# System maintenance

## System overview

This is a maintenance manual for the newly designed ski Rossendale booking system section of the website. This is an integrated part of the site as a whole and will begin use on 1st August 2014. In this document is a detailed manual of the workings behind each page for input by the clients and the staff will be using. This system is designed for anybody after reading this manual to be able to fully understand the system in a professional and effective manor, and if required make the necessary alterations;

|  |  |
| --- | --- |
| Page Name | Comments |
| DATA ENTRY SCREENS. | At the top of every page are 3 linked files, an include file, to check that the user is logged on if they want to view the page, otherwise they are directed to the log on page, secondly is the external scripts file, which will be explained in detail below, this contains all of the validation for all of the data input screens, finally, there is a css file which keeps all pages looking the same. |
| ADDGROUPCOURSE.ASP | This is used by administrators only.  This page is used for staff to add a new group course for members to book onto.  Firstly displayed is the courseno, which is the course’s individual id. This is found by an SQL function called TOP 1 to find the next value that will be assigned when this course is written.  There is then a form containing data entry text boxes where admin staff can input the course values, when the submit button is pressed the page will go to “sendgroupcourse.asp” and the following values will be passed to it via the post method from the form.  Activity  Instructor  Skill(skill level)  Date  Time  Duration  Max entrants(maxent)  Slope  Age range |
| ADDNEWADMIN.ASP | This is used by administrators only.  This page is for adding a new admin to the system, an administrator must be logged on to do this, on this page is a form with the new administrators name and password, the post method is used to send these two variables to the page where the data is stored on the database, called sendadmin.asp. |
| ADMINLOGON.ASP | This is used by administrators only.  Very similar to the above page, this page is a simple log on page, with 2 form text inputs, one for name and one for password, this form also contains a hyperlink to the user log on page which is handy in case the unlikely event should arise that a customer ends up here. The values of this page are carried via the post method to adminlogon1.asp |
| BOOKGROUP.ASP | This is a very simple page for the user to define which sort of courses they want to see on the aft page, there is a dropdown combo box with values populated by the sql SELECT DISTINCT query allowing them to view courses only of the activity they defined, this value chosen is passed to the next page via the post method, showgroup.asp |
| BOOKPARTY.ASP | This page is used for party bookings , firstly, the customer is shown the individual course id of the course they are booking, for reference purposes, this is calculated using the SQL function called TOP 1 to find the next value that will be assigned when this course is written. The user then has a choice of activities to choose from , skiing , snowboarding or tubing, the three activities the club offer, upon choosing this, they can then choose a skill level that they are all at in this particular activity, however if they select tubing, the boxes where skill is chosen are disabled, as tubing does not have a skill level, after this, the user can then state the amount of people coming on the course, this is a very easy thing for the user to do thanks to the scroller boxes I have made, on these scroll boxes, the down arrow is disabled when the number of people is equal to 0, this prevents people from putting negative amounts of people on a course, which could cause problems, furthermore this is validated so that across the 4 fields there must be at least one person combined on the course. At the end of the form the user can choose the date, time, and duration of the course, which is fully validated using REGEX to ensure dates are in the correct format, and to ensure the date has not already passed, furthermore the time field is validated to ensure they separate the time by colon, and the time must be on the hour between 12 and 5.  All of these values are then passed to the next page, sendparty.asp, where the values are written to the database,  The values sent are as follows:  Activity  Skill  Number of adults  Number infants  Numberchildren  Numberteens  Date  Time  duration |
| BOOKPRIV.ASP | This page is used for private 1 on 1 bookings , firstly, the customer is shown the individual course id of the course they are booking, for reference purposes, this is calculated using the SQL function called TOP 1 to find the next value that will be assigned when this course is written.  The user then chooses their activity, snowboarding or skiing, the user then chooses their skill level, beginner, improver, intermediate, advanced and expert.  The user then states the age range that the person taking the course would be.  At the end of the form the user can choose the date, time, and duration of the course, which is fully validated using REGEX to ensure dates are in the correct format, and to ensure the date has not already passed, furthermore the time field is validated to ensure they separate the time by colon, and the time must be on the hour between 12 and 5.  These values are then passed via the most method to the next page,sendpriv.asp:  Activity  Skill  Age range  Date  Time  Duration |
| INPUTCUSTOMER.ASP | This page is used for users to register to the website as a member, firstly, the customer is shown the individual client id for their use on the site whenever they want to use it in future for making bookings, the course they are booking, for reference purposes, this is calculated using the SQL function called TOP 1 to find the next value that will be assigned when this course is written. This page contains a form which values will be carried using the post method to storeinputcustomer.asp, the values are as follows:  First name: txtfirstname  Surname: txtlastname  Address line 1:txtaddressln1  Address line 2:txtaddressln2  Postcode: txtpostcode  Email address: txtemail  Date of Birth: txtdob  Contact no: txtphoneno  Emergency : txtemergency  Password: txtpassword |
| LOGONREAL.ASP | This page is a simple log on page, with 2 form text inputs, one for ID and one for password, this form also contains a hyperlink to the user registration page which is handy in case the user has not used the service before. The values of this page are carried via the post method to logonreal1.asp |
| REMOVECUST1.ASP | This is used by administrators only.  On this page is a simple form where the administrator can enter a customers id so it can be removed from the system, the value client id, is passed to the next page via the post method, removecust2.asp |
| REMOVECUSTGRPMEM1.ASP | This is used by administrators only. On this page is a simple form where the administrator can enter a customers id and a group course number so their booking to that course can be removed from the system, the value client id and courseno, is passed to the next page via the post method, removecustgrpmem2.asp |
| REMOVEGRPCOURSE1.ASP | This is used by administrators only. On this page is a simple form where the administrator can enter a group course number so it can be removed from the system, the value courseno, is passed to the next page via the post method, removegrpcourse2.asp |
| REMOVEPARTYCUST1.ASP | This is used by administrators only. On this page is a simple form where the administrator can enter a course id so it can be removed from the system, the value courseno, is passed to the next page via the post method, removepartycust2.asp |
| REMOVEPRIVCOURSE1.ASP | This is used by administrators only. On this page is a simple form where the administrator can enter a course id so it can be removed from the system, the value courseno, is passed to the next page via the post method, removeprivcourse2.asp |
| SEARCHCUSTOMERDETAILS.ASP | This is used by administrators only.  On this page there is a single form , where administrative staff can input a customers unique id, and then via the post method it is carried to the next page getcustomerdetails.asp containing SQL queries to show the customers details. |
| SEARCHPARTY.ASP | This is used by administrators only.  This is one of the main query pages for the administrative staff, they will be using this and the page below mainly, as these pages contains all search queries for the user to see.  The first option is to search for courses by activity, to do this there is a combo box which is populated using the SQL SELECT DISTINCT activity From TBLPARTY, this means only values of that sort will come up, the user can then choose one and press sumbit, then they will be directed to the page “getactivityparty.asp” which runs an sql query based on the search criteria and displays the results.  The second option is to search for all courses made by one customer, to do this there is a text box where the user inputs a customers id, the user can then press sumbit, then they will be directed to the page “getcustomerparty.asp” which runs an sql query based on the search criteria and displays the results.  The final option is to search for courses by date, or to be specific between 2 chosen dates, to do this there are 2 combo boxes which are populated using the SQL SELECT DISTINCT date From TBLPARTY, this means only values of that sort will come up, the user can then choose both(Different) and press sumbit, then they will be directed to the page “getdatesparty.asp” which runs an sql query based on the search criteria and displays all of the results. |
| SEARCHPRIVCOURSE.ASP | This is used by administrators only.  This is used by administrators only.  This is the other query page for the administrative staff, they will be using this and the page above mainly, as these pages contains all search queries for the user to see.  The first thing available to search by is instructor, to do this there is a combo box which is populated using the SQL SELECT DISTINCT instructor From TBLPrivate, this means only values of that sort will come up, the user can then choose one and press sumbit, then they will be directed to the page “getinstructorpriv.asp” which runs an sql query based on the search criteria and displays the results.  The second option is to search for courses by activity, to do this there is a combo box which is populated using the SQL SELECT DISTINCT activity From TBLPrivate, this means only values of that sort will come up, the user can then choose one and press sumbit, then they will be directed to the page “getactivitypriv.asp” which runs an sql query based on the search criteria and displays the results.  The penultimate option is to search for all courses made by one customer, to do this there is a text box where the user inputs a customers id, the user can then press sumbit, then they will be directed to the page “getcustomerpriv.asp” which runs an sql query based on the search criteria and displays the results.  The final option is to search for courses by date, or to be specific between 2 chosen dates, to do this there are 2 combo boxes which are populated using the SQL SELECT DISTINCT date From TBLprivate, this means only values of that sort will come up, the user can then choose both(Different) and press sumbit, then they will be directed to the page “getdatespriv.asp” which runs an sql query based on the search criteria and displays all of the results. |
| SHOWGROUP.ASP | This is the most complex page on the site, it features a detailed set of SQL codes within loops ,  Firstly on the page asp will calculate the current date and save it as a variable called currdate.  The code will then retrieve the variable activity from the previous page”bookgroup.asp” following this, the SQL code will be executed, getting all courses that are of the selected activity type, and are after todays date(currdate). And then proceed to write the headings for the table,  After this the do while not eof loop begins, which will allow entries tto be continuously written to the table until it is the end of the file,  Within this loop is another sql query which is executed, which selects coursenumber and count of client id from tblcustgroup, and stores it as a variable called countofclientid, where courseno(from tblgroup) from the previous sql code and courseno (from tblcustgroup) are equal, this allows the code to only show courses that are fully booked by only writing the information to the table if the countofclient id is smaller than the maximum amount of entries to each course from the previous search.  Then the entries have passed through the criteria, if successful, will be written to the table, otherwise the code will go to the next entry in the table and try again, until it reached the end of the loop.  Following this information being displayed, the user then chooses a course they wish to do and enters the coursenumber into the text box at the button, the result of which is then passed via the post method to sendgroup.asp. |
| UPDATEDETAILS.ASP | This page is used for users to update their individual contact data Eg email, password etc. This page contains a form which values will be carried using the post method to updatedetails2.asp, the values are as follows:  First name: txtfirstname  Surname: txtlastname  Address line 1:txtaddressln1  Address line 2:txtaddressln2  Postcode: txtpostcode  Email address: txtemail  Date of Birth: txtdob  Contact no: txtphoneno  Emergency : txtemergency  Password: txtpassword |
| UPDATESINSTRUCTORS1.ASP | This is used by administrators only.  This is where they choose the instructors for the week ahead, there is one form with 14 entries, the first seven are the snowboarding instructors for each day, these variables are:  Bimon  Bitues  Biwed  Bithurs  Bifri  Bisat  Bisun  All of these stand for boarding instructor then the day,  Following this is the skiing instructors, this is:  Skimon  Skitues  Skiwed  Skithurs  Skifri  Skisat  Skisun  These are the variable for the instructors for skiing each week day, these are passed via the post method to the next page, updatesinstructors2.asp |

## 

## VALIDATION FILE SCRIPTS.JS

function valgroup()<!-- stops user from not entering blank data on the showgroup.asp form -->

{

var courseno=document.getElementById('courseno').value;

if (courseno==null || courseno=="")

{

alert ("Please choose a course number before sumbitting!");

return false;

}

}

function actskill() <!-- disables skill level input when tubing is selected -->

{

if(document.getElementById('a3').checked)

{

activity = document.getElementById('a3').value;

skill\_level = "n/a"

document.getElementById("s1").disabled=true;

document.getElementById("s2").disabled=true;

document.getElementById("s3").disabled=true;

}

else

{

document.getElementById("s1").disabled=false;

document.getElementById("s2").disabled=false;

document.getElementById("s3").disabled=false;

}

}

function negative1() <!-- validates 1st number box -->

{ numberadults=document.forms["formparty"]["numberadults"].value;

if (numberadults>0)

{

document.getElementById("down1").disabled=false;

}

else

{

document.getElementById("down1").disabled=true;

}

}

function negative2() <!-- validates 2nd number box -->

{ numberinfants=document.forms["formparty"]["numberinfants"].value;

if (numberinfants>0)

{

document.getElementById("down2").disabled=false;

}

else

{

document.getElementById("down2").disabled=true;

}

}

function negative3() <!-- validates 3rd number box -->

{ numberchildren=document.forms["formparty"]["numberchildren"].value;

if (numberchildren>0)

{

document.getElementById("down3").disabled=false;

}

else

{

document.getElementById("down3").disabled=true;

}

}

function negative4() <!-- validates 4th number box -->

{ numberteens=document.forms["formparty"]["numberteens"].value;

if (numberteens>0)

{

document.getElementById("down4").disabled=false;

}

else

{

document.getElementById("down4").disabled=true;

}

}

function validateuser1()<!-- form validation for the inputcustomer.asp page -->

{

txtfirstname=document.forms["frmAddCustomer"]["txtFirstName"].value;

txtlastname=document.forms["frmAddCustomer"]["txtlastname"].value;

txtaddressln1=document.forms["frmAddCustomer"]["txtaddressln1"].value;

txtaddressln2=document.forms["frmAddCustomer"]["txtaddressln2"].value;

txtpostcode=document.forms["frmAddCustomer"]["txtpostcode"].value;

txtemail=document.forms["frmAddCustomer"]["txtemail"].value;

txtdob=document.forms["frmAddCustomer"]["txtdob"].value;

txtphoneno=document.forms["frmAddCustomer"]["txtphoneno"].value;

txtemergency=document.forms["frmAddCustomer"]["txtemergency"].value;

txtpassword=document.forms["frmAddCustomer"]["txtpassword"].value;

atpos=txtemail.indexOf("@");

dotpos=txtemail.lastIndexOf(".");

if (txtfirstname==null || txtfirstname=="")<!-- stops the text box from being empty -->

{

alert ("first name must be filled out");

return false;

}

if (txtlastname==null || txtlastname=="")<!-- stops the text box from being empty -->

{

alert ("last name must be filled out");

return false;

}

if (txtaddressln1==null || txtaddressln1=="" ||txtaddressln2==null || txtaddressln2=="")<!-- stops the text box from being empty -->

{

alert ("address must be correctly filled out");

return false;

}

if (txtpostcode==null || txtpostcode=="")<!-- stops the text box from being empty -->

{

alert ("postcode must be filled out");

return false;

}

if (txtemail==null || txtemail=="" || atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)<!-- stops the text box from being empty and email validation to ensure there is an "@" and a "." -->

{

alert("invalid or empty e-mail address, please try again");

return false;

}

if (txtphoneno==null || txtphoneno=="")<!-- stops the text box from being empty -->

{

alert ("phone number must be filled in");

return false;

}

if (txtemergency==null || txtemergency=="")<!-- stops the text box from being empty -->

{

alert ("emergency contact number must be filled in");

return false;

}

if (txtpassword==null || txtpassword=="")<!-- stops the text box from being empty -->

{

alert ("password must be filled in");

return false;

}

if (txtdob==null || txtdob=="" || txtdob.length>7 || txtdob.length<10)<!-- stops the text box from being empty and ensures it is of a valid length -->

{

alert ("date of birth must be filled in correctly");

return false;

}

}

function validateuser2()<!-- validation for the second user information form updatedetails.asp -->

{

txtfirstname=document.forms["updateuser"]["txtFirstName"].value;

txtlastname=document.forms["updateuser"]["txtlastname"].value;

txtaddressln1=document.forms["updateuser"]["txtaddressln1"].value;

txtaddressln2=document.forms["updateuser"]["txtaddressln2"].value;

txtpostcode=document.forms["updateuser"]["txtpostcode"].value;

txtemail=document.forms["updateuser"]["txtemail"].value;

txtdob=document.forms["updateuser"]["txtdob"].value;

txtphoneno=document.forms["updateuser"]["txtphoneno"].value;

txtemergency=document.forms["updateuser"]["txtemergency"].value;

txtpassword=document.forms["updateuser"]["txtpassword"].value;

atpos=txtemail.indexOf("@");

dotpos=txtemail.lastIndexOf(".");

if (txtfirstname==null || txtfirstname=="")<!-- stops the text box from being empty -->

{

alert ("first name must be filled out");

return false;

}

if (txtlastname==null || txtlastname=="")<!-- stops the text box from being empty -->

{

alert ("last name must be filled out");

return false;

}

if (txtaddressln1==null || txtaddressln1=="" ||txtaddressln2==null || txtaddressln2=="")<!-- stops the text box from being empty -->

{

alert ("address must be correctly filled out");

return false;

}

if (txtpostcode==null || txtpostcode=="")<!-- stops the text box from being empty -->

{

alert ("postcode must be filled out");

return false;

}

if (txtemail==null || txtemail=="" || atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length)<!-- stops the text box from being empty and email validation to ensure there is an "@" and a "." -->

{

alert("invalid or empty e-mail address, please try again");

return false;

}

if (txtphoneno==null || txtphoneno=="")<!-- stops the text box from being empty -->

{

alert ("phone number must be filled in");

return false;

}

if (txtemergency==null || txtemergency=="")<!-- stops the text box from being empty -->

{

alert ("emergency contact number must be filled in");

return false;

}

if (txtpassword==null || txtpassword=="")<!-- stops the text box from being empty -->

{

alert ("password must be filled in");

return false;

}

if (txtdob==null || txtdob=="" || txtdob.length>7 || txtdob.length<10)<!-- stops the text box from being empty and ensures it is of a valid length -->

{

alert ("date of birth must be filled in correctly");

return false;

}

}

function validateFormparty() <!-- validates fields in bookparty.asp-->

{

dateinput=document.forms["formparty"]["mydate"].value;

hrs=document.forms["formparty"]["mytime"].value

hours1=hrs.split(":")[0]

mins=hrs.split(":")[1]

validformat=/^\d{1,2}\/\d{1,2}\/\d{4}$/

dayfield=dateinput.split("/")[0];

monthfield=dateinput.split("/")[1];

yearfield=dateinput.split("/")[2];

dayobj = new Date(yearfield,monthfield-1,dayfield);

var result=validformat.test(dateinput)

numberadults=document.forms["formparty"]["numberadults"].value;

numberinfants=document.forms["formparty"]["numberinfants"].value;

numberchildren=document.forms["formparty"]["numberchildren"].value;

numberteens=document.forms["formparty"]["numberteens"].value;

if (document.getElementById('a1').checked) <!-- stops the activity radio buttons from being empty -->

{

activity = document.getElementById('a1').value;

}

else if (document.getElementById('a2').checked)

{

activity = document.getElementById('a2').value;

}

else if(document.getElementById('a3').checked)

{

activity = document.getElementById('a3').value;

}

else

{

alert ("please choose an activity");

return false;

}

if (activity=="skiing" || activity=="snowboarding")<!-- if tubing is not selected, ensures the user also chooses a skill level -->

{

if (document.getElementById('s1').checked)

{

skill\_level = document.getElementById('s1').value;

}

else if (document.getElementById('s2').checked)

{

skill\_level = document.getElementById('s2').value;

}

else if(document.getElementById('s3').checked)

{

skill\_level = document.getElementById('s3').value;

}

else

{

alert ("please select a skill level");

return false;

}

}

if (numberadults=="0" && numberinfants=="0" && numberchildren=="0" && numberteens=="0")<!-- checks there is a min of 1 person in total on the course-->

{

alert ("invalid! nobody is attending the course!");

return false;

}

if (result==false)<!-- regex validation for date , basic format check -->

{

alert("Invalid Date Format. Please correct and submit again.")

return false

}

else if (dayobj<currdate)<!-- ensures chosen date is after the current date -->

{

alert("the date you have selected has already passed- please try again.");

return false;

}

if (mins!=0)<!-- ensures the time of the course is on the hour -->

{

alert ("only on the hour. please change time/ seperate the time by a colon, (:)");

return false;

}

if (hours1<12 || hours1>17)<!--stops a time entry that is not in centre operating hours-->

{

alert ("invalid time! must be between 12.00 and 17.00!");

return false;

}

}

function validateFormpriv() <!-- validates empty fields in page-->

{

dateinput=document.forms["formpriv"]["mydate"].value;

hrs=document.forms["formpriv"]["mytime"].value

validformat=/^\d{1,2}\/\d{1,2}\/\d{4}$/

dayfield=dateinput.split("/")[0];

monthfield=dateinput.split("/")[1];

yearfield=dateinput.split("/")[2];

hours1=hrs.split(":")[0]

mins=hrs.split(":")[1]

dayobj = new Date(yearfield,monthfield-1,dayfield);

var result=validformat.test(dateinput)

age=document.getElementById('selectage').value;

skill\_level=document.getElementById('selectskill').value;

if (document.getElementById('a1').checked) <!-- stops the activity radio buttons from being empty -->

{

activity = document.getElementById('a1').value;

}

else if (document.getElementById('a2').checked)

{

activity = document.getElementById('a2').value;

}

else

{

alert ("please choose an activity");

return false;

}

if (skill\_level== null ||skill\_level=="") <!-- stops the skill level combo box from being empty -->

{

alert ("please select a skill level");

return false;

}

if (age== null || age=="") <!-- stops the age range combo box from being empty -->

{

alert ("please select an age range");

return false;

}

if (mins!=0)<!-- ensures the time of the course is on the hour -->

{

alert ("only on the hour. please change time/ seperate the time by a colon, (:)");

return false;

}

if (hours1<12 || hours1>17)<!--stops a time entry that is not in centre operating hours-->

{

alert ("invalid time! must be between 12.00 and 17.00!");

return false;

}

if (result==false)<!-- regex validation for date , basic format check -->

{

alert("Invalid Date Format. Please correct and submit again.")

return false;

}

else if (dayobj<currdate)<!-- ensures chosen date is after the current date -->

{

alert("the date you have selected has already passed- please try again.");

return false;

}

}

## A sample of detailed algorithm design

This is the code from sendpriv.asp, this uses the variables passed from bookpriv.asp as explained above, and ultimately writes them to the database, during this process, 2 other processes must occur prior to this, firstly the slope the course will be occurring on will have to be decided, this is the highlighted code in yellow that determines slope dependant on activity/skill chosen on the previous page.

The majority of the designed algorithms are stored in the design section; these are merely pieces of code from the system being explained.

The next piece of code is to calculate the instructor who will be running the course, highlighted in green, firstly the code calculates the day of the week that the course will be on by the function instructorday= WeekdayName(weekday(date1))

Date1 as specified earlier is the date input by the user, upon this calculation the result is stored as instructorday, the next code to be done in relationship to this is the code:

instructorsql="select \* FROM tblinstructor AS instructor WHERE dayavailable='"&instructorday&"' AND activity='"&activity&"';"

instructorcheck.open instructorsql,db

instructor=INSTRUCTORCHECK("instructor")

this checks tblinstuctor to see which instructor is assigned to that activity on that day of the week, and saves it as a variable name “instructor”,

Next, the code runs a query to check if the specified is already running a course at the specified time, this prevents double bookings with the following code:

chksql="select courseno AS cavailable FROM tblprivate where instructor='"&instructor&"' AND date=#"&date1&"# AND time=#"&time1&"#;"

coursecheck.open chksql,db

available=coursecheck("cavailable")

if isnull(available) then

the chksql query checks wether the specified instructor is running a course at the set time, and stores it as a variable “cavailable” which is then stored in asp as “available”.

Naturally if the instructor is not running a course at the time, the value of “available” will be null, as no results will have been returned from the query. So the next portion uses the function ISNULL to see if the variable is null, if this is the case, then all of the specified values are written to the database, as shown in the INSERT INTO string above. Otherwise if a course is running, the user Is redirected to the previous page after 5 seconds to change the booking.

<%

activity=request.form("activity")

skill=request.form("selectskill")

agerange=request.form("selectage")

duration=request.form("duration")

date1=request.form("mydate")

time1=request.form("mytime")

clientID=session("ClientID")

instructorday= WeekdayName(weekday(date1))

if activity="snowboarding" then slope="funpark" else if skill="beginner" then slope="beginner slope" else slope="main slope"

dim orseof

dim instructorcheck

set db=server.createobject("adodb.connection")

set orseof=server.createobject("adodb.recordset")

set instructorcheck=server.createobject("adodb.recordset")

set coursecheck=server.createobject("adodb.recordset")

db.open "DRIVER={Microsoft Access Driver (\*.mdb)}; DBQ=" & Server.MapPath("../linked/skiRossy.mdb")

instructorsql="select \* FROM tblinstructor AS instructor WHERE dayavailable='"&instructorday&"' AND activity='"&activity&"';"

instructorcheck.open instructorsql,db

instructor=INSTRUCTORCHECK("instructor")

chksql="select courseno AS cavailable FROM tblprivate where instructor='"&instructor&"' AND date=#"&date1&"# AND time=#"&time1&"#;"

coursecheck.open chksql,db

available=coursecheck("cavailable")

if isnull(available) then

strSQL = "INSERT INTO tblpriv (clientID,activity,instructor,skill,date,time,duration,agerange,slope) values ("&clientID&",'"&activity&"','"&skill&"','"&instructor&"',#"&date1&"#,#"&time1&"#,"&duration&",'"&agerange&"','"&slope&"');"

orseof.open strSQL, db

else

response.Write("there is already a course running at this time, please try again, you will be returned to the previous page in 5 seconds. if you are not , please click")%> <a href "bookpriv.asp">here</a><%

end if

%>

Getdatesparty.asp-

This code is for an admin based query- used to see a list of courses between 2 set dates, the variables, date1 and date2 are retrieved from the previous form, searchparty.asp.

The SELECT function is used to return all values from the database that fulfil the criteria specified, here, as you can see, it shows any with a date larger than date1 and smaller than date2.

Following this, a do until end of file loop is used, to continuously write lines of data returned until the end of the file.

<%

date1 = Request.Form("selectdate1")

date2 = Request.Form("selectdate2")

response.expires=-1

set db=server.createobject("adodb.connection")

set orseof=server.createobject("adodb.recordset")

db.open "DRIVER={Microsoft Access Driver (\*.mdb)}; DBQ=" & Server.MapPath("../linked/skiRossy.mdb")

sql="SELECT \* FROM tblparty WHERE date>=#"&date1&"# AND date<=#"&date2&"# ORDER BY date ASC"

orseof.Open sql,db

do until orseof.EOF

response.write("<table>")

for each x in orseof.Fields

response.write("<tr><td><b>" & x.name & "</b></td>")

response.write("<td>" & x.value & "</td></tr>")

next

response.write("<br>")

response.write("</table>")

orseof.MoveNext

loop

%>

Removeprivcourse2.asp-

This is another admin based task- removing an item from the database,<%

courseno=request.form("courseno")

dim orseof

set db=server.createobject("adodb.connection")

set orseof=server.createobject("adodb.recordset")

db.open "DRIVER={Microsoft Access Driver (\*.mdb)}; DBQ=" & Server.MapPath("../linked/skiRossy.mdb")

strSQL = "DELETE FROM tblprivate WHERE courseno="&courseno&";"

orseof.Open strSQL, db

%>

This used the sql DELETE FROM function

## Procedure and variable lists/descriptions for programs OR list of package items developed

var txtfirstname;

var txtlastname;

var txtaddressln1;

var txtaddressln2;

var txtpostcode;

var txtemail;

var txtdob;

var txtphoneno;

var txtemergency;

var txtpassword;

var atpos;

var dotpos;

var currdate = new Date();

var duration;

var numberadults;

var numberinfants;

var numberchildren;

var numberteens;

var skill\_level;

var activity;

var age;

var hours1

var dateinput

var validformat

var dayobj

var dayfield

var monthfield

var yearfield

var hrs

var mins

var courseno

var result

used for user information(inputcustomer.asp|updatedetails.asp)

used on private and party course pages(bookpriv.asp|bookparty.asp)

used on party course page only(bookparty)

used on private course page only