COMP1690: Component Programming

Trevor Kiggundu: 001001720
April 2020
BEng Software Engineering

A report submitted in fulfilment of the requirements for the module, Component Programming, Computing and Information Systems Department, University of Greenwich.

Contents

PART A: Group Documentation:

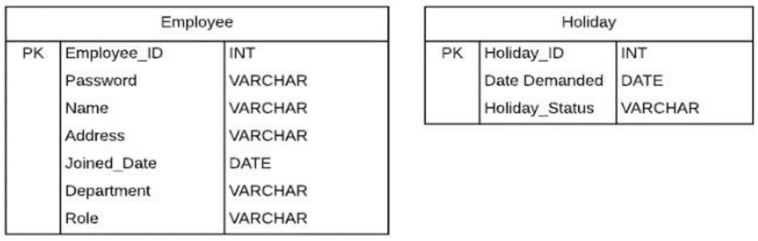
Team Database Design:

LucidChart was used to design the initial ERD. We originally attempted to use Microsoft Access for database management, but we decided to settle for the SQL Server system as we figured it was easier to use with the variety of information available online. LINQ was also a powerful tool that easily allowed us to do what we wished with the commands available.

Normalization of design:

Our original database was flawed, redundant and relied too heavily on large tables. The lack of normalization was evident as our tables lacked the appropriate foreign keys and relationships between them. However, after consultation with the lecturer, we were able to decompose the fields in each table until they were in an atomic state. The 'Employee' and 'Holiday' tables are shown below. Our 'Employee' table was very populated with attributes that could have been listed as their own entities. The 'Role' and 'Department' tables were originally part of the given table.

Initial Idea/Design Without Normalisation:

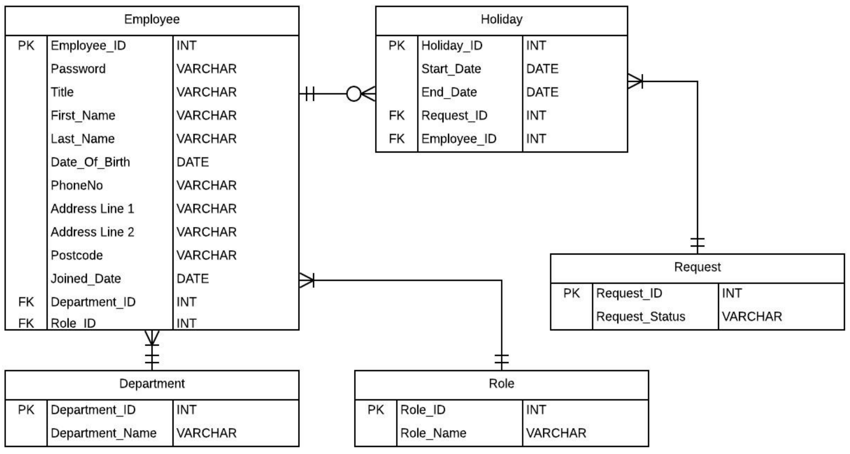


Employee		
PK	Employee_ID	INT
	Password	VARCHAR
	Name	VARCHAR
	Address	VARCHAR
	Joined_Date	DATE
	Department	VARCHAR
	Role	VARCHAR

Holiday		
PK	Holiday_ID	INT
	Date Demanded	DATE
	Holiday_Status	VARCHAR

Final Entity Relationship Diagram (ERD) with Normalization:

After normalising the graph, we split data fields to separate columns and tables so that each piece of data can be represented with corresponding data. Also, we added extra fields to capture the right data with attention to detail. For example: Address Line 1, Address Line 2, Postcode etc.



Employee		
PK	Employee_ID	INT
	Password	VARCHAR
	Title	VARCHAR
	First_Name	VARCHAR
	Last_Name	VARCHAR
	Date_Of_Birth	DATE
	PhoneNo	VARCHAR
	Address Line 1	VARCHAR
	Address Line 2	VARCHAR
	Postcode	VARCHAR
	Joined_Date	DATE
FK	Department_ID	INT
FK	Role_ID	INT

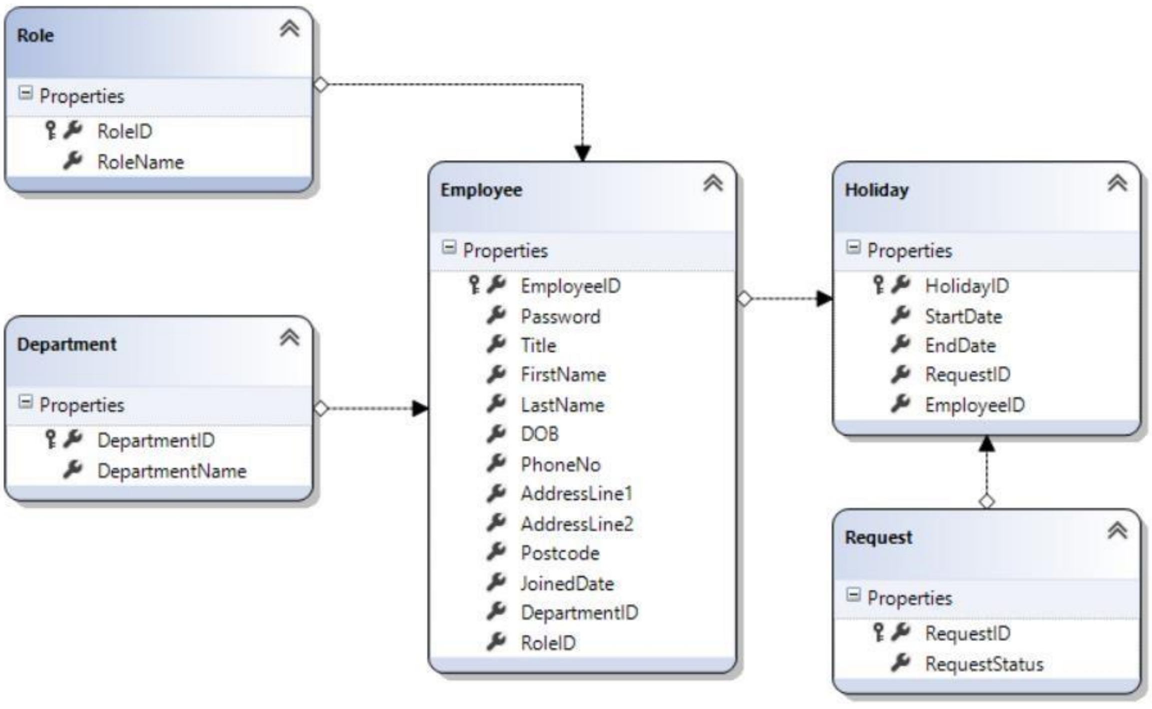
Holiday		
PK	Holiday_ID	INT
	Start_Date	DATE
	End_Date	DATE
FK	Request_ID	INT
FK	Employee_ID	INT

Request		
PK	Request_ID	INT
	Request_Status	VARCHAR

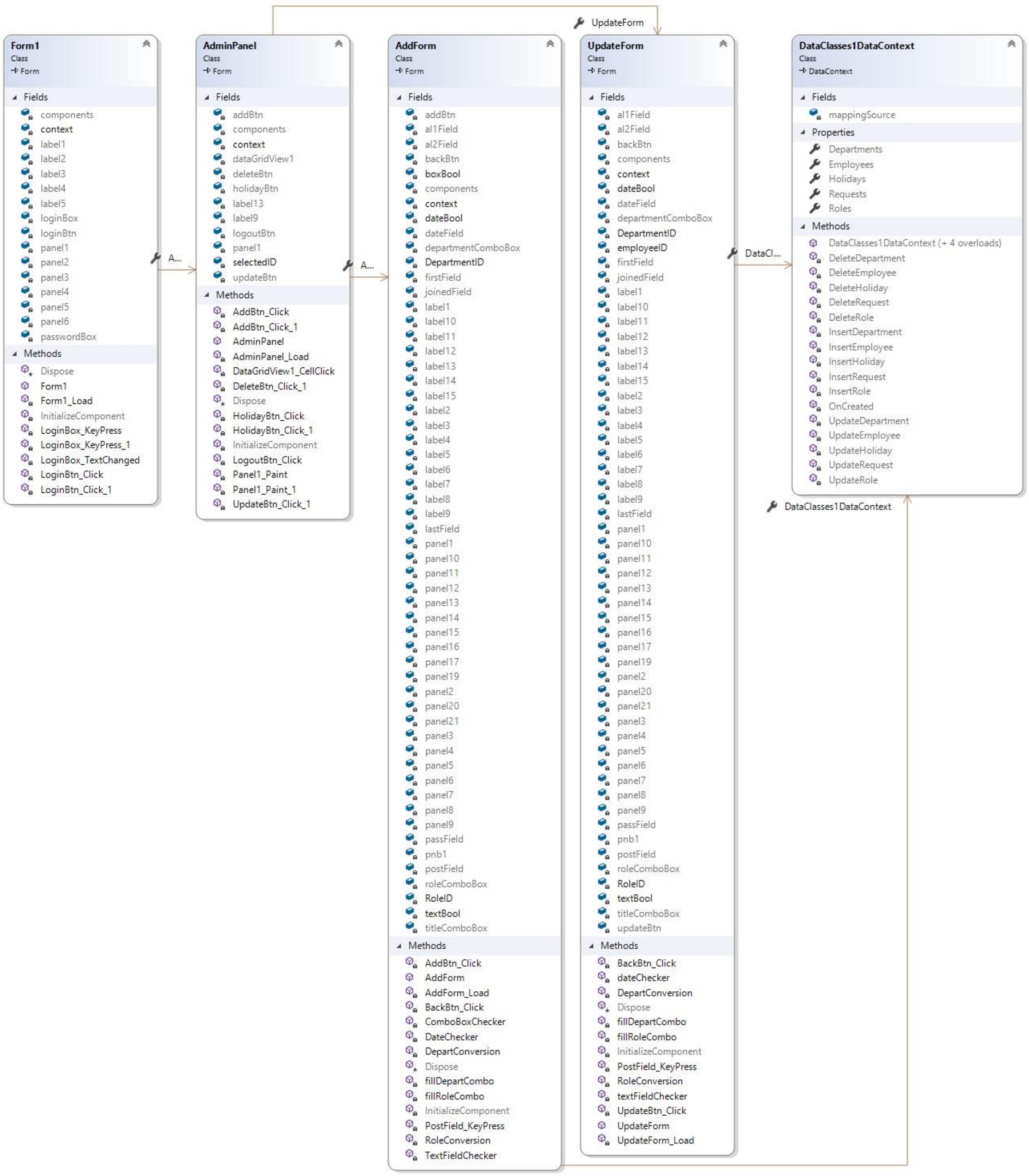
Department		
PK	Department_ID	INT
	Department_Name	VARCHAR

Role		
PK	Role_ID	INT
	Role_Name	VARCHAR

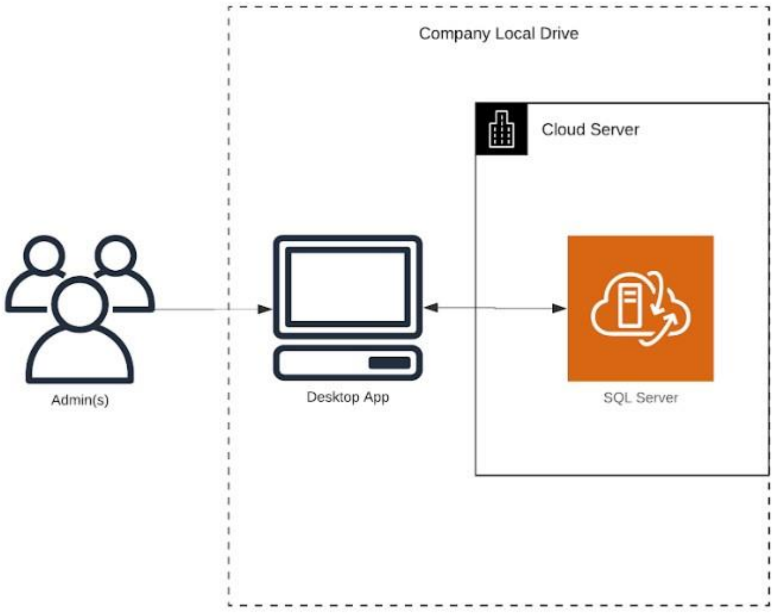
Entity/Model Classes:



Class Diagrams:



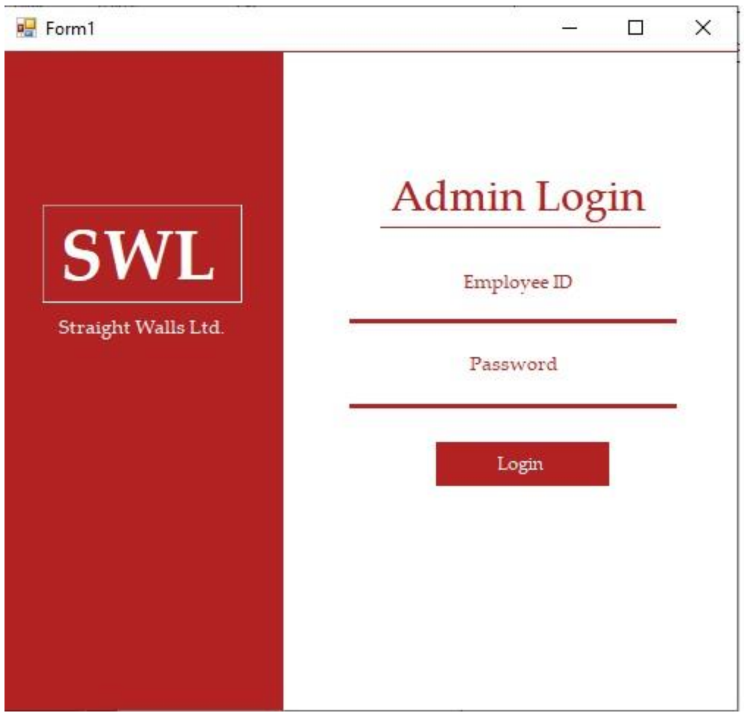
Architecture Diagram:



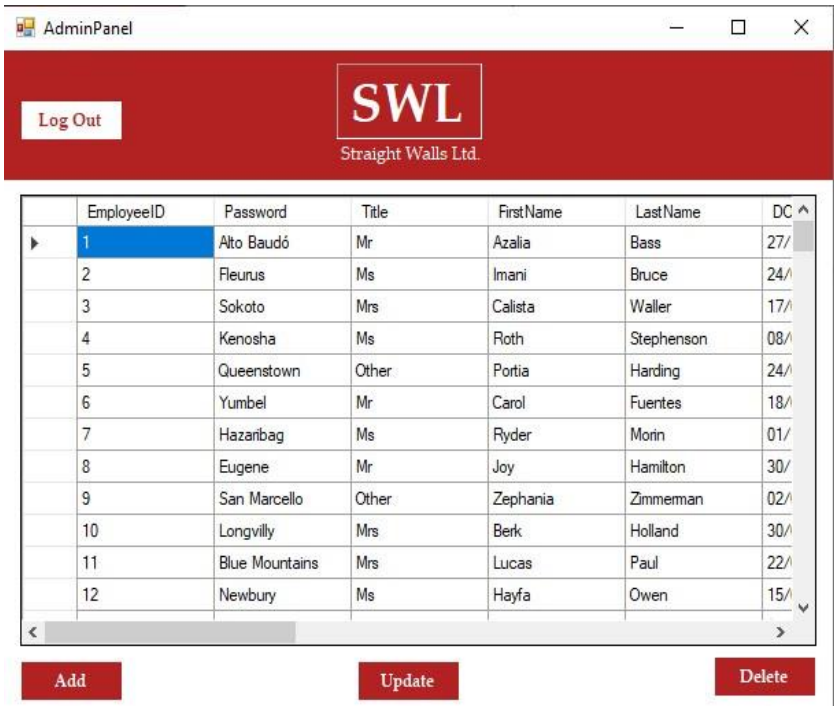
Admin Desktop Application (with screenshots):

Below are the screenshots and caption for each feature of the basic Admin Desktop Application.

Admin Login:

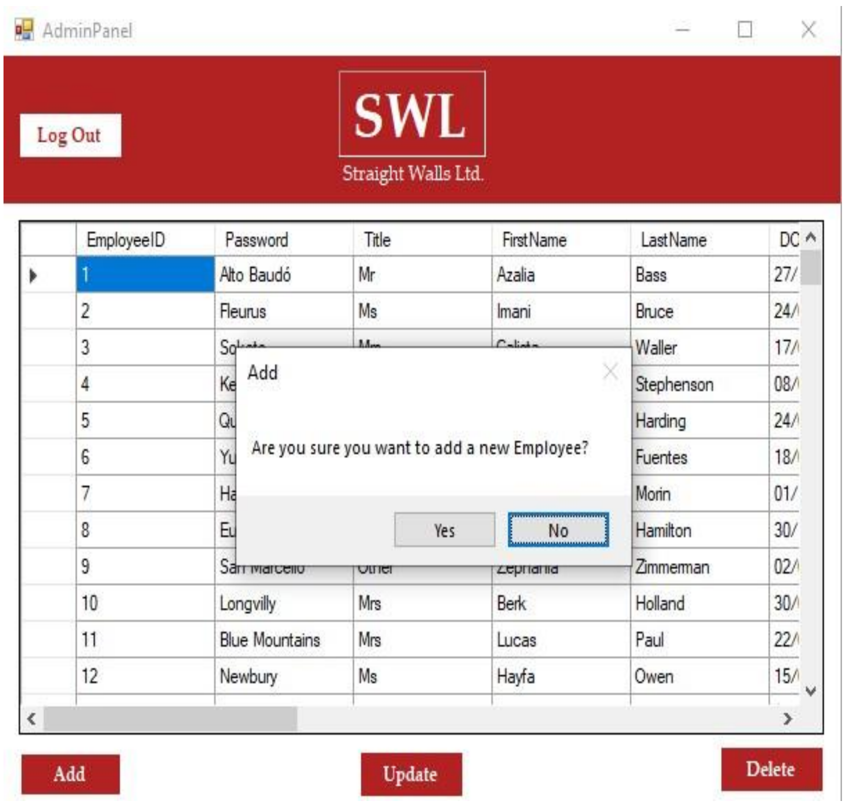


User Management Enabling An Admin User To Create/Edit/Delete Employees:



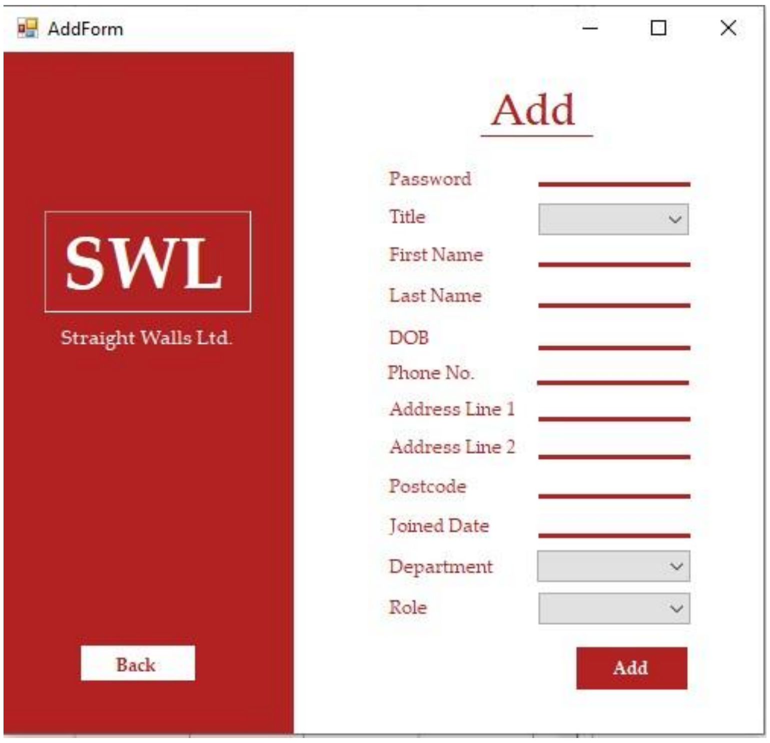
	EmployeeID	Password	Title	FirstName	LastName	DC
▶	1	Alto Baudó	Mr	Azalia	Bass	27/
	2	Fleurus	Ms	Imani	Bruce	24/
	3	Sokoto	Mrs	Calista	Waller	17/
	4	Kenosha	Ms	Roth	Stephenson	08/
	5	Queenstown	Other	Portia	Harding	24/
	6	Yumbel	Mr	Carol	Fuentes	18/
	7	Hazaribag	Ms	Ryder	Morin	01/
	8	Eugene	Mr	Joy	Hamilton	30/
	9	San Marcello	Other	Zephania	Zimmerman	02/
	10	Longvilly	Mrs	Berk	Holland	30/
	11	Blue Mountains	Mrs	Lucas	Paul	22/
	12	Newbury	Ms	Hayfa	Owen	15/

Add Employee Validation:



Are you sure you want to add a new Employee?

It Must Be Possible To Allocate An Employee To A Department And Assign Him/Her A Role:


AddForm

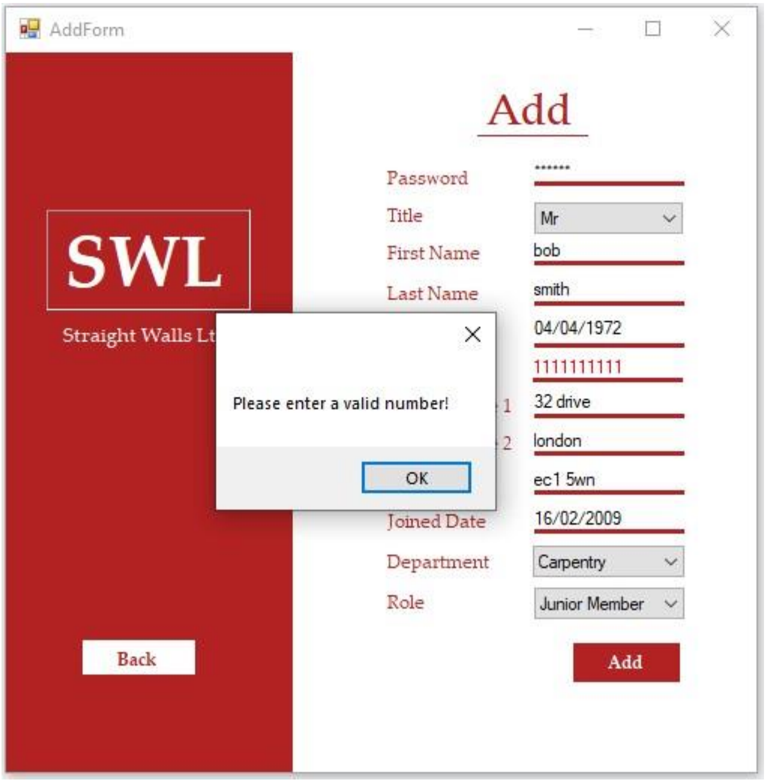
SWL
Straight Walls Ltd.

Add

Password	
Title	
First Name	
Last Name	
DOB	
Phone No.	
Address Line 1	
Address Line 2	
Postcode	
Joined Date	
Department	
Role	

Back
Add

10 Numbers:


AddForm

SWL
Straight Walls Lt

Add

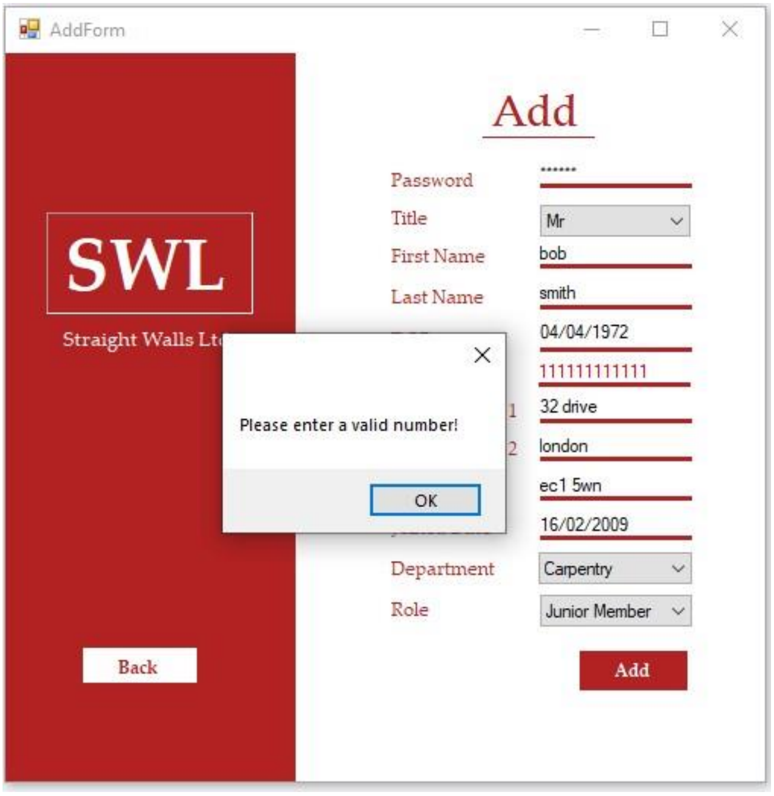
Password	••••••
Title	Mr
First Name	bob
Last Name	smith
	04/04/1972
	1111111111
1	32 drive
2	london
	ec1 5wn
Joined Date	16/02/2009
Department	Carpentry
Role	Junior Member

Please enter a valid number!

OK

Back
Add

12 numbers:


AddForm

SWL

Straight Walls Ltd

Add

Password	******
Title	Mr
First Name	bob
Last Name	smith
	04/04/1972
	111111111111
1	32 drive
2	london
	ec1 5wn
	16/02/2009
Department	Carpentry
Role	Junior Member

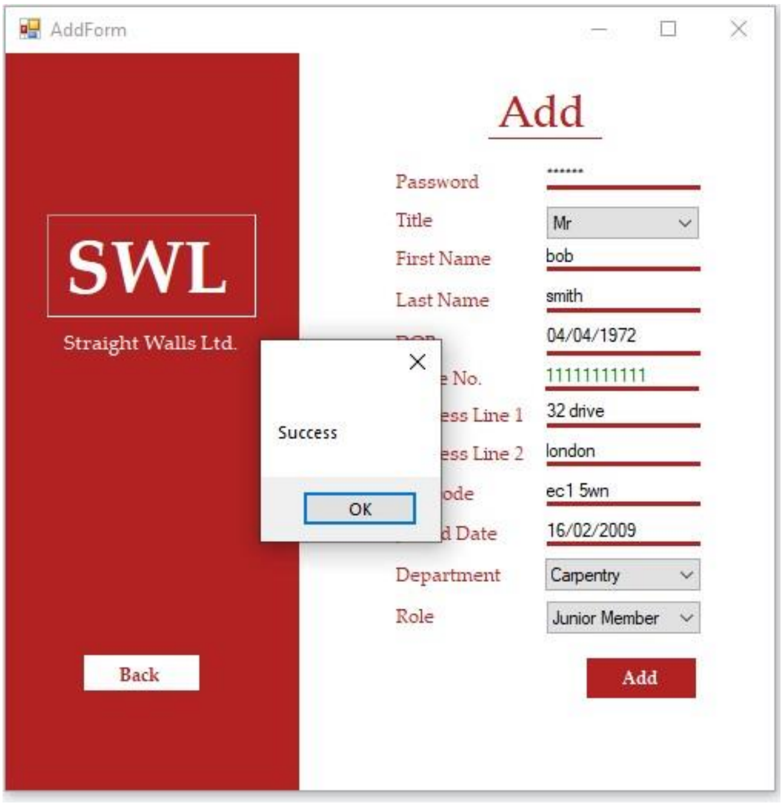
Please enter a valid number!

OK

Back

Add

11 numbers:


AddForm

SWL

Straight Walls Ltd.

Add

Password	******
Title	Mr
First Name	bob
Last Name	smith
	04/04/1972
e No.	11111111111
ess Line 1	32 drive
ess Line 2	london
ode	ec1 5wn
d Date	16/02/2009
Department	Carpentry
Role	Junior Member

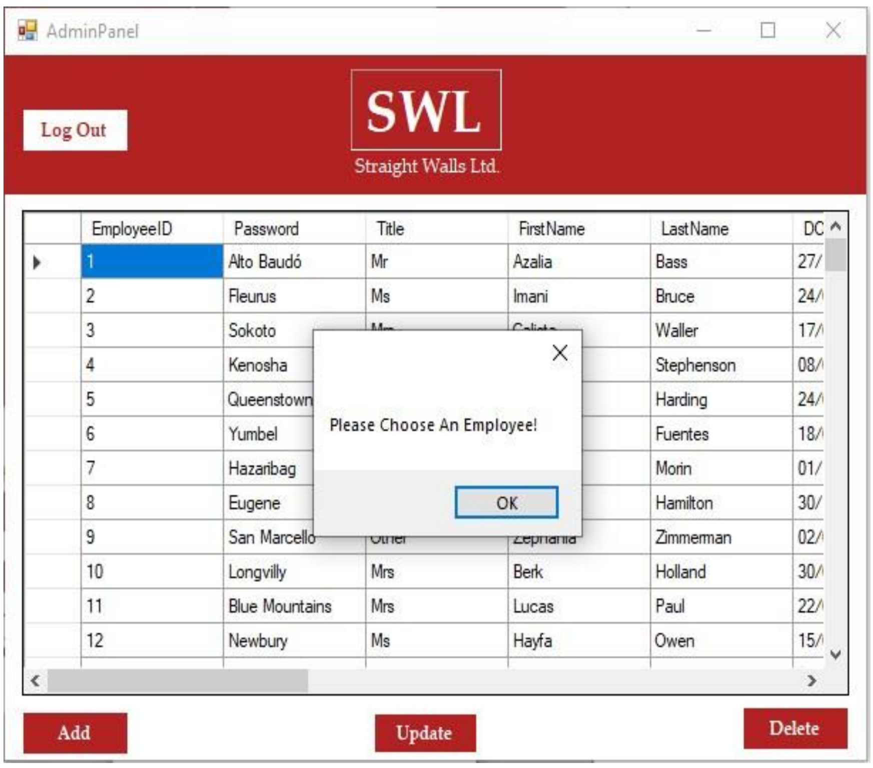
Success

OK

Back

Add

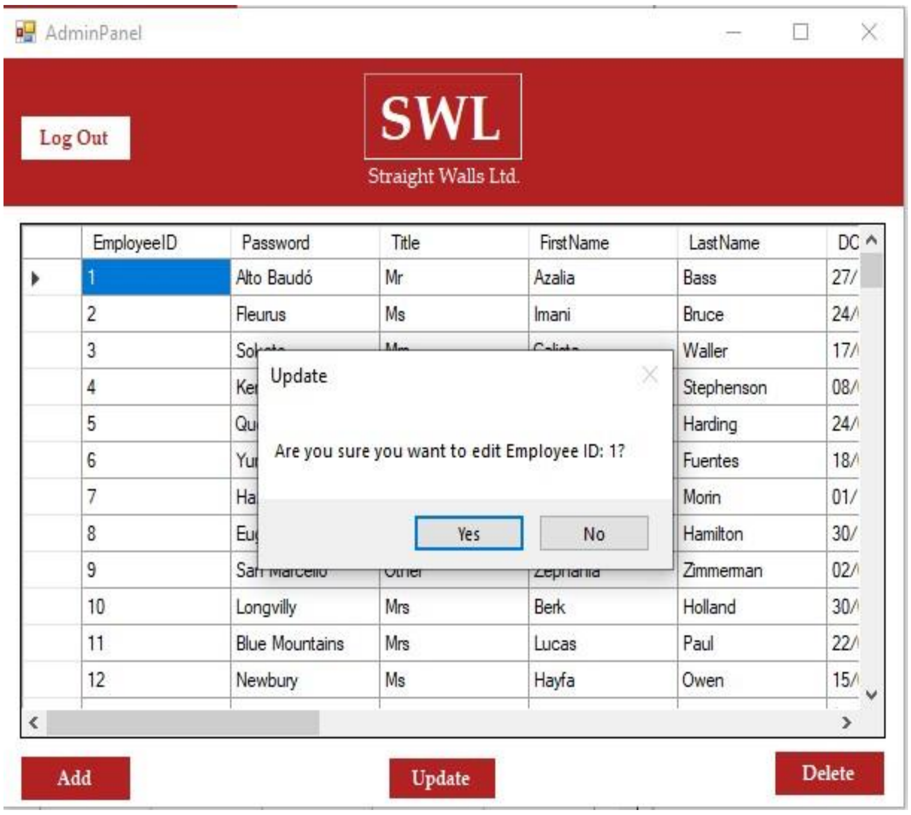
Update/Delete Employee Validation:



EmployeeID	Password	Title	FirstName	LastName	DC
1	Alto Baudó	Mr	Azalia	Bass	27/
2	Fleurus	Ms	Imani	Bruce	24/
3	Sokoto			Waller	17/
4	Kenosha			Stephenson	08/
5	Queenstown			Harding	24/
6	Yumbel			Fuentes	18/
7	Hazaribag			Morin	01/
8	Eugene			Hamilton	30/
9	San Marcello			Zimmerman	02/
10	Longvilly	Mrs	Berk	Holland	30/
11	Blue Mountains	Mrs	Lucas	Paul	22/
12	Newbury	Ms	Hayfa	Owen	15/

Please Choose An Employee!

Update Employee Validation:

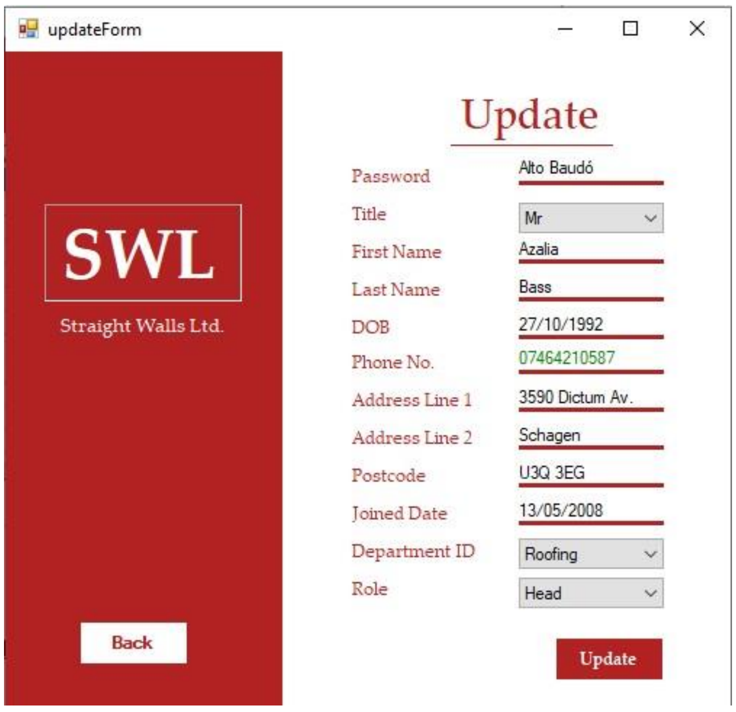


EmployeeID	Password	Title	FirstName	LastName	DC
1	Alto Baudó	Mr	Azalia	Bass	27/
2	Fleurus	Ms	Imani	Bruce	24/
3				Waller	17/
4	Ke			Stephenson	08/
5	Qu			Harding	24/
6	Yu			Fuentes	18/
7	Ha			Morin	01/
8	Eu			Hamilton	30/
9				Zimmerman	02/
10	Longvilly	Mrs	Berk	Holland	30/
11	Blue Mountains	Mrs	Lucas	Paul	22/
12	Newbury	Ms	Hayfa	Owen	15/

Update

Are you sure you want to edit Employee ID: 1?

Update Employee Page:

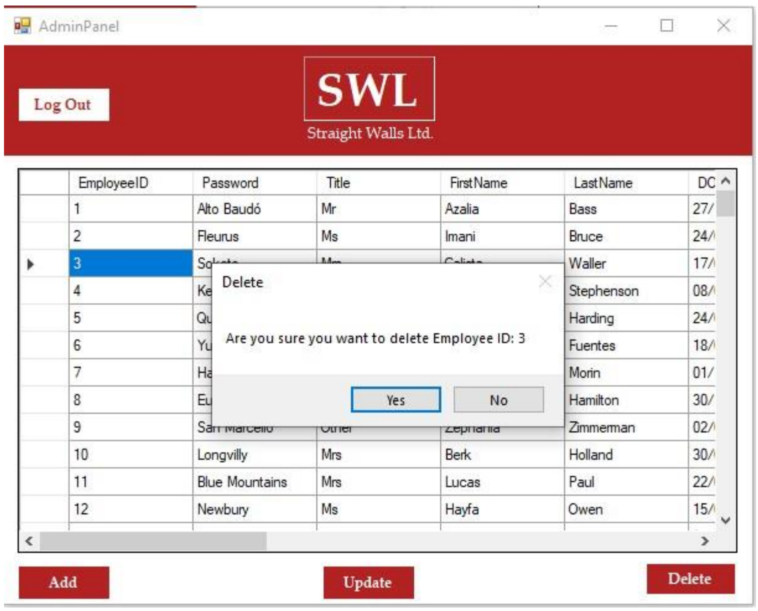


updateForm

Update

Password	Alto Baudó
Title	Mr
First Name	Azalia
Last Name	Bass
DOB	27/10/1992
Phone No.	07464210587
Address Line 1	3590 Dictum Av.
Address Line 2	Schagen
Postcode	U3Q 3EG
Joined Date	13/05/2008
Department ID	Roofing
Role	Head

Delete Employee Validation When Button Is Clicked:



AdminPanel

EmployeeID	Password	Title	FirstName	LastName	DC
1	Alto Baudó	Mr	Azalia	Bass	27/
2	Fleurus	Ms	Imani	Bruce	24/
3	So[illegible]	[illegible]	[illegible]	Waller	17/
4	Ke			Stephenson	08/
5	Qu			Harding	24/
6	Yu			Fuentes	18/
7	Ha			Morin	01/
8	Eu			Hamilton	30/
9	San Marcello	Other	Zephania	Zimmerman	02/
10	Longvilly	Mrs	Berk	Holland	30/
11	Blue Mountains	Mrs	Lucas	Paul	22/
12	Newbury	Ms	Hayfa	Owen	15/

Delete

Are you sure you want to delete Employee ID: 3

Individual Report Section:

I chose to work with Maruf and Rathusan as I felt most comfortable working with them. Rathusan is a competent programmer and Maruf had previous UX/UI experience with his previous modules, so my role was to help both in the creation of the desktop application. We all collaborated to create the entity diagrams using LucidChart, as we were familiar with the technology from previous years. Rathusan did a lot of the initial LINQ and C# programming, something that I had to catch up on later, as I has never used LINQ or SQL Server for database management before. However, I was able to catch up and help design the database and populate the tables effectively. Maruf and I also made the GUI elements for the admin applications, as Visual Studio made it easy due to its drag and drop elements. I learned a lot from Maruf's past UI experience as well. We were then able to implement the program with the redesigned database and get rid of our old program that lacked any significant UI features. Using SQL Server was a hard task at first, due to the fact that I was so used to the MYSQL database management system. However, with the help of my group members, I was able to understand and use my knowledge for the rest of the coursework duration.

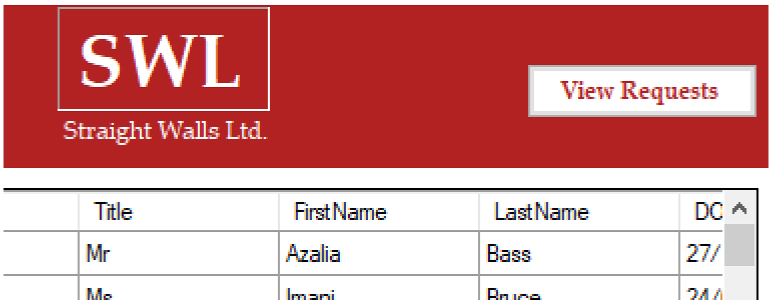
PART B: Individual Technical Documentation:

Annotated Screenshots:

Functionality A:

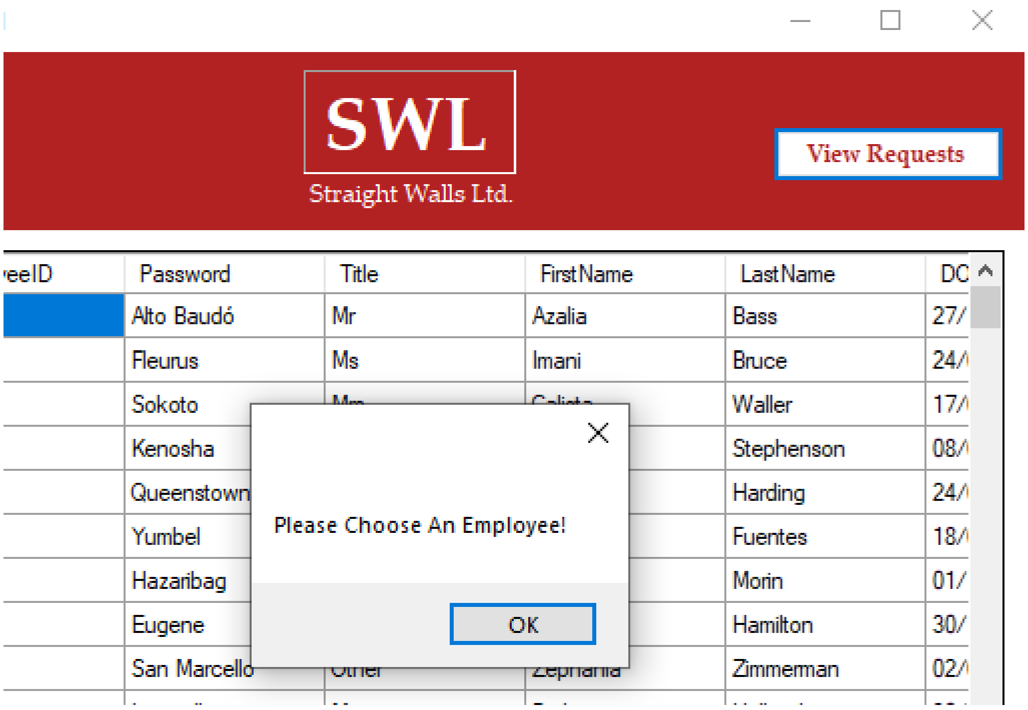
View a list of outstanding holiday requests:

A button called ‘View Requests’ was added to the top right corner of the Admin Form. This was created to lead the user to the ‘Holiday Form’ where the Admin was able to view holiday requests from other employees.



	Title	FirstName	LastName	DC
	Mr	Azalia	Bass	27/
	Ms	Imani	Bruce	24/

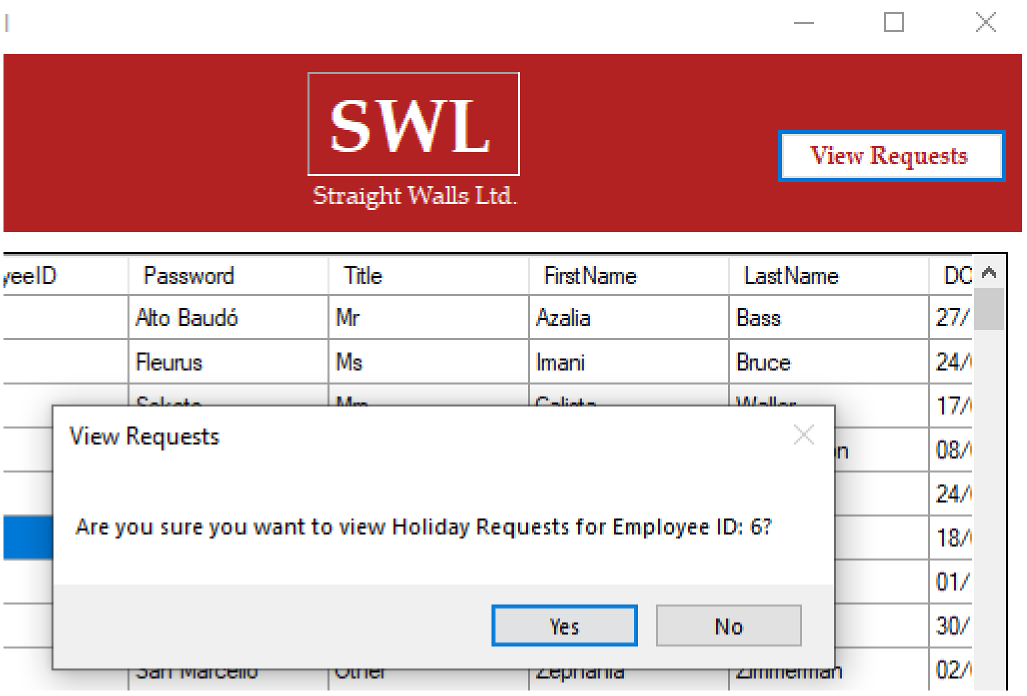
Added validation means that the user must choose an employee, or an error message box will be prompted.



eeID	Password	Title	FirstName	LastName	DC
	Alto Baudó	Mr	Azalia	Bass	27/
	Fleurus	Ms	Imani	Bruce	24/
	Sokoto	Mrs	Calista	Waller	17/
	Kenosha			Stephenson	08/
	Queenstown			Harding	24/
	Yumbel			Fuentes	18/
	Hazaribag			Morin	01/
	Eugene			Hamilton	30/
	San Marcello	Other	Zephania	Zimmerman	02/

Please Choose An Employee!

OK

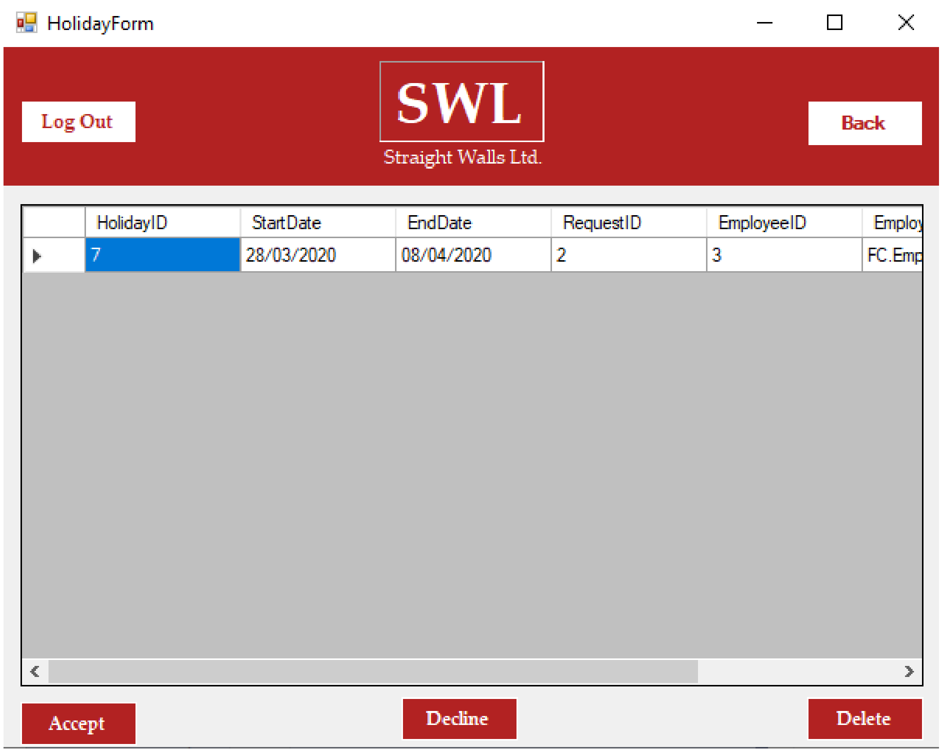


yeeID	Password	Title	FirstName	LastName	DC
	Alto Baudó	Mr	Azalia	Bass	27/
	Fleurus	Ms	Imani	Bruce	24/
	Sakata	Mrs	Calista	Waller	17/
				n	08/
					24/
					18/
					01/
					30/
	San Marcello	Other	Zephania	Zimmerman	02/

View Requests

Are you sure you want to view Holiday Requests for Employee ID: 6?

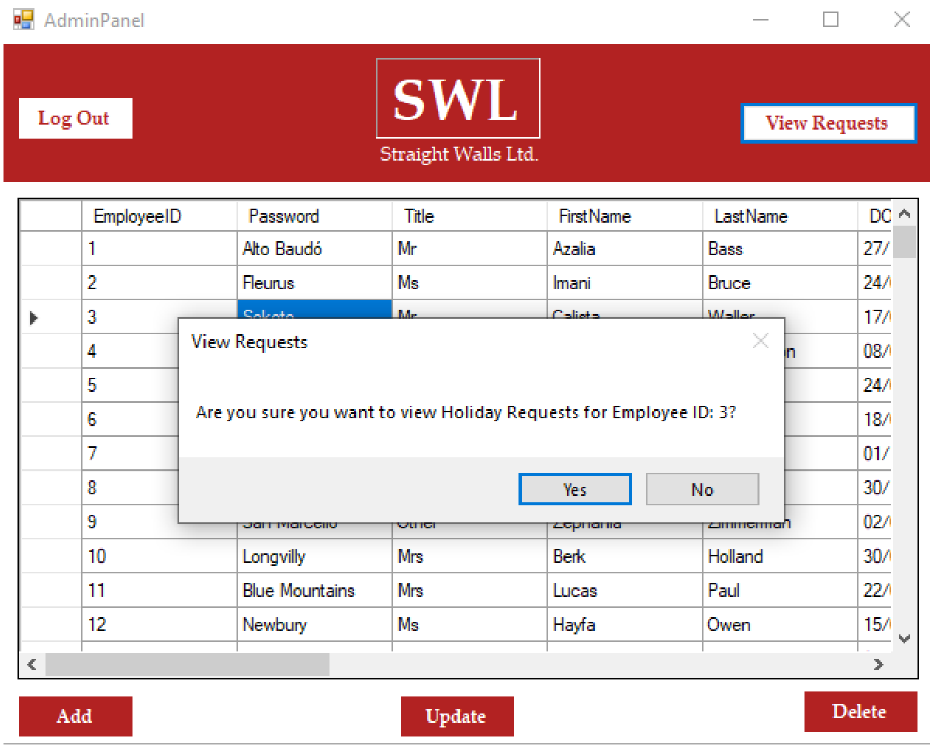
Clicking the ‘Yes’ button will take the user to the ‘HolidayForm’ where the specific employee’s requests and bookings will be seen:



HolidayID	StartDate	EndDate	RequestID	EmployeeID	Employ
7	28/03/2020	08/04/2020	2	3	FC.Emp

View A List Of All Holiday Bookings And Filter Them By Employee:

The holiday bookings are already filtered by employee due to the validation added in the screenshots above. The ‘HolidayForm’ will only show a single employee’s information at a time:

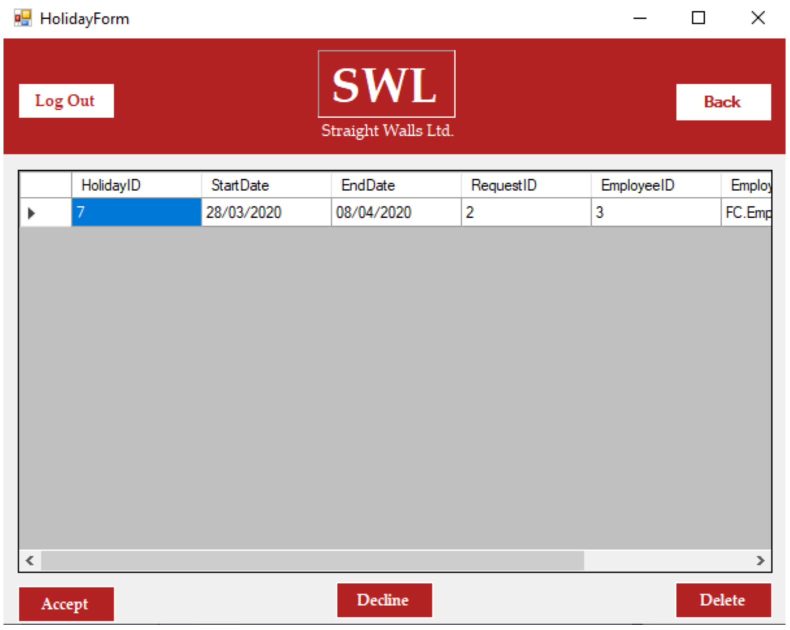


Are you sure you want to view Holiday Requests for Employee ID: 3?

EmployeeID	Password	Title	FirstName	LastName	DC
1	Alto Baudó	Mr	Azalia	Bass	27/
2	Fleurus	Ms	Imani	Bruce	24/
3	Sokoto	Mr	Calista	Waller	17/
4				n	08/
5					24/
6					18/
7					01/
8					30/
9	San Marcello	Other	Zephania	Zimmerman	02/
10	Longvilly	Mrs	Berk	Holland	30/
11	Blue Mountains	Mrs	Lucas	Paul	22/
12	Newbury	Ms	Hayfa	Owen	15/

Accept/Reject a Request:

The ‘HolidayForm’ has three buttons added to the bottom named ‘Accept’, ‘Decline’ and Delete. It is to be assumed by the employee that their request is still pending if it has not been accepted/declined.

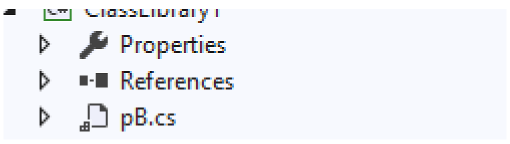


HolidayID	StartDate	EndDate	RequestID	EmployeeID	Employ
7	28/03/2020	08/04/2020	2	3	FC.Emp

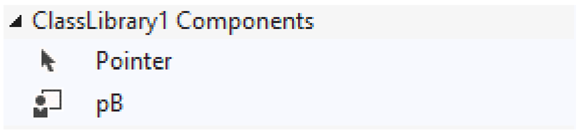
Functionality B:

Number Component: Phone Textbox:

This functionality required us to implement a phone textbox component into the group admin application that displayed the numbers in red if they surpassed 11. A new text box named ‘pB’ short for phoneBox was created and built, inheriting from ‘ClassLibrary1’.

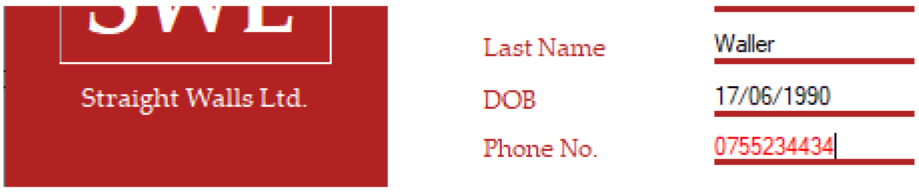


The text box was then dragged and dropped from the ‘Toolbox’ menu and placed in the ‘AddForm’ and ‘updateForm’ forms. The old hardcoded text box was then removed and replaced.



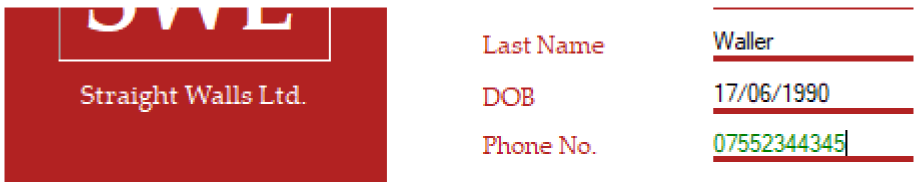
Shown below is the added validation for the Phone Textbox. The numbers in the textbox will turn red if they precede or exceed the number 11.

10 numbers:



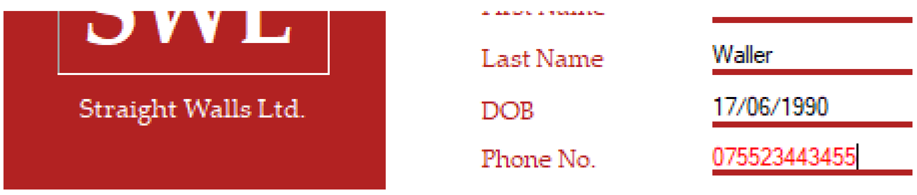
Last Name	Waller
DOB	17/06/1990
Phone No.	0755234434

11 numbers (correct input):



Last Name	Waller
DOB	17/06/1990
Phone No.	07552344345

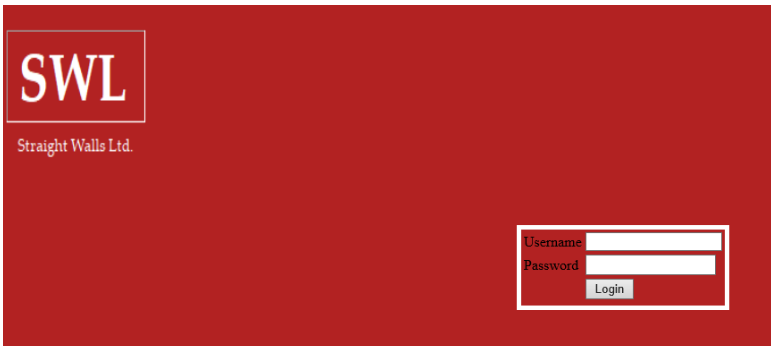
12 numbers:



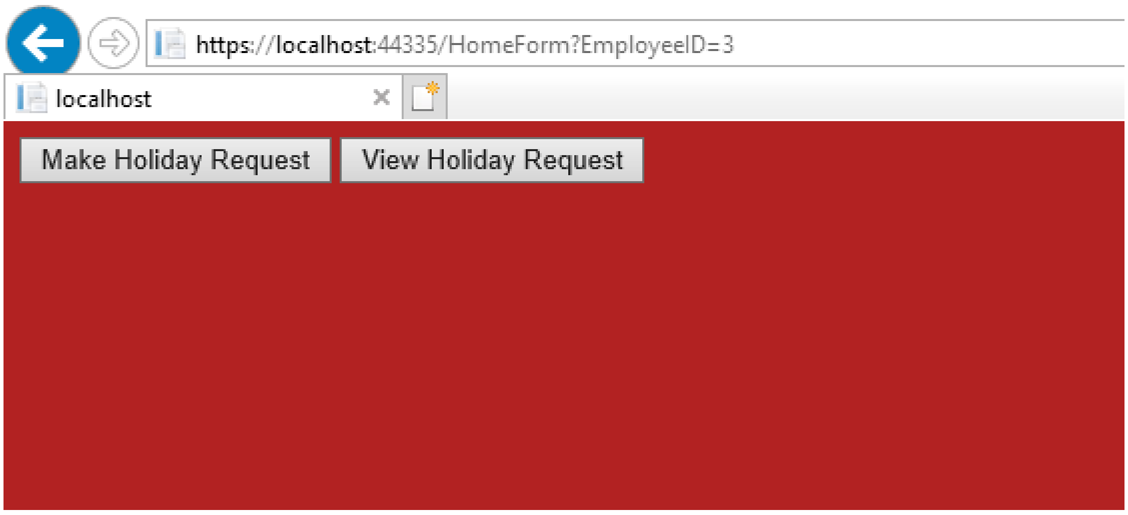
Last Name	Waller
DOB	17/06/1990
Phone No.	075523443455

ASP.NET Employee Login Website:

Shown below is the login page for the ASP.NET web application:

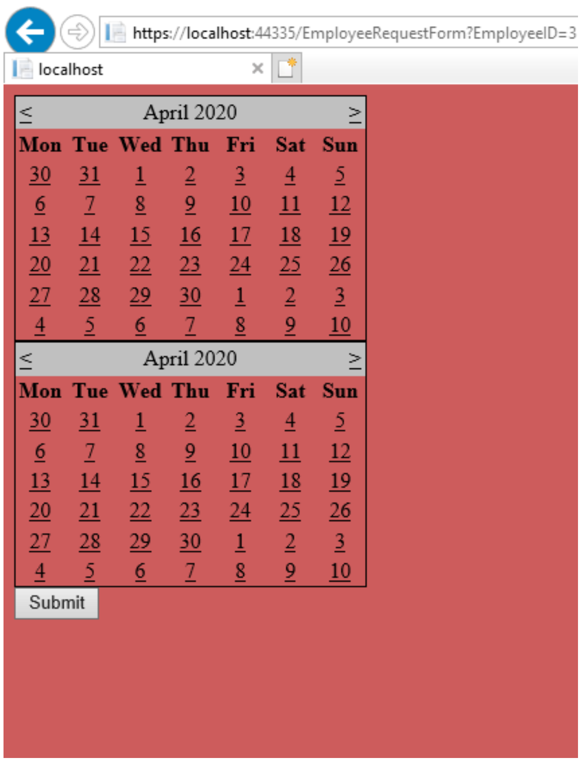


Signing in successfully will bring the user to the home page, where they will be prompted to either submit or view any pending requests:



Submit A Holiday Request:

Clicking the ‘Make Holiday Request’ button will take the user to the ‘EmployeeRequestForm’, where the employee will find 2 calendars to select their start and end date for their request. Clicking the ‘submit’ button will send the request into the database.


https://localhost:44335/EmployeeRequestForm?EmployeeID=3

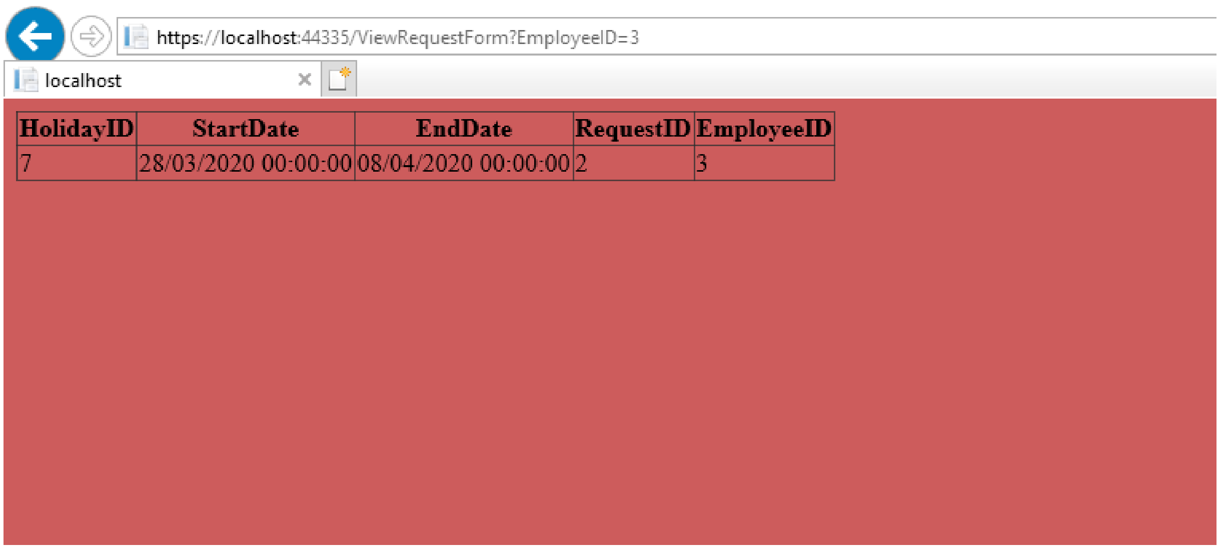
localhost

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

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

View A List Of Existing Holiday Requests, Which Also Shows Whether They Were Approved/Rejected:

Clicking the ‘View Holiday Requests’ button will take the user to the ‘ViewRequestForm’, where they will be able to view any holiday requests they have made. The request ID for the employee shown below is 2, meaning that it is still pending approval.


https://localhost:44335/ViewRequestForm?EmployeeID=3

localhost

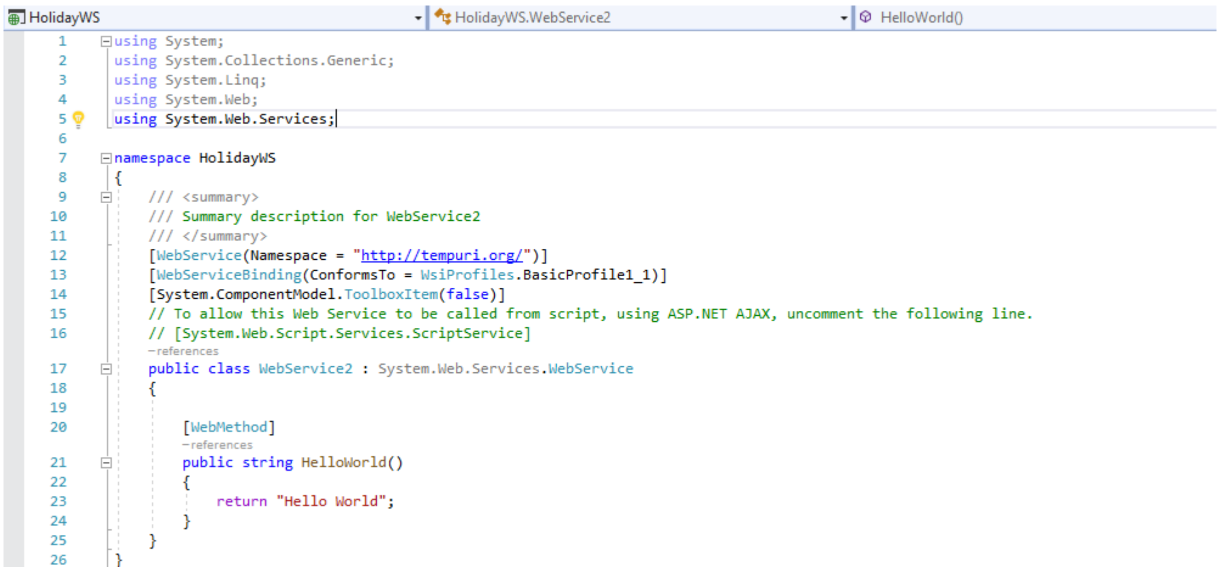
HolidayID	StartDate	EndDate	RequestID	EmployeeID
7	28/03/2020 00:00:00	08/04/2020 00:00:00	2	3

Functionality D:

Web Service:

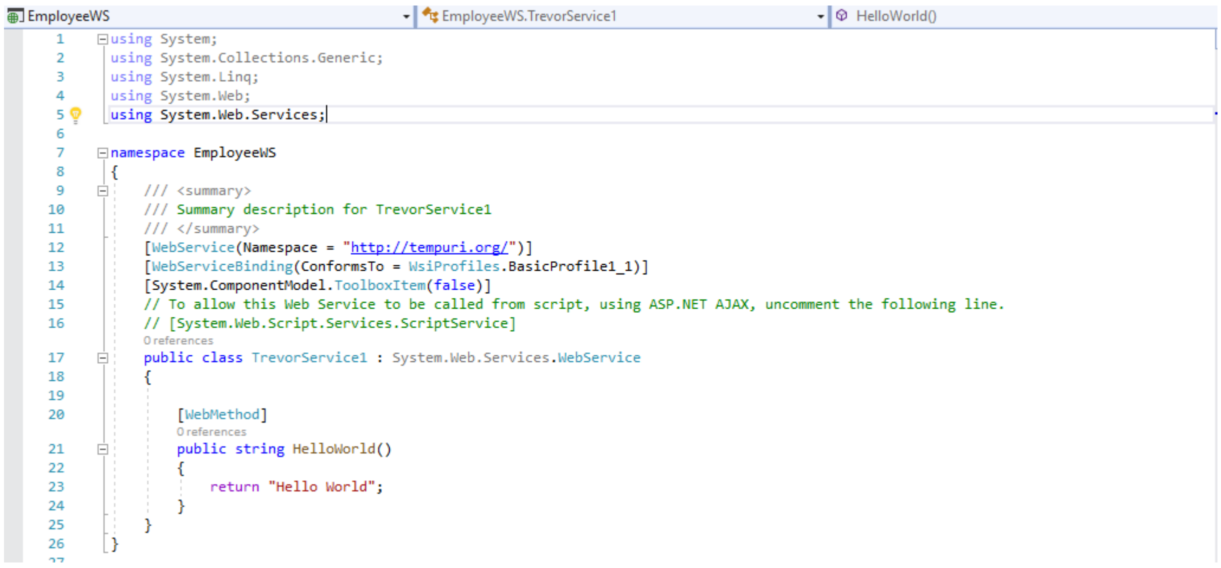
The web service was tasked to us in order to act as a middle ground for handling all communication between the client app and the database. Unfortunately, I was only able to have limited functionality regarding the web service as it was interfering with my main application and had to be moved into a different folder to ensure the rest of the application worked. I will include the files for the web service in a different zip folder.

Holiday Web Service:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

namespace HolidayWS
{
    /// <summary>
    /// Summary description for WebService2
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
    // [System.Web.Script.Services.ScriptService]
    -references
    public class WebService2 : System.Web.Services.WebService
    {

        [WebMethod]
        -references
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
}
```

Employee Web Service:



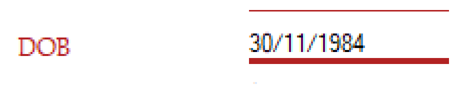
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

namespace EmployeeWS
{
    /// <summary>
    /// Summary description for TrevorService1
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    [System.ComponentModel.ToolboxItem(false)]
    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
    // [System.Web.Script.Services.ScriptService]
    0 references
    public class TrevorService1 : System.Web.Services.WebService
    {

        [WebMethod]
        0 references
        public string HelloWorld()
        {
            return "Hello World";
        }
    }
}
```

Functionality F:

Visualization Component:

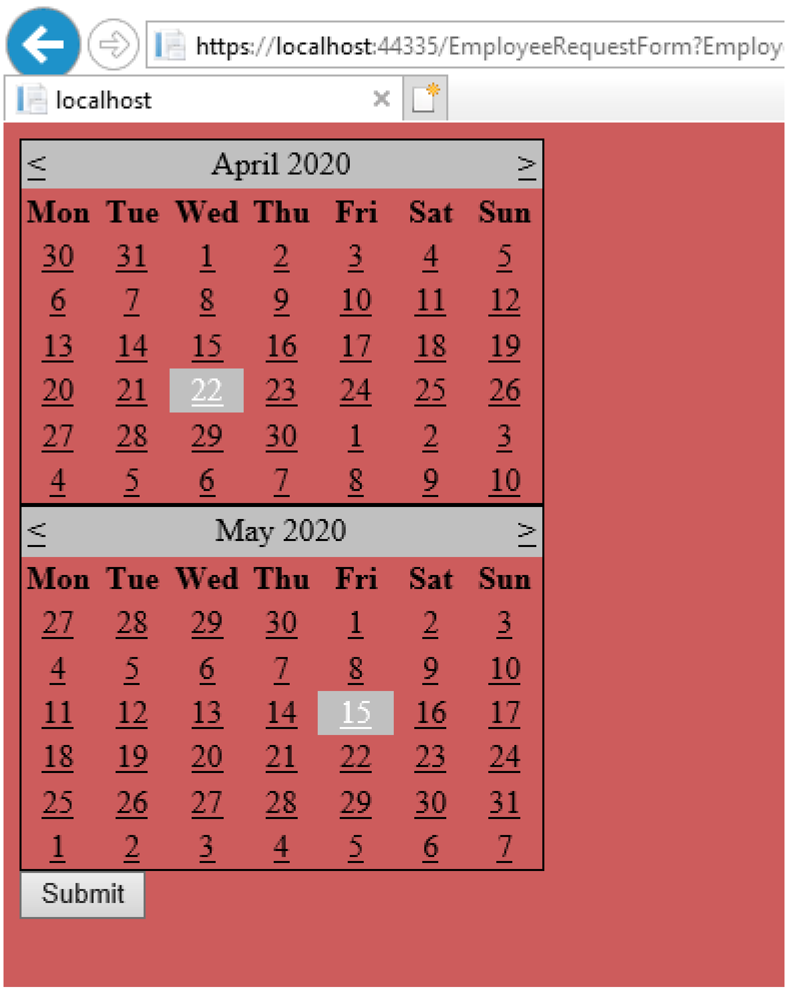
It is possible for the user to see a visual depiction of which days they want to book leave on the ASP.NET application, an alternative to other systems implemented elsewhere in the application, such as the admin form, where the date of birth has to been entered instead of select feature on the calendar. However, this functionality has not been fully implemented as a component, so it can only count for partial marks.

Admin App:



DOB 30/11/1984

Web App:



https://localhost:44335/EmployeeRequestForm?Employ

localhost

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

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

Algorithm:

Functionality A:

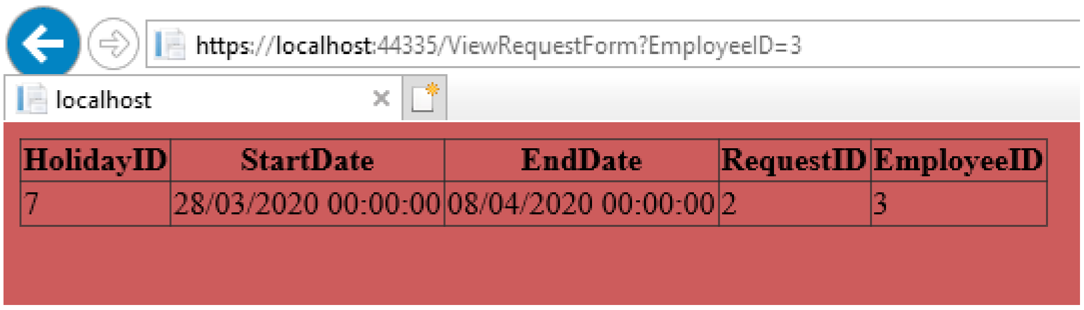
View A List of Outstanding Holiday Requests:

This was done by writing a query selecting the holiday requests from the database where the employee ID matched that one being parsed from the previous page, eliminating the need for hard coding. I was then able to display this data on a data grip view on the website.

Pseudo Code Example:

```
currentUserID = Request.QueryString["EmployeeID"];
EmployeeID = Int32.Parse(currentUserID);

var holidayquery = (from x in context.Holidays where x.EmployeeID == EmployeeID select x).ToList();
GridView1.DataSource = holidayquery;
GridView1.DataBind();
```

Diagram:



https://localhost:44335/ViewRequestForm?EmployeeID=3

localhost

HolidayID	StartDate	EndDate	RequestID	EmployeeID
7	28/03/2020 00:00:00	08/04/2020 00:00:00	2	3

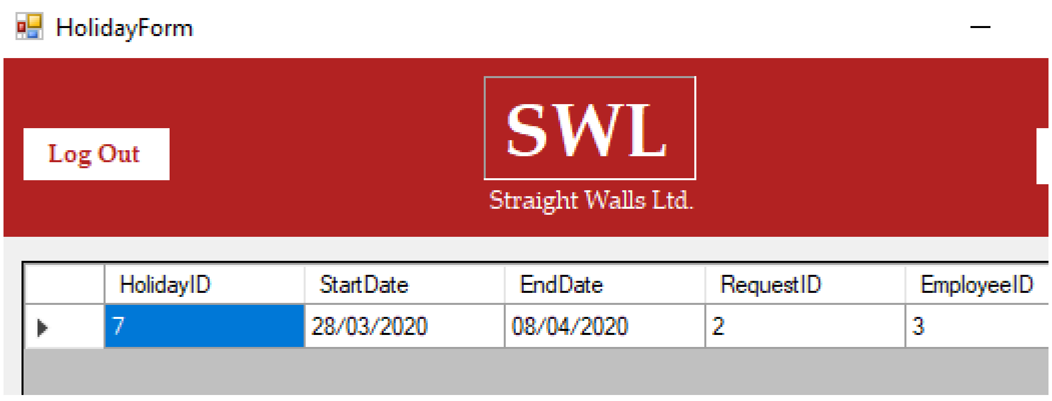
View A List of All Holiday Bookings and Filter Them By Employee:

This was done by extending the functionality given in the admin form to include database selection from a different table than before, this time, selecting from the 'Holiday' table as in the previous example. The where statement was also used to make sure that only one EmployeeID was selected.

Code Example:

```
private void HolidayForm_Load(object sender, EventArgs e)
{
    var q = from p in context.GetTable<Holiday>()
            where p.EmployeeID == employeeID
            select p;

    dataGridView2.DataSource = q;
}
```

Diagram:


HolidayForm

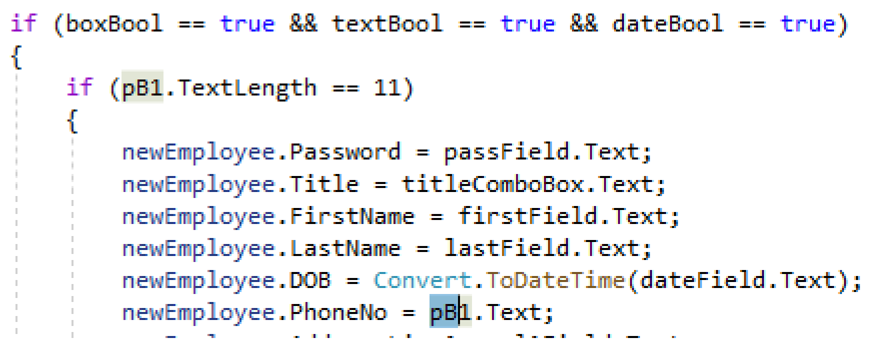
	HolidayID	StartDate	EndDate	RequestID	EmployeeID
▶	7	28/03/2020	08/04/2020	2	3

Functionality B:

Number Component: Phone TextBox:

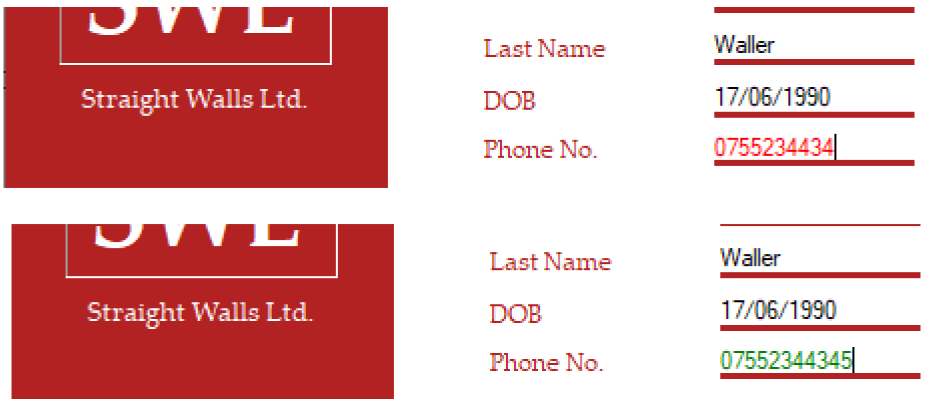
This was done by creating a new phone box class that the phone boxes on the add and update forms would inherit from. Anything other than the desired input would be turned into red text.

Code Example:



```
if (boxBool == true && textBool == true && dateBool == true)
{
    if (pB1.TextLength == 11)
    {
        newEmployee.Password = passField.Text;
        newEmployee.Title = titleComboBox.Text;
        newEmployee.FirstName = firstField.Text;
        newEmployee.LastName = lastField.Text;
        newEmployee.DOB = Convert.ToDateTime(dateField.Text);
        newEmployee.PhoneNo = pB1.Text;
```

Diagram:



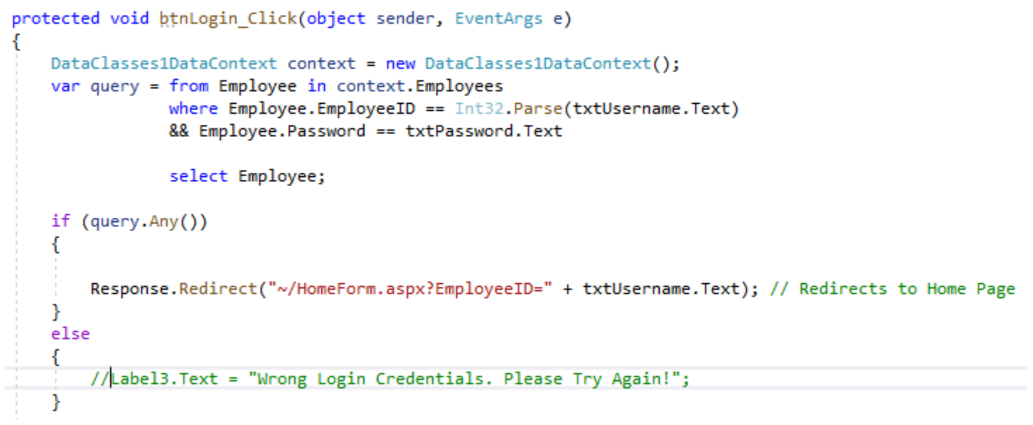
Last Name	Waller
DOB	17/06/1990
Phone No.	0755234434

Last Name	Waller
DOB	17/06/1990
Phone No.	07552344345

ASP.NET Login

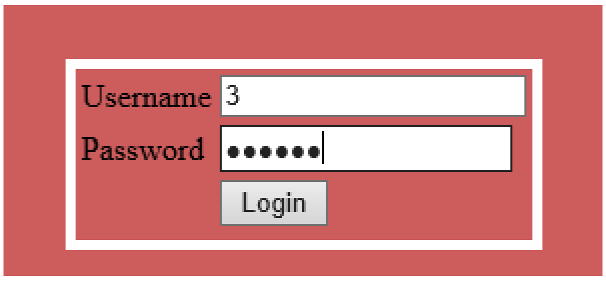
This was done by making sure to select the EmployeeID and password from the database that matches the one typed into the login form. A query was written to do this and on successful logon after the database check, the page redirects to the home page. The employee ID is also parsed onto the next page and is also visible in the url to ensure that the process has been executed correctly.

Code Example:



```
protected void btnLogin_Click(object sender, EventArgs e)
{
    DataClasses1DataContext context = new DataClasses1DataContext();
    var query = from Employee in context.Employees
                where Employee.EmployeeID == Int32.Parse(txtUsername.Text)
                && Employee.Password == txtPassword.Text

                select Employee;

    if (query.Any())
    {

        Response.Redirect("~/HomeForm.aspx?EmployeeID=" + txtUsername.Text); // Redirects to Home Page
    }
    else
    {
        //Label3.Text = "Wrong Login Credentials. Please Try Again!";
    }
```

Diagram:



Username	3
Password	●●●●●●

Submit A Holiday Request:

This was done by parsing the EmployeeID to the home page, then the make holiday request form. The employee chooses a date from each calendar and the start and end date for the request, which are saved and stored in the holiday table once the submit button is clicked. The DateTime varaiable can be stored as a string so it makes database integration much easier.

Code Example:

```
protected void Button1_Click(object sender, EventArgs e)
{
   selectedStartDate = Calendar1.SelectedDate;
   selectedEndDate = Calendar2.SelectedDate;

    if (Calendar1.SelectedDate < Calendar2.SelectedDate)
    {
        Holiday newEmployeeRequest = new Holiday();

        newEmployeeRequest.StartDate = selectedStartDate;
        newEmployeeRequest.EndDate = selectedEndDate;
        newEmployeeRequest.RequestID = 2;
        newEmployeeRequest.EmployeeID = Int32.Parse(currentUserID);

        context.Holidays.InsertOnSubmit(newEmployeeRequest);
        context.SubmitChanges();

        ScriptManager.RegisterClientScriptBlock(this, this.GetType(), "alertMessage", "alert('Request Submitted')", true);
```

Functionality F:

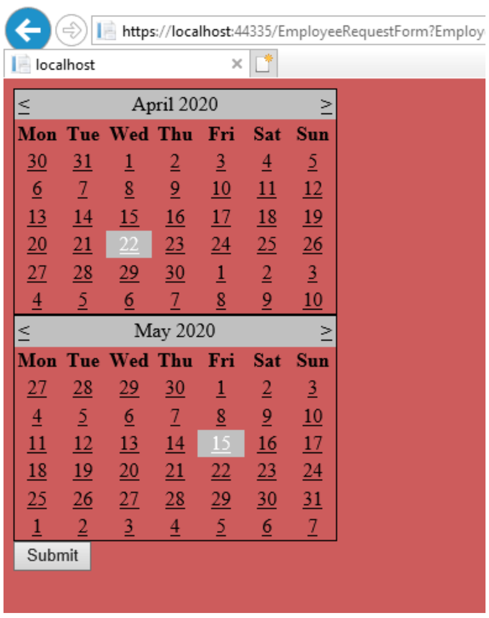
Calendar Visualization:

The drag and drop toolbox calendar allows for a visual representation of selected dates.

Code Example:

```
<form id="form1" runat="server">
    <div>
        <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
    </div>

    <div>
        <asp:Calendar ID="Calendar2" runat="server"></asp:Calendar>
    </div>
```

Diagram:


https://localhost:44335/EmployeeRequestForm?Employ

localhost

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

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

Evaluation:

Functionalities A and B were somewhat easy to implement given the base that had already been set during the group part of the coursework. The program structure remained the same and I was able to cope with what I was being asked to do. I had severe difficulties implementing most of the other features though, as I faced a bit of coursework clash due to the deadline extension, and I had progressed much more on this assignment than the other, meaning that it took a back seat. Time management was definitely an issue. Detailed below is a critical evaluation of my program(s).

Positives:

The remainder of the individual admin application was very well validated, as it involved a lot of code reuse from the group application. Every single button, text field, and grid view were all validated and ran as they were supposed to, ensuring that there would be no confusion had the system run into any problems. The GUI was also superb and continued the pattern and colour-scheme set by the earlier parts of application, allowing for an easy user experience. The database was also well-designed, so it allowed for any changeable factors in the future to be easy to implement, such as an addition of roles or a completely new department. Certain features of the web application were also good, such as validation to not allow any data to be entered into the database without matching the proper criteria involved. Validation such as making sure that the selected end date for the request was later than the selected start date helped to ensure that the employee request dates made sense and did not cause any errors in the database.

Negatives:

There are a lot of negatives regarding my program/coursework starting with the fact that I was unable to implement some of the features asked for in the specification, leaving a lot to be asked for. My inability to fully develop the web service means that my system remains exposed due to the lack of a third-party web server moderation. This would have made my applications and database a lot more secure, and it is something that I plan on learning for my Final Year Project. Another disappointing feature is the inadequate GUI of the web application, especially compared to that of the admin desktop application, which was very good. I created my ASP.NET web application without the pre-generated UI features that a default apsx web forms page does, and I ran out of time to go back and make the necessary changes to make it look much nicer. Given more time, the web application would have looked very similar to the admin one, but I had to settle with what I had and improvise. Some of my code re-usage could have also been done using methods and other types of inheritance, but I wanted to make sure everything worked correctly, and not to overcomplicate things. I sincerely do believe that given more time and without the coursework clash, I would have been able to produce a much better piece of coursework. However, I am happy with the skills that I’ve learned from Terry and Markus, especially with ASP.NET and SQL Server database management, and I will definitely use them to complete my Final Year Project.