)
         {
            if (($dayFound>1)&&($dayFound<8))
            {
               $q++;
               $weekBeginning2[$q]=$previousWeekBeginning;
               $caption2[$q]=$previousPrice;
               $bookingCode2[$q]=$previousAvailable;
            }
         }
         if ($monthFound==$nextMonth)
         {
              $q++;
              $weekBeginning2[$q]=$weekBeginning;
              $caption2[$q]=$price;
              $bookingCode2[$q]=$available;
         }
         $previousWeekBeginning=$weekBeginning;
         $previousAvailable=$available;
         $previousPrice=$price;
      }
      $dt = $yearFound."-".$monthWanted."-01";
      $firstDay1= date("D", strtotime($dt));
      $lastDay1= cal_days_in_month(CAL_GREGORIAN,$monthWanted,$yearFound);
      $dt = $yearFound."-".$nextMonth."-01";
      $firstDay2= date("D", strtotime($dt));
      $lastDay2= cal_days_in_month(CAL_GREGORIAN,$nextMonth,$yearFound);
  ?>
```

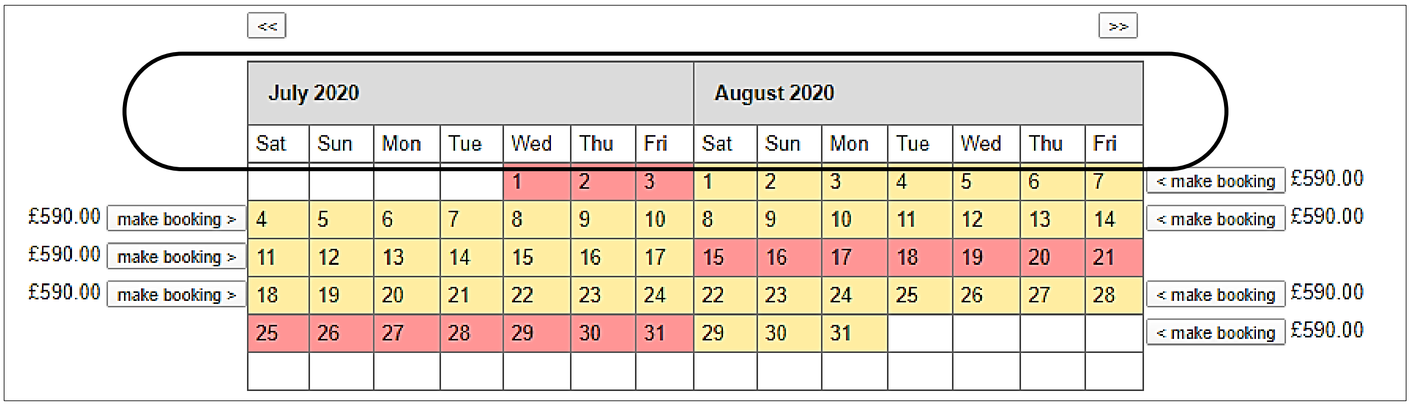
At the end of this section, lines of code have been added to determine the day name for the first day of the month, and the number of days in the month. In the case of the calendar displayed above, the results returned will be:

July 2020	first day = Wed	last day = 31
August 2020	first day = Sat	last day = 31

The next task is to update the months displayed when either the **forward** or **back** button is clicked. Go to the beginning of the **bookings.php** file and add lines of code to do this.

```
<html>
 <head>
  <title> Celtic Holidays </title>
  <?
    $caravanNoWanted=$_REQUEST['caravanNoWanted'];
    $monthWanted=$_REQUEST['monthWanted'];
    $changeMonth=$_REQUEST['changeMonth'];
    if ($changeMonth=='>>')
    {
        $monthWanted++;
          if ($monthWanted>11)
             $monthWanted=11;
    }
    if ($changeMonth=='<<')
    {
        $monthWanted--;
          if ($monthWanted<1)
             $monthWanted=1;
    }
  ?>
  <style>
```

We will now continue to develop the calendar display, beginning with the month and day headings.



		July 2020							August 2020								
		Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri		
						1	2	3	1	2	3	4	5	6	7	< make booking	£590.00
£590.00	make booking >	4	5	6	7	8	9	10	8	9	10	11	12	13	14	< make booking	£590.00
£590.00	make booking >	11	12	13	14	15	16	17	15	16	17	18	19	20	21		
£590.00	make booking >	18	19	20	21	22	23	24	22	23	24	25	26	27	28	< make booking	£590.00
		25	26	27	28	29	30	31	29	30	31					< make booking	£590.00

Two month headings will be needed, along with the sequence of days from Saturday to Friday within each booking week.

Near the end of the **bookings.php** file, insert the block of program code shown below. This begins with a function to convert the month number into the equivalent month name. Loops are then used to display the month and day headings.

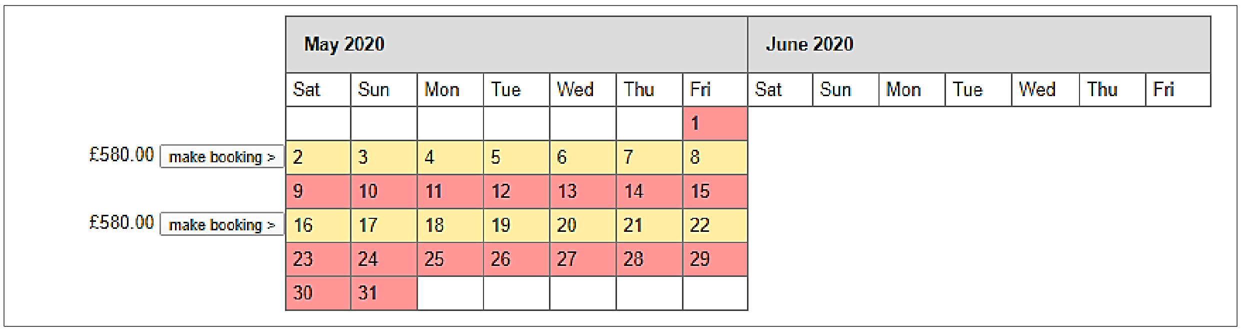
```
        $dt = $yearFound."-".$nextMonth."-01";
        $firstDay2= date("D", strtotime($dt));
        $lastDay2= cal_days_in_month(CAL_GREGORIAN,$nextMonth,$yearFound);
    ?>
    </form>
    <?
       function getMonthName($month)
       {
          switch($month)
          {
             case 1:$monthName='January';break;
             case 2:$monthName='February';break;
             case 3:$monthName='March';break;
             case 4:$monthName='April';break;
             case 5:$monthName='May';break;
             case 6:$monthName='June';break;
             case 7:$monthName='July';break;
             case 8:$monthName='August';break;
             case 9:$monthName='September';break;
             case 10:$monthName='October';break;
             case 11:$monthName='November';break;
             case 12:$monthName='December';break;
           }
           return $monthName;
       }
       echo"<form method=post action='customerDetails.php?caravanNoWanted="
                                                .$caravanNoWanted."'>";
       echo"<table class='cal' width=1100>";
       echo"<tr><td>";
       $firstMonth=getMonthName($monthWanted);
       $nextMonth=intval($monthWanted+1);
       $secondMonth=getMonthName($nextMonth);
       $year = CaravanWeek::$week[1]->getYearBeginning();
       echo"<th class='cal' colspan=7 >".$firstMonth." ".$year."</th>";
       echo"<th class='cal' colspan=7 >".$secondMonth." ".$year."</th><th>";
       $dayName[1]='Sat';   $dayName[2]='Sun';
       $dayName[3]='Mon';   $dayName[4]='Tue';
       $dayName[5]='Wed';   $dayName[6]='Thu';
       $dayName[7]='Fri';
       echo"<tr><td>";
       for($n=1;$n<=2;$n++)
       {
          for($d=1;$d<=7;$d++)
          {
               echo"<td class='cal' width=50>".$dayName[$d];
          }
       }
       echo"</form>";
       ?>
   </body>
```

Save the **bookings.php** file and copy it to the server. Run the website and go to the bookings page. Check that the calendar headings are displayed correctly.

Re-open the **bookings.php** file and add lines of program to display the day entries for the first month.

```
   for($n=1;$n<=2;$n++)

   {
      for($d=1;$d<=7;$d++)
      {
         echo"<td class='cal' width=50>".$dayName[$d];
      }
   }
   $count1=0;
   $count2=0;
   $display1=false;
   $display2=false;
   for ($w=1;$w<=6;$w++)
   {
      echo"<tr>";
      if ($bookingCode1[$w]==1)
         echo"<td align=right width=320> £".$caption1[$w].
          " <button name='week' value='$weekBeginning1[$w]£$caption1[$w]'>
                                               make booking > </button>";
      else
      echo"<td width=320>";
      for ($d=1;$d<=7;$d++)
      {
         if ( $firstDay1==$dayName[$d])
             $display1=true;
         if ($count1>=$lastDay1)
             $display1=false;
         if ($display1==true)
         {
             $count1++;
             if ($bookingCode1[$w]==1)
                 echo"<td class='av' width=50>".$count1;
             else
                 if (($bookingCode1[$w]==0)||($bookingCode1[$w]==2))
                     echo"<td class='bk' width=50>".$count1;
                 else
                     echo"<td class='cal' width=50>".$count1;
         }
         else
             echo "<td class='cal' width=50>&nbsp; ";
      }
   }
   echo"</form>";
   ?>
 </body>
</html>
```

Save the **bookings.php** file and copy it to the server. Run the website and go to the bookings page. The left-hand month calendar should now be displayed.



	May 2020							June 2020						
	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri
							1							
£580.00 make booking >	2	3	4	5	6	7	8							
	9	10	11	12	13	14	15							
£580.00 make booking >	16	17	18	19	20	21	22							
	23	24	25	26	27	28	29							
	30	31												

Six rows are provided in the calendar to allow for a possible maximum of six separate weeks, as in the example above. Day numbering begins on the starting day of the month, which in this case is a Friday. Numbering of the cells continues until the final day of the month is reached: in this case, day 31.

Each booking week begins on a Saturday. **Availability** data is used to set the colour background for the week, and is accessed from the array created earlier. In the case of the first week actually beginning in the previous month, the availability data from this earlier Saturday date will be used to set the background colour.

Where a week is available, the hire charge is shown and a button is provided to make a booking. We will link the button to another web page where the customer will enter their contact details and make payment.

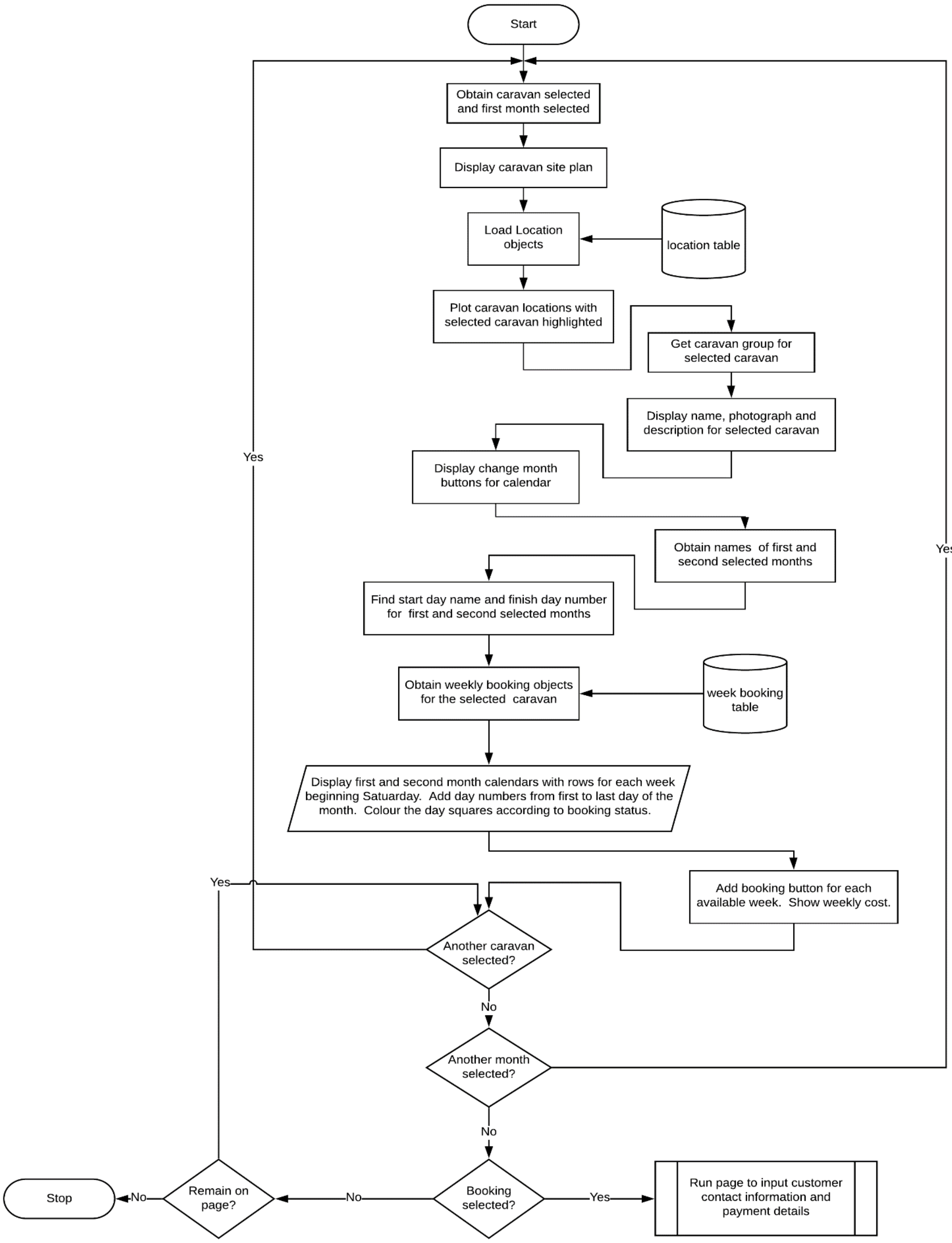
Check that the **backward<<** and **forward>>** buttons allow the month to be changed. Weeks will be shown randomly as available or unavailable. The earliest month which can be reached should be January, and the latest month November.

Re-open the **bookings.php** file and add the similar block of code listed below to create the right-hand calendar display.

```
                    if (($bookingCode1[$w]==0)||($bookingCode1[$w]==2))
                         echo"<td class='bk' width=50>".$count1;
                     else
                         echo"<td class='cal' width=50>".$count1;
             }
             else
                  echo "<td class='cal' width=50>&nbsp; ";
        }
        for ($d=1;$d<=7;$d++)
        {
             if ( $firstDay2==$dayName[$d])
                  $display2=true;
             if ($count2>=$lastDay2)
                  $display2=false;
             if ($display2==true)
             {
                  $count2++;
                  if ($bookingCode2[$w]==1)
                       echo"<td class='av' width=50>".$count2;
                  else
                      if (($bookingCode2[$w]==0)||($bookingCode2[$w]==2))
                         echo"<td class='bk' width=50>".$count2;
                  else
                         echo"<td class='cal' width=50>".$count2;
             }
             else
                  echo "<td class='cal' width=50>&nbsp; ";
         }
         if ($bookingCode2[$w]==1)
             echo"<td width=320><button name='week'
                     value='$weekBeginning2[$w]£$caption2[$w]'>
                            make booking </button>  £".$caption2[$w];
         else
             echo"<td width=320>";
    }
    echo"</form>";
    ?>
  </body>
```

Save the **bookings.php** file and copy it to the server. Run the website and go to the bookings page. Check that both calendar months are now displayed, and that the month can be changed by means of the forward and back buttons. Click on different caravans on the site map. The pattern of bookings should be randomly different for each caravan.

A flow chart summarising the sequence of events during a booking is given below.


Start
Obtain caravan selected and first month selected
Display caravan site plan
Load Location objects
location table
Plot caravan locations with selected caravan highlighted
Get caravan group for selected caravan
Display name, photograph and description for selected caravan
Display change month buttons for calendar
Obtain names of first and second selected months
Find start day name and finish day number for first and second selected months
Obtain weekly booking objects for the selected caravan
week booking table
Display first and second month calendars with rows for each week beginning Satuarday. Add day numbers from first to last day of the month. Colour the day squares according to booking status.
Add booking button for each available week. Show weekly cost.
Another caravan selected?
Yes
No
Another month selected?
Yes
No
Booking selected?
Yes
No
Run page to input customer contact information and payment details
Remain on page?
Yes
No
Stop

We can now move ahead to produce a customer booking page. Open a blank file and add the program code below.

```
<html>
<head>
   <title> Celtic Holidays </title>
   <style>
      body {font-family: Arial, Helvetica, sans-serif; color: black;}
   </style>
 </head>
 <body>
   <?
     $text=$_REQUEST['week'];
     $data=explode("£",$text);
     $weekBeginning=$data[0];
     $price=$data[1];
     $caravanNo=$_REQUEST['caravanNoWanted'];
     echo"<form method=post action='confirm.php?caravan=$caravanNo
                                                 &week=$weekBeginning'>";
   ?>
   <table cellpadding=4>
   <tr>
       <td colspan=3><h3>Booking</td></tr>
   <tr><td><td width=200>Caravan:
       <td>
       <?
          include ('Location.php');
          Location::loadLocations();
          $caravanName=Location::getName($caravanNo);
          echo $caravanName;
       ?>
    <tr><td><td width=200>Week beginning:
       <td>
       <?
          echo $weekBeginning;
       ?>
    </table>
    </form>
 </body>
 </html>
```

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

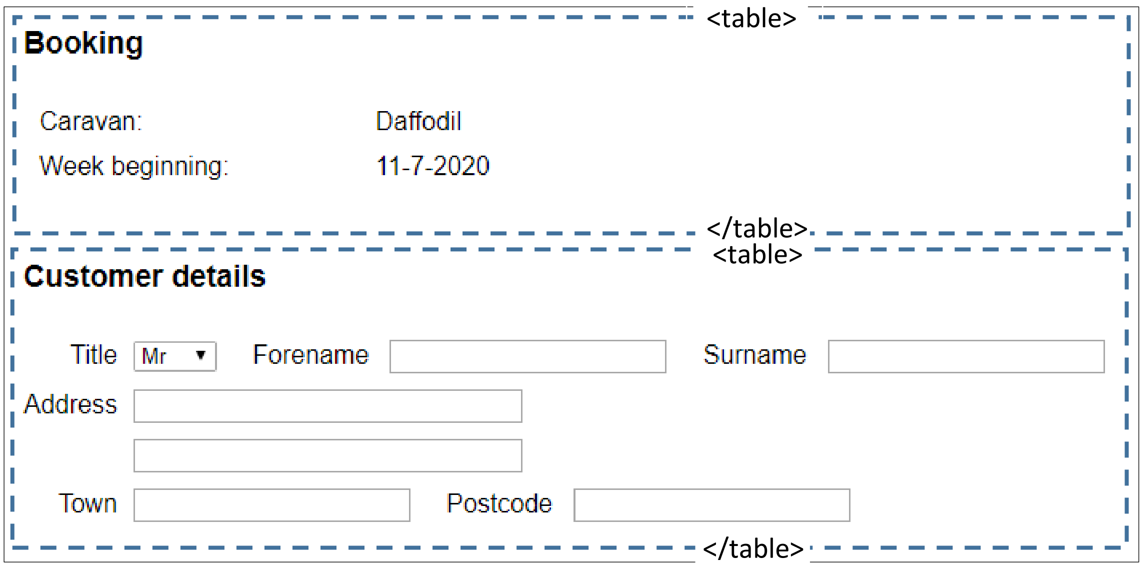
The name of the caravan should be displayed on the customer booking page, rather than its reference number. We will add methods to the **Location** class file to obtain the caravan name and number. Open the **Location.php** file and add the **getName( )** method shown below.

```
   public static function getName($numberWanted)
   {
        for ($i=1;$i<=Location::$caravanCount;$i++)
        {
            if (Location::$caravan[$i]->caravanNo == $numberWanted)
             {
                 $nameWanted=Location::$caravan[$i]->caravanName;
             }
        }
        return $nameWanted;
   }
}
?>
```

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

Run the website. Go to the booking page. Select a caravan and calendar month, then click the 'make booking' button for an available week. Check that the customer details page opens, and that the caravan name and booking week are displayed correctly.

We will now add input boxes for the customer's name and address, as shown below. Note that two tables are used for the **booking** and **customer details** sections of the page.



Re-open the **customerDetails.php** file and add the lines of code shown in the two boxes below.

```
   <tr>
      <td><td width=200>Week beginning:
      <td>
      <?
          echo $weekBeginning;
      ?>
   </table>
   <table cellpadding=4>
   <tr>
      <td colspan=3><br><h3>Customer details</td></tr>
   <tr>
      <td align='right'>Title
      <td>
      <?
         $t[0]='Mr';
         $t[1]='Mrs';
         $t[2]='Miss';
         $t[3]='Ms';
         $t[4]='Dr';
         echo"<select name='title'>";
         for ($i=0; $i<=4; $i++)
         {
            if ($title == $t[$i])
               echo"<option selected>".$t[$i];
            else
               echo"<option>".$t[$i];
         }
         echo"</select>";
```

```
          }
          echo"</select>";
      ?>
        &nbsp;&nbsp;&nbsp;
        Forename
        &nbsp;
      <?
        echo"<input type=text name=forename id=forename value='$forename'>";
      ?>
        &nbsp;&nbsp;&nbsp;
        Surname
        &nbsp;
      <?
        echo"<input type=text name=surname id=surname value='$surname'>";
      ?>
   </table>
   </form>
</body>
</html>
```

Save the **customerDetails.php** file and copy it to the server. Run the website, go to the bookings page and click a 'make booking' button. Check that the customer name input boxes are displayed correctly, including a drop-down selection for 'Title'.
Re-open the **customerDetails.php** file. Add the further lines of code shown below which input the customer's address.

```
      <?
        echo"<input type=text name=surname id=surname value='$surname'>";
      ?>
      <tr>
          <td align='right'>Address
          <td>
          <?
             echo"<input type=text name=address1 size=30 value='$address1'>";
          ?>
      <tr>
          <td><td>
          <?
             echo"<input type=text name=address2 size=30 value='$address2'>";
          ?>
      <tr>
          <td align='right'>Town
          <td>
          <?
             echo"<input type=text name=town value='$town'>";
          ?>
          &nbsp;&nbsp;&nbsp;
          Postcode
          &nbsp;
          <?
             echo"<input type=text name=postcode value='$postcode'>";
          ?>
    </table>
    </form>
  </body>
  </html>
```

Save the **customerDetails.php** file and copy it to the server. Run the website and make a booking. Check that the address input boxes are displayed correctly.

Notice that non-breaking space characters **(&nbsp;**) have been used to separate the input boxes and captions. Multiple blank spaces in HTML code are normally ignored, but space characters allow multiple spaces to be included in the page layout.

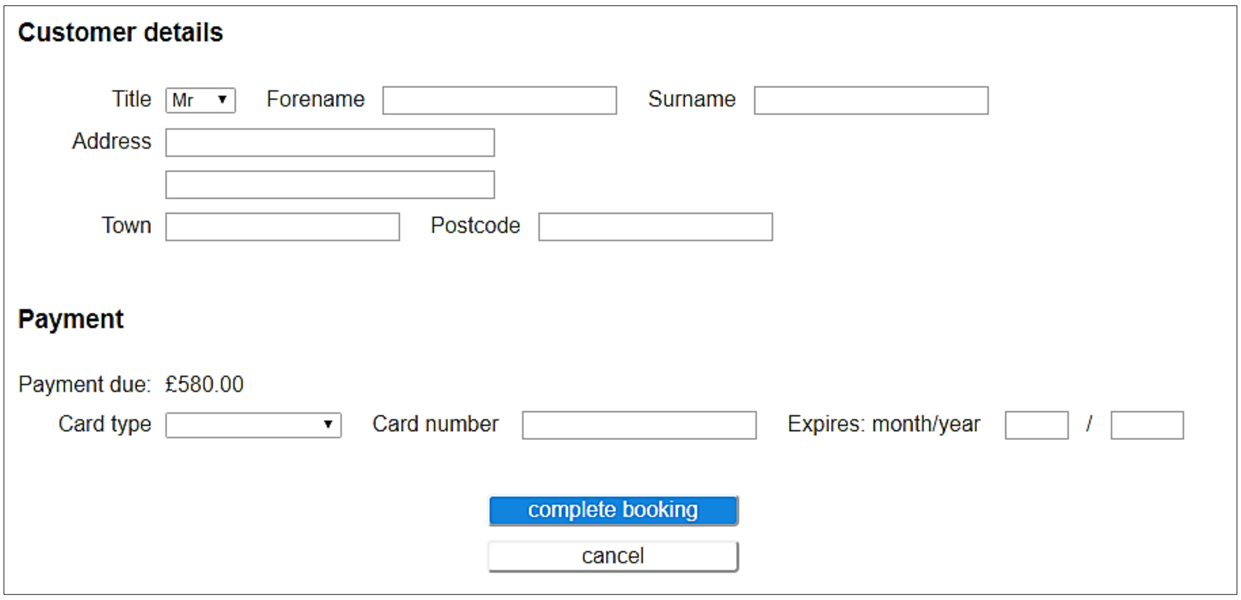
Re-open the **customerDetails.php** file. The final section of the page will display the hire charge for the caravan week selected and request the user to enter payment details. Add the lines of code shown below.

```
        Postcode
        &nbsp;
        <?
           echo"<input type=text name=postcode value='$postcode'>";
        ?>
        <tr>
           <td colspan=3><br><h3>Payment</td></tr>
        <tr>
           <td align='right'>Payment due:
           <td>
           <?
              echo "£".$price;
              echo"<input type=hidden name='paymentAmount' value='".$price."'>";
           ?>
        <tr>
           <td align='right'>Card type
           <td>
           <?
              $t[0]='';
              $t[1]='Visa Credit';
              $t[2]='Visa Debit';
              $t[3]='Mastercard Credit';
              $t[4]='Mastercard Debit';
              echo"<select name='cardType'>";
              for ($i=0; $i<5; $i++)
              {
                 if ($cardType == $t[$i])
                    echo"<option selected>".$t[$i];
                 else
                    echo"<option>".$t[$i];
              }
              echo"</select>";
           ?>
           &nbsp;&nbsp;&nbsp; Card number &nbsp;&nbsp;
           <?
              echo"<input type=text name='cardNumber' value='$cardNumber'>";
           ?>
           &nbsp;&nbsp;&nbsp; Expires: month/year &nbsp;&nbsp;
           <?
             echo"<input type=text name='expireMonth' size=2 value='$expireMonth'>";
           ?>
             &nbsp; / &nbsp;
           <?
             echo"<input type=text name='expireYear' size=3 value='$expireYear'>";
           ?>
        </table>
        </form>
      </body>
```

To finish the page display, add the lines of code below to create two buttons labelled '**complete booking**' and '**cancel**'.

```
          <?
             echo"<input type=text name='expireYear' size=3 value='$expireYear'>";
          ?>

      </table>
      <p><br>
      <button style='position:absolute;left:360px; font-size:16px;
                   border-radius: 4px;width: 186px;background-color: #1184DF;
                                         color: white'>complete booking</button>
    </form>
      <br>
      <form method='post' action='index.php'>
      <button style='position:absolute;left:360px; font-size: 16px;
                   border-radius: 4px;width: 186px;background-color: white;
                                                  color: black'>cancel</button>
   </form>
  </body>
```

Save the **customerDetails.php** file and copy it to the server. Run the website and make a booking. Check that all input boxes are displayed correctly, and that the two buttons appear at the bottom of the page.



Customer details

Title Mr ▾ Forename Surname

Address

Town Postcode

Payment

Payment due: £580.00

Card type ▾ Card number Expires: month/year /

complete booking

cancel

Clicking 'cancel' will return the user to the home page. The next step is to program the 'complete booking' button. This will carry out some error checking before saving the booking record.

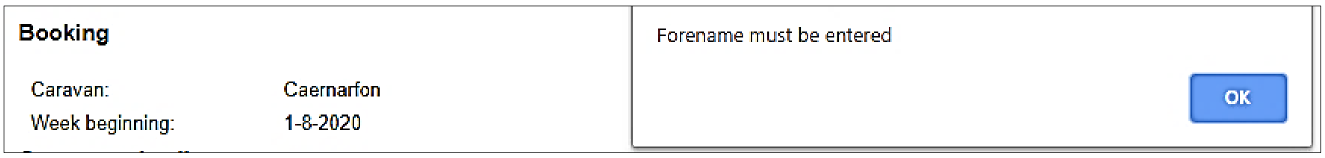
Re-open the **customerDetails.php** file. Modify the **<form>** command to run a JavaScript function **checkInput( )**:

```
       $weekBeginning=$data[0];
       $price=$data[1];
       $caravanNo=$_REQUEST['caravanNoWanted'];
       echo"<form method=post action='confirm.php?caravan=$caravanNo&week
                     =$weekBeginning' onsubmit='return checkInput()'>";
    ?>
```

Add a **<script>** block containing this function.

```
        $weekBeginning=$data[0];
        $price=$data[1];
        $caravanNo=$_REQUEST['caravanNoWanted'];
        echo"<form method=post action='confirm.php?caravan=$caravanNo
                        &week=$weekBeginning' onsubmit='return checkInput()'>";
    ?>
    <script>
        function checkInput()
        {
           forename = document.getElementById("forename").value;
           surname = document.getElementById("surname").value;
           var error=false;
           var n = forename.length;
           if (n<1)
           {
               alert("Forename must be entered");
               error=true;
           }
           n = surname.length;
           if (n<1)
           {
               alert("Surname must be entered");
               error=true;
           }
           if (error==true)
               result=false;
           else
               result = true;
           return result;
        }
    </script>
    <table cellpadding=4>
    <tr>
       <td colspan=3><h3>Booking</h3></td></tr>
```

Save the **customerDetails.php** file and copy it to the server. Run the website and make a booking. Click the 'complete booking' button without entering any data. Warning messages should appear to indicate that Forename and Surname entries are missing. Additional validation could be added if required.


Booking

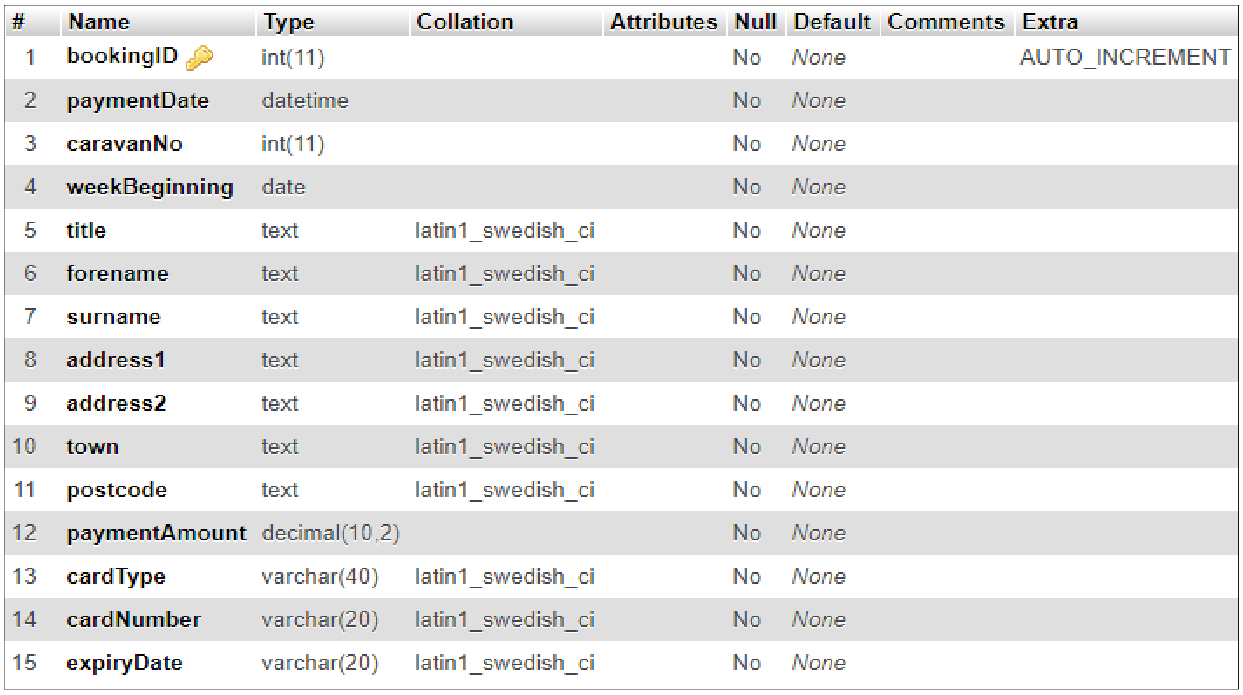
Caravan: Caernarfon

Week beginning: 1-8-2020

Forename must be entered

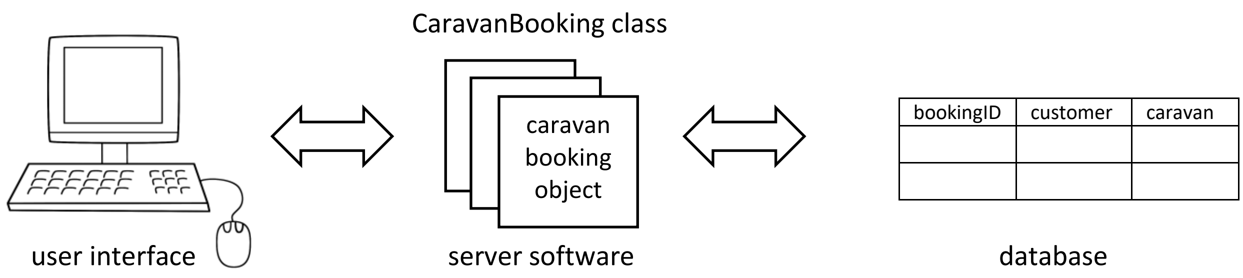
The next step is to create a database table to store bookings. Go to the PHP MyAdmin web site and log-in to the database. List the existing database tables and select the 'new' option.

Set up a table with the name '**caravanBooking**'. Add fields with the names and data types sown below. Set the bookingID field to auto-increment as records are added.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	bookingID	int(11)			No	*None*		AUTO_INCREMENT
2	paymentDate	datetime			No	*None*		
3	caravanNo	int(11)			No	*None*		
4	weekBeginning	date			No	*None*		
5	title	text	latin1_swedish_ci		No	*None*		
6	forename	text	latin1_swedish_ci		No	*None*		
7	surname	text	latin1_swedish_ci		No	*None*		
8	address1	text	latin1_swedish_ci		No	*None*		
9	address2	text	latin1_swedish_ci		No	*None*		
10	town	text	latin1_swedish_ci		No	*None*		
11	postcode	text	latin1_swedish_ci		No	*None*		
12	paymentAmount	decimal(10,2)			No	*None*		
13	cardType	varchar(40)	latin1_swedish_ci		No	*None*		
14	cardNumber	varchar(20)	latin1_swedish_ci		No	*None*		
15	expiryDate	varchar(20)	latin1_swedish_ci		No	*None*		

As with previous tables, we will create a class file to act as a link to the web page.



Open a blank file and add lines of program code as shown below. These begin by defining the attributes of a **CaravanBooking object**, corresponding to the fields of the caravanBooking table. A public static array **$booking[ ]** will identify individual objects.

```
<?
class CaravanBooking
{
  public static $booking = array();
  public static $bookingCount;
  private $bookingID;
  private $paymentDate;
  private $caravanNo;
  private $weekBeginning;
  private $title;
  private $forename;
  private $surname;
  private $address1;
  private $address2;
  private $town;
  private $postcode;
  private $paymentAmount;
  private $cardType;
  private $cardNumber;
  private $expiryDate;

}
?>
```

Add a **constructor** method.

```
public function __construct($bookingID, $paymentDate, $caravanNo,$weekBeginning,
                      $title,$forename, $surname, $address1, $address2, $town,
                 $postcode,$paymentAmount, $cardType, $cardNumber, $expiryDate)
{
   $this->bookingID = $bookingID;
   $this->paymentDate = $paymentDate;
   $this->caravanNo = $caravanNo;
   $this->weekBeginning = $weekBeginning;
   $this->title = $title;
   $this->forename = $forename;
   $this->surname = $surname;
   $this->address1 = $address1;
   $this->address2 = $address2;
   $this->town = $town;
   $this->postcode = $postcode;
   $this->paymentAmount = $paymentAmount;
   $this->cardType = $cardType;
   $this->cardNumber = $cardNumber;
   $this->expiryDate = $expiryDate;
}
}
?>
```

Save the file as **CaravanBooking.php**.

We will now produce a web page which will be loaded when the customer clicks the 'complete booking' button. This will save the booking details into the database, and display a message confirming the booking. Open a blank file and add the program code shown below. This begins by obtaining the customer name, address and payment entries from the input boxes on the customerDetails page.

```
<html>
<head>
   <title> Celtic Holidays </title>
   <style>
     body {font-family: Arial, Helvetica, sans-serif;  color: black; }
   </style>
</head>
<body>
  <p><img src="title.jpg">
  <?
     $title = $_REQUEST["title"];
     $forename = $_REQUEST["forename"];
     $surname = $_REQUEST["surname"];
     $address1 = $_REQUEST["address1"];
     $address2 = $_REQUEST["address2"];
     $town = $_REQUEST["town"];
     $postcode = $_REQUEST["postcode"];
     $email = $_REQUEST["email"];
     $cardType = $_REQUEST["cardType"];
     $cardNumber = $_REQUEST["cardNumber"];
     $expireMonth = $_REQUEST["expireMonth"];
     $expireYear = $_REQUEST["expireYear"];
     $expiryDate = $expireMonth."/".$expireYear;
     $paymentAmount = $_REQUEST["paymentAmount"];
  ?>
</body>
</html>
```

Add a table:

```
        $expiryDate = $expireMonth."/".$expireYear;
        $paymentAmount = $_REQUEST["paymentAmount"];
        ?>
        <table cellpadding=10>
        <form method=post action='index.php'>
        <tr>
            <th>Your booking is confirmed</th>
        </tr>
        </form>
        </table>
       </body>
     </html>
```

Save the file as **confirm.php**.

To record a booking, two actions must be carried out:

- The customer name, address and payment details must be saved into the **caravanBooking** table in the database. A **bookingID** will be allocated.
- The corresponding record must be updated in the **caravanWeeks** table to indicate that the caravan is now booked for the selected week. The **bookingID** will be added to the **caravanWeeks** record, so that the customer details are linked to the caravan hire.

Open the **CaravanBooking.php** file and add a **makeBooking( )** method.

```
    public static function makeBooking($caravanSelected,$weekSelected,$title,
                        $forename,$surname, $address1,$address2,$town,$postcode,
                                  $paymentAmount,$cardType,$cardNumber,$expiryDate)
    {
        include('user.inc');
        $paymentDate = date("Y-m-d H:i:00");
        $conn = new mysqli(localhost, $username, $password, $database);
        if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
        $query="INSERT INTO caravanBooking VALUES('$bookingID','$paymentDate',
                          '$caravanSelected','$weekSelected','$title','$forename',
                         '$surname','$address1','$address2', '$town',  '$postcode',
                        '$paymentAmount','$cardType', '$cardNumber', '$expiryDate')";
        $result=mysqli_query($conn, $query);
        $bookingID = mysqli_insert_id($conn);
        CaravanWeek::confirmBooking($caravanSelected, $weekSelected, $bookingID);
        mysqli_close($conn);
    }

  }
  ?>
```

Save **CaravanBooking.php** and copy it to the server. Notice that the **makeBooking( )** method calls a **confirmBooking( )** method in the CaravanWeek class. We will add this method next.

Open **CaravanWeek.php** and add the **confirmBooking( )** method shown below. This takes the weekBeginning date and splits it into day, month and year. These date values are combined with the caravan identification number to select the required caravan week record.

```
 public static function confirmBooking($caravanSelected, $weekSelected, $bookingID)
 {
    $values=explode("-",$weekSelected);
    $day=$values[2];
    $month=$values[1];
    $year=$values[0];
    include ('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    $query="UPDATE caravanWeeks SET available='0',bookingID='".$bookingID.
                  "'WHERE caravanNo='".$caravanSelected."' AND year='".$year.
                            "' AND month = '".$month."' AND day = '".$day."'";
    $result=mysqli_query($conn, $query);
    mysqli_close($conn);
 }
}
?>
```

Save **CaravanWeek.php** and copy it to the server.

The final step in handling the booking is to call the methods to update the database tables. The booking details will also be displayed on the confirmation page.

Go to **confirm.php** and add the lines of program code shown below.

```
    <form method=post action='index.php'>
    <tr>
       <th>Your booking is confirmed</th>
    </tr>
    <tr>
      <td>
      <?
          $caravanSelected=$_REQUEST['caravan'];
          $weekBeginning=$_REQUEST['week'];
          include ('Location.php');
          Location::loadLocations();
          $caravanName=Location::getName($caravanSelected);
          echo "Caravan: ".$caravanName;
          echo "<p>Week beginning: ". $weekBeginning;
          $data=explode("-",$weekBeginning);
          $weekDate=$data[2]."-".$data[1]."-".$data[0];
          include('CaravanBooking.php');
          include('CaravanWeek.php');
          CaravanBooking::makeBooking($caravanSelected, $weekDate, $title,
                 $forename, $surname, $address1, $address2, $town, $postcode,
                            $paymentAmount, $cardType, $cardNumber, $expiryDate);
          echo"<p>". $title." ".$forename." ".$surname;
          echo"<br>".$address1;
          echo"<br>".$address2;
          echo"<br>".$town.", ".$postcode;
      ?>
      </td></tr>
   <tr>
     <td></td><td><br><input type=submit value='return to home page'></td>
   </tr>
  </form>
</table>
</body>
```

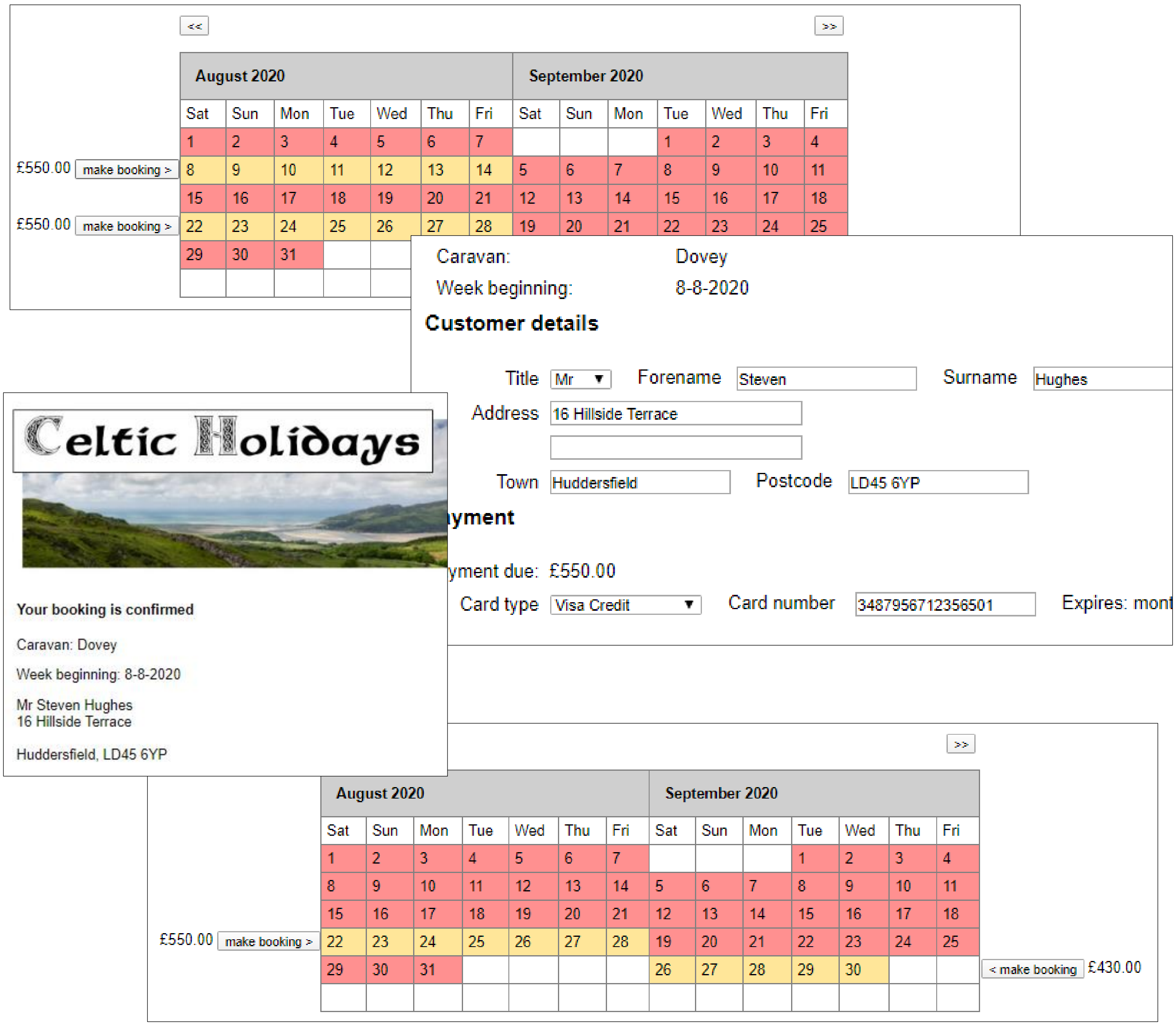
Save **confirm.php** and copy it to the server.

This completes the booking procedure. Careful testing is now necessary.

Run the website and go to the bookings page. Select a caravan and available week, making a note of the selection. Click the **'make booking'** button and enter full name, address and payment details for a customer. Click the '**complete booking**' button.

Return to the bookings page and check that the selected caravan week is now shown as unavailable. Open the PHP MyAdmin web page and check that the booking details appear correctly in the **caravanBooking** table. Make a note of the bookingID which was allocated.

Go to the **caravanLocation** table to obtain the caravan number, then find the required week and caravan in the **caravanWeeks** table. Check that the **available** value is now set to 0, and the **bookingID** value has been inserted correctly.



	August 2020							September 2020						
	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri
	1	2	3	4	5	6	7				1	2	3	4
£550.00 make booking >	8	9	10	11	12	13	14	5	6	7	8	9	10	11
	15	16	17	18	19	20	21	12	13	14	15	16	17	18
£550.00 make booking >	22	23	24	25	26	27	28	19	20	21	22	23	24	25
	29	30	31											

Caravan: Dovey
Week beginning: 8-8-2020

Customer details

Title Mr ▼ Forename Steven Surname Hughes
Address 16 Hillside Terrace
Town Huddersfield Postcode LD45 6YP

yment

yment due: £550.00
Card type Visa Credit ▼ Card number 3487956712356501 Expires: mont

Celtic Holidays

Your booking is confirmed

Caravan: Dovey

Week beginning: 8-8-2020

Mr Steven Hughes
16 Hillside Terrace

Huddersfield, LD45 6YP

	August 2020							September 2020							
	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	
	1	2	3	4	5	6	7				1	2	3	4	
	8	9	10	11	12	13	14	5	6	7	8	9	10	11	
	15	16	17	18	19	20	21	12	13	14	15	16	17	18	
£550.00 make booking >	22	23	24	25	26	27	28	19	20	21	22	23	24	25	
	29	30	31					26	27	28	29	30			< make booking £430.00

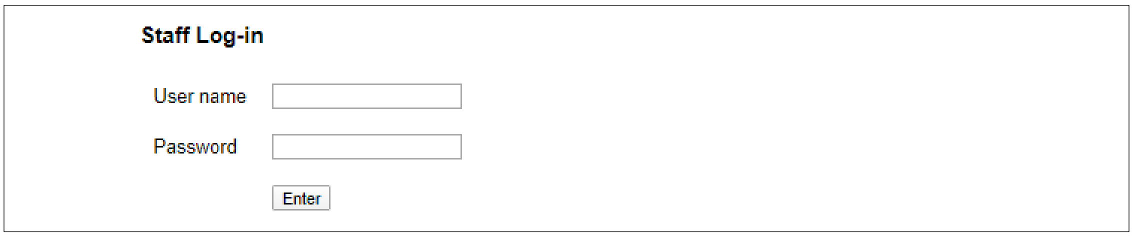
With the public web pages completed, we can now move on to produce staff administration pages. These should be password protected.

Open a blank file and add the program code below to produce a staff log-in page.

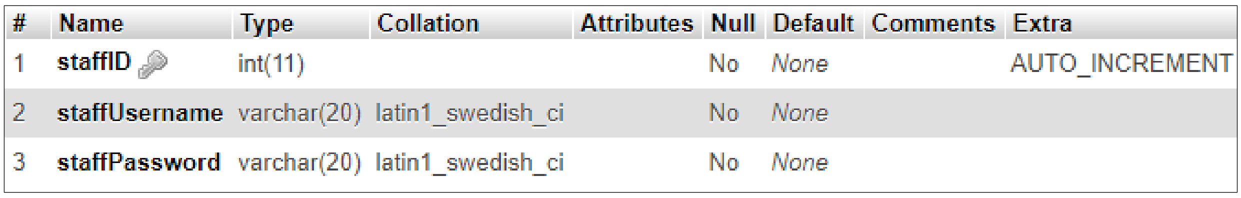
```
    <?
        session_start();
        $_SESSION['login']='NO';
    ?>
    <html>
    <head>
       <title>Celtic Holidays</title>
    </head>
    <body>
      <style>
        body {
           font-family: Arial, Helvetica, sans-serif;
           color: black;
           font-size: 12pt;
        }
      </style>
      <img src ='title.jpg' >
      <form action="staffBookings.php?caravanNoWanted=1&monthWanted=6" method="post">
      <h3>Staff Log-in</h3>
      <table border="0" cellpadding="10">
      <tr>
        <td>User name</td>
        <td>
        <?
           echo "<input type=text size=20 name=user >";
        ?>
        </td></tr>
      <tr>
        <td>Password</td>
        <td>
        <?
           echo "<input type=password size=20 name=pass >";
        ?>
        </td></tr>
      <tr>
        <td></td>
        <td>
           <input type=submit value="Enter">
        </td></tr>
     </table>
     </form>
   </body>
   </html>
```

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

Run the website, specifying **staffLogin.php** in the URL. Check that input boxes for user name and password are displayed as shown below. The creation of a staff login system was explained in more detail previously in chapter 2: Hardware Store project.



Go to the PHP MyAdmin website and open the database. A table with the name '**staff**' is required, containing fields as shown below. The **staffID** field should be set to auto-increment as records are added.



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

Select the database option to add records, and enter usernames and passwords for several members of staff.

A **Staff object class** will now be created to link the log-in screen to the database table. Open a new file and add the program code below. Save the file as **Staff.php**.

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

The program above defines the user name and password attributes for a Staff object, then provides a **constructor** method to create the object. A **checkUser( )** method then determines whether a particular Staff object matches the log-in data entered.

Add a further method **checkPassword( )** as shown in the two boxes below. This will access the database to create a set of Staff objects, then uses a loop to check each object in turn for valid log-in values.

```
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"];
```

```
        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;
    }
}
?>
```

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

When a member of staff clicks the 'Enter' button on the log-in screen, they will be taken to a booking calendar page very similar to the public booking page. We can therefore save programming time by copying this existing file. Open **bookings.php** and re-save it as **staffBookings.php**.

Some changes then need to be made. We will begin by setting up the staff log-in system. Add the block of PHP code shown below to the start of the **staffBookings.php** file.

```
<?
   session_start();
   $user=$_REQUEST['user'];
   $pass=$_REQUEST['pass'];
   $login=$_SESSION['login'];
   if (!($_SESSION['login']=='YES'))
   {
      include('Staff.php');
      if (Staff::checkPassword($user,$pass)==false)
          header('Location: staffLogin.php');
      else
              $_SESSION['login']='YES';
   }
?>
<html>
<head>
    <title> Celtic Holidays </title>
    <?
       $caravanNoWanted=$_REQUEST['caravanNoWanted'];
       $monthWanted=$_REQUEST['monthWanted'];
```

Save **staffBookings.php** and copy it to the server. Test the log-in system by running the website staffLogin page. If a correct staff user name and password are entered, the booking calendar page should open. However, an incorrect entry will immediately return the user to the log-in page.

Re-open **staffBookings.php.** Change the address shown in the **window.location** and **<form>** commands, replacing 'bookings.php' with 'staffBookings.php' as shown below. This ensures that the page is reloaded correctly when a different caravan or calendar month is selected.

```
            xdiff = Math.abs(x-mx);
            ydiff = Math.abs(y-my);
            if ((xdiff<20)&&(ydiff<20))
            {
               caravanNoWanted=caravan[i].caravanNo;
               markCaravan(caravanNoWanted);
               window.location = "staffBookings.php?caravanNoWanted="
                                  +caravanNoWanted+"&monthWanted="+monthWanted;
             }
          }
        }
        </script>
        <?
           echo"<form method=post action='staffBookings.php?caravanNoWanted="
                             .$caravanNoWanted."&monthWanted=".$monthWanted."'>";
        ?>
        <table width=900>
        <tr><td width=320>
```

On the public booking calendar, colour coding only distinguished between weeks available for booking and weeks when the caravan was not available. On the staff calendar, however, we will further distinguish between weeks when a booking has been made (red), weeks when the caravan is still available for hire (cream), and weeks when the caravan is not offered for hire (grey) – for example, if maintenance is being carried out. To allow for the additional coloured background, go to the **<style>** block near the start of the **staffBookings.php** file and update the entries as shown below.

```
        table.cal {
           border-collapse: collapse;
        }
        th.cal, td.cal, td.av, td.bk, td.cl {
           border: 1px solid gray;
        }
        th.cal {
           background-color: #cfcfcf;
           color: black;
           padding: 15px;
           text-align: left;
        }
        td.cl {
           padding: 5px;
           text-align: left;
           background-color: #cccccc;
        }
        td.cal {
           padding: 5px;
           text-align: left;
           background-color: #ffffff;
        }
```

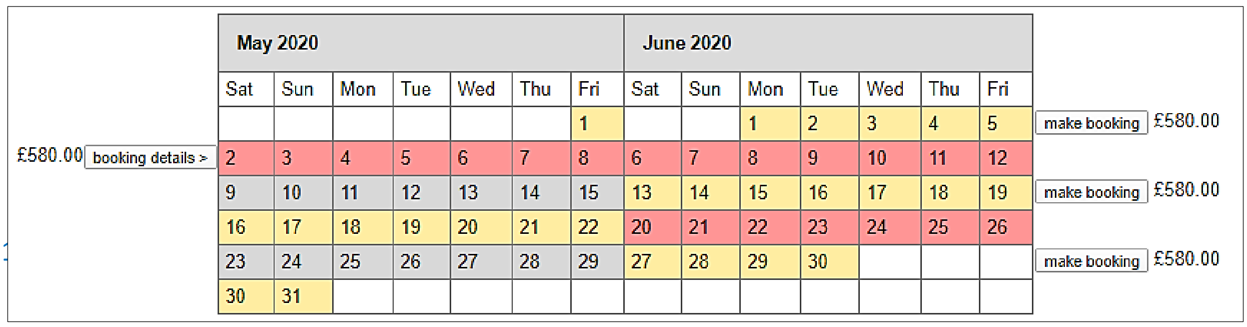
Make further changes to **staffBookings.php**, replacing lines of code as shown below:

The first correction involves the buttons which appear alongside available caravan weeks and offer a 'make booking' option. For staff, the buttons should appear alongside booked weeks, and allow the customer and payment details to be displayed.

The second correction relates to the display styles for the cells in the calendar table. Available weeks continue to be shown with a cream background, identified as class 'av'. Other weeks are now divided into bookings shown in red and identified as class 'bk', and closed weeks shown in grey and identified as class 'cl'.

```
$count1=0;
$count2=0;
$display1=false;
$display2=false;
for ($w=1;$w<=6;$w++)
{
     echo"<tr>";
     if (($bookingCode1[$w]==0)&&(strlen($caption1[$w])>1))
        echo"<td align=right width=320> £".$caption1[$w]."<button name='week'
                     value='$weekBeginning1[$w]'>booking details > </button>";
     else
     echo"<td width=320>";
     for ($d=1;$d<=7;$d++)
     {
         if ( $firstDay1==$dayName[$d])
         {
             $display1=true;
         }
         if ($count1>=$lastDay1)
         {
             $display1=false;
         }
         if ($display1==true)
         {
             $count1++;
             if ($bookingCode1[$w]==1)
                 echo"<td class='av' width=50>".$count1;
             else if ($bookingCode1[$w]==0)
                 echo"<td class='bk' width=50>".$count1;
             else if ($bookingCode1[$w]==2)
                 echo"<td class='cl' width=50>".$count1;
             else
                 echo"<td class='cal' width=50>".$count1;
         }
         else
             echo "<td class='cal' width=50>&nbsp; ";
```

Save **staffBookings.php** and copy it to the server. Run the website and log-in as a member of staff. The changes outlined above should now have been applied to the left-hand calendar month.



	May 2020							June 2020							
	Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri	
							1			1	2	3	4	5	make booking £580.00
£580.00 booking details >	2	3	4	5	6	7	8	6	7	8	9	10	11	12	
	9	10	11	12	13	14	15	13	14	15	16	17	18	19	make booking £580.00
	16	17	18	19	20	21	22	20	21	22	23	24	25	26	
	23	24	25	26	27	28	29	27	28	29	30				make booking £580.00
	30	31													

Re-open **staffBookings.php** and make similar changes for the right-hand month display, as shown below.

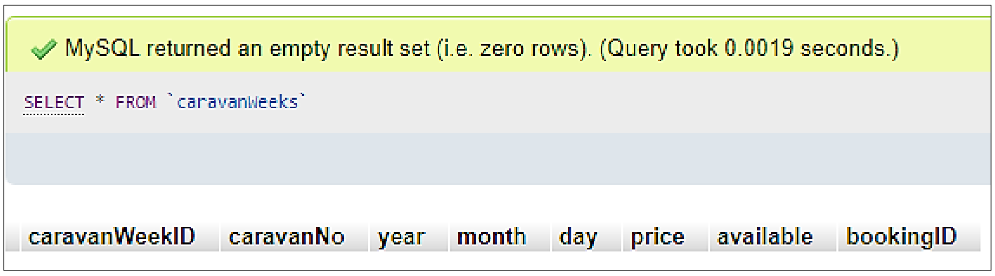
```
        for ($d=1;$d<=7;$d++)
        {
             if ( $firstDay2==$dayName[$d])
             {
                 $display2=true;
             }
             if ($count2>=$lastDay2)
             {
                 $display2=false;
             }
             if ($display2==true)
             {
                 $count2++;
                 if ($bookingCode2[$w]==1)
                      echo"<td class='av' width=50>".$count2;
                 else if ($bookingCode2[$w]==0)
                      echo"<td class='bk' width=50>".$count2;
                 else if ($bookingCode2[$w]==2)
                       echo"<td class='cl' width=50>".$count2;
                 else
                       echo"<td class='cal' width=50>".$count2;
             }
             else
                   echo "<td class='cal' width=50>&nbsp; ";
        }
        if (($bookingCode2[$w]==0)&&(strlen($caption2[$w])>1))
            echo"<td width=320><button name='week' value='$weekBeginning2[$w]'>
                                   < booking details </button>  £".$caption2[$w];
        else
             echo"<td width=320>";
    }
```

Save **staffBookings.php** and copy it to the server. Run the website again as a member of staff. The calendar display should now show closed weeks in grey for both months, with buttons appearing alongside booked weeks.

Re-open the **staffBookings.php** file and make a change to the **echo"<form>"** line shown below, so that a new page **staffDisplayBooking.php** will be loaded when a 'booking details' button is clicked. Save the updated **staffBookings.php** file.

```
             case 10:$monthName='October';break;
             case 11:$monthName='November';break;
             case 12:$monthName='December';break;
          }
          return $monthName;
       }
       echo"<form method=post action='staffDisplayBooking.php
                              ?caravanNoWanted=".$caravanNoWanted."'>";
       echo"<table class='cal' width=1100>";
       echo"<tr><td>";
       $firstMonth=getMonthName($monthWanted);
       $nextMonth=intval($monthWanted+1);
```

Before proceeding any further with the staff website, we should delete the random data that was used for testing the calendar display and re-initialise the **caravanWeeks** table.
Go to the PHP MyAdmin web page and open the **caravanWeeks** table. Select the 'Operations' option at the top of the page, then click 'Empty the table (TRUNCATE)'. Return to the data display and check that the table is now empty. Use the same procedure to empty the **caravanBooking** table.



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0019 seconds.)

```
SELECT * FROM `caravanWeeks`
```

caravanWeekID	caravanNo	year	month	day	price	available	bookingID

Open the **CaravanWeek.php** class file. Locate the **initialiseBookings( )** method, and change the line which sets the value for **$bookingCode**. This change removes the random( ) function.

```
                        else
                           $price = 440.00;
                 }
                      }
                    }
                  }
                  $bookingCode=1;
                  $query="INSERT INTO caravanWeeks VALUES ('','$van','$year',
                                      '$month', '$i','$price','$bookingCode','0')";
                  echo"<br>".$query;
                  $result=mysql_query($query);
               }
             }
           }
```

Save **CaravanWeek.php** and copy it to the server. When the **initialiseBookings( )** method is run, all caravan weeks will be set to booking code 1, indicating that the caravan is available for hire.

Open the **staffBookings.php** file and add lines of code at the beginning of the program to run the **initialiseBookings( )** method.

```
   <?
      session_start();
      include('CaravanWeek.php');
      CaravanWeek::initialiseBookings(2020);
      $user=$_REQUEST['user'];
      $pass=$_REQUEST['pass'];
      $login=$_SESSION['login'];
       if (!($_SESSION['login']=='YES'))
       {
```

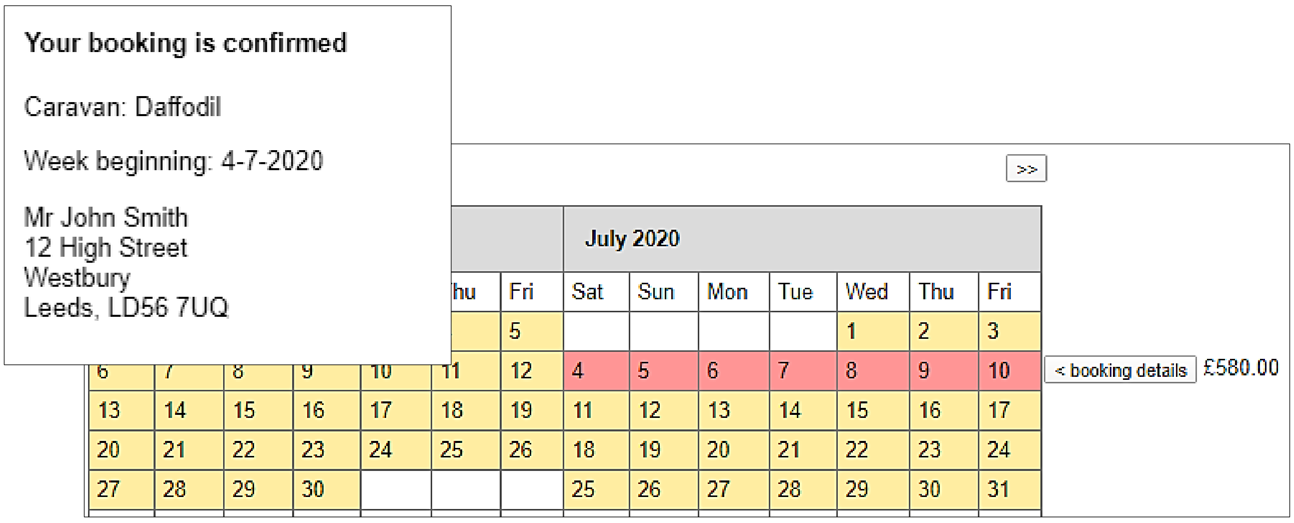
Set to the required calendar year, e.g. 2022

Save **staffBookings.php** and copy it to the server.

Load the website staff log-in page. Enter the user name and password for a member of staff. The **staffBookings** page will then load and the **initialiseBookings( )** method will run. Immediately close the web page by means of the cross symbol on the browser tab.

Go to the PHP MyAdmin page and check that a set of 1560 caravan week records has been created, with all entries in the **available** field set to a value of 1. Re-open **staffBookings.php** and remove the two lines of code outlined above, so that the **initialiseBookings( )** method will not be re-run. Save **staffBookings.php** and copy it to the server.

Run the website, this time as a customer. Go to the bookings page and check that all caravans are shown as available. Make a booking, keeping a note of the caravan, week, and customer details.



Your booking is confirmed

Caravan: Daffodil

Week beginning: 4-7-2020

Mr John Smith
12 High Street
Westbury
Leeds, LD56 7UQ

>>

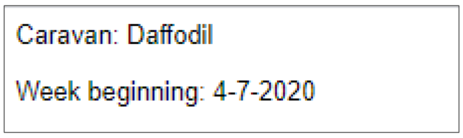
							July 2020								
					hu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri		
						5					1	2	3		
6	7	8	9	10	11	12	4	5	6	7	8	9	10	< booking details	£580.00
13	14	15	16	17	18	19	11	12	13	14	15	16	17		
20	21	22	23	24	25	26	18	19	20	21	22	23	24		
27	28	29	30				25	26	27	28	29	30	31		

Our next objective is to produce a staff page to display details of the booking just entered. Open a blank file and add the lines of program code shown below.

```
<html>
 <head>
   <title> Celtic Holidays </title>
     <style>
       body
       {
           font-family: Arial, Helvetica, sans-serif;
           color: black;
       }
     </style>
  </head>
  <body>
  <?
     $caravanSelected=$_REQUEST['caravanNoWanted'];
     $weekBeginning=$_REQUEST['week'];
     include ('Location.php');
     Location::loadLocations();
     $caravanName=Location::getName($caravanSelected);
     echo "Caravan: ".$caravanName;
     echo "<p>Week beginning: ". $weekBeginning;
  ?>
  </body>
  </html>
```

Save the file as **staffDisplayBooking.php** and copy this to the server.

Run the website staff log-in page and enter a staff username and password. When the staff bookings page has loaded, locate the booking on the calendar and click the 'booking details' button. The name of the caravan and week beginning date should be displayed.



Caravan: Daffodil

Week beginning: 4-7-2020

When this is working correctly, return to the **staffDisplayBooking.php** file and remove the two 'echo' lines outputting the caravan name and week. These were for test purposes only. Re-save the file.

Further details of the booking can now be displayed. To do this, we must add methods to the **CaravanBooking.php** class file as shown below:

- A series of **get( )** methods will allow access to the private attributes of the Booking object.
- The **loadBookingByID( )** method will access the caravanBooking table in the database and obtain the record with the required **bookingID** value. The record is then used to construct a Booking object, identified as the array member **$booking[0]**.

```
    public function getBookingID(){return $this->bookingID;}
    public function getPaymentDate(){return $this->paymentDate;}
    public function getCaravanNo(){return $this->caravanNo;}
    public function getWeekBeginning(){return $this->weekBeginning;}
    public function getTitle(){return $this->title;}
    public function getForename(){return $this->forename;}
    public function getSurname(){return $this->surname;}
    public function getAddress1(){return $this->address1;}
    public function getAddress2(){return $this->address2;}
    public function getTown(){return $this->town;}
    public function getPostcode(){return $this->postcode;}
    public function getPaymentAmount(){return $this->paymentAmount;}
    public function getCardType(){return $this->cardType;}
    public function getCardNumber(){return $this->cardNumber;}
    public function getExpiryDate(){return $this->expiryDate;}

    public static function loadBookingById($bookingIDwanted)
    {
        include ('user.inc');
        $conn = new mysqli(localhost, $username, $password, $database);
        if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
        $query="SELECT * FROM caravanBooking WHERE bookingID='".$bookingIDwanted."'";
        $result=mysqli_query($conn, $query);
        mysqli_close($conn);
        $row=mysqli_fetch_assoc($result);
        $paymentDate=$row["paymentDate"];
        $caravanNo=$row["caravanNo"];
        $weekBeginning=$row["weekBeginning"];
        $title==$row["title"];
        $forename=$row["forename"];
        $surname=$row["surname"];
        $address1=$row["address1"];
        $address2=$row["address2"];
        $town=$row["town"];
        $postcode=$row["postcode"];
        $paymentAmount=$row["paymentAmount"];
        $cardType=$row["cardType"];
        $cardNumber=$row["cardNumber"];
        $expiryDate=$row["expiryDate"];
        $obj = new CaravanBooking($bookingIDwanted,$paymentDate,$caravanNo,
               $weekBeginning,$title,$forename,$surname,$address1,$address2,
               $town,$postcode, $paymentAmount,$cardType,$cardNumber,$expiryDate);
        CaravanBooking::$booking[0] = $obj;
   }
}
?>
```

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

Return to the **staffDisplayBooking.php** file and add lines of program code to display the customer name, address and payment details as shown on the following two pages, then save the file.

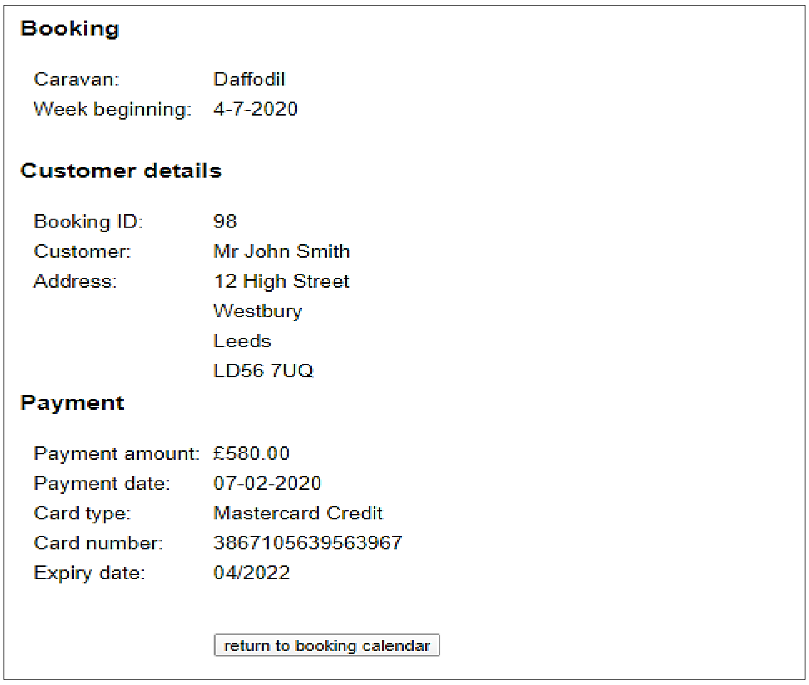
```
    <?
          $caravanSelected=$_REQUEST['caravanNoWanted'];
          $weekBeginning=$_REQUEST['week'];
          include ('Location.php');
          Location::loadLocations();
          $caravanName=Location::getName($caravanSelected);
          $data = explode("-",$weekBeginning);
          $month=$data[1];
          include ('CaravanBooking.php');
          include('CaravanWeek.php');
          $bookingIDwanted=CaravanWeek::loadBookingID($caravanSelected,
                                                        $weekBeginning);
          CaravanBooking::loadBookingByID($bookingIDwanted);
          echo"<form method=post  action='staffBookings.php?caravanNoWanted=".
                                  $caravanSelected."&monthWanted=".$month."'>";
     ?>
     <table cellpadding=4>
     <tr>
          <td colspan=3><h3>Booking</td></tr>
     <tr><td></td><td>
          Caravan: </td>
          <td>
          <?
             echo  $caravanName;
          ?> </td></tr>
     <tr><td></td><td>
          Week beginning: </td>
          <td>
          <?
             echo $weekBeginning;
          ?> </td></tr>
     <tr><td><br></td></tr>
     <tr>
          <td colspan=3><h3>Customer details</td></tr>
     <tr><td></td><td>Booking ID: </td>
          <td>
          <?
              echo $bookingIDwanted;
          ?> </td></tr>
     <tr><td></td><td>Customer: </td>
          <td>
          <?
              echo CaravanBooking::$booking[0]->getTitle()." ".
                             CaravanBooking::$booking[0]->getForename()." ".
                                      CaravanBooking::$booking[0]->getSurname();
          ?> </td></tr>
     <tr><td></td><td>Address: </td>
           <td>
           <?
                echo CaravanBooking::$booking[0]->getAddress1()."</td></tr>";
                $address2 = CaravanBooking::$booking[0]->getAddress2();
                if (strlen($address2)>0)
                     echo"<tr><td></td><td></td><td>".$address2."</td></tr>";
           ?>
      <tr><td></td><td></td><td>
           <?
                echo CaravanBooking::$booking[0]->getTown()."</td></tr>";
           ?>
      <tr><td></td><td></td><td>
           <?
                echo CaravanBooking::$booking[0]->getPostcode()."</td></tr>";
           ?>
      <tr><td colspan=3><h3>Payment</td></tr>
      <tr><td></td><td>Payment amount: </td>
           <td>
           <?
              echo "£".CaravanBooking::$booking[0]->getPaymentAmount()."</td></tr>";
           ?>
      <tr><td></td><td>Payment date: </td>
           <td>
           <?
                $paymentDate=CaravanBooking::$booking[0]->getPaymentDate();
                echo substr($paymentDate,8,2)."-".substr($paymentDate,5,2).
                                     "-".substr($paymentDate,0,4) ."</td></tr>";
           ?>
       <tr><td></td><td>Card type: </td>
           <td>
           <?
                echo CaravanBooking::$booking[0]->getCardType()."</td></tr>";
           ?>
       <tr><td></td><td>Card number: </td>
           <td>
           <?
                echo CaravanBooking::$booking[0]->getCardNumber()."</td></tr>";
           ?>
       <tr><td></td><td>Expiry date: </td>
           <td>
           <?
                echo CaravanBooking::$booking[0]->getExpiryDate()."</td></tr>";
           ?>
       <tr><td></td><td></td><td>
           <br><br><input type=submit value='return to booking calendar'></td></tr>
       </table>
      </form>
    </body>
    </html>
```

One final task is to add a method to the CaravanWeek class file to obtain the bookingID for the required caravan and week. Open the file **CaravanWeek.php** and add the **loadBookingID( )** method shown in the two boxes below, then save the file.

```
public static function loadBookingID($caravanNoWanted,$weekBeginning)
{
    $data = explode("-",$weekBeginning);
    $day=$data[0];
    $month=$data[1];
    $year=$data[2];
    include ('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
```

```
        if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }

        $query="SELECT * FROM caravanWeeks WHERE caravanNo=".$caravanNoWanted.
                       " AND year=".$year." AND month=".$month." AND day=".$day;
        $result=mysqli_query($conn, $query);
        mysqli_close($conn);
        $row=mysqli_fetch_assoc($result);
        $bookingID=$row["bookingID"];
        return $bookingID;
    }
  }
  ?>
```

Copy the **staffDisplayBooking.php** and **CaravanWeek.php** files to the server. Run the website and log-in as a member of staff. Select the booking made earlier and check that all details are now displayed correctly.



Booking

Caravan:	Daffodil
Week beginning:	4-7-2020

Customer details

Booking ID:	98
Customer:	Mr John Smith
Address:	12 High Street Westbury Leeds LD56 7UQ

Payment

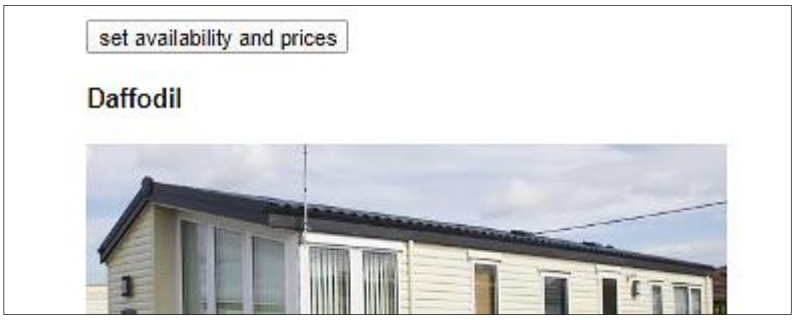
Payment amount:	£580.00
Payment date:	07-02-2020
Card type:	Mastercard Credit
Card number:	3867105639563967
Expiry date:	04/2022

Staff using the booking system will require a facility to change the hire charges for particular caravans, or show caravans as unavailable if maintenance is required. We will add a page now to provide these functions.

Re-open **staffBookings.php** and add lines of program code to provide a 'set availability and prices' button.

```
        <canvas id="myCanvas" width="510" height="400"
                 style="border:1px solid #d3d3d3;" onmousedown="mouseDown()">
        </canvas>
        </td>
        <td width=100></td>
        <td width = 400 style="vertical-align:top">
        <?
            echo"<form method=post action='setPrices.php?caravanID=".
                                                 $caravanNoWanted."'>";
        ?>
        <input type=submit value="set availability and prices">
        </form>
        <h3><p id="caravanName" ></p></h3>
        <img id="caravanImage" src=""  width = 400>
```

Save the **staffBookings.php** file and copy it to the server. Run the website and log-in as a member of staff. Check that the button has been added above the caravan image.



Open a blank file and add the program code below. This takes the number of the currently selected caravan and determines the number and name of the caravan group to which it belongs. The numbers of the first and last caravan within the group are also recorded.

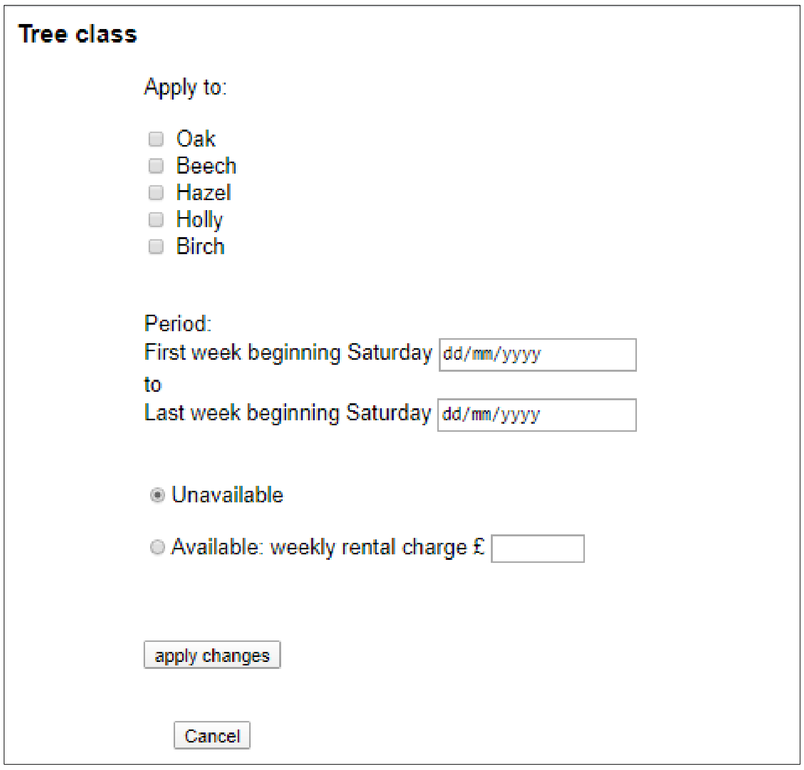
Save the file as **setPrices.php**.

```
<?
   $caravanID=$_REQUEST["caravanID"];
   include('Location.php');
   Location::loadLocations();
?>
<html>
 <head>
 <title> Celtic Holidays </title>
  <style>
       body  {font-family: Arial, Helvetica, sans-serif; color: black; }
       td    {font-size: 12pt;}
 </style>
 </head>
 <body>
  <?
     echo"<form method=post action='updatePrices.php?caravanID=".
                      $caravanID."' onsubmit='return submitForm(this)'>";
     $first=0;
     $last=0;
       if ($caravanID<=9)
       {      $groupWanted=1;  $caravanClass="Flower";
              $first=1; $last=9;   }
       else if ($caravanID<=14)
       {      $groupWanted=2;  $caravanClass="Tree";
              $first=10; $last=14; }
       else if ($caravanID<=20)
       {      $groupWanted=3;  $caravanClass="River";
              $first=15; $last=20; }
       else if ($caravanID<=24)
       {      $groupWanted=4; $caravanClass="Mountain";
              $first=21; $last=24; }
       else
       {      $groupWanted=5; $caravanClass= "Castle";
              $first=25; $last=30; }
    ?>
    </body>
  </html>
```

We will now add input boxes to create a dialogue page. The user will be able to select particular caravans within the currently selected group by means of a set of check boxes.

```
     {     $groupWanted=5; $caravanClass= "Castle";
           $first=25; $last=30;  }

     echo"<input type='hidden' name='first' value='".$first."'>";
     echo"<input type='hidden' name='last' value='".$last."'>";
     echo"<table><tr>";
     echo"<td><h3>".$caravanClass." class</h3></td></tr>";
     echo"<tr><td></td><td>Apply to:<br><br></td></tr>";
     for ($i=1;$i<=Location::$caravanCount; $i++)
     {
        $number = Location::$caravan[$i]->getCaravanNo();
        if (($number>=$first)&&($number<=$last))
        {
           echo"<tr><td>";
           $caravanName = Location::getName($number);
           echo"<td><input type='checkbox' name='caravan".$number.
                                                 "' value='YES'> ".$caravanName;
           echo"</td></tr>";
        }
     }
     echo"<tr><td></td><td><br><br>Period: </td></tr>";
     echo"<tr><td></td><td>First week beginning Saturday <input type=date
                                                    name=firstWeek></td></tr>";
     echo"<tr><td></td><td>to</td></tr>";
     echo"<tr><td></td><td>Last week beginning Saturday <input type=date
                                                     name=lastWeek></td></tr>";
     echo"<tr><td></td><td><br><br>";
     echo"<input type='radio' name='available' id='unavailable' value='NO'
                                                checked>Unavailable</td></tr>";
     echo"<tr><td></td><td><br>";
     echo"<input type='radio' name='available' value='YES' >Available:";
     echo" weekly rental charge £ ";
     echo"<input type=text size=6 id='weekCost' name ='weekCost'></td></tr>";
     echo"<tr><td></td><td>";
     echo"<br><br><br>";
     echo"<input type='submit' value='apply changes'></td></tr>";
  ?>
  </form>
  <tr><td></td><td><br><br>
  <?
     echo"<form method=post action='staffBookings.php?caravanNoWanted=".
                                                 $caravanID."&monthWanted=6'>";
  ?>
  &nbsp;&nbsp;&nbsp;&nbsp;
  <input type=submit value='Cancel'></td></tr>
  </form>
  </table>
 </body>
</html>
```

Save the **setPrices.php** file and copy it to the server. Run the website and log-in as a member of staff. On the bookings page, select a caravan from any group then click the 'set availability and prices' button. Check that the list of caravans for the selected group are shown with check boxes.



Tree class

Apply to:

- ☐ Oak
- ☐ Beech
- ☐ Hazel
- ☐ Holly
- ☐ Birch

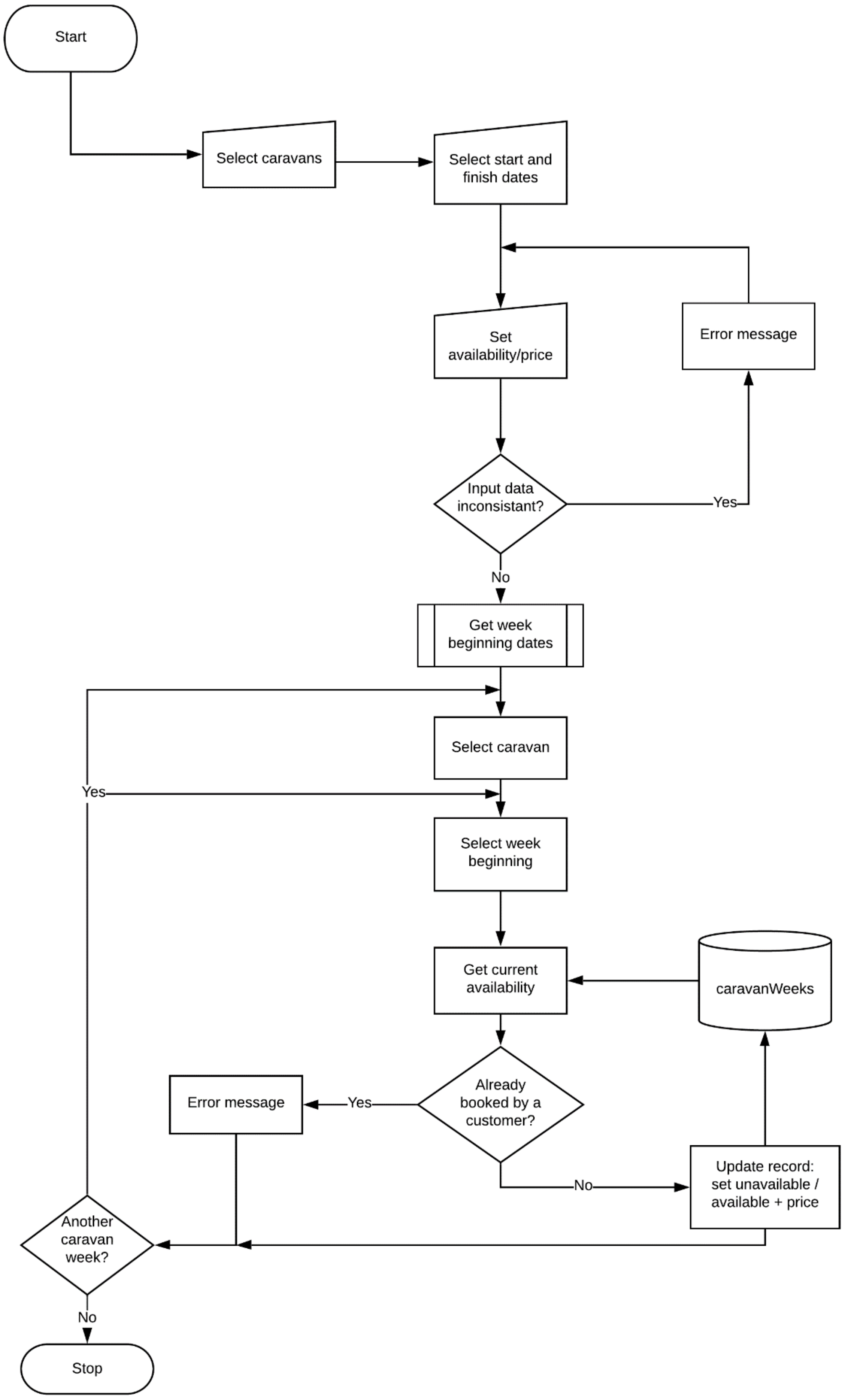
Period:
First week beginning Saturday dd/mm/yyyy
to
Last week beginning Saturday dd/mm/yyyy

● Unavailable

○ Available: weekly rental charge £

The processing of the availability and price update is quite complex, so a flowchart of the sequence is given on the next page.

- When the user clicks the 'apply changes' button, the data entries will be checked for consistency. Error messages will be displayed if:
  Unavailable is selected, but a weekly rental charge has been entered.
  Available is selected, but no weekly rental charge has been entered.
- After confirming that the user wishes changes to be applied to the database, a new page **updatePrices.php** will be loaded. A function will use the specified start and finish dates to make a list of the Saturday week beginning dates which are to be included.
- A loop will operate for each caravan selected. Within this, an inner loop will repeat for each week beginning date.
- The **availability code** for the current caravan and week will be obtained from the database. If the caravan availability or price is to be changed, but the code indicates that it has already been booked to a customer, then an error message will be displayed.
- Where no error is detected, the **caravanWeek** record will be updated to show the requested availability and weekly hire charge values.

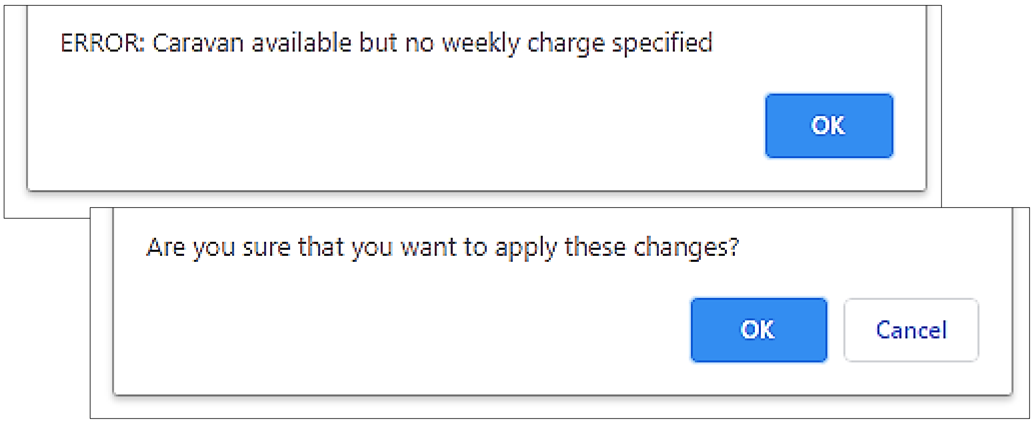

Start
Select caravans
Select start and finish dates
Set availability/price
Error message
Input data inconsistant?
Yes
No
Get week beginning dates
Select caravan
Yes
Select week beginning
Get current availability
caravanWeeks
Already booked by a customer?
Yes
Error message
No
Update record: set unavailable / available + price
Another caravan week?
No
Stop

Re-open the **setPrices.php** file and add a JavaScript function at the end of the file. This checks the input data when the user clicks the 'apply changes' button. Two error conditions are:

- Setting a hire charge for a week when the caravan is marked as 'unavailable'.
- Specifying an available carvan week, but not entering a hire charge.

```
    <input type=submit value='Cancel'>
  </form>
  </table>
  <script>
  function submitForm()
  {
     unavailable=document.getElementById("unavailable").checked;
     weekCost=document.getElementById("weekCost").value;
     if ((unavailable==true)&&(weekCost>0))
     {
        alert('ERROR: Caravan unavailable but a weekly charge has been set');
        return false;
     }
     else
        if ((unavailable==false)&&(weekCost==0))
        {
           alert('ERROR: Caravan available but no weekly charge specified');
           return false;
        }
        else
           return confirm('Are you sure that you want to apply these changes?');
  }
  </script>
  </body>
 </html>
```

Save the **setPrices.php** file and copy it to the server. Run the website as staff and continue to the 'set availability and prices' option. Enter inconsistent data and check that error messages are displayed. Enter valid data and check that a confirm dialogue is displayed.


ERROR: Caravan available but no weekly charge specified

Are you sure that you want to apply these changes?

Open a new blank file and add the program code below.

```
<?
   $available=$_REQUEST["available"];
    $weekCost=$_REQUEST["weekCost"];
    $firstWeek=$_REQUEST["firstWeek"];
    $lastWeek=$_REQUEST["lastWeek"];
    $first=$_REQUEST["first"];
    $last=$_REQUEST["last"];
    $caravanID=$_REQUEST["caravanID"];
?>
<html>
<head>
   <title> Celtic Holidays </title>
   <style>
    body {
         font-family: Arial, Helvetica, sans-serif;
         color: black;
         font-size: 14pt;
    }
   </style>
</head>
<body>
 <?
   echo"Start date: ".$firstWeek;
   echo"<p>Finish date: ".$lastWeek;
 ?>
</body>
</html>
```

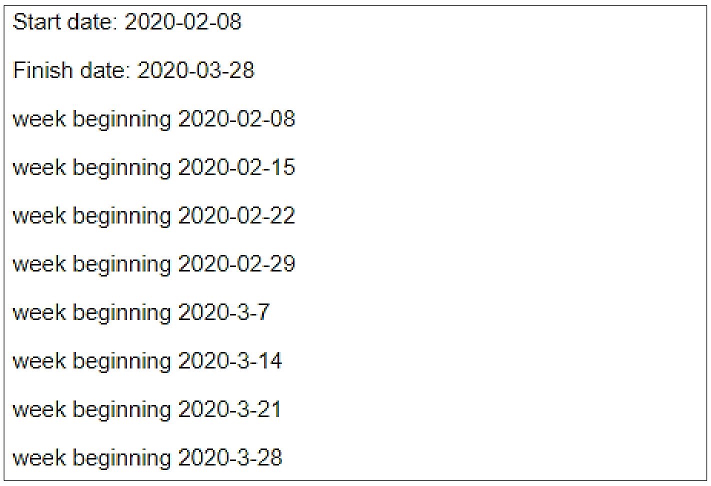
Save the file as **updatePrices.php** and copy it to the server. Run the website and log-in as a staff member. Go to the bookings page and click the 'set availability and prices' button. Complete the data entry form, including start and finish dates for the update period, then click 'apply changes'. For test purposes, the two dates are displayed in year-month-day format. Check that these are correct.

```
Start date: 2020-02-08

Finish date: 2020-02-29
```

Re-open the **updatePrices.php** file. Go to the **<body>** section and add a function **findWeekBeginning( )** as shown in the two boxes below. This will determine all Saturday week starting dates during the period specified. These dates are stored in the array **$weekBeginning** in year-month-day format, which is returned by the function. Other lines added to the program will call the **findWeekBeginning( )** function, and then display the list of dates for test purposes.

```
<body>
<?
   echo"Start date: ".$firstWeek;
   echo"<p>Finish date: ".$lastWeek;
   $weekBeginning=findWeekBeginning($firstWeek,$lastWeek);
   $weekCount=sizeof($weekBeginning);
   for ($i=1; $i<=$weekCount;$i++)
   {
      echo"<p>week beginning ".$weekBeginning[$i];
   }

   function findWeekBeginning( $firstWeek,$lastWeek)
   {
  function findWeekBeginning( $firstWeek,$lastWeek)
  {
    $year1=substr($firstWeek,0,4);   $year2=substr($lastWeek,0,4);
    $month1=substr($firstWeek,5,2);  $month2=substr($lastWeek,5,2);
     $day1=substr($firstWeek,8,2);   $day2=substr($lastWeek,8,2);
     $count=0;
    for ($year=$year1; $year<=$year2; $year++)
     {
       for($month=$month1; $month<=$month2; $month++)
       {
          if ($month>$month1)
               $dayStart=1;
           else
               $dayStart=$day1;
           if ($month<$month2)
               $dayFinish=31;
           else
               $dayFinish=$day2;
           for($day=$dayStart; $day<=$dayFinish; $day++)
          {
             $dayname = date("D", strtotime($year."-".$month."-".$day));
             if ($dayname == "Sat")
             {
                $count++;
                $weekBeginning[$count] = $year."-".$month."-".$day;
             }
          }
       }
     }
     return $weekBeginning;
   }
 ?>
 </body>
 </html>
```

Save the **updatePrices.php** file and copy it to the server. Run the website as previously and navigate to the 'set availability and prices' page. Complete the data entry form and then click 'apply changes'. Check that all Saturday week start dates during the selected period are listed, as in the example below.

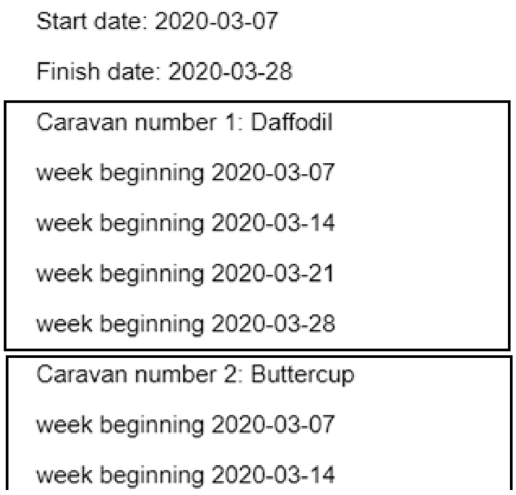


```
Start date: 2020-02-08

Finish date: 2020-03-28

week beginning 2020-02-08

week beginning 2020-02-15

week beginning 2020-02-22

week beginning 2020-02-29

week beginning 2020-3-7

week beginning 2020-3-14

week beginning 2020-3-21

week beginning 2020-3-28
```

Reopen **updatePrices.php.** The next step is to list the caravans which will be included in the update procedure. Add the lines of program code shown below. These collect results from the set of check boxes on the input page, then use these to list the caravan numbers and names.

```
<body>
<?
    echo"Start date: ".$firstWeek;
    echo"<p>Finish date: ".$lastWeek;
    $weekBeginning=findWeekBeginning($firstWeek,$lastWeek);
    $weekCount=sizeof($weekBeginning);
    include('Location.php');
    Location::loadLocations();
    for ($c=$first; $c<=$last; $c++)
     {
        $caravanWanted="caravan".$c;
        $includeCaravan = $_REQUEST[$caravanWanted];
        if ($includeCaravan=='YES')
        {
            $caravanName=Location::getName($c);
            echo"<p>Caravan number ".$c.": ".$caravanName;
            for ($i=1; $i<=$weekCount;$i++)
            {
                echo"<p>week beginning ".$weekBeginning[$i];
            }
        }
      }

     function findWeekBeginning( $firstWeek,$lastWeek)
     {
```

Save **updatePrices.php** and copy it to the server. Run the website, navigate to the update page and enter test data for caravans and dates as previously. On selecting 'apply changes', each caravan should be listed, along with the week beginning dates for which the caravan's records will be updated.



```
Start date: 2020-03-07

Finish date: 2020-03-28

Caravan number 1: Daffodil

week beginning 2020-03-07

week beginning 2020-03-14

week beginning 2020-03-21

week beginning 2020-03-28

Caravan number 2: Buttercup

week beginning 2020-03-07

week beginning 2020-03-14
```

The program will now call a method in the **CaravanWeek class** file to carry out the actual database update. Re-open the **updatePrices.php** file and add a line to include the CaravanWeek class and call a **setAvailability( )** method. This method has input parameters representing the **caravan number**, **week beginning date**, **availability** and **weekly hire charge**. This information is needed to identify the correct record in the caravanWeeks table, then update the availability status and/or weekly cost. Remove the 'echo' lines which were included for test purposes, and insert the lines of program code shown below.

```
    $weekBeginning=findWeekBeginning($firstWeek,$lastWeek);
    $weekCount=sizeof($weekBeginning);
    include('Location.php');
    Location::loadLocations();
    include('CaravanWeek.php');
    for ($c=$first; $c<=$last; $c++)
    {
       $caravanWanted="caravan".$c;
       $includeCaravan = $_REQUEST[$caravanWanted];
       if ($includeCaravan=='YES')
       {
          $caravanName=Location::getName($c);
          for ($i=1; $i<=$weekCount;$i++)
          {
             CaravanWeek::setAvailability($c,$weekBeginning[$i],$available,$weekCost);
          }
       }
    }
```

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

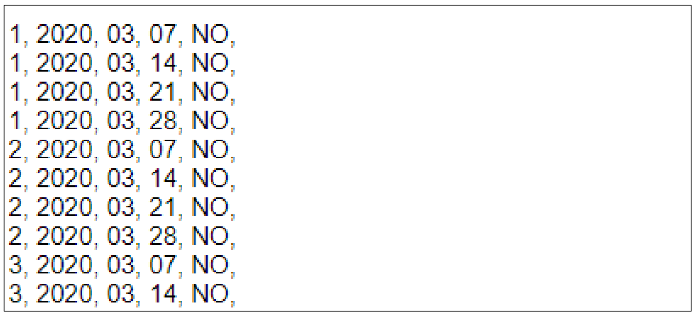
Open the **CaravanWeek.php** class file and add a new method **setAvailability( )**. Week-beginning dates in the caravanWeeks table are represented as three separate fields for year, month and day. The method begins by splitting the date into these fields. We then add a test line to check that all the data required for the update is available.

```
     public static function setAvailability($caravanNo, $weekBeginning,
                                                     $available,$weekCost)
     {
        $data = explode("-",$weekBeginning);
        $year=$data[0];
        $month=$data[1];
        $day=$data[2];
        echo"<br>".$caravanNo.", ".$year.", ".$month.", ".$day.", ".
                                                 $available.", ".$weekCost;
      }
   }
   ?>
```

Save **CaravanWeek.php** and copy it to the server.

Run the website, navigate to the update page and enter test data as previously. On selecting 'apply changes', a series of lines of data should be displayed representing:

caravanNo, week beginning: **year**, **month**, **day**, **availability**: YES or NO, and **price**(if specified)



```
1, 2020, 03, 07, NO,
1, 2020, 03, 14, NO,
1, 2020, 03, 21, NO,
1, 2020, 03, 28, NO,
2, 2020, 03, 07, NO,
2, 2020, 03, 14, NO,
2, 2020, 03, 21, NO,
2, 2020, 03, 28, NO,
3, 2020, 03, 07, NO,
3, 2020, 03, 14, NO,
```

We are almost ready to update the database record, but a couple of tasks remain.

- The existing database record should be checked in case the caravan has already been booked by a customer for the specified week. In this situation, the update will not be carried out and a warning message will be displayed.
- A request to make the caravan unavailable, currently recorded as a 'NO' value, should be entered as code 2 in the database table. A request for the caravan to be available, currently recorded as 'YES', should be entered as code 1.

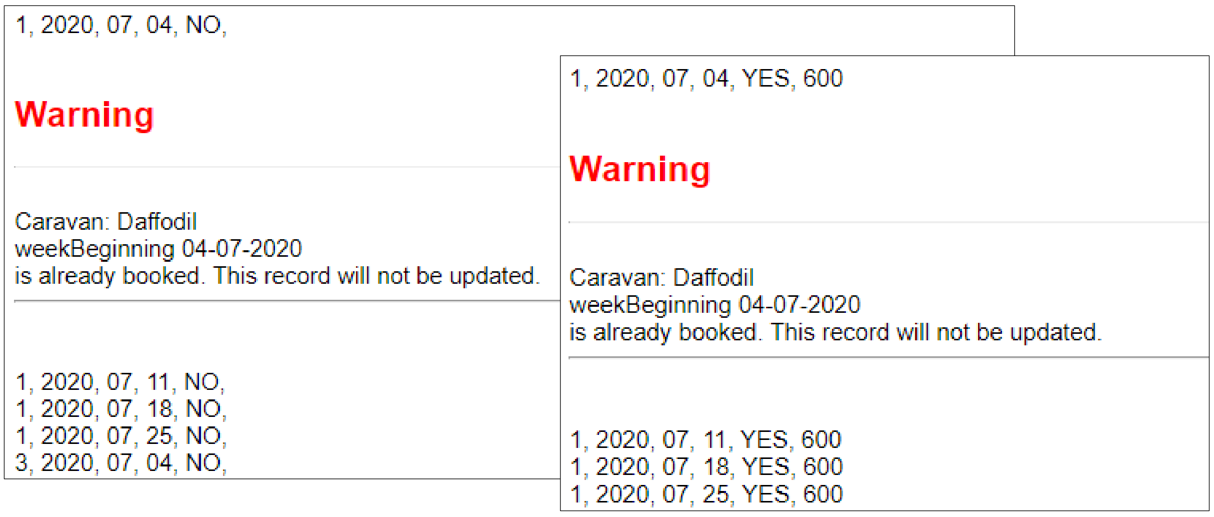
Re-open the **CaravanWeek.php** file and add the program code shown below to the **setAvailability( )** method. This accesses the existing caravanWeek record to obtain the availability code. If this has a value of 0, a booking already exists and an error message is displayed.

```
      public static function setAvailability($caravanNo, $weekBeginning,
                                                   $available,$weekCost)
      {
           $data = explode("-",$weekBeginning);
           $year=$data[0];
           $month=$data[1];
           $day=$data[2];
           echo"<br>".$caravanNo.", ".$year.", ".$month.", ".$day.", ".
                                                   $available.", ".$weekCost;

           Location::loadLocations();
           include ('user.inc');
           $conn = new mysqli(localhost, $username, $password, $database);
           if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
           $query="SELECT * FROM caravanWeeks WHERE caravanNo='".$caravanNo.
                             "' AND year = '".$year."' AND month = '".$month.
                                                   "' AND day = '".$day."'";
           $result=mysqli_query($conn, $query);
           $row=mysqli_fetch_assoc($result);
           $currentAv=$row["available"];
           if ($currentAv==0)
           {
               echo"<br><br><h2 style='color:red;'>Warning</h2>";
               $caravanName=Location::getName($caravanNo);
               echo"<hr>";
               echo"<br>Caravan:  ".$caravanName;
               echo"<br>weekBeginning ".(string)$day."-".(string)$month.
                                                   "-".(string)$year;
               echo"<br>is already booked. This record will not be updated.";
               echo"<br><hr><br>";
           }
           mysqli_close($conn);

      }
  }
  ?>
```

Re-save **CaravanWeek.php** and copy it to the server.

Run the website, navigate to the update page. Enter test data which includes a caravan week which is already shown on the calendar as booked. Click the 'apply changes' button and check that a warning message is displayed as below.

The test should be repeated for both an attempt to make a booked caravan unavailable, and an attempt to change the weekly hire charge for a booked caravan.



```
1, 2020, 07, 04, NO,

Warning

Caravan: Daffodil
weekBeginning 04-07-2020
is already booked. This record will not be updated.

1, 2020, 07, 11, NO,
1, 2020, 07, 18, NO,
1, 2020, 07, 25, NO,
3, 2020, 07, 04, NO,
```

```
1, 2020, 07, 04, YES, 600

Warning

Caravan: Daffodil
weekBeginning 04-07-2020
is already booked. This record will not be updated.

1, 2020, 07, 11, YES, 600
1, 2020, 07, 18, YES, 600
1, 2020, 07, 25, YES, 600
```

Assuming that no error has been found, the caravan week record can now be updated. Re-open the **CaravanWeek.php** file and add the program code shown below to the **setAvailability( )** method.

```
      if ($currentAv==0)
      {
            echo"<br><br><h2 style='color:red;'>Warning</h2>";
            $caravanName=Location::getName($caravanNo);
            echo"<hr>";
            echo"<br>Caravan:  ".$caravanName;
            echo"<br>weekBeginning ".(string)$day."-".(string)$month.
                                                   "-".(string)$year;
            echo"<br>is booked. This record will not be set as unavailable.";
            echo"<br><hr><br>";
      }
      else
      {
            if ($available=='YES')
                  $av = 1;
            if ($available=='NO')
                  $av = 2;
            $query="UPDATE caravanWeeks SET price='".$weekCost."',available='".
                           $av."' WHERE caravanNo='".$caravanNo. "' AND year = '".
                           $year."' AND month = '".$month."' AND day = '".$day."'";
            $result=mysqli_query($conn, $query);
      }
      mysqli_close($conn);
   }
}
?>
```

The 'echo' line near the start of the **setAvailability( )** method:

```
        echo"<br>".$caravanNo.", ".$year.", ".$month.", ".$day.", ".
                                    $available.", ".$weekCost;
```

displaying the sets of weekly data was for testing purposes only, and can now be deleted. Re-save **CaravanWeek.php** and copy it to the server.

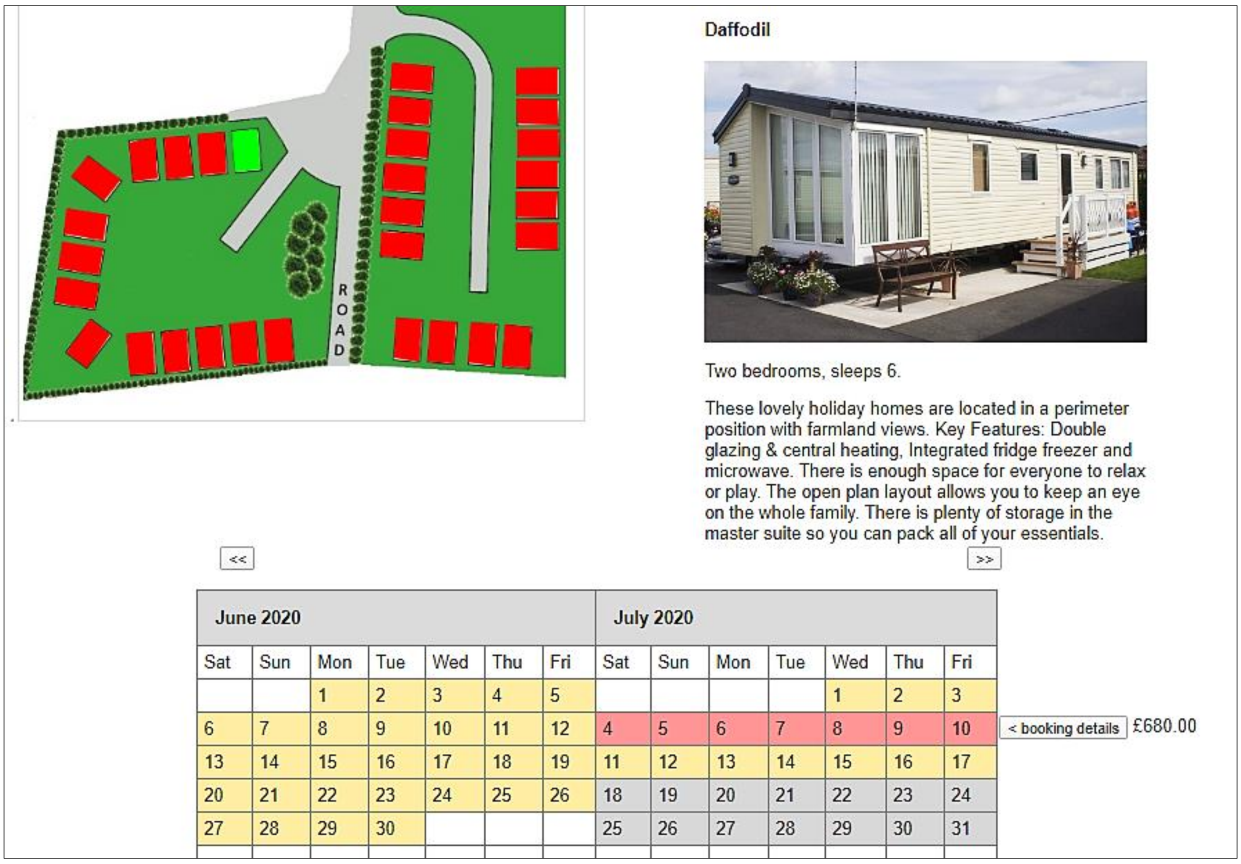
After updating the caravan week records, the program will return to the **updatePrices** page. Here we can display a message to confirm that the update has taken place.

Re-open **updatePrices.php** and add a message and 'continue' button.

```
    if ($includeCaravan=='YES')
    {
       $caravanName=Location::getName($c);
       for ($i=1; $i<=$weekCount;$i++)
       {
        CaravanWeek::setAvailability($c,$weekBeginning[$i], $available,$weekCost);
       }
    }
}
?>
<form method=post action="staffBookings.php?caravanNoWanted=1&monthWanted=6">
<br><br><br>Update completed
<br><br><input type='submit' value='continue'>
</form>
<?

function findWeekBeginning( $firstWeek,$lastWeek)
{
       $year1=substr($firstWeek,0,4);   $year2=substr($lastWeek,0,4);
       $month1=substr($firstWeek,5,2);  $month2=substr($lastWeek,5,2);
```

Save **updatePrices.php** and copy it to the server. Run the website with correct and incorrect test data. Check that the completion message and 'continue' button are displayed, along with any error message, and the button takes the user back to the bookings page.

Carry out a systematic series of tests to check that caravans can be set as unavailable for particular weeks (shown in grey on the calendar display) or the weekly hire charges can be changed for particular weeks.



Daffodil

Two bedrooms, sleeps 6.

These lovely holiday homes are located in a perimeter position with farmland views. Key Features: Double glazing & central heating, Integrated fridge freezer and microwave. There is enough space for everyone to relax or play. The open plan layout allows you to keep an eye on the whole family. There is plenty of storage in the master suite so you can pack all of your essentials.

June 2020							July 2020								
Sat	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Mon	Tue	Wed	Thu	Fri		
		1	2	3	4	5					1	2	3		
6	7	8	9	10	11	12	4	5	6	7	8	9	10	< booking details	£680.00
13	14	15	16	17	18	19	11	12	13	14	15	16	17		
20	21	22	23	24	25	26	18	19	20	21	22	23	24		
27	28	29	30				25	26	27	28	29	30	31		

Further development

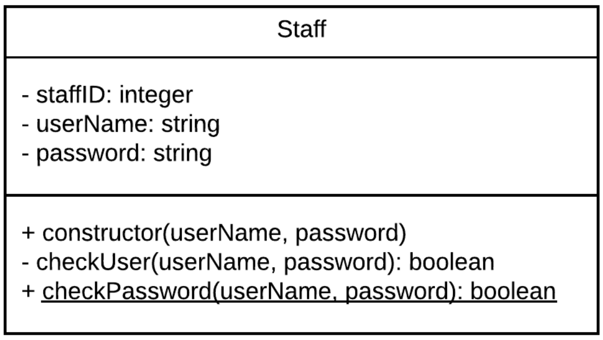
A variety of functions have been added to the caravan booking system, but this is still some distance away from completion. For example, a fully functioning system should allow for cancellation of bookings. Booking confirmation would be sent by e-mail, as in the Airline Booking system in chapter 2 of this book. A graphical input screen may be needed to allow for the easy addition or removal of caravans from the site map.

A variety of booking system websites might use a calendar display similar to this caravan booking system. These might include hotels, restaurants or package holidays. The map display selection function might be used for allocation of pitches for tents or touring caravans in a camp site.

Summary of the object structures

Staff

A Staff object contains the staffID which is set by the database as an auto-number, along with the user name an password. Two methods are included which check the input data for valid log-in details. The public method **checkPassword()** calls the private method **checkUser()** to examine each Staff object in turn, then returns an overall true/false result depending on whether valid log-in details were found.


Staff
- staffID: integer
- userName: string
- password: string
+ constructor(userName, password)
- checkUser(userName, password): boolean
+ checkPassword(userName, password): boolean

Location

A Location object is created for each individual caravan. Attributes specify the caravan number and name, and give (x,y) coordinates for the four corners of a rectangle to represent the caravan on the site plan. A method allows a caravan name to be retrieved by specifying the caravan number. The set of Location objects are created in PHP from the database table, and are then copied to an equivalent set of objects in JavaScript for use in interactive graphics on the web page.

Description

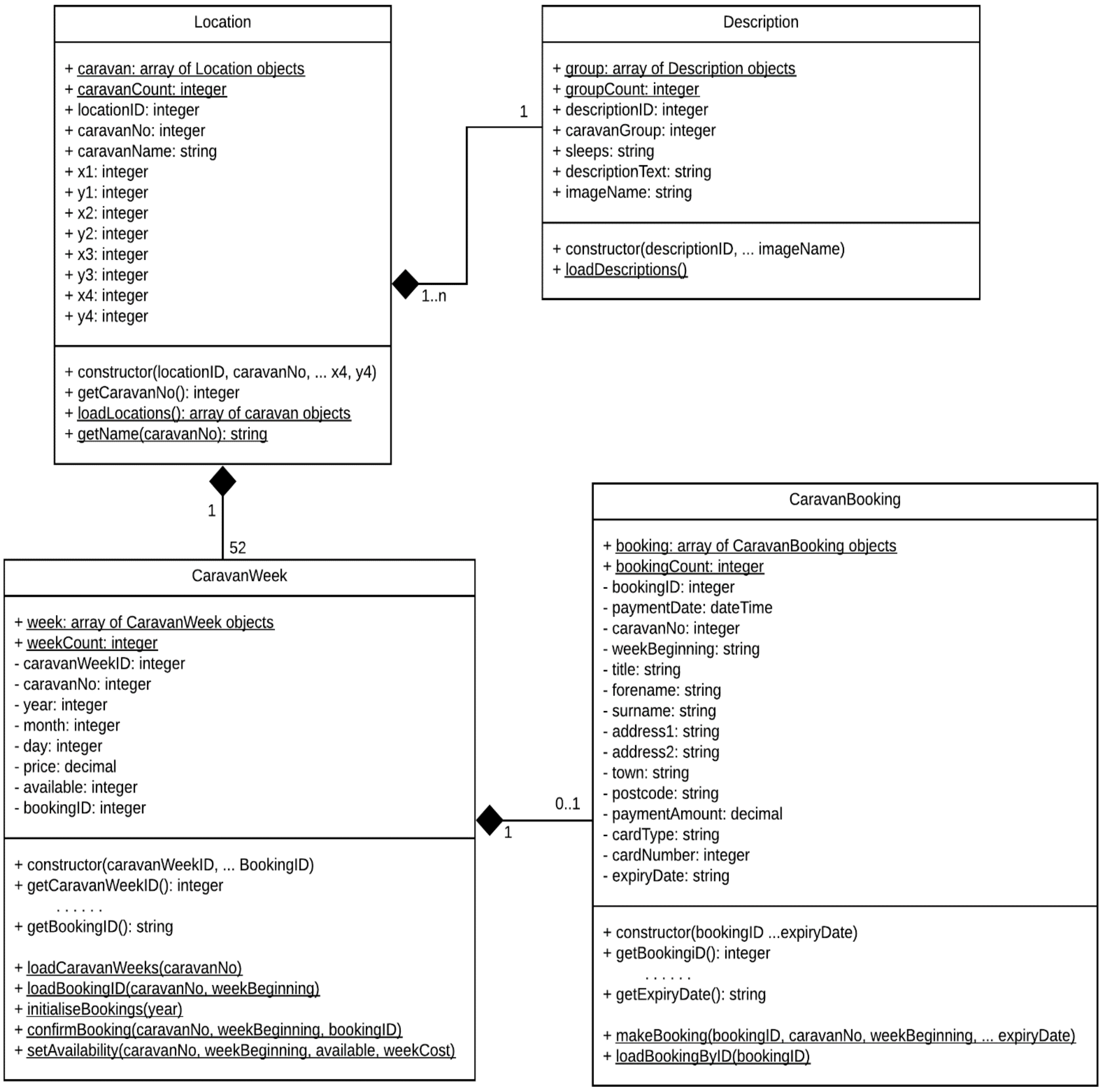
A Description object is provided for each caravan group. It has attributes specifying the number of persons that the caravan can accommodate, a text description of the caravan and its facilities, and the file name for a photograph of a caravan which is representative of the group. The set of Description objects are created in PHP from the database table, and are then copied to an equivalent set of objects in JavaScript for use in interactive graphics on the web page.

CaravanWeek

A CaravanBooking object can be created for each week of the year for each caravan. The object attributes specify the hire charge for the week, the current booking status, and can provide a link to the customer booking details if the caravan is booked for the week. An initialisation method allows the set of CaravanWeek objects to be created for a calendar year, including the setting of hire charges. Methods allow caravan weeks to be specified as available, booked, or unavailable e.g. during maintenance, and for hire charges to be re-set. Methods allow the loading of all caravan week objects for a particular caravan for the year, or just for a specified week.

CaravanBooking

Attributes of CaravanBooking objects record the contact details and payment details for customers. Methods allow a booking to be added to the database, and for a particular booking identified by the bookingID to be retrieved and displayed.



-	private
+	public
underlined	static