

Airline booking system

3

Scenario

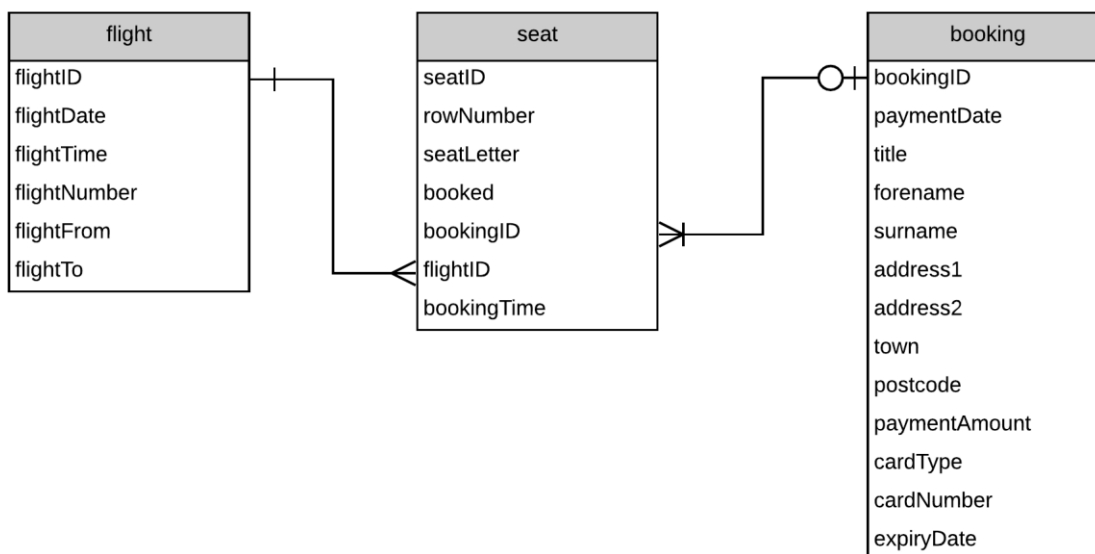
Cambrian Air is a small airline which links Caernarfon airport in North Wales with the larger regional airports in Manchester and Cardiff, allowing passengers to connect with international flights. *Cambrian Air* operates several aircraft with the same 30-seat cabin layout. Daily flights are scheduled in each direction between Caernarfon and Manchester, and Caernarfon and Cardiff.

It is intended that customers may use the web site to select a flight, check the availability of seats, choose and reserve seats, then enter their contact details and make payment. Customers will be allowed 10 minutes to complete the booking process, with reserved seats being released for re-sale if payment is not received by this time.

Design

The program will be constructed around an on-line database containing three principal tables:

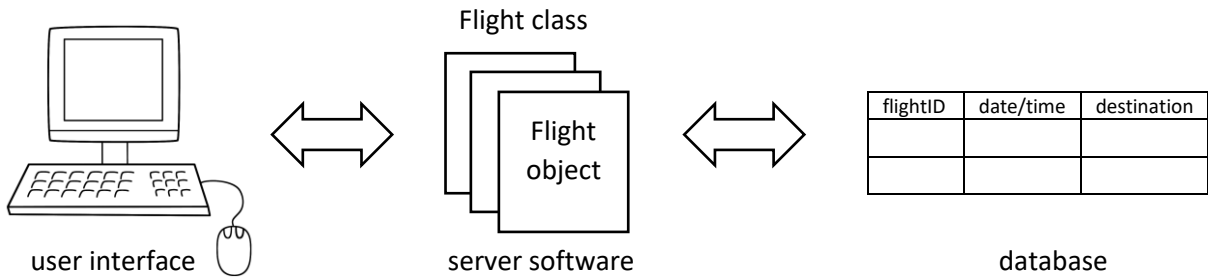
The **flight** table will record the details of individual flights, including the date and time, and the route. The **flightNumber** provided to customers when booking, such as 'CA23', cannot be used as a primary key as this may be re-used by the airline for a particular service each week.



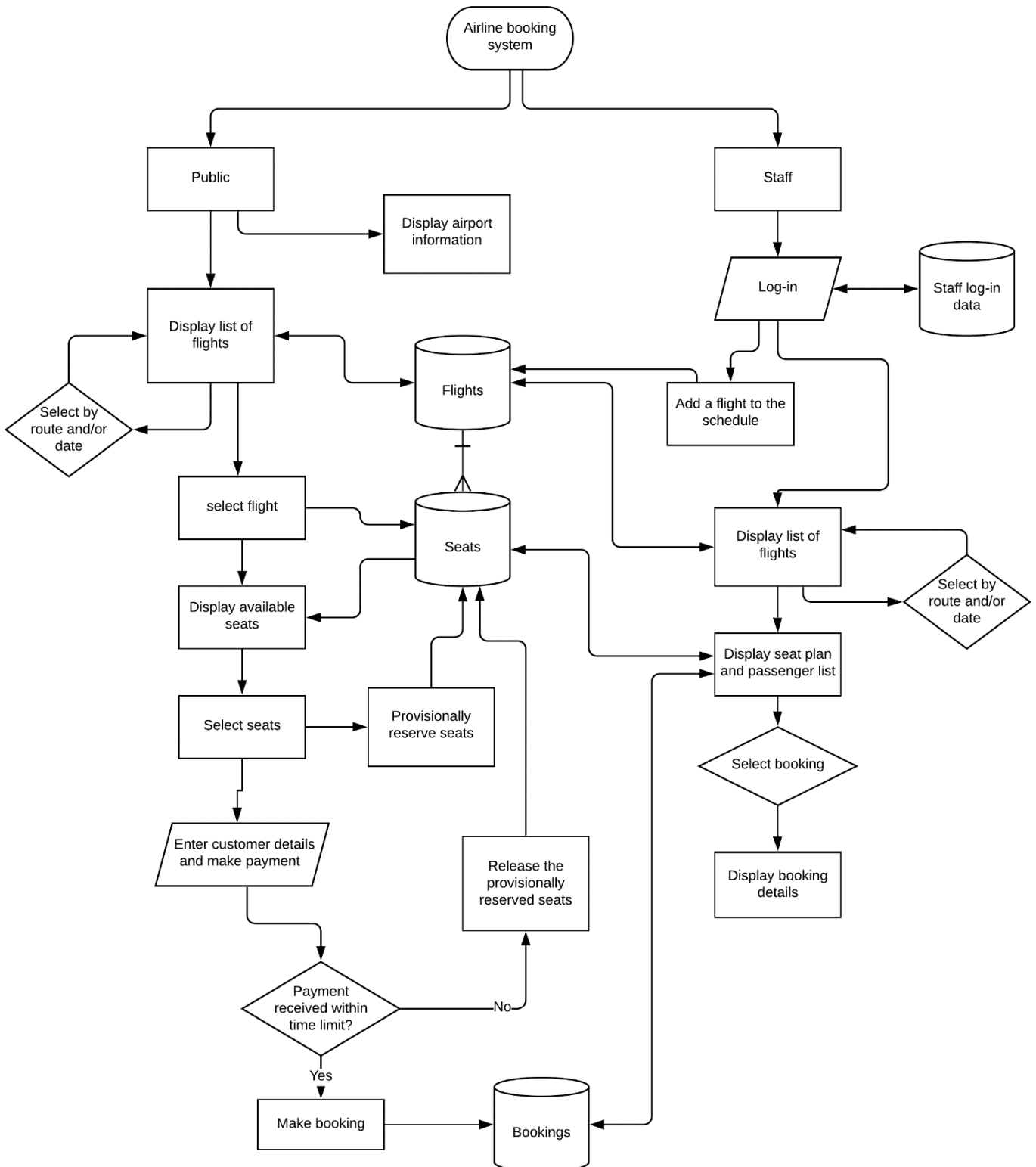
Each flight will be linked to a set of 30 **seat** records, with each seat record relating to only one flight. The **seat** record will uniquely identify the flight by its **flightID**, and show the row and seat letter. A variable will record whether the seat is currently booked, available, or temporarily reserved whilst a customer is completing their on-line booking. When booked, the seat record will link to the **bookingID** to allow access to passenger information.

As in the Hardware Store project in the previous chapter, we will use **classes of objects** as intermediate links between the on-line database and the web page displays.

The class will contain **properties** which are variables describing each object, such as the date and time for a particular flight. The class will also contain **methods** which are functions which can be applied to the objects, such as selecting the seat bookings for a particular flight.



In addition to the public web pages, a password protected section of the site will be available to the airline staff. This will allow passenger bookings to be examined, and staff can set up flights with groups of seats which can be booked.



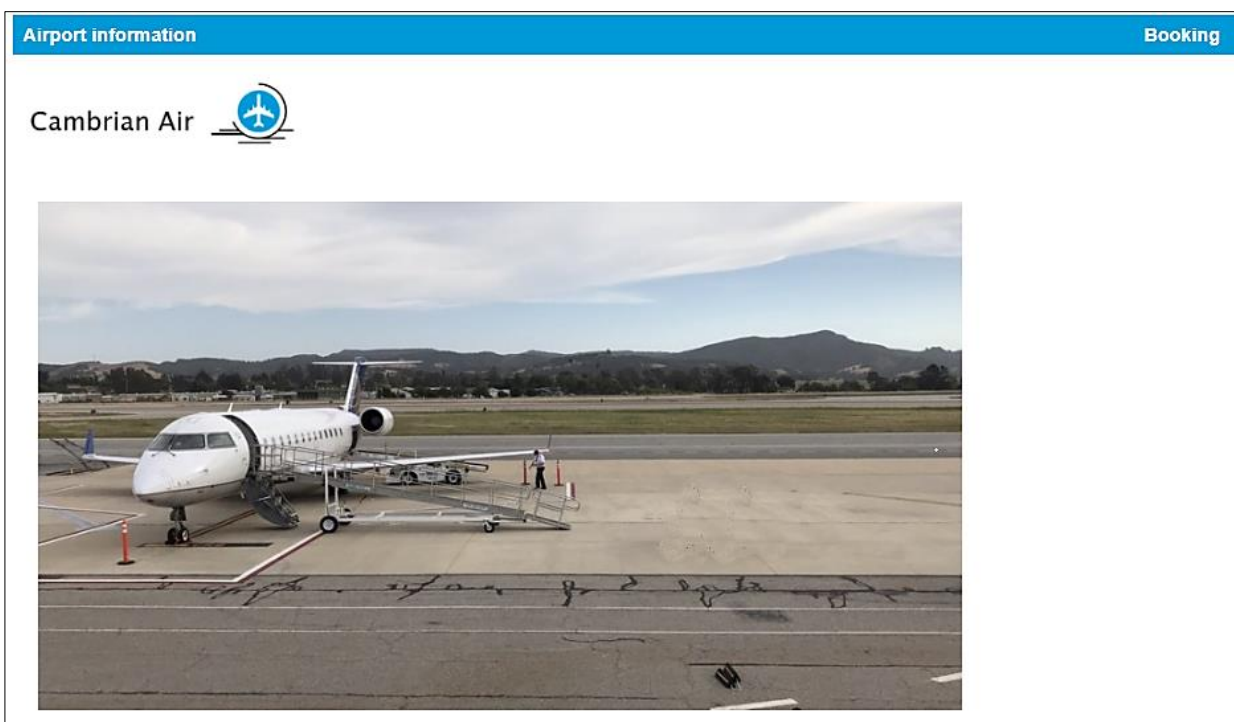
Programming techniques

Pages will be produced using HTML augmented by a style sheet. The database will be accessed with PHP and SQL code. A timer in PHP will operate the real time temporary seat reservation as customers complete their bookings.

The booking system centres around a seating plan for the aircraft. An array of HTML button components will represent seats, and can be displayed in colour according to availability.

Method

The web site will begin with a home page providing general information about the airport. From this page, customers will be able to choose a 'Booking' menu option to obtain information about flight times and seat availability.



Begin by setting up a new folder on your local computer and on the server with the name 'airline'. Create a logo for the company similar to the heading above, and save this as a graphics image **logo.png**. Obtain a suitable photograph of a small passenger aircraft, and save this as **aircraft.jpg**. Copy the two graphics files to the server.

Open a blank text file and set up a style sheet with the lines of code shown in the two boxes below.

```
body
{
  font-family: arial, sans-serif;
}
table.menu
{
  border-collapse: collapse;
  width: 100%;
}
```

```

th.menu
{
  text-align: left;
  padding: 8px;
  background-color: rgb(0, 153, 216);
  color: white;
}
a:link, a:visited
{
  color: white;
  text-decoration: none;
}

```

Save the file as **styleSheet.css** and copy it to the server.

Open a blank file and add the lines of code below. Save the file as **index.php**, then copy this to the server.

```

<html>
<head>
  <title> Cambrian Air </title>
  <link rel="Stylesheet" type="text/css" href="styleSheet.css" />
</head>
<body>
  <table class=menu>
    <tr><th class=menu>
      <a href=index.php>
        Airport information
      </a>
    <th class=menu>
      <a href=selectFlight.php>
        Booking
      </a>
    </tr>
  </table>
  <p>
  
  <table cellpadding=20>
    <tr>
      <td>
        <img src='aircraft.jpg'>
      </td>
    </tr>
  </table>
  </body>
</html>

```

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

www.website.com/airline

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.

Before developing the public booking system further, we will create a staff section for the web site where flights can be set up and seats made available for booking. The staff pages will be password protected in a similar way to the staff section of the Hardware Store project in the chapter 2.

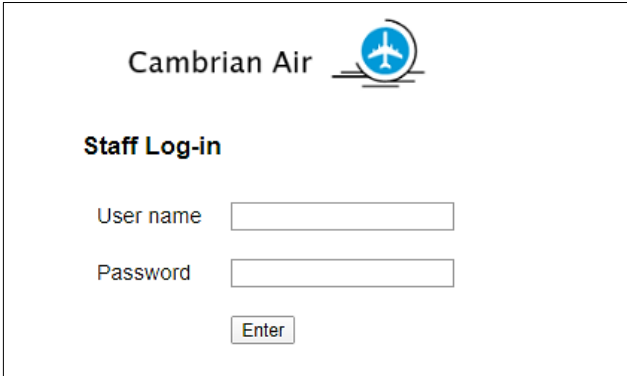
Begin by creating a staff log-in screen. Open a blank file and add the lines of code below.

```

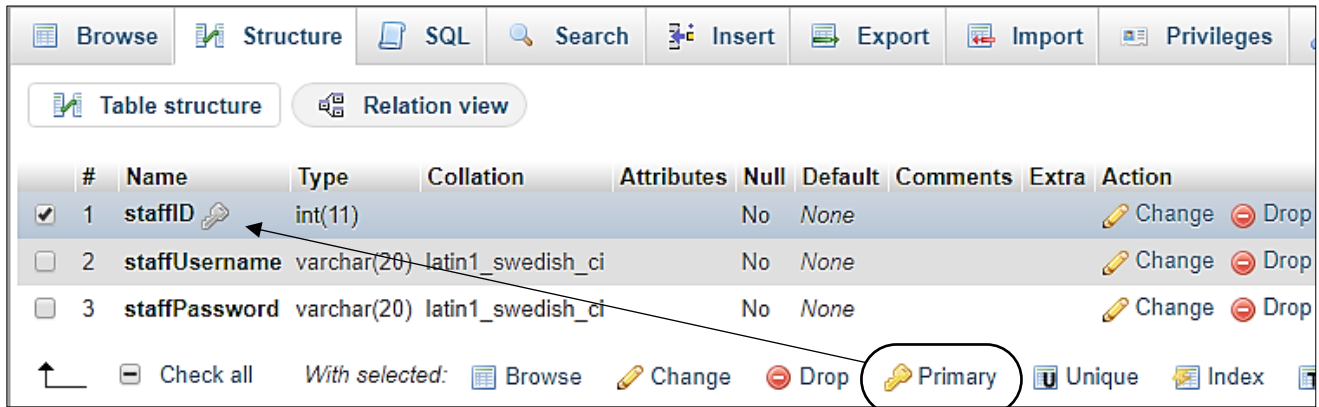
<?
  session_start();
  $_SESSION['login']='NO';
?>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="styleSheet.css" />
  <title>Cambrian Air</title>
</head>
<body>
  <center>
    <form action="staffDisplayBookings.php" method="post">
      
      <table cellpadding=20>
        <tr><td>
          <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></td></tr>
        </table>
      </form>
    </body>
  </html>

```

Save the file as **staffLogin.php** and copy this to the server. Run the **staffLogin.php** page. Entry boxes should appear. Notice that the password box is set to conceal the text which is being entered.



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



Set the **staffID** field to be the primary key. Click the Change option on the **staffID** line, then tick the auto increment (**A_I**) box. Further information about setting up the staff table will be found in the Hardware Store project in Chapter 2.

Use the **Insert** option to add several members of staff as test data.

We will now create the Staff class which will act as an interface between the database and the local computer. Open a new file and save this as **Staff.php**. By convention, the names of classes begin with an upper case letter. Add the lines of code shown in the boxes below:

```
<?
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;
    }

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

Continued:

```

        $found=false;
        for ($i=1;$i<=$num;$i++)
        {
            $answer= $staff[$i]->checkUser($userWanted,$passWanted);
            if ($answer==true)
            {
                $found=true;
            }
        }
        return $found;
    }
}
?>

```

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

After logging in, the first web page that a member of staff will access is **staffDisplayBookings.php**, where we can provide a list of flights and allow staff to inspect the aircraft seat plan and bookings for any flight.

Open a blank file and save this as **staffDisplayBookings.php**. Add the program code below.

```

<?
    session_start();
    $user=$_REQUEST['user'];
    $pass=$_REQUEST['pass'];
    $login=$_SESSION['login'];
?>
<html>
<head>
    <title> Cambrian Air </title>
    <link rel="stylesheet" type="text/css" href="styleSheet.css" />
</head>
<body>
    <?
        if (!($_SESSION['login']=='YES'))
        {
            include('Staff.php');
            if (Staff::checkPassword($user,$pass)==false)
                header('Location: staffLogin.php');
            else
                $_SESSION['login']='YES';
        }
        include('staffMenu.php');
    ?>
    <p>
</body>
</html>

```

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

On loading this page, the user name and password entered on the log-in screen will be accessed, then passed to a **Staff class file** for checking against the database table. The **checkPassword()** function will return a **true** value if the log-in details are valid, or **false** if incorrect. A result of **false** will cause the program to return immediately to the staff log-in page.

All pages within the staff section of the web site will display the same menu options along the top of the page. We can save repetition by creating a separate file for the menu program code, which can be included in each staff page.

Open a blank file. Add the program code shown below, then save this as **staffMenu.php**. Copy the file to the server.

```
<table class=menu>
  <tr><th class=menu>
    STAFF OPTIONS
  <th class=menu>
    <a href=scheduleFlight.php>
      Add flight to schedule
    </a>
  <th class=menu>
    <a href=staffDisplayBookings.php>
      Display flight schedule
    </a>
</table>
```

Before running the staff log-in system, 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 **staffLogin.php** and enter a correct user name and password. The staff menu should be displayed.

STAFF OPTIONS	Add flight to schedule	Display flight schedule

If an incorrect user name or password is entered, the program should return to the staff log-in page.

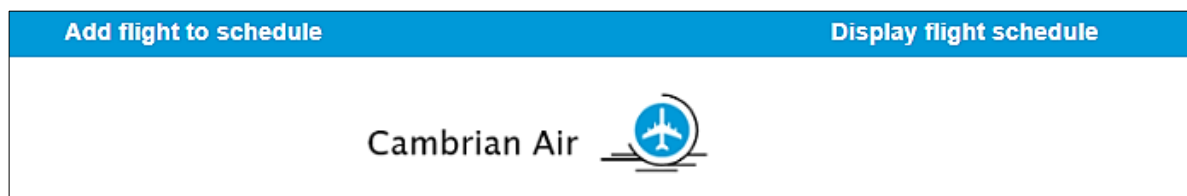
We can now work on the 'Add flight to schedule' option. Open a blank file and save this as **scheduleFlight.php**. Add the lines of program code shown below. Re-save the updated file and copy it to the server.


```

<html>
<head>
  <title> Cambrian Air </title>
  <link rel="stylesheet" type="text/css" href="styleSheet.css" />
</head>
<body>
  <?
    include('staffMenu.php');
  ?>
  <center>
  <p>
  
  </center>
</body>
</html>

```

Run the staff web site and select the 'Add flight to schedule' option. The **scheduleFlight.php** page should open, displaying the menu and company logo.



We will now add components to allow the input of flight details. Add the lines of program code shown on the next page to the **scheduleFlight.php** file. Save the updated file and copy it to the server.

Run the staff web site. Select the 'Add flight to schedule' option, then check that the components on the input page are displayed correctly.

The date input will open a calendar to allow a date to be easily selected. The flight departure and destination airports will be selected from drop-down lists.

```

<center>
<p>

<p>
<form method=post action='addFlight.php' onsubmit='return submitForm(this)'\>
<table cellpadding=20>
<tr>
<td><table cellspacing=20></td>
</tr>
<tr>
<td>Date</td>
<td><input type="date" name="flightDate"></td>
</tr>
<tr>
<td>Departure time</td>
<td><input type="time" name="flightTime"></td>
</tr>
<tr>
<td>Flight number</td>
<td><input type="text" name="flightNumber" size=12></td>
</tr>
<tr>
<td>From</td>
<td><select name="flightFrom">
<option></option>
<option>Caernarfon</option>
<option>Cardiff</option>
<option>Manchester</option>
</select></td>
</tr>
<tr>
<td>To</td>
<td><select name="flightTo">
<option></option>
<option>Caernarfon</option>
<option>Cardiff</option>
<option>Manchester</option>
</select></td>
</tr>
<tr>
<td><td><input type='submit' value='Add flight to schedule' ></td>
</tr>
</table>
</form>
</center>
</body>
</html>

```

Return to the **scheduleFlight.php** file and add lines of code to produce a JavaScript confirm box, which will appear when the 'Add flight' button is clicked.

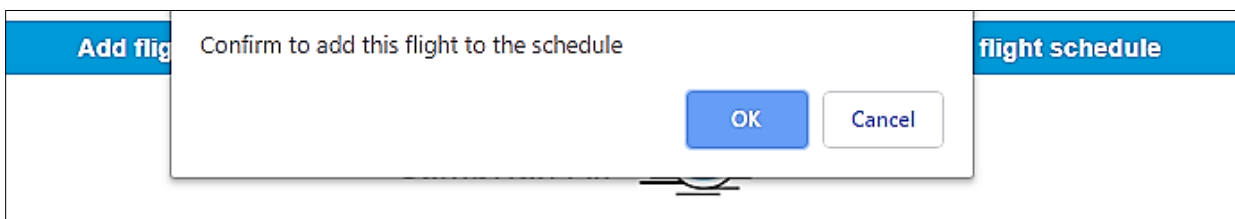
```

</table>
</form>

<script>
  function submitForm()
  {
    return confirm('Confirm to add this flight to the schedule')
  }
</script>
</center>
</body>
</html>

```

Save the updated **scheduleFlight.php** file and copy it to the server. Run the scheduleFlight page. Click the 'Add flight' button, check that the confirm message is displayed and then select 'Cancel'.




In order to store flight details, a database table will be needed. Open the PHP MyAdmin site for your database and add a new table. Give this the name '**flight**' and add fields as shown below: **flightID** is identified as the primary key and set as an auto-number; **flightDate** and **flightTime** have date and time data types; and **flightNumber**, **flightFrom** and **flightTo** are set as varchar with sizes of 12, 30 and 30 characters respectively.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	flightID	int(11)			No	None		AUTO_INCREMENT
2	flightDate	date			No	None		
3	flightTime	time			No	None		
4	flightNumber	varchar(12)	latin1_swedish_ci		No	None		
5	flightFrom	varchar(30)	latin1_swedish_ci		No	None		
6	flightTo	varchar(30)	latin1_swedish_ci		No	None		

As each new flight is added to the schedule, a set of 30 seats must be created and made available for booking. To do this, we must also add a seat table to the database. Create another new table and give this the name '**seats**'. Add fields as shown below.

- **seatID** is identified as the primary key and set as an auto-number.
- **rowNumber** is an integer and **seatLetter** is varchar with a length of 1. Together they make up the seat location, such as seat 6B.
- **booked** is specified as a tiny integer. It will take values of 0 (seat available), 1 (seat temporarily reserved during the booking procedure) or 2 (booked).
- **journeyBookingID** and **flightID** are integers, whilst **bookingTime** is datetime data type.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	seatID 	int(11)			No	None		AUTO_INCREMENT
2	rowNumber	int(11)			No	None		
3	seatLetter	varchar(1)	latin1_swedish_ci		No	None		
4	booked	tinyint(1)			No	None		
5	journeyBookingID	int(11)			No	None		
6	flightID	int(11)			No	None		
7	bookingTime	datetime			No	None		

In this project we will use an object oriented approach, with object classes forming interfaces between the database and the local computer display. Begin by creating a **Flight** class. Open a blank file and add the program code below.

```
<?
class Flight
{
    private $flightID;
    private $flightDate;
    private $flightTime;
    private $flightNumber;
    private $flightFrom;
    private $flightTo;
    private function __construct($flightID, $flightDate, $flightTime,
                                $flightNumber, $flightFrom, $flightTo)
    {
        $this->flightID = $flightID;
        $this->flightDate = $flightDate;
        $this->flightTime = $flightTime;
        $this->flightNumber = $flightNumber;
        $this->flightFrom = $flightFrom;
        $this->flightTo = $flightTo;
    }
}
?>
```

Please note that the long line at the start of the constructor function should be entered by continuous typing without a line break. Save the file as **Flight.php** and copy it to the server.

A **Seat** class will also be required. Open another blank file and add the program code in the two boxes below.

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

```
<?
class Seat
{
    public static $seatObj = array();
    private $seatID;
    private $rowNumber;
    private $seatLetter;
    private $booked;
    private $bookingID;
    private $flightID;
    private $bookingTime;
```

Continued:

```

public function __construct($seatID, $rowNumber, $seatLetter, $booked,
                           $bookingID, $flightID, $bookingTime)
{
    $this->seatID = $seatID;
    $this->rowNumber = $rowNumber;
    $this->seatLetter = $seatLetter;
    $this->booked = $booked;
    $this->bookingID = $bookingID;
    $this->flightID = $flightID;
    $this->bookingTime = $bookingTime;
}
}
?>

```

We will now create an **addFlight.php** file which will be activated when the 'Add flight' button is clicked on the flight input screen. Open a blank file and add the program code below.

```

<?
$flightDate=$_REQUEST['flightDate'];
$flightTime=$_REQUEST['flightTime'];
$flightNumber=$_REQUEST['flightNumber'];
$flightFrom=$_REQUEST['flightFrom'];
$flightTo=$_REQUEST['flightTo'];

echo"<p>flight date: ".$flightDate;
echo"<p>flight time: ".$flightTime;
echo"<p>flight number: ".$flightNumber;
echo"<p>flight from: ".$flightFrom;
echo"<p>flight to: ".$flightTo;
?>

```

Save the file as **addFlight.php** and copy it to the server. Run the staff web site and select the 'Add flight to schedule' option. Enter details for a flight, then click OK to add the flight. Check that the addFlight.php page is then loaded and details of the flight have been carried over correctly.

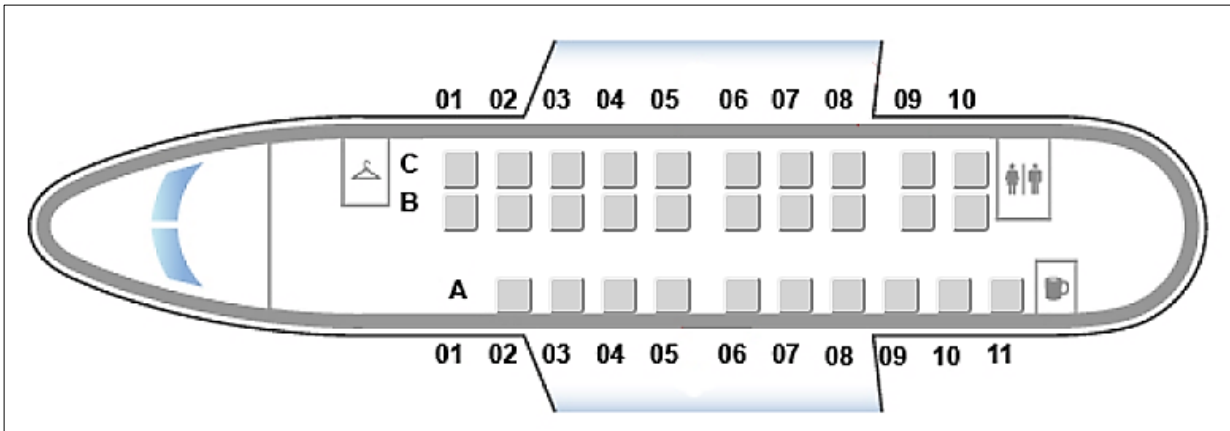
The screenshot displays a web interface for adding a flight to a schedule. On the left, a form titled 'Add flight' contains the following fields:

- Date:
- Departure time:
- Flight number:
- From: - To:

Below the form is an 'Add flight to schedule' button. A modal dialog box is open over the form, titled 'Confirm to add this flight to the schedule', with 'OK' and 'Cancel' buttons. On the right, a 'flight schedule' box displays the following details:

- flight date: 2020-02-06
- flight time: 10:30
- flight number: CA29
- flight from: Caernarfon
- flight to: Cardiff

Flights can now be set up and sets of available seats created for the booking system. We will assume that the type of aircraft operated by Cambrian Air has seats in 11 numbered rows, with three seats A to C in each row. However, due to the layout of the aircraft, there are no seats located at positions 1A, 11B and 11C.



Re-open the **Flight.php** class file and insert an **addFlight()** method. Please note the long lines, marked by the curved arrows below, which should be inserted without line breaks.

```

public static function addFlight($flightID, $flightDate, $flightTime,
                                $flightNumber, $flightFrom, $flightTo)
{
    include('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    $query="INSERT INTO flight VALUES ('$flightID','$flightDate','$flightTime',
                                        '$flightNumber','$flightFrom','$flightTo)";
    $result=mysqli_query($conn, $query);
    $flightID = mysqli_insert_id($conn);
    mysqli_close($conn);
    Seat::createSeats($flightID);
}
}
?>

```

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

The **addFlight()** method will in turn call a method to create the set of 30 seats for the flight. Re-open the **Seat.php** class file and insert a **createSeats()** method as shown in the two boxes below. The program code uses IF... operators to avoid saving records for seats 1A, 11B and 11C.

```

public static function createSeats($flightID)
{
    include('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    for ($r = 1; $r <= 11; $r++)
    {

```

Continued:

```

        if ($r > 1)
        {
            $query="INSERT INTO seats VALUES ('','$r','A','0','0',
                                                '$flightID','0000-00-00 00:00:00')";
            $result=mysqli_query($conn, $query);
        }
        if ($r < 11)
        {
            $query="INSERT INTO seats VALUES ('','$r','B','0','0',
                                                '$flightID','0000-00-00 00:00:00')";
            $result=mysqli_query($conn, $query);
            $query="INSERT INTO seats VALUES ('','$r','C','0','0',
                                                '$flightID','0000-00-00 00:00:00')";
            $result=mysqli_query($conn, $query);
        }
    }
    mysqli_close($conn);
}
?>

```

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

The final step in adding a flight is to call the methods in the Flight and Seat class files which will add the necessary records to the database tables.

Re-open the **addFlight.php** file. The 'echo' display lines can now be removed and replaced by code to call the **addFlight()** method in the Flight class. Update the file as shown below:

```

<?
    $flightDate=$_REQUEST['flightDate'];
    $flightTime=$_REQUEST['flightTime'];
    $flightNumber=$_REQUEST['flightNumber'];
    $flightFrom=$_REQUEST['flightFrom'];
    $flightTo=$_REQUEST['flightTo'];

    include('Flight.php');
    include('Seat.php');
    Flight::addFlight('', $flightDate, $flightTime, $flightNumber,
                    $flightFrom, $flightTo);
    header('Location: staffDisplayBookings.php');
?>

```

Save the **addFlight.php** file and copy it to the server. Run the staff web site, select the 'Add flight to schedule' option and enter the details of a flight. The program should return to the staffDisplayBookings page. Go to the database.

Check that the flight details have been uploaded correctly to the **flight** table, and that a set of seats from 1B to 11A has been created in the **seats** table. If all is working correctly, add several more flights.

We can now work on display of flight information. Re-open the **staffDisplayBookings.php** file and add the lines of program code below. These create a drop-down selection for the departure airport.

```

    <th class=menu>
      <a href=staffDisplayBookings.php>
        Display flight schedule
      </a>
    </th>
  </table>
  <p>
  <table cellpadding=20>
  <tr>
    <td>
      <form method='post' action='staffDisplayBookings.php'>
      <table cellspacing=20>
        <tr>
          <td colspan=2>
            Search using one or more criteria:</td>
        </tr>
        <tr>
          <td>From</td>
          <td>
            <?
              $airport[0]='';
              $airport[1]='Caernarfon';
              $airport[2]='Cardiff';
              $airport[3]='Manchester';
              $wantedFlightFrom=$_REQUEST['flightFrom'];
              $wantedFlightTo=$_REQUEST['flightTo'];
              $wantedFlightDate=$_REQUEST['flightDate'];
              echo"<select name='flightFrom'>";
              for ($i=0;$i<=3; $i++)
              {
                if ($wantedFlightFrom==$airport[$i])
                  echo"<option selected>";
                else
                  echo"<option>";
                echo $airport[$i]."</option>";
              }
              echo"</select>";
            <?>
          </td>
        </tr>
        <tr>
          <td></td>
          <td>
            <input type='submit' value='Find flight'></td>
          </tr>
        </table>
      </form>
    </td>
  </table>
</body>
</html>

```

Continue by adding the further lines of program code below, which allow the destination airport and flight date to be selected.


```

        echo"</select>";
    ?>
</td>
</tr>

<tr>
    <td>To</td>
    <td>
        <?
            echo"<select name='flightTo'>";
            for ($i=0;$i<=3; $i++)
            {
                if ($wantedFlightTo==$airport[$i])
                    echo"<option selected>";
                else
                    echo"<option>";
                echo $airport[$i]."</option>";
            }
            echo"</select>";
        ?>
    </td>
</tr>
<tr>
    <td>Date</td>
    <td>
        <?
            echo"<input type='date' name='flightDate' value='$wantedFlightDate'>";
        ?>
    </td>
</tr>

<tr>
    <td></td>
    <td>
        <input type='submit' value='Find flight'>
    </td>
</tr>

```

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

Open the **staffDisplayBookings** page. Make a selection of airports and date, then click the 'find flights' button. When the button is clicked, the page is reloaded. Check that the flight requirements are still displayed correctly in the input boxes.

Reloading the page allows PHP code to be run on the server to select flight information which we can display in a table.

STAFF OPTIONS		Add flight to schedule
Search using one or more criteria:		
From	<input type="text" value="Caernarfon"/>	
To	<input type="text" value="Manchester"/>	
Date	<input type="text" value="16/04/2020"/>	
		<input type="button" value="Find flight"/>

Return to the **staffDisplayBookings.php** file and add the program code below to create headings for the table of flights.

```

        <input type='submit' value='Find flight'>
    </td>
</tr>
</table>
</form>

</td>
<td>
    <form method='post' action='staffDisplayBookings.php'>
    <table border=0>
    <tr>
        <td width=200>Flight schedule</td>
        <td><input type='submit' value='Show all flights'></td>
    </table>
    </form>
    <form method='post' action='staffDisplayBookings2.php'>
    <p>
    <table class = f>
    <tr>
        <td>Date</td>
        <td>Time</td>
        <td>Flight number &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</td>
        <td>From</td>
        <td>To</td>
    </tr>
    </table>
    </td></tr>
</table>
</body>
</html>

```

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

The next step is to display flight details in the table which meet the required search criteria.

Search using one or more criteria:

From

To

Date

Flight schedule

Date	Time	Flight number	From	To
02/09/2019	10:00:00	CA22	Caernarfon	Manchester
04/09/2019	10:30:00	CA12	Caernarfon	Cardiff
04/09/2019	15:45:00	CA14	Caernarfon	Cardiff
05/09/2019	11:20:00	CA32	Caernarfon	Manchester
05/11/2019	10:17:00	CA50	Caernarfon	Cardiff
14/11/2019	09:15:00	CA51	Caernarfon	Manchester

The table can be formatted with alternate lines shaded in grey. To do this, open the **styleSheet.css** file and add the commands shown below. These will only apply to the flight information table which we have designated as **'class=f'** within the program listing.

```

table.f
{
    font-family: arial, sans-serif;
    border-collapse: collapse;
    width: 100%;
}
td.f, th.f
{
    border: 1px solid #dddddd;
    text-align: left;
    padding: 8px;
}
tr.f:nth-child(even)
{
    background-color: #dddddd;
}

```

Save the updated **styleSheet.css** file and copy it to the server.

The database search will be carried out by a method which we will add to the Flight class. Open the **Flight.php** file and begin by adding the lines of code shown below. These create two new variables:

- **\$itemcount**, which is the number of flight records found which meet the specified search criteria. These records will be made into objects.
- **\$flightObj[]**, which is an array for identifying the objects produced from the flight records. The first object will be named **\$flightObj[1]**, the second will be **\$flightObj[2]**, etc.

```

<?
class Flight
{
    public static $itemCount = 0;
    public static $flightObj = array();

    private $flightID;
    private $flightDate;
    private $flightTime;
    private $flightNumber;
    private $flightFrom;
    private $flightTo;
}

```

Move now to the end of the **Flight.php** file and insert a **selectFlights()** method as shown in the two boxes below.

```

public static function selectFlights($wantedFlightDate, $wantedFlightFrom,
                                     $wantedFlightTo)
{
    include ('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    $query="SELECT * FROM flight ORDER BY flightDate";
    $result=mysqli_query($conn, $query);
    $num=mysqli_num_rows($result);
    mysqli_close($conn);
    $found=false;
    $i=1;
    $itemCount=0;
}

```

Continued:

```

while ($i <= $num)
{
    $row=mysqli_fetch_assoc($result);
    $flightID=$row["flightID"];
    $flightDate=$row["flightDate"];
    $flightTime=$row["flightTime"];
    $flightNumber=$row["flightNumber"];
    $flightFrom=$row["flightFrom"];
    $flightTo=$row["flightTo"];
    $i++;
}
}
?>

```

This method begins by opening the **flight** table in the database and loading all the records. A loop then obtains the field values for each of the records in turn. To carry out a search for suitable flights, the departure and destination airports and the flight date must be checked if the user has specified a preference. Add the following lines of code within the loop.

```

while ($i <= $num)
{
    $flightFrom=$row["flightFrom"];
    $flightTo=$row["flightTo"];

    $found=true;
    if (strlen($wantedFlightFrom)>2)
    {
        if ($wantedFlightFrom != $flightFrom)
            $found = false;
    }
    if (strlen($wantedFlightTo)>2)
    {
        if ($wantedFlightTo != $flightTo)
            $found = false;
    }
    if (strlen($wantedFlightDate)>2)
    {
        if ($wantedFlightDate != $flightDate)
            $found = false;
    }
    if ($found==true)
    {
        $obj = new Flight($flightID, $flightDate, $flightTime, $flightNumber,
            $flightFrom, $flightTo);
        $itemCount++;
        Flight::$flightObj[$itemCount] = $obj;
    }
    $i++;
}

Flight::$itemCount=$itemCount;
return $itemCount;
}
}
?>

```


Run the staff web site. Use the 'Add flight' option to create some additional flights.

Enter various flight selections and check that suitable flights are displayed. Also check that the 'Show all flights' option works correctly.

Search using one or more criteria:

From

To

Date

Flight schedule

Date	Time	Flight number	From	To	
04/09/2020	10:30:00	CA12	Caernarfon	Cardiff	<input type="button" value="display bookings"/>
04/09/2020	15:45:00	CA14	Caernarfon	Cardiff	<input type="button" value="display bookings"/>
05/11/2020	10:17:00	CA50	Caernarfon	Cardiff	<input type="button" value="display bookings"/>
06/02/2021	10:30:00	CA29	Caernarfon	Cardiff	<input type="button" value="display bookings"/>

The buttons alongside each of the flight records will display the bookings made for that flight. Before working on this option, we will return to the public section of the website to develop the on-line booking system and enter test data for seat bookings.

We will now create a page which will be loaded when a customer selects the 'Booking' option from the public menu. This page will display a flight list very similar to the staff flight list. The only differences will be the menu options, and the button alongside each flight record which now offers a 'make booking' option:

Airport information
Booking

Cambrian Air

Search using one or more criteria:

From

To

Date

Flight schedule

Date	Time	Flight number	From	To	
02/09/2020	14:00:00	CA21	Cardiff	Caernarfon	<input type="button" value="make booking"/>
02/09/2020	16:00:00	CA16	Cardiff	Caernarfon	<input type="button" value="make booking"/>
07/09/2020	08:10:00	CA19	Cardiff	Caernarfon	<input type="button" value="make booking"/>

We can save time by creating the new page from **staffDisplay Bookings.php** and making a few alterations where necessary.

Load the file **staffDisplay Bookings.php** into a text editor, then re-save this with the file name **selectFlight.php**. Make the series of changes shown below:

```

<html>
<head>
  <title> Cambrian Air </title>
  <link rel="stylesheet" type="text/css" href="styleSheet.css" />
</head>
<body>

```

remove the lines of PHP code before the opening <html> tag

```

<html>
<head>
  <title> Cambrian Air </title>
  <link rel="stylesheet" type="text/css" href="styleSheet.css" />
</head>
<body>
  <table class=menu>
  <tr>
    <th class=menu>
      <a href=index.php>Airport information</a></th>
    <th class=menu>
      <a href=selectFlight.php>Booking</a></th>
  </tr>
</table>

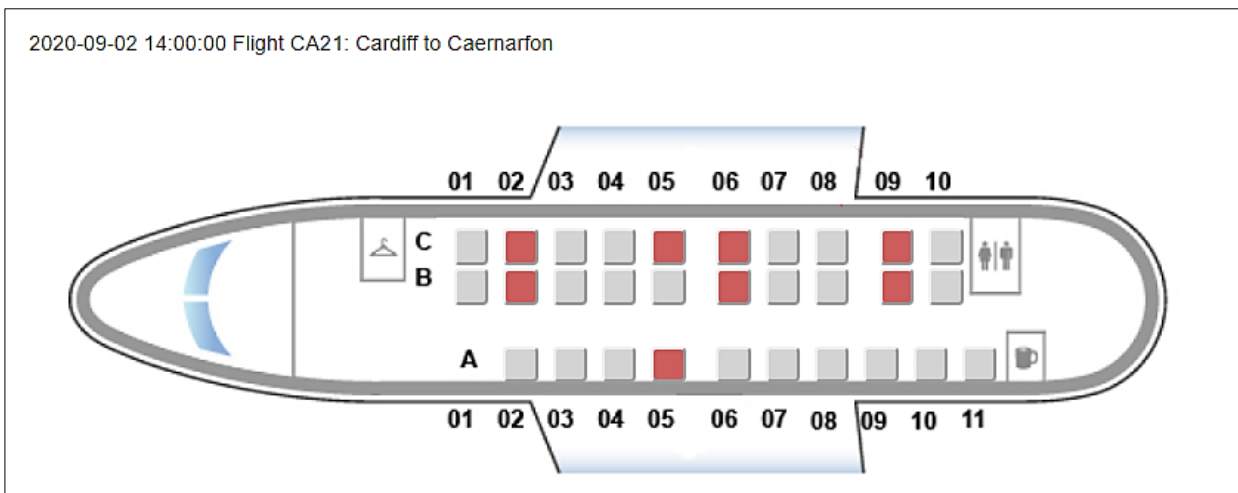
  <p>
  <table cellpadding=20>
  <tr>
  <td>
    <form method='post' action='selectFlight.php'>
      <table cellspacing=20>
      <tr>
        <td colspan=2>
          Search using one or more criteria:
        </td>
        . . . . .
        <input type='submit' value='Find flight'>
      </td>
      </tr>
      </table>
    </form>
  </td>
  <td>
    <form method='post' action='selectFlight.php'>
      <table border=0>
      <tr>
        <td width=200>Flight schedule</td>
        <td><input type='submit' value='Show all flights'></td>
      </table>
    </form>
    <form method='post' action='displayBookings.php'>
      <p>
      <table class = f>
        <tr>
          <td>Date</td>
          <td class = f>".Flight::$flightObj[$k]->getFlightFrom()."</td>";
          <td class = f>".Flight::$flightObj[$k]->getFlightTo()."</td>";
          <td class = f>";
          <td class = f>".Flight::$flightObj[$k]->getFlightID().">make booking</td>";
        </tr>";
      </table>
    </form>
  </td>
  </tr>
  </table>

```

- The PHP code block which appears above the `<html>` line has been removed.
- The PHP code block at the start of `<body>` which checks the staff password has been removed.
- The staff menu has been replaced by the public menu.
- The first `<form>` command has been amended to re-load the page 'selectFlight.php' when the 'Find flight' button is clicked.
- Similarly, the second `<form>` command has been amended to re-load the 'selectFlight.php' page when the 'Show all flights' button is clicked .
- The third `<form>` command has been amended to load the page 'displayBookings.php' if the 'make booking' button is clicked alongside any of the flight records.
- The button caption has been changed from 'display bookings' to 'make booking'.

Save the updated **selectFlight.php** file and copy it to the server. Run the public homepage and select the 'Booking' option. Check that the table of flights is displayed correctly, and can be searched according to route or flight date.

We will now produce a page to show the seating plan for the aircraft, with available seats for the selected flight colour coded in grey.



Open a new blank file and save this as **displayBookings.php**. Add the program code shown below. This begins by displaying the menu and company logo as on previous pages. The **flightID** number of the flight selected from the table is then displayed. A temporary 'echo' line shows this number for test purposes.

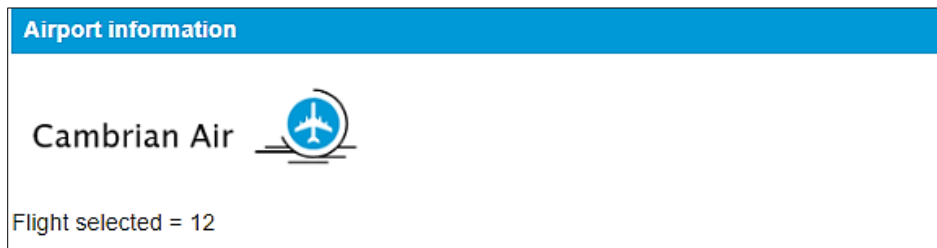
```

<html>
<head>
  <title> Cambrian Air </title>
  <link rel="stylesheet" type="text/css" href="styleSheet.css" />
</head>
<body>
  <table class=menu>
  <tr>
    <th class=menu>
      <a href=index.php>Airport information</a>
    <th class=menu>
      <a href=selectFlight.php>Booking</a>
  </table>
  <p>
  <?
    $flightSelected=$_REQUEST['flightSelected'];
    echo"Flight selected = ". $flightSelected;
  ?>
</body>
</html>

```


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

Run the public web site. Select the 'Booking' option. Choose a flight from the table display, then click the 'make booking' button. The **displayBookings.php** page should open, with the flightID value displayed.



Go to the PHP MyAdmin page and check in **flight** table of the database that the correct **flightID** number is shown for the selected flight.

The next step is to display the date, time and route for the selected flight. Re-open the **Flight.php** class file and add the **selectFlightByID()** method shown below. This will use the **flightID** number to create an object containing details of the flight.

```
public static function selectFlightByID($flightIDwanted)
{
    include ('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    $query="SELECT * FROM flight WHERE flightID=".$flightIDwanted;
    $result=mysqli_query($conn, $query);
    mysqli_close($conn);
    $row=mysqli_fetch_assoc($result);
    $flightID=$row["flightID"];
    $flightDate=$row["flightDate"];
    $flightTime=$row["flightTime"];
    $flightNumber=$row["flightNumber"];
    $flightFrom=$row["flightFrom"];
    $flightTo=$row["flightTo"];
    $obj = new Flight($flightID, $flightDate, $flightTime, $flightNumber,
        $flightFrom, $flightTo);
    Flight::$flightObj[0] = $obj;
}
?>
```

Save the updated **Flight.php** class file and copy this to the server.

Return to the **displayBookings.php** file. Remove the PHP code which displayed the flightSelected during testing, and replace it with the block of code on the next page. Save the updated **displayBookings.php** file and copy it to the server.

The code calls the **selectFlightByID()** method in the Flight class, which then creates an object **\$flightObj[0]** for the selected flight. The attributes of the object are then displayed by means of **get()** methods.

Run the public web page and go to the table of flights. Select a flight and click the 'make booking' button.

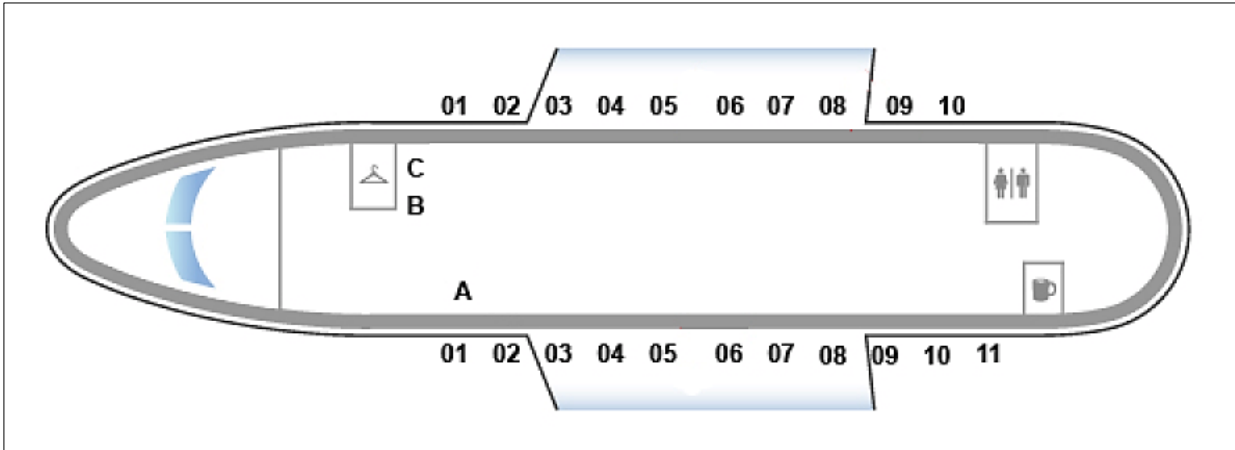


```

</table>
<p>
<table cellpadding=10>
<tr><td>
<?
    $flightSelected=$_REQUEST['flightSelected'];
    include('Flight.php');
    Flight::selectFlightByID($flightSelected);
    $flightDate=Flight::$flightObj[0]->getFlightDate();
    $dateFormatted = substr($flightDate,8,2)."/".substr($flightDate,5,2).
        "/" .substr($flightDate,0,4);
    $flightTime=Flight::$flightObj[0]->getFlightTime();
    $flightNumber=Flight::$flightObj[0]->getFlightNumber();
    $flightFrom=Flight::$flightObj[0]->getFlightFrom();
    $flightTo=Flight::$flightObj[0]->getFlightTo();
    echo $dateFormatted." ".$flightTime." Flight ".$flightNumber.
        " from ".$flightFrom." to ".$flightTo;
?>
</td></tr>
</table>
</body>
</html>

```

The main purpose of the **displayBookings.php** page is to display a plan of the aircraft, showing seats available. Open a suitable graphics application such as PhotoShop or Microsoft Word. Using the illustration below as a guide, create an outline diagram of the aircraft. The outer rectangle bordering the diagram has a size of 620 by 230 pixels, with a distance from the nose of the aircraft to the tail of 570 pixels. Save your diagram as 'seating plan.png' and copy it to the server.



Add lines of code to display the aircraft outline on the **displayBookings.php** page.

```

    echo $dateFormatted." ".$flightTime." Flight ".$flightNumber.
        " from ".$flightFrom." to ".$flightTo;
?>
</td></tr>
<tr>
    <td>
</td></tr>
</table>
</body>

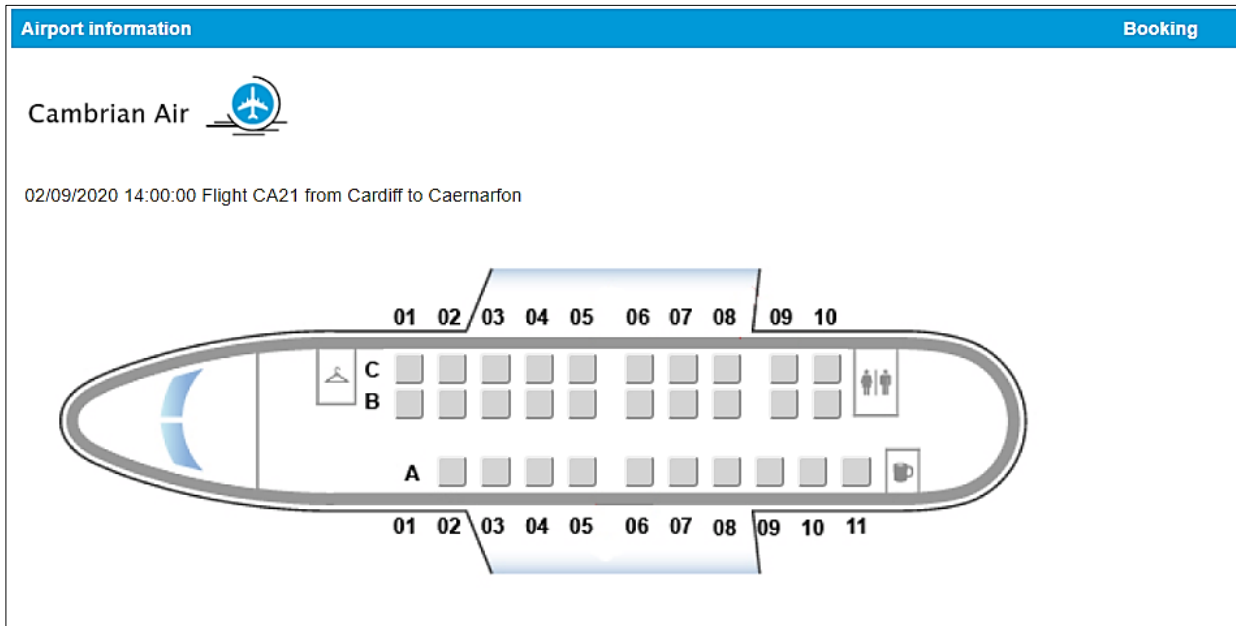
```

Program code can now be added to create the seat display using buttons. Add the lines of code below, which adjust the pixel positions of the seats to match the aircraft layout. It is important that the long 'echo' line producing a button is entered by continuous typing with no line breaks.

```
|  |
| --- |
|  |

```

Save the updated **displayBookings.php** file and copy this to the server. Run the public web site, select a flight, then click the 'make booking' button. The complete plan of the aircraft should now be displayed, with all seats coloured grey. If the array of seats does not correctly align with the outline of the aircraft, the position can be adjusted by altering the **\$Yoffset** and/or **\$Xoffset** values in the first line of program code which you added on the previous page.



The next step is to colour code the seats which are already booked. To do this, we will create test data to simulate booked seats in the database.

Open the PHP MyAdmin web site for your database and go to the **seats** table. All seats should display a **booked** value of 0, showing that they are currently available. Within the group of seats for a particular flight, change the **booked** value for some seats from **0** to **2** to specify that they are booked:

rowNumber	seatLetter	booked	journeyBookingID	flightID
1	B	2	0	9
1	C	2	0	9
2	A	0	0	9
2	B	0	0	9
2	C	2	0	9
3	A	0	0	9

The next step is to add a method to the **Seat** class which will access the seat records for the required flight and create a set of seat objects.

Open the **Seat.php** class file and add a set of **get()** methods at the end, so that the private attributes of the seat objects can be accessed from the main program.

```

}
public function getSeatID(){return $this->seatID;}
public function getRowNumber(){return $this->rowNumber;}
public function getSeatLetter(){return $this->seatLetter;}
public function getBooked(){return $this->booked;}
public function getBookingID(){return $this->bookingID;}
public function getFlightID(){return $this->flightID;}
public function getBookingTime(){return $this->bookingTime;}
}
?>

```

Also add a **loadSeats()** at the end of the **Seat.php** class file.

```

public function getBookingID(){return $this->bookingID;}
public function getFlightID(){return $this->flightID;}
public function getBookingTime(){return $this->bookingTime;}

public static function loadSeats($flightIDwanted)
{
    include ('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    $query="SELECT * FROM seats WHERE flightID='".$flightIDwanted.'";
    $result=mysqli_query($conn, $query);
    $num=mysqli_num_rows($result);
    mysqli_close($conn);
    $i=1;
    while ($i <= $num)
    {
        $row=mysqli_fetch_assoc($result);
        $seatID=$row["seatID"];
        $rowNumber=$row["rowNumber"];
        $seatLetter=$row["seatLetter"];
        $booked=$row["booked"];
        $bookingID=$row["bookingID"];
        $bookingID=$row["journeyBookingID"];
        $bookingTime=$row["bookingTime"];
        $obj = new Seat($seatID, $rowNumber, $seatLetter, $booked,
                       $bookingID, $bookingID, $bookingTime);
        Seat::$seatObj[$i] = $obj;
        $i++;
    }
}
}
?>

```

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

Re-open the **displayBookings.php** file. Locate the block of PHP code at the start of the <body> section and add lines of code to call the **loadSeats()** method in the Seat class.

```

<td>
    <?
        $flightSelected=$_REQUEST['flightSelected'];
        include('Flight.php');
        include('Seat.php');
        Seat::loadSeats($flightSelected);
        Flight::selectFlightByID($flightSelected);
        $flightDate=Flight::$flightObj[0]->flightDate;
    </?

```

Add program code to the **displayBookings.php** file as shown below. This will check the booking state for each seat, and select a red button colour if the seat is booked.

```

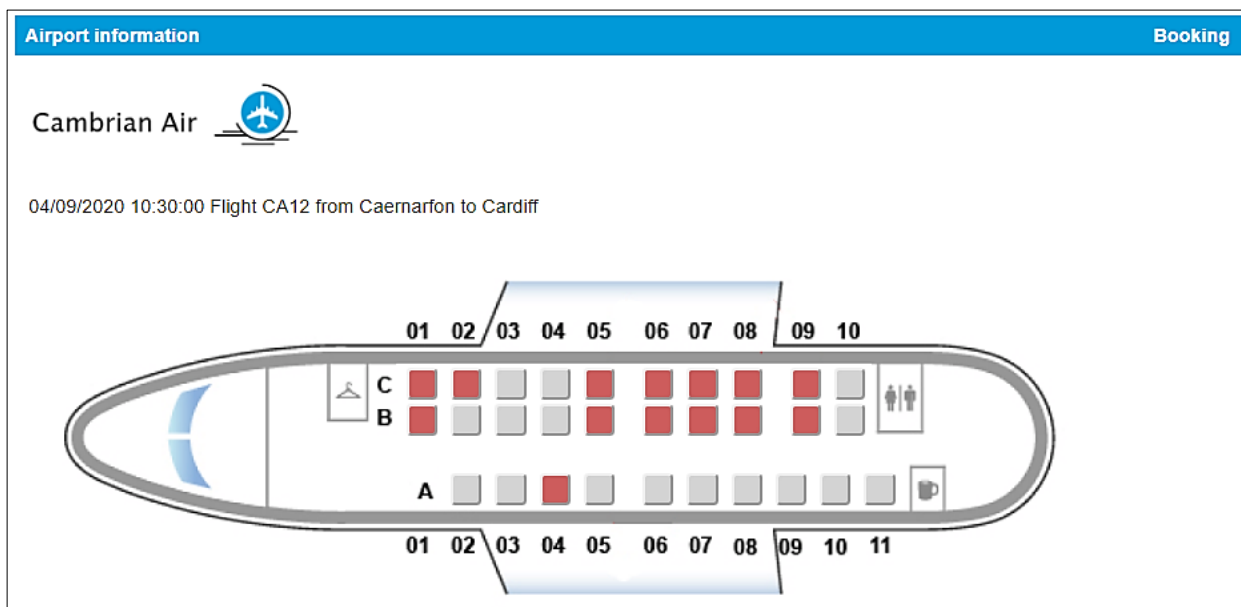
for ($i=0; $i<11; $i++)
{
    $seat = $i+1;
    $across = 323 + $Xoffset+ $i * $Xgap;
    if ($i>4)
        $across = $across +12;
    if (($i>7)&&($j<2))
        $across = $across +12;
    $c='#D3D3D3';

    for ($count=1; $count<=30; $count++)
    {
        $rowNumber=Seat::$seatObj[$count]->getRowNumber();
        $seatLetter=Seat::$seatObj[$count]->getSeatLetter();
        $booked=Seat::$seatObj[$count]->getBooked();
        if (((($i+1)==$rowNumber)&&($r==$seatLetter))
            {
                if ($booked>0)
                {
                    $c='#cd5c5c';
                }
            }
        }

    if(($i<10)||($j==2))
    {
        if (($j<2)||($i>0))
        {
            echo"<button name='submit' value="."$seat.$r.

```

Save the **displayBookings.php** file and copy it to the server. Run the public web site and select the flight which has seat booking codes set to 2. Check that the booked seats appear in red on the aircraft diagram.



We must now deal with the selection of seats by the customer.

Return to the **displayBookings.php** file. Go to the end of the first block of PHP code and add a **<form>** command. This will make the page reload when a seat button is clicked.

```

$flightTime=Flight::$flightObj[0]->flightTime;
$flightNumber=Flight::$flightObj[0]->flightNumber;
$flightFrom=Flight::$flightObj[0]->flightFrom;
$flightTo=Flight::$flightObj[0]->flightTo;
echo $dateFormatted." ".$flightTime." Flight ".$flightNumber.
        " from ".$flightFrom." to ".$flightTo;
echo"<form method='post' action='displayBookings.php?flight=$flightSelected'>";
?>
</td>
</tr>

```

At the end of the page, close the form.

```

    }
  }
?>
</td></tr>
</form>
</table>

```

Return to the first section of PHP code and add lines to obtain the flightID and the selected seat location when the page is reloaded.

```

<table cellpadding=10>
<tr>
<td>
<?
    $flightSelected=$_REQUEST['flightSelected'];
    include('Flight.php');
    include('Seat.php');

    $flightID=$_REQUEST['flight'];
    if ($flightID>0)
    {
        $flightSelected=$flightID;
        $seatSelected=$_REQUEST['seatSelected'];
        $seatLetterWanted=substr($seatSelected,-1,1);
        $seatRowWanted=substr($seatSelected,0,-1);
    }

    Seat::loadSeats($flightSelected);
    Flight::selectFlightByID($flightSelected);
    $flightDate=Flight::$flightObj[0]->flightDate;

```

In order to test the program, go to the end of the **displayBookings.php** file and add temporary lines to display the location for the seat selected.

```

    }
?>
</td></tr>
</form>
</table>

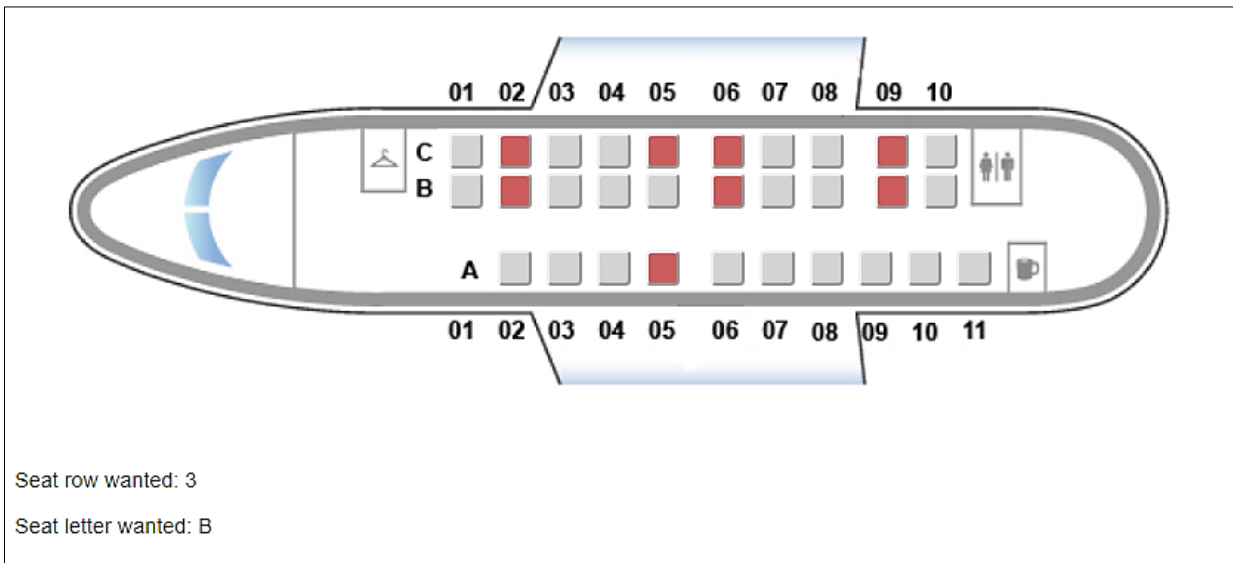
<?
    echo"<p>Seat row wanted: ".$seatRowWanted;
    echo"<p>Seat letter wanted: ".$seatLetterWanted;
?>

</body>
</html>

```

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

Run the aircraft seating plan page. Click on various seats and check that the seat row and letter selected are displayed correctly.



It is necessary to keep a record of each seat which is selected. This can be done as a string list of the seat row numbers and seat letters. Return to the **displayBookings.php** file and modify the block of code around the `<form>` command.

```

$flightNumber=Flight::$flightObj[0]->flightNumber;
$flightFrom=Flight::$flightObj[0]->flightFrom;
$flightTo=Flight::$flightObj[0]->flightTo;
echo $dateFormatted." ".$flightTime." Flight ".$flightNumber.
        " from ".$flightFrom." to ".$flightTo;

$seatlist = $_REQUEST['seatlist'];
$seatlist = $seatlist.$seatRowWanted.$seatLetterWanted."*";

echo"<form method='post' action='displayBookings.php?flight=
        $flightSelected&seatlist=$seatlist'>";
$seatsSelected = explode("*", $seatlist);
?>
</td>
<tr>
<td>


```

The purpose of these program lines is to:

- Obtain the current list of seats **\$seatlist** selected by the customer, e.g. *6B*6C*.
- Add the newly selected seat to the list, including a * character as a separator, e.g. *6B*6C*7A*
- Convert the single string of seat locations into an array **\$seatsSelected** of the individual seat locations, e.g. \$seatsSelected[1]=6B, \$seatsSelected[2]=6C, \$seatsSelected[3]=7A .

Modify the colour selection section to include a green colour code for selected seats.

```

if (($i+1)==$rowNumber)&&($r==$seatLetter))
{
    if ($booked>0)
    {
        $c='#cd5c5c';
    }
    else
    {
        $currentSeat=$rowNumber.$seatLetter;
        for($k = 0; $k < count($seatsSelected); $k++)
        {
            if($seatsSelected[$k]== $currentSeat)
            {
                $c='#2ECC71';
            }
        }
    }
}

```

Change the test display at the end of the page.

```

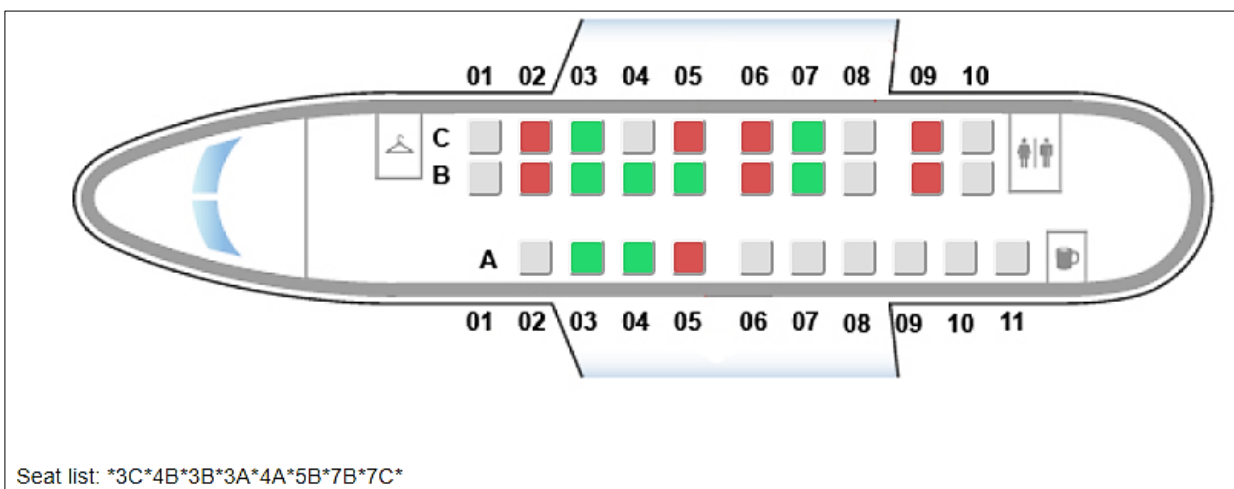
</td></tr>
</form>
</table>

<?
    echo"<p>Seat list: ".$seatlist;
?>

</body>
</html>

```

Save the **displayBookings.php** file and copy it to the server. It should now be possible to select seats, show the seat colour in green, and add the seat to the list displayed at the bottom of the page.



We do, however, still need to correct a couple of problems:

- It is possible to select a seat which is shown in red as already booked.
- It is possible to select a seat more than once, with the seat identifier being added multiple times to the seat list.

Return to the **displayBookings.php** file and add the program code shown below. Save the updated file and copy it to the server. Run the seat bookings display and check that these two problems have been solved.

```

$flightTo=Flight::$flightObj[0]->flightTo;
echo $dateFormatted." ".$flightTime." Flight ".$flightNumber.
        " from ".$flightFrom." to ".$flightTo;
$seatlist = $_REQUEST['seatlist'];

$seatsSelected = explode(" ", $seatlist);
$wantedSeat=$seatRowWanted.$seatLetterWanted;
$found=false;
for ($count=1; $count<=30; $count++)
{
    $rowNumber=Seat::$seatObj[$count]->getRowNumber();
    $seatLetter=Seat::$seatObj[$count]->getSeatLetter();
    $booked=Seat::$seatObj[$count]->getBooked();
    $currentSeat =$rowNumber.$seatLetter;
    if (($currentSeat==$wantedSeat)&&($booked==2))
        $found=true;
}
$duplicate=false;
for($k = 0; $k < count($seatsSelected); $k++)
{
    if($seatsSelected[$k]== $wantedSeat)
    {
        $found=true;
        $seatsSelected[$k]='';
        $duplicate=true;
    }
}
if ($duplicate==true)
{
    $seatlist = '*';
    for($k = 0; $k < count($seatsSelected); $k++)
    {
        if (strlen($seatsSelected[$k])>0)
            $seatlist = $seatlist.$seatsSelected[$k]."*";
    }
}
if ($found==false)
    $seatlist = $seatlist.$seatRowWanted.$seatLetterWanted.*";
echo"<form method='post' action='displayBookings.php?flight=
        $flightSelected&seatlist=$seatlist'>";
$seatsSelected = explode(" ", $seatlist);

```

Notice that the program now allows the user to de-select a seat by clicking on it a second time.

Return to **displayBookings.php**. Complete the page by replacing the 'echo' command with the lines of program shown below. These display instructions for the user, the list of the seats currently selected, and a button to continue to the payment page.

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

```

        </td></tr>
    </form>
</table>

<?
echo"<form method=post action='customerDetails.php?flight=
                $flightSelected&seatlist=$seatlist'>";
$output = str_replace("*"," ",$seatlist);
echo"<p>Seats selected <input type='text' id='seats' size=24; value='".$output."'>";
echo"&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&";
echo"Seats coloured grey are currently available. Click to reserve or cancel.";
echo"<button style='position:absolute;left:760px; top: 580px;font-size:16px;
                border-radius: 4px;width: 86px; background-color: #1184DF;
                color: white'>continue</button>";

?>
</form>

</body>
</html>

```

Run the page and check that seat bookings are listed correctly in the text display at the bottom of the page.

02/09/2020 14:00:00 Flight CA21 from Cardiff to Caernarfon

Seats selected Seats coloured grey are currently available. Click to reserve or cancel.

continue

We will now move on to enter the customer's contact and payment details. Open a new file and enter the lines of program code shown on the next page. These will create input boxes for entering the customer's name.

Cambrian Air

Customer details

Title **Mr** Forename Surname


```

<head>
  <title> Cambrian Air </title>
  <link rel="stylesheet" type="text/css" href="styleSheet.css" />
</head>
<body>
  <?
    $flightSelected=$_REQUEST['flight'];
    $seatsSelected=$_REQUEST['seatlist'];
    include('Flight.php');
    include('Seat.php');
    Flight::selectFlightByID($flightSelected);
    $flightDate=Flight::$flightObj[0]->getFlightDate();
    $dateFormatted = substr($flightDate,8,2)."/".substr($flightDate,5,2).
    " ".substr($flightDate,0,4);
    $flightTime=Flight::$flightObj[0]->getFlightTime();
    $flightNumber=Flight::$flightObj[0]->getFlightNumber();
    $flightFrom=Flight::$flightObj[0]->getFlightFrom();
    $flightTo=Flight::$flightObj[0]->getFlightTo();
  ?>
  <table class=menu>
  <tr>

```

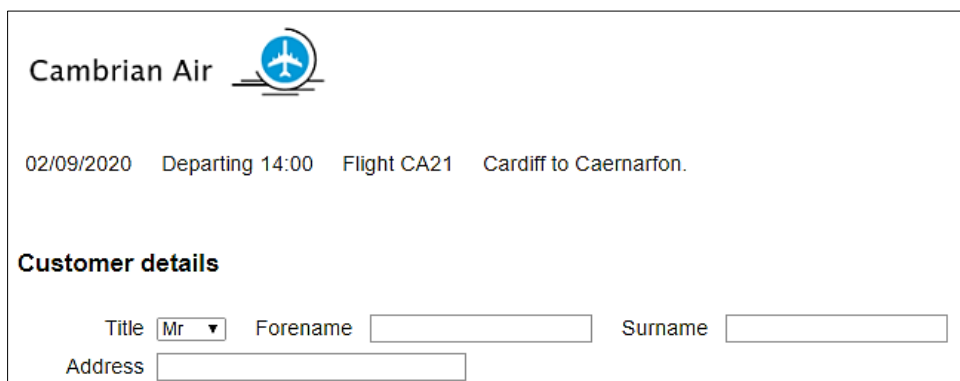
Add the lines of code below to display the flight details.

```

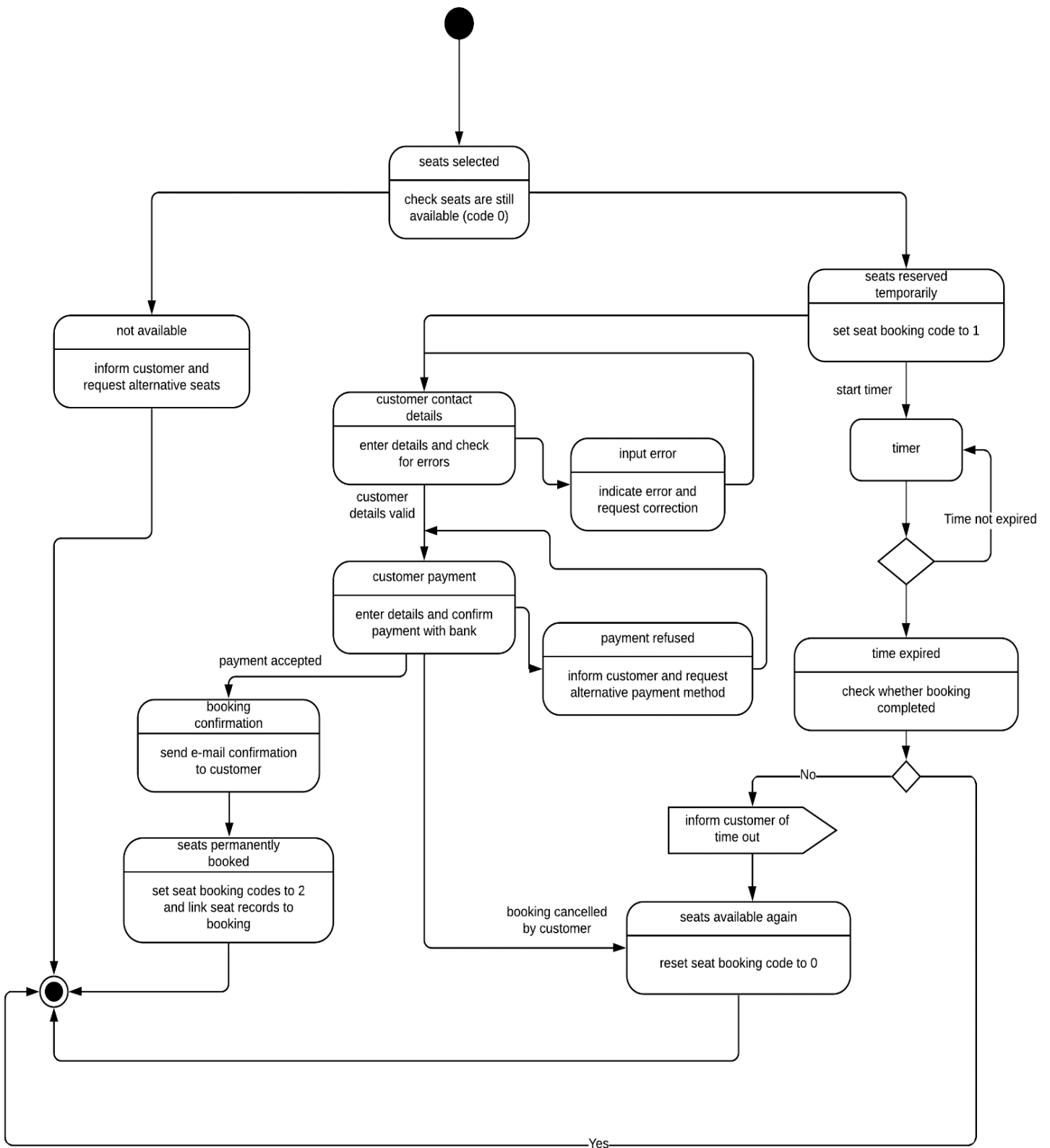

<?
  $bookingTime = date("Y-m-d H:i:00");
  echo"<form method=post
    action='confirm.php?flight=$flightSelected&seatsSelected=$seatsSelected'>";
?>
<table cellpadding=10>
  <tr>
  <?
    echo "<td>".$dateFormatted."</td>";
    echo "<td>Departing ".substr($flightTime, 0, 5) ."</td>";
    echo "<td>Flight ".$flightNumber."</td>";
    echo "<td>".$flightFrom." to ".$flightTo."</td>";
  ?>
  </tr>
</table>
<table cellpadding=4>
<tr>
  <td colspan=3><h3>Customer details</td></tr>

```

Save the updated **customerDetails.php** file and copy it to the server. Run the web site. Select a flight and seats, then check that the flight details are displayed correctly.



At this point we must consider the requirements of the real-time system to avoid double booking of seats by customers using different computers on-line at the same time. A **state diagram** for the processing of bookings is shown below, and will be followed as we program the web page.



Several customers might be using the web site at the same time to book seats from different locations. When a customer first proceeds to the payment page, we must check that the required seats are still available. To do this, open the **Seat.php** class file and add the method shown below.

```

public static function checkAvailability($flightIDwanted, $seatlist)
{
    Seat::loadSeats($flightIDwanted);
    $seatsSelected = explode("*", $seatlist);
    $available = true;
    for ($count=1; $count<=30; $count++)
    {
        $rowNumber=Seat::$seatObj[$count]->rowNumber;
        $seatLetter=Seat::$seatObj[$count]->seatLetter;
        $booked=Seat::$seatObj[$count]->booked;
        $currentSeat = $rowNumber.$seatLetter;
        for($k = 0; $k < count($seatsSelected); $k++)
        {
            if(($seatsSelected[$k]==$currentSeat)&&($booked>0))
            {
                $available = false;
            }
        }
    }
    return $available;
}
?>

```

Save the updated **Seat.php** file and copy it to the server. Re-open **customerDetails.php**. Insert lines of code to run the checkAvailability() method and respond to the result of the check.

```


<?
$available=Seat::checkAvailability($flightSelected, $seatsSelected);
if ($available==false)
{
    echo"<form method='post' action = 'displayBookings.php?flight=$flightSelected'>";
    echo"<p>A seat you selected has just been booked by another customer. ↩";
    echo"<p><input type=submit value='continue'>";
    echo"</form>";
}
else
{
    $bookingTime = date("Y-m-d H:i:00");
    echo"<form method=post action='confirm.php?flight=$flightSelected

```

Add a closing PHP bracket at the end of the page. Save the updated file and copy it to the server.

```

</table>
</form>
<?
}
?>
</body>
</html>

```


Re-open the **Seat.php** class file. Assuming that the selected seats are available, a temporary reservation is made by setting the booked code to 1 for each of the required seats and inserting the current time in the bookingTime field. Add a method to do this, save the file and copy it to the server.


```
public static function reserveSeats($flightID,$seatlist,$bookingTime)
{
    include ('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    $seatsSelected = explode(" ", $seatlist);
    for($k = 0; $k < count($seatsSelected); $k++)
    {
        $seatLetterWanted=substr($seatsSelected[$k], -1,1);
        $seatRowWanted=substr($seatsSelected[$k],0, -1);
        $query="UPDATE seats SET booked='1',bookingTime='$bookingTime'
              WHERE flightID='".$flightID.'" AND rowNumber='".$seatRowWanted.'"
              AND seatLetter = '".$seatLetterWanted.'";
        $result=mysqli_query($conn, $query);
    }
    mysqli_close($conn);
}
?>
```

Re-open the **customerDetails.php** file. Add a line of code to call the **reserveSeats()** method in the Seat class. Save the updated **customerDetails.php** file and copy it to the server.

```
else
{
    $bookingTime = date("Y-m-d H:i:00");
    Seat::reserveSeats($flightSelected,$seatsSelected,$bookingTime);
    echo"<form method=post action='confirm.php?bookingTime=".$bookingTime."'>";
?>
<table cellpadding=10>
```

To test the method, it will be necessary to run the web site on more than one computer. Select the same flight, then the same seats on each screen. Click the 'continue' button on one of the pages, and the customer name and address input boxes should be displayed. Now click 'continue' on the other computer. A message should be displayed to indicate that the seats are no longer available.

Airport information


Cambrian Air

A seat you selected has just been booked by another customer. Please return to the flight page and select an alternative seat.

Return to the **customerDetails.php** file. The list of seats booked can now be displayed. Add lines of program code to do this, as shown below.

```

<table cellpadding=10>
<tr>
<?
    echo "<td>".$dateFormatted."</td>";
    echo "<td>Departing ".$substr($flightTime, 0, 5) . "</td>";
    echo "<td>Flight ".$flightNumber."</td>";
    echo "<td>".$flightFrom." to ".$flightTo."</td>";

    $output = str_replace("*", " ", $seatsSelected);
    echo "<td>Seats selected:".$output."</td>";
?>
</tr>

```

The total ticket price can then be calculated by determining the number of seats selected. Add the lines of code to do this. Save the **customerDetails.php** file and copy it to the server.

```

<tr>
<td align='right'>Payment due
<td>
<?
    $seat = explode(" ", $seatsSelected);
    $seatCount=0;
    for ($i=0;$i<count($seat);$i++)
        if (strlen($seat[$i])>0)
            $seatCount++;

    $payment=48.00 * $seatCount;
    echo"<input type='text' size = 60 value='".$seatCount." seats @ £48.00:
        total ticket cost £".$payment.".00'>";

```

Run the booking page and check that the seats selected are now listed, and that the payment due has been calculated correctly.

Cambrian Air

02/09/2020 Departing 10:00 Flight CA22 Caernarfon to Manchester. Seats selected: 9C 9B 8C 8B

Customer details

Title Forename Surname

Address

Town Postcode

E-mail

Payment

Payment due

Re-open the **customerDetails.php** file. The final entries required are the customer's credit card details. Buttons will be added to complete or cancel the booking.

Town	<input type="text"/>	Postcode	<input type="text"/>
E-mail	<input type="text"/>		
Payment			
Payment due	2 seats @ £48.00: total ticket cost £96.00		
Card type	<input type="text"/>	Card number	<input type="text"/>
	Expires: month/year	<input type="text"/>	/ <input type="text"/>
<input type="button" value="complete booking"/>		<input type="button" value="cancel transaction"/>	

Re-open the **Seat.php** class file. We will consider the case of a customer clicking the 'cancel' button on the booking page. Add a method to the Seat object which will reset the reserved seats as available. Save the **Seat.php** file and copy it to the server.

```

public static function cancelReservation($flightID,$seatlist)
{
    include ('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    $seatsSelected = explode("*", $seatlist);
    for($k = 0; $k < count($seatsSelected); $k++)
    {
        $seatLetterWanted=substr($seatsSelected[$k],-1,1);
        $seatRowWanted=substr($seatsSelected[$k],0,-1);
        $query="UPDATE seats SET booked='0',bookingTime='0000-00-00 00:00:00'
            WHERE flightID='".$flightID.'" AND rowNumber='".$seatRowWanted.'"
            AND seatLetter = '".$seatLetterWanted.'";
        $result=mysqli_query($conn, $query);
    }
    mysqli_close($conn);
}
}
?>

```

Open the **index.php** file. If a customer cancels their transaction, they will be returned to this page.

Add a block of code at the start of **index.php** which collects details of the flight and seats selected, then calls the **cancelReservation()** method in the Seat class file to re-set the seats as available.

```

<?
include('Seat.php');
$cancel=$_REQUEST['cancel'];
if ($cancel=='YES')
{
    $flightSelected=$_REQUEST['flight'];
    $seatsSelected=$_REQUEST['seats'];
    Seat::cancelReservation($flightSelected,$seatsSelected);
}
?>

<html>
<head>
<title> Cambrian Air </title>
<link rel="stylesheet" type="text/css" href="styleSheet.css" />

```

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

Run the web site. Select a flight and seats, then continue to the payment page. Click the '**cancel transaction**' button. The program should return to the homepage. If the same flight is now selected, the seats reserved earlier should now be available again.

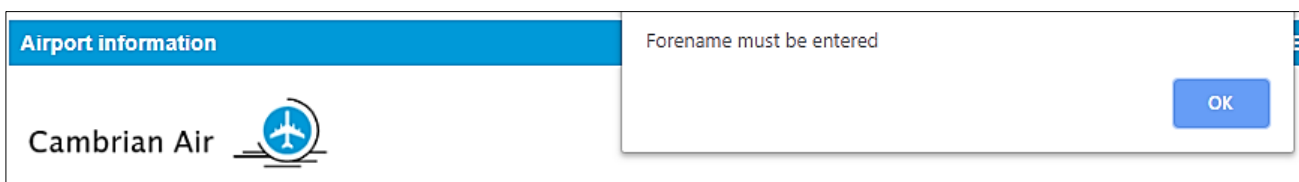
Re-open the **customerDetails.php** file. We will now consider the situation when the customer enters payment details and selects the '**complete booking**' button. Validation checks may be carried out on the text entries. As an example, we will carry out presence checks for the forename and surname fields. Replace the '**echo <form>**' command with the line of code shown, then add the **<script>** block. Save the **customerDetails.php** file and copy it to the server.

```

else
{
    $bookingTime = date("Y-m-d H:i:00");
    Seat::reserveSeats($flightSelected, $seatsSelected,$bookingTime);
    echo"<form method=post action='confirm.php?flight=$flightSelected&
        seatsSelected=$seatsSelected' 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=10>
<tr>
    <?
        echo "<td>".$dateFormatted;

```

Run the website, select a flight and seats, then click the 'continue' button. The customer details page will open. Click the 'complete booking' button without entering a customer name. An error message should be displayed.



Before the booking record is entered into the database, the Javascript function **checkInput()** is called. This contains code to carry out the presence checks. Further validation code can be added for other fields as required. The next web page **confirm.php** will only be loaded if no errors are found.

Continue work on **customerDetails.php**. In a real system, the web page would at this point contact a bank to authorise the customer's payment. A result would be returned to indicate whether payment had been accepted. We will simulate this with a Javascript function. Replace the **else** condition with the line of code shown, and add the **bankCheck()** function to the `<script>` block.

```

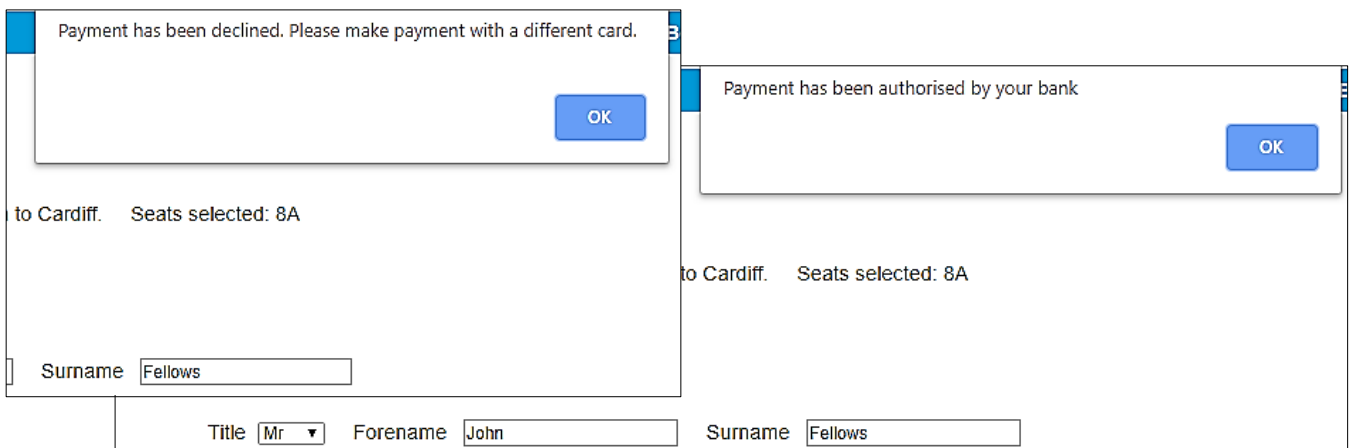
        alert("Surname must be entered");
        error=true;
    }
    if (error==true)
        result=false;
    else
        result = bankCheck();
    return result;
}

function bankCheck()
{
    var r = Math.random();
    if (r<0.75)
    {
        alert("Payment has been authorised by your bank");
        return true;
    }
    else
    {
        alert("Payment has been declined.
                Please make payment with a different card.");
        return false;
    }
}
</script>
<table cellpadding=10>

```

Save the updated **customerDetails.php** file and copy it to the server. Select a flight and seats, enter customer details, then click the 'complete booking' button.

The **bankCheck()** function generates a random decimal fraction between 0.0 and 1.0. If this value is 0.75 or greater, the function simulates the customer's payment being refused; this represents a probability of one in four.



Return to the **customerDetails.php** file.

Referring back to the state diagram above, we should terminate the booking procedure if the allocated time for entering contact information and making payment is exceeded. For test purposes, we will set this time to be 10 minutes. Insert a timer function at the start of the <body> block.

```
<title> Cambrian Air </title>
<link rel="stylesheet" type="text/css" href="styleSheet.css" />
</head>
<body>
<script>
    setTimeout(function()
    {
        alert("Time expired. Please re-enter your booking");
        window.location = "selectFlight.php";
    },
    600000);
</script>
<?
    $flightSelected=$_REQUEST['flight'];
    $seatsSelected=$_REQUEST['seatlist'];
```

Save the **customerDetails.php** file and copy it to the server. Run the website, go to the customer details input page, then leave the page open. After 10 minutes, the **setTimeout()** function should operate and display an alert message that the allowed time for booking has expired. The **selectFlight** page is then reloaded, and the customer has the option to repeat the booking process.

Re-open the **Seat.php** class file. We must now consider the possibility that the customer simply decides not to continue with the booking and switches off their computer. In this circumstance, the seats which were temporarily reserved must be released for resale once the 10 minute booking period has expired. To do this, add a **timeOut()** method to the **Seat.php** class file as shown in the two boxes below:

```
public static function timeOut()
{
    include ('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    $query="SELECT * FROM seats WHERE booked = '1'";
    $result=mysqli_query($conn, $query);
    $num=mysqli_num_rows($result);
    $timeNow = date("Y-m-d H:i:00");
    $hour2=substr($timeNow,11,2);
    $minute2=substr($timeNow,14,2);
    $minutes2=$hour2*60 + $minute2;
    $i=0;
    while ($i < $num)
    {
        $row=mysqli_fetch_assoc($result);
        $seatID=$row["seatID"];
        $bookingTime=$row["bookingTime"];
        $hour1=substr($bookingTime,11,2);
        $minute1=substr($bookingTime,14,2);
        $minutes1=$hour1*60 + $minute1;
        if ($minutes1>$minutes2)
            $minutes2=$minutes2+24*60;
        $timeDiff=$minutes2 - $minutes1;
```

Continued:

```

        if ($timeDiff>10)
        {
            $query2="UPDATE seats SET booked='0', bookingTime=null
                    WHERE seatID='$seatID'";
            $result2=mysqli_query($conn, $query2);
        }
        $i++;
    }
    mysqli_close($conn);
}
?>

```

Save the **Seat.php** file and copy it to the server. The **timeOut()** function begins by extracting all records from the **seat** table in the database which are temporarily reserved with a **booked** code of 1. Records contain the exact times when the booking procedure commenced. These values are converted to the number of minutes after the start of the current day, then compared against the current time. If the booking commenced 10 minutes or more before the current time, the seat record is released by resetting the **booked** code to 0 and cancelling the booking time. The seat is now available again for booking.

Re-open the **selectFlight.php** file. We will call the **timeOut()** method each time the selectFlight page is loaded. This ensures that customers will view the up-to-date seat availability for any flight selected. Add the function at the start of **selectFlight.php**, save the file and copy it to the server.

```

<?
    include('Seat.php');
    Seat::timeOut();
?>

<html>
<head>
    <title> Cambrian Air </title>

```

Save the **selectFlight.php** file and copy it to the server. Run the website, select a flight and seats, then continue to the customer details page. Close the website, wait for 10 minutes then reload the home page. Select the same flight and check that the seats you selected previously are available again.

We will now move on to complete the booking. Open the PHP MyAdmin web site for the database. Set up an **airBooking** table ready to handle the transfer of bookings from the web page.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	bookingID	int(11)			No	None		AUTO_INCREMENT
2	paymentDate	datetime			No	None		
3	flightID	int(11)			No	None		
4	title	text	latin1_swedish_ci		No	None		
5	forename	text	latin1_swedish_ci		No	None		
6	surname	text	latin1_swedish_ci		No	None		
7	address1	text	latin1_swedish_ci		No	None		
8	address2	text	latin1_swedish_ci		No	None		
9	town	text	latin1_swedish_ci		No	None		
10	postcode	text	latin1_swedish_ci		No	None		
11	email	varchar(40)	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		

Add fields as shown above. The **bookingID field** is designated as the primary key and set to auto-increment.

Open a blank file and save this as **Booking.php**. Add the lines of code below to produce a Booking class. Attributes correspond to the database fields, and a constructor method is included. Save the **Booking.php** file and copy it to the server.

```
<?
class Booking
{

    public static $booking = array();
    public static $bookingCount;
    private $bookingID;
    private $paymentDate;
    private $flightID;
    private $title;
    private $forename;
    private $surname;
    private $address1;
    private $address2;
    private $town;
    private $postcode;
    private $email;
    private $paymentAmount;
    private $cardType;
    private $cardNumber;
    private $expiryDate;

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

}
?>
```

The next step is to produce the booking confirmation page. Open a blank file and add the code below. Save this as **confirm.php** then copy it to the server.

```

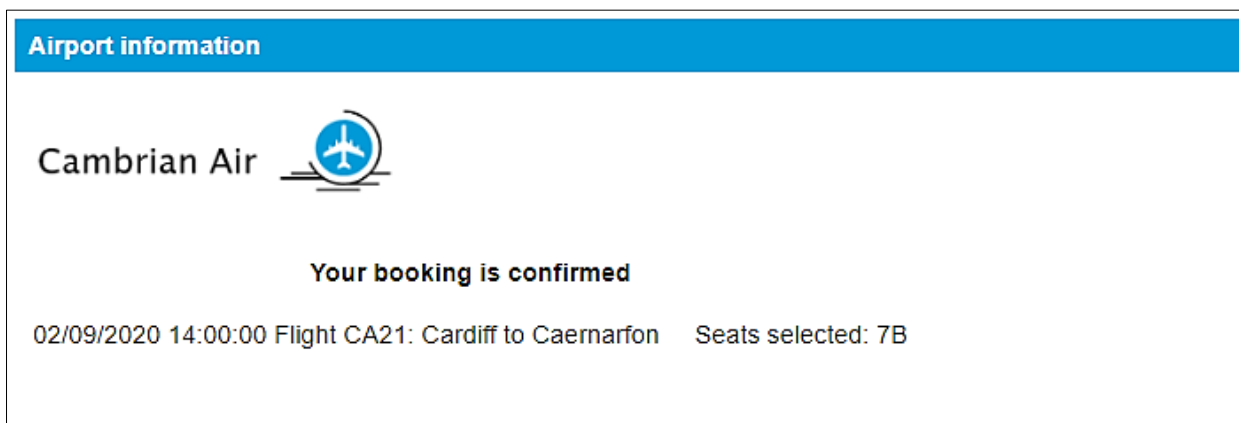
<html>
<head>
  <title> Cambrian Air </title>
  <link rel="stylesheet" type="text/css" href="styleSheet.css" />
</head>
<table class=menu>
<tr><th class=menu>
<a href=index.php>
Airport information
</a>
<th class=menu>
<a href=selectFlight.php>
Booking
</a>
</tr>
</table>
<p>

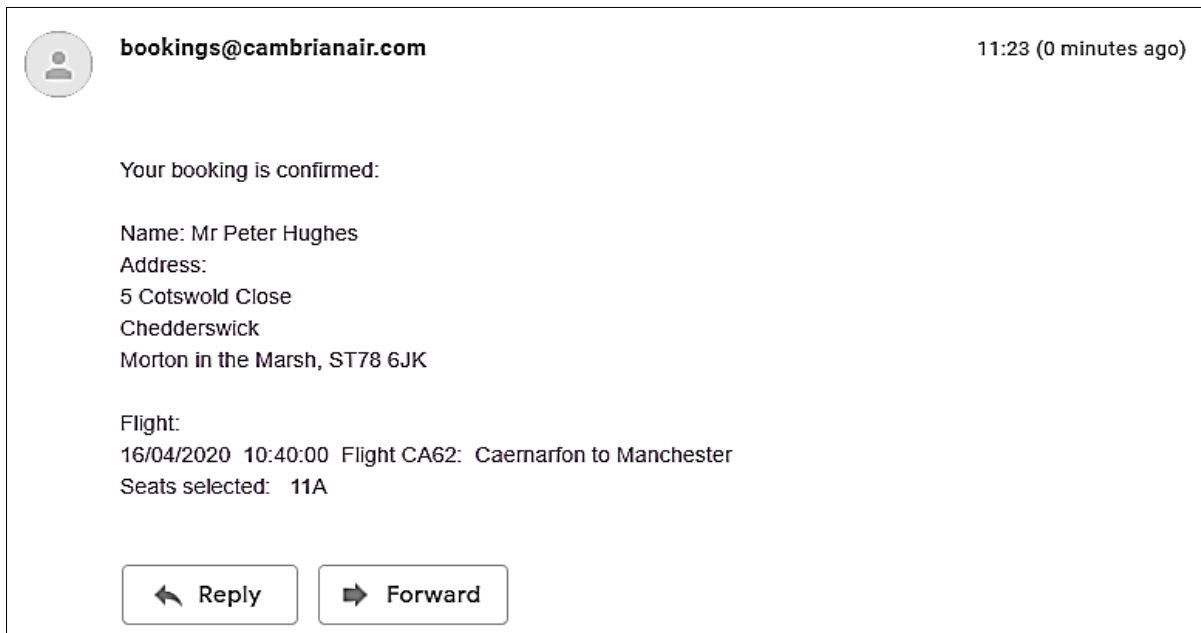
<?
$title = $_REQUEST["title"];
$forename = $_REQUEST["forename"];
$surname = $_REQUEST["surname"];
$address1 = $_REQUEST["address1"];
$address2 = $_REQUEST["address2"];
$town = $_REQUEST["town"];
$postcode = $_REQUEST["postcode"];
$email = $_REQUEST["email"];
$cardType = $_REQUEST["cardType"];
$cardNumber = $_REQUEST["cardNumber"];
$expireMonth = $_REQUEST["expireMonth"];
$expireYear = $_REQUEST["expireYear"];
$expiryDate = $expireMonth."/".$expireYear;
$paymentAmount = $_REQUEST["paymentAmount"];
?>
</body>
</html>

```

The page displays menu options and the company logo, then accesses customer contact and payment information from the input boxes on the previous page. Add code which will display details of the flight and seats booked, as shown on the next page.

Save the **confirm.php** file and copy it to the server. Run the public web site, make a booking and click the complete booking button. Check that the flight details are shown correctly.





Re-open the **confirm.php** file and add the lines of code below. The program sets up an e-mail message and then sends it to the e-mail address entered by the customer. Save the **confirm.php** file and copy it to the server.

```
include('Booking.php');
Booking::makeBooking($flightSelected, $seatsSelected, $paymentDate, $title,$forename,
$surname, $address1, $address2, $town, $postcode, $email,$paymentAmount, $cardType,
$cardNumber, $expiryDate);

echo"<tr><td> An e-mail has been sent to you to confirm your booking.";
$customer=$title." ".$forename." ".$surname;
$address3=$town.", ".$postcode;
$subject="Cambrian Airways booking confirmation ";
$body='Your booking is confirmed: ';
$body=$body."\n". "\nName: ".$customer;
$body=$body."\n". "Address: ";
$body=$body."\n".$address1;
$body=$body."\n".$address2;
$body=$body."\n".$address3;
$body=$body."\n". "\nFlight: ";
$body=$body."\n".$dateFormatted ." ".$flightTime." Flight ".$flightNumber."
": ".$flightFrom." to ".$flightTo;
$body=$body."\nSeats selected: ".$output;
$from= "bookings@cambridgianair.com";
$headers="From: ".$from;
$to=$email;
mail($to,$subject,$body,$headers);

?>
</table>
</body>
</html>
```

Run the public web site, select a flight and seats. Continue to the customer details page and enter a customer name, address and payment details. Add your own e-mail address. Click the 'complete booking' button to load the **confirm.php** page.

Access your e-mail system. Check that a confirmation e-mail has been received and contains the customer's name, address and flight details. (if the e-mail does not arrive in your in-box, please check the spam folder.)

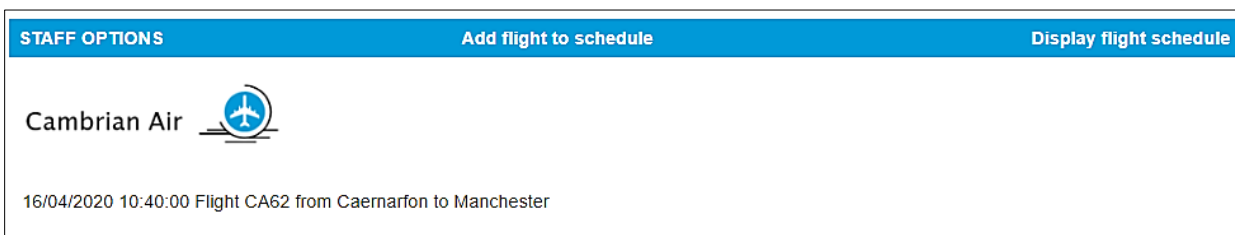
This completes the customer booking system. We can now return to the staff section of the website to finish the flight information pages.

When a member of staff logs-in to the website, the flights are listed with a 'display bookings' button alongside each flight record. These buttons have been set up to load a **staffDisplayBookings2.php** page which we will now create. This will display a seating plan and show the bookings for the flight. Open a blank file and add the lines of program code below.

```
<html>
<head>
  <title> Cambrian Air </title>
  <link rel="stylesheet" type="text/css" href="styleSheet.css" />
</head>
<body>
<?
  include('staffMenu.php');
?>
<p>
<table cellpadding=10>
<tr><td>
<?
  $flightSelected=$_REQUEST['flightSelected'];
  include('Flight.php');
  include('Seat.php');
  Seat::loadSeats($flightSelected);
  Flight::selectFlightByID($flightSelected);
  $flightDate=Flight::$flightObj[0]->getFlightDate();
  $dateFormatted = substr($flightDate,8,2)."/".substr($flightDate,5,2).
                    "/".substr($flightDate,0,4);
  $flightTime=Flight::$flightObj[0]->getFlightTime();
  $flightNumber=Flight::$flightObj[0]->getFlightNumber();
  $flightFrom=Flight::$flightObj[0]->getFlightFrom();
  $flightTo=Flight::$flightObj[0]->getFlightTo();
  echo $dateFormatted." ".$flightTime." Flight ".$flightNumber.
                    " from ".$flightFrom." to ".$flightTo;
?>
</td>
</tr>
</table>
</body>
</html>
```

Save the file as **staffDisplayBookings2.php** and copy it to the server.

Log-in to the staff web site, select a flight and click the 'display bookings' button. The new page should open to display the staff menu, company logo and details of the flight selected.



Return to **staffDisplayBookings2.php** and add the lines of code below. These display a seating plan for the aircraft in a very similar way to the booking page of the public web site.

Please note that the values of the variables **\$Yoffset** and **\$Xoffset** at the start of the code block can be adjusted if necessary so that the pattern of seats aligns correctly with the aircraft outline.

```

    $flightTo=Flight::$flightObj[0]->getFlightTo();
    echo $dateFormatted." ".$flightTime." Flight ".$flightNumber.
        " from ".$flightFrom." to ".$flightTo;
?>
<tr><td>
<?
    $Yoffset =170; $Xoffset = 11;
    $down = 114;
    $Xgap = 37; $Ygap = 30;
    $down = $down + $Yoffset;
    for ($j=0; $j<3; $j++)
    {
        switch ($j)
        {
            case 0: $r="C";break;
            case 1: $r="B";break;
            case 2: $r="A";break;
        }
        $down = $down + $j * $Ygap;
        if ($j==2)
            $down = $down - 2;
        for ($i=0; $i<11; $i++)
        {
            $seat = $i+1;
            $across = 323 + $Xoffset+ $i * $Xgap;
            if ($i>4)
                $across = $across +12;
            if (($i>7)&&($j<2))
                $across = $across +12;
            $c='#D3D3D3';
            if(($i<10)||($j==2))
            {
                if (($j<2)||($i>0))
                {
                    echo"<button style='position:absolute;left:". $across
                        ."px;top:". $down."px;font-size:18px;border-radius:4px;
                            width:26px;background-color:". $c.";'>&nbsp;&nbsp;&nbsp;</button>";
                }
            }
        }
    }
?>
</td></tr>
</table>
</body>
</html>

```

Save the **staffDisplayBookings2.php** file and copy it to the server. Run the 'display bookings' option again. The seating plan should now be shown, but with all seats shown in grey.

Return to the **staffDisplayBookings2.php** file and add the block of code below. This uses a loop to locate the **Seat** object corresponding to each seat before it is displayed. If the **booked** attribute of the seat is set to 2, the display colour is changed from grey to red.

```

if (($i>7)&&($j<2))
    $across = $across +12;
$c='#D3D3D3';

for ($count=1; $count<=30; $count++)
{
    $rowNumber=Seat::$seatObj[$count]->getRowNumber();
    $seatLetter=Seat::$seatObj[$count]->getSeatLetter();
    $booked=Seat::$seatObj[$count]->getBooked();
    if ((($i+1)==$rowNumber)&&($r==$seatLetter))
    {
        if ($booked==2)
        {
            $c='#cd5c5c';
        }
    }
}

if(($i<10)||($j==2))
{
    if (($j<2)||($i>0))
    {

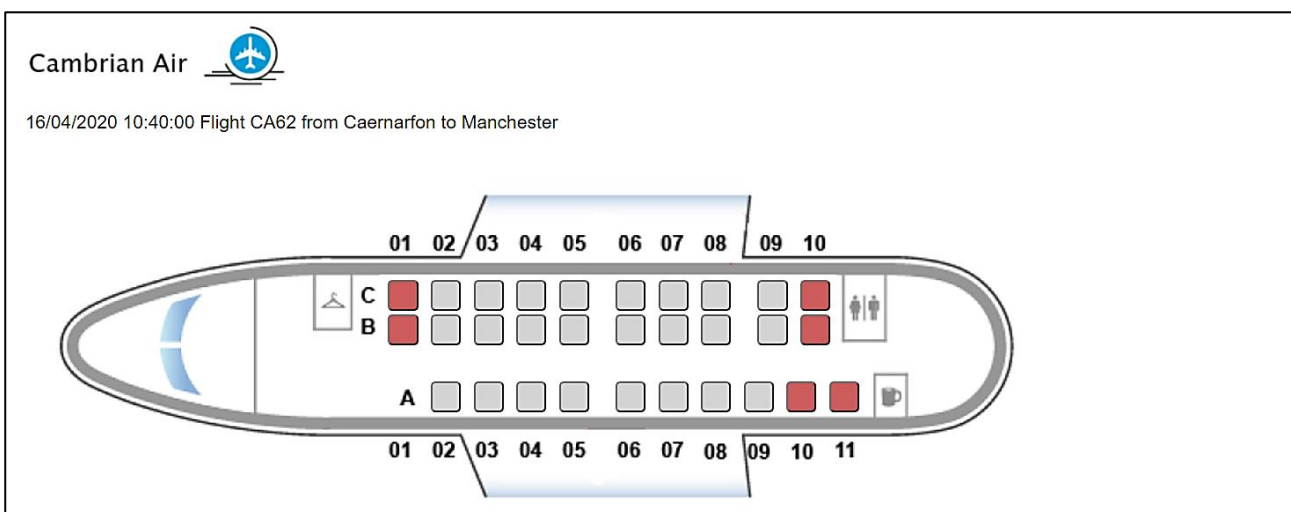
```

Save the **staffDisplayBookings2.php** file and copy it to the server.

Go to the PHP MyAdmin website and clear the previous test data from the **flight**, **seats** and **airBooking** tables. This can be done by selecting the **Operations** menu option and clicking **'Empty table (TRUNCATE)'**.

Enter test data for one flight with seats booked by several customers. To do this:

- Run the website and log-in as staff. Enter details for one flight. A set of empty seat records will be created automatically.
- Run the website again as a member of the public. Go to the bookings page, select several seats, then complete the booking procedure by entering the customer's name, address, e-mail and payment information. Repeat this for several more customers booking seats on the same flight.
- Run the website and log-in as staff. Select the flight entered earlier. Check that the seating diagram appears, with the booked seats highlighted in red.



A final step is to add a table of bookings for the flight. Return to the **staffDisplayBookings2.php** file and add the block of program code shown below. This begins by displaying the column headings for the table, then calls a **displayFlightBookings()** method which we will add to the Booking class file.


```

        echo"<button style='position:absolute;left:". $across
            ."px;top:". $down."px;font-size:18px;border-radius:4px;
            width:26px;background-color:". $c."'>&nbsp;&nbsp;&nbsp;</button>";
    }
}
}
?>

```

```

<h3>Bookings</h3>
<table class = f>
  <tr>
    <td></td>
  </tr>
  <tr>
    <td>Row</td>
    <td>Seat</td>
    <td>BookingID</td>
    <td>Title</td>
    <td>Forename</td>
    <td>Surname</td>
    <?
      include('Booking.php');
      Booking::displayFlightBookings($flightSelected);
    ?>
  </tr>
</table>
</td></tr>
</table>
</body>
</html>

```

Save the **staffDisplayBookings2.php** and copy it to the server.

Open the **Booking.php** class file and add the **displayFlightBookings ()** method as shown in the two boxes below.

```

public static function displayFlightBookings($flightSelected)
{
    include ('user.inc');
    $conn = new mysqli(localhost, $username, $password, $database);
    if (!$conn) {die("Connection failed: ".mysqli_connect_error()); }
    for($i=1;$i<=30; $i++)
    {
        $rowNumber =Seat::$seatObj[$i]->getRowNumber();
        $seatLetter=Seat::$seatObj[$i]->getSeatLetter();
        $booked=Seat::$seatObj[$i]->getBooked();
        $bookingID=Seat::$seatObj[$i]->getBookingID();
        if ($booked==2)
        {
            $query="SELECT * FROM airBooking WHERE bookingID = '". $bookingID."'";
            $result=mysqli_query($conn, $query);
            $row=mysqli_fetch_assoc($result);
            $title=$row["title"];
            $forename=$row["forename"];
            $surname=$row["surname"];
            echo "<tr class = f>";
            echo "<td class = f>". $rowNumber."</td>";

```

```

        echo "<td class = f>".$seatLetter."</td>";
        echo "<td class = f>".$bookingID."</td>";
        echo "<td class = f>".$title."</td>";
        echo "<td class = f>".$forename."</td>";
        echo "<td class = f>".$surname."</td>";
        echo "<td class = f><button name='bookingSelected'
            value='".$bookingID."'>booking details</button>";
    }
}
mysqli_close($conn);
}
?>

```

The **displayFlightBookings()** method examines the **booked** attribute for each of the Seat objects. If **booked** is set to 2, the **bookingID** attribute is used to obtain the corresponding customer name from the **airBooking** table in the database.

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

Return to the staff bookings page of the website and select the flight entered earlier. The seating diagram will be displayed along with a table showing the customer name and bookingID for each of the booked seats, as in the example below.

The **staffDisplayBookings2** web page calls the **displayFlightBookings()** method in the **Booking** class, which loads all bookings for the selected flight. This includes the customer's address, e-mail and payment details. A button has been added alongside each booking record in the display table. It is left as a further programming exercise to add another web page which will display this extra information if required.

Row	Seat	BookingID	Title	Forename	Surname	
2	B	44	Mr	Dewi	Morris	booking details
2	C	44	Mr	Dewi	Morris	booking details
5	A	43	Mr	Geoff	Mullins	booking details

Re-open the **staffDisplayBookings.php** file. One final task is to add the **timeOut()** function. This will ensure that any abandoned bookings are deleted and the data is correct when the aircraft seating plans and

passenger lists are displayed. Add the code below to the PHP block at the start of **staffDisplayBookings.php**. Save the file and copy it to the server.

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

    include('Seat.php');
    Seat::timeOut();
?>
```

Further development

The airline booking system demonstrated here is considerably simplified:

Access to the banking system is simulated; the arrangements and protocols provided by banks for on-line payment to businesses might be investigated by students.

The names of all travelling passengers would be required in a real system, rather than just the customer making the booking. The name of the passenger allocated to each seat on the aircraft would be recorded. This requirement could be implemented as a further development of the program.

Other travel booking systems might be developed, for example: for coach tours, boat excursions, or heritage steam railway events. Real time seat booking systems could be developed for theatre productions, music concerts or sports events.

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 and 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 + <u>checkPassword(userName, password): boolean</u>

Flight

Objects in the Flight class are identified as elements of the **flightObj[]** array, with the total number of flight objects stored as the public static attribute **itemcount**. Attributes of individual Flight objects can be accessed through a set of **get()** methods. Methods are included to add a new flight to the database, to create objects for all flights on a particular date or route, and to create a single object for a flight with a specified FlightID value.

Seat

Seat objects have attributes specifying the row and seat letter, and a booking code to indicate whether the seat is available, booked, or currently reserved during the booking procedure. Methods create a set of seats for a new flight, load all the seat bookings for a particular flight, reserve seats temporarily or make definite bookings. Methods can also cancel seat bookings on request, or when the allowed booking time has expired.

Booking

Attributes of Booking objects record the contact details and payment details for customers. Methods allow a booking to be added to the database, and all bookings for a particular flight to be retrieved and displayed.

