**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: 3](#_Toc36843752)

[Team Database Design: 3](#_Toc36843753)

[Normalization of design: 3](#_Toc36843754)

[Entity/Model Classes: 4](#_Toc36843755)

[Class Diagrams: 5](#_Toc36843756)

[Architecture Diagram: 6](#_Toc36843757)

[Individual Report Section: 12](#_Toc36843758)

[PART B: Individual Technical Documentation: 13](#_Toc36843759)

[Annotated Screenshots: 13](#_Toc36843760)

[Functionality A: 13](#_Toc36843761)

[View a list of outstanding holiday requests: 13](#_Toc36843762)

[View A List Of All Holiday Bookings And Filter Them By Employee: 15](#_Toc36843763)

[Accept/Reject a Request: 15](#_Toc36843764)

[Functionality B: 16](#_Toc36843765)

[Number Component: Phone Textbox: 16](#_Toc36843766)

[ASP.NET Employee Login Website: 17](#_Toc36843767)

[Submit A Holiday Request: 17](#_Toc36843768)

[Functionality D: 19](#_Toc36843769)

[Web Service: 19](#_Toc36843770)

[Functionality F: 20](#_Toc36843771)

[Visualization Component: 20](#_Toc36843772)

[Algorithm: 21](#_Toc36843773)

[Functionality A: 21](#_Toc36843774)

[View A List of Outstanding Holiday Requests: 21](#_Toc36843775)

[View A List of All Holiday Bookings and Filter Them By Employee: 21](#_Toc36843776)

[Functionality B: 22](#_Toc36843777)

[Number Component: Phone TextBox: 22](#_Toc36843778)

[ASP.NET Login 23](#_Toc36843779)

[Submit A Holiday Request: 23](#_Toc36843780)

[Functionality F: 24](#_Toc36843781)

[Calendar Visualization: 24](#_Toc36843782)

[Evaluation: 25](#_Toc36843783)

# PART A: Group Documentation:

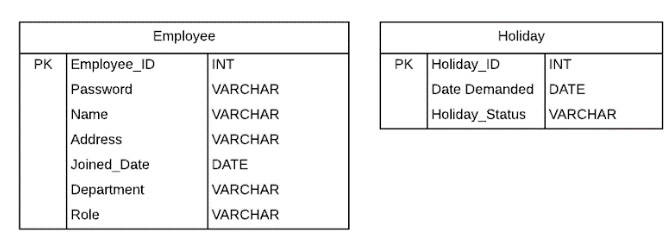
## Team Database Design:

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:

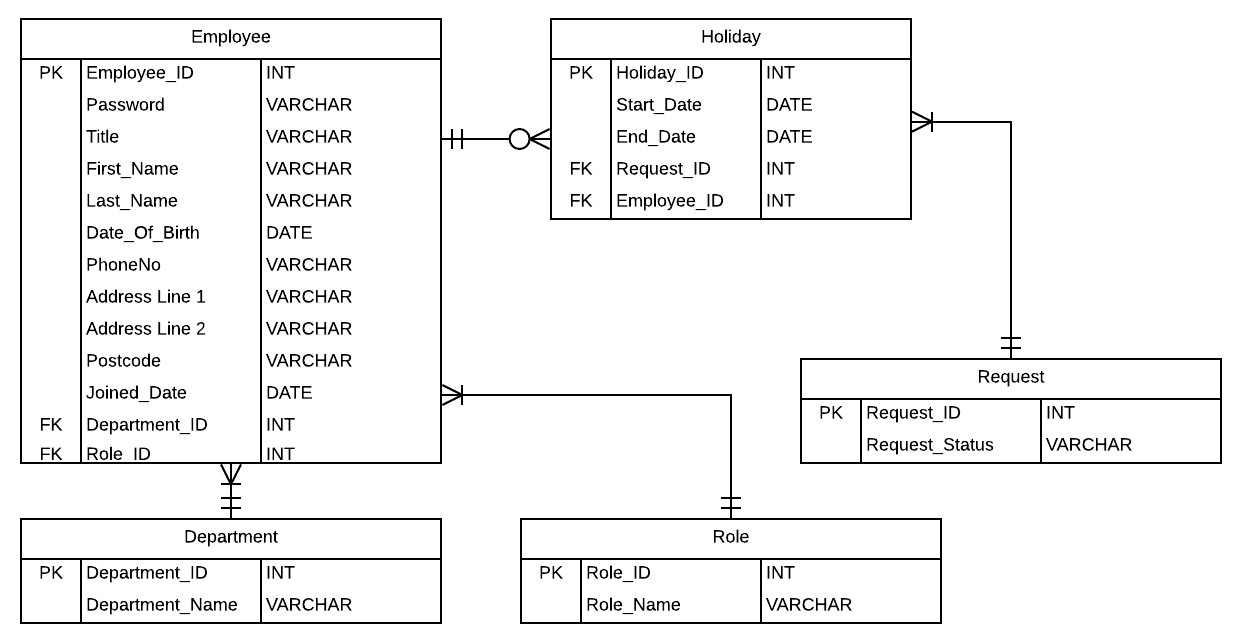
Our original database was flawed, redundant and relied too heavily on large tables. The lack of normalization was evident as our tables lacked the appropriate foreign keys and relationships between them. However, after consultation with the lecturer, we were able to decompose the fields in each table 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:

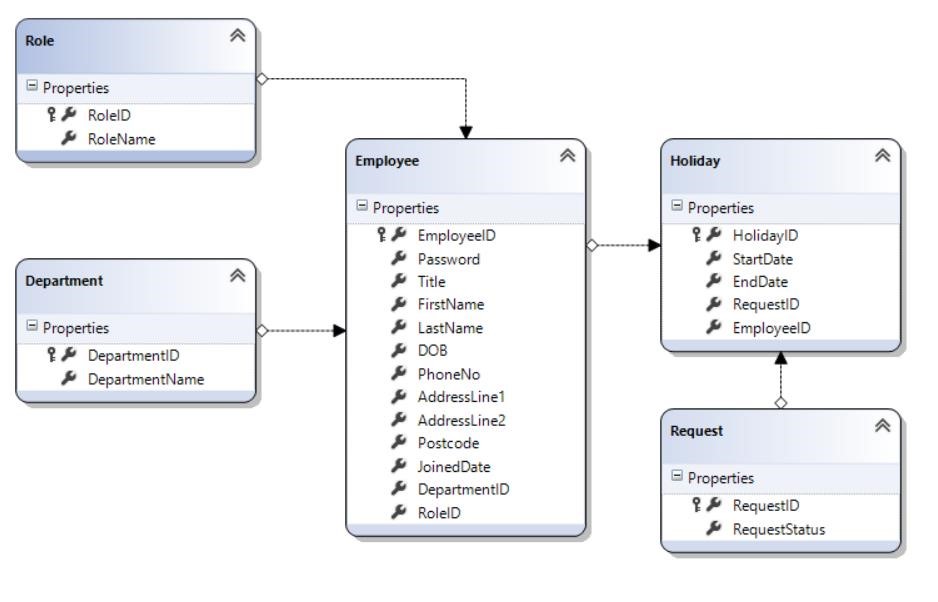


Final Entity Relationship Diagram (ERD) with Normalization:

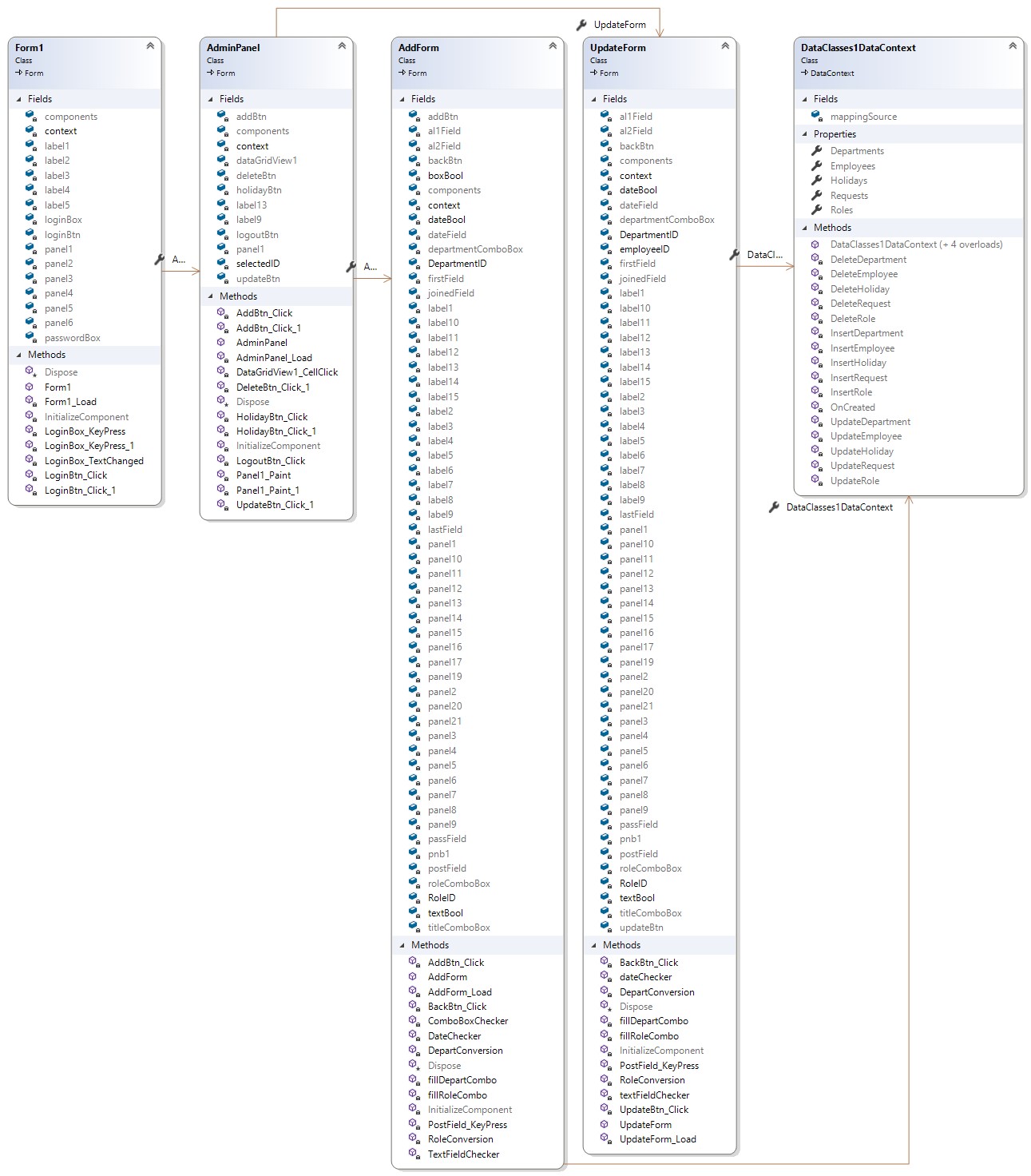
After normalising the graph, we split data fields to separate columns and tables so that each piece of data can be represented with corresponding data. Also, we added extra fields to capture the right data with attention to detail. For example: Address Line 1, Address Line 2, Postcode etc.



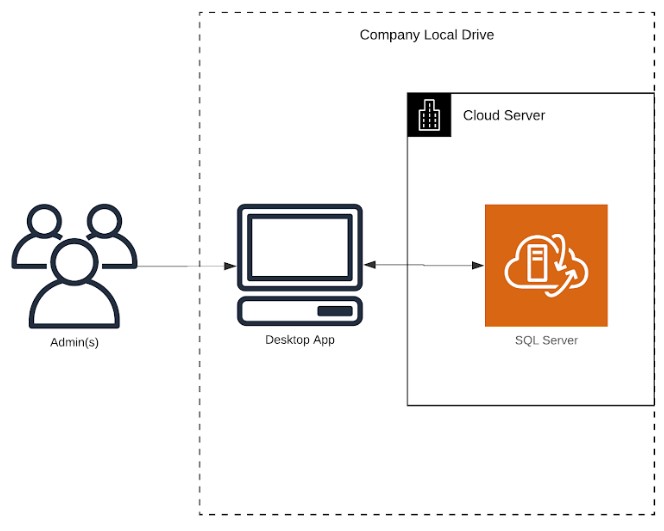
## Entity/Model Classes:



## Class Diagrams:



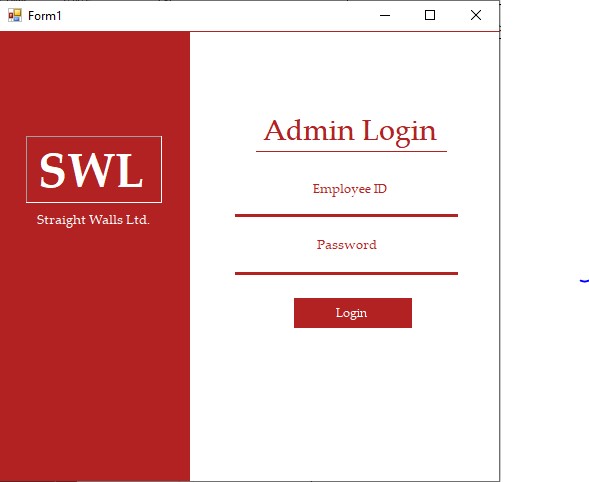
## Architecture Diagram:



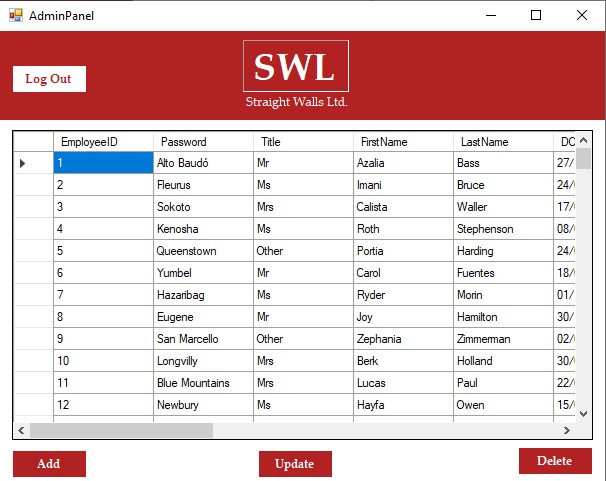
Admin Desktop Application (with screenshots):

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

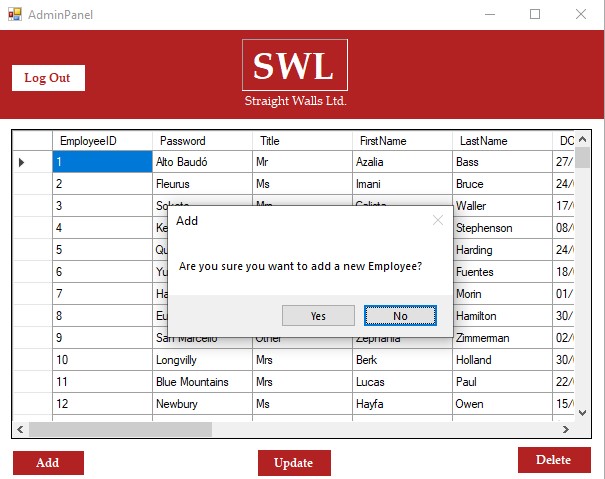
Admin Login:



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



Add Employee Validation:



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



10 Numbers:



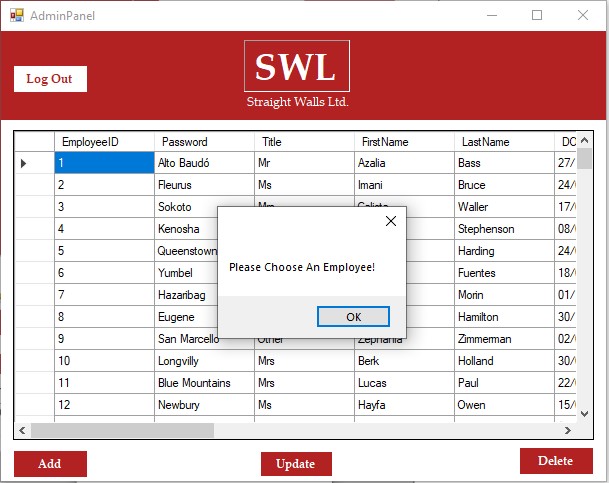
12 numbers:



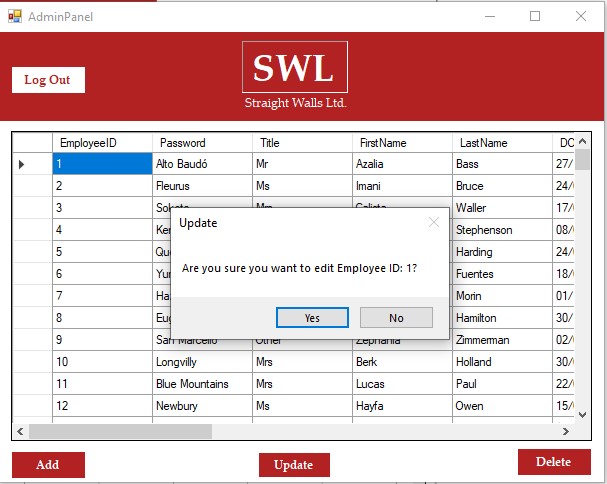
11 numbers:



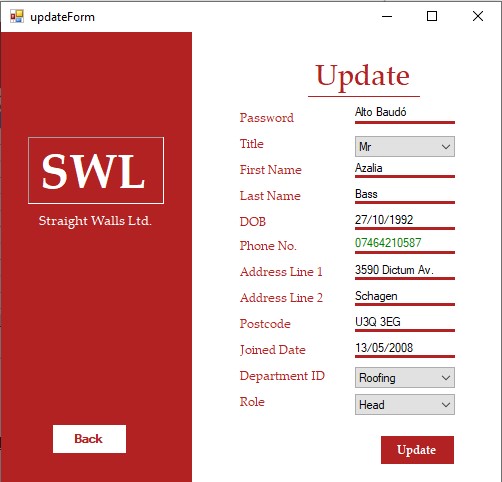
Update/Delete Employee Validation:



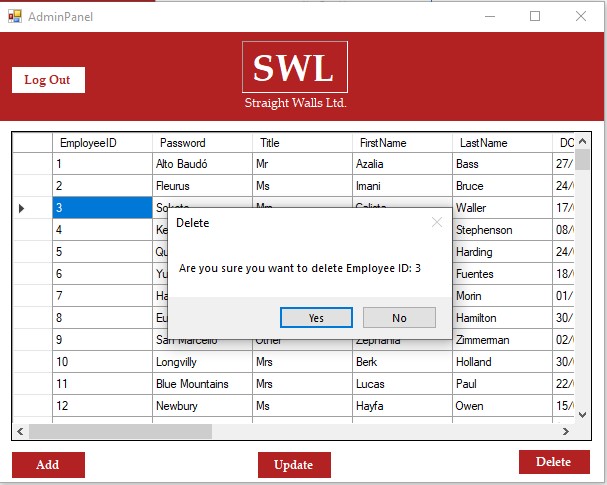
Update Employee Validation:



Update Employee Page:



Delete Employee Validation When Button Is Clicked:



## 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:

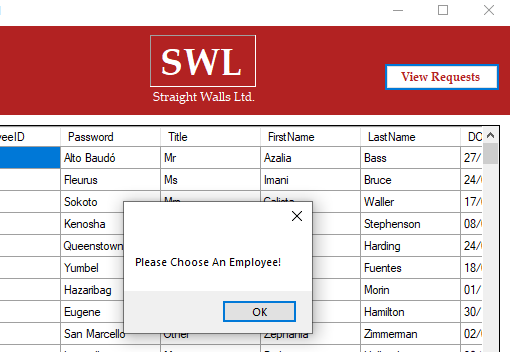
### Functionality A:

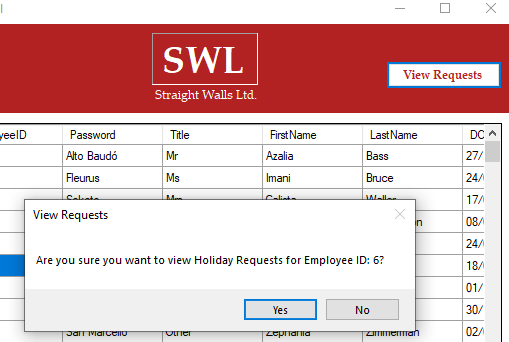
### View a list of outstanding holiday requests:

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.

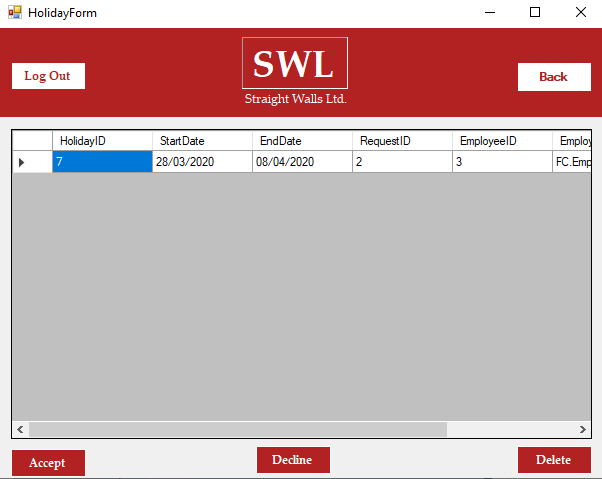
****

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

****

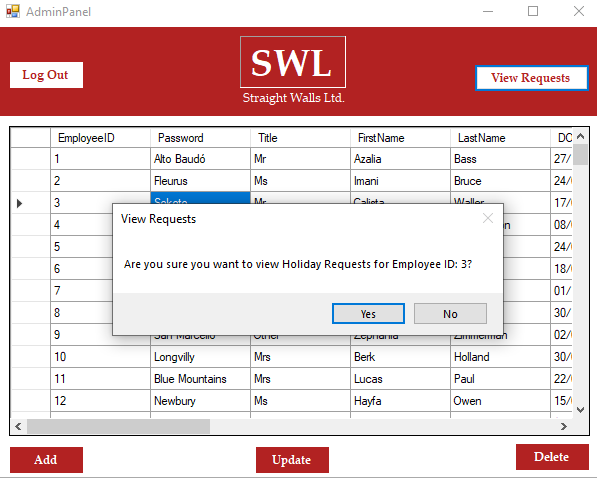
****

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

****

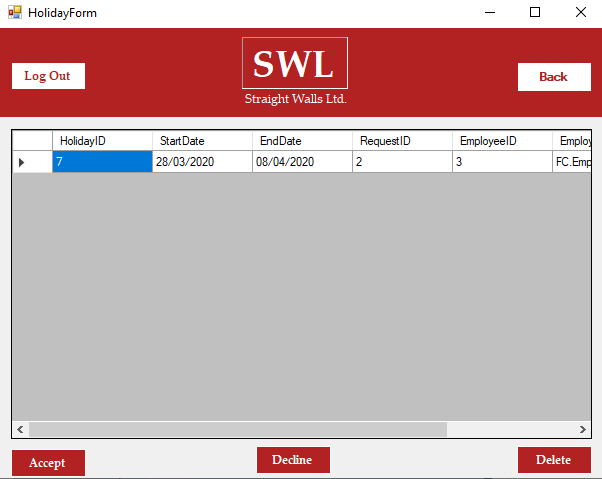
### View A List Of All Holiday Bookings And Filter Them By Employee:

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:



### Accept/Reject a Request:

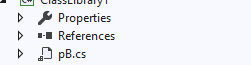
The ‘HolidayForm’ has three buttons added to the bottom named ‘Accept’, ‘Decline’ and Delete. It is to be assumed by the employee that their request is still pending if it has not been accepted/declined.

****

### 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’.

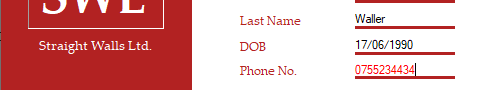


The text box was then dragged and dropped from the ‘Toolbox’ menu and placed in the ‘AddForm’ and ‘updateForm’ forms. The old hardcoded text box was then removed and replaced.

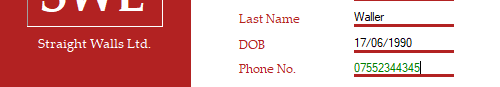


Shown below is the added validation for the Phone Textbox. The numbers in the textbox will turn red if they precede or exceed the number 11.

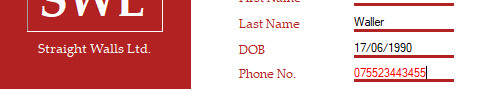
10 numbers:



11 numbers (correct input):



12 numbers:

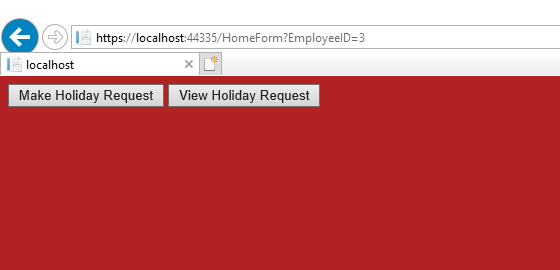


### ASP.NET Employee Login Website:

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

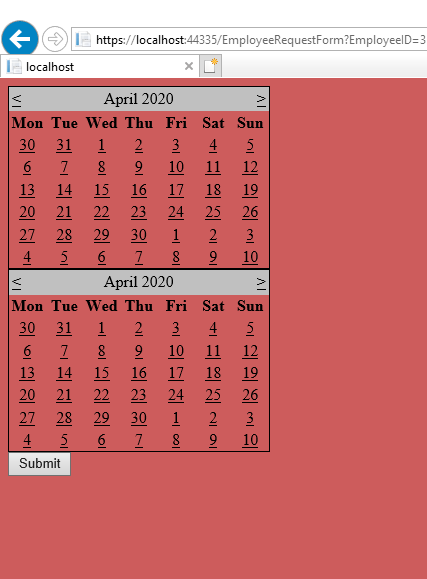


Signing in successfully will bring the user to the home page, where they will be prompted to either submit or view any pending requests:



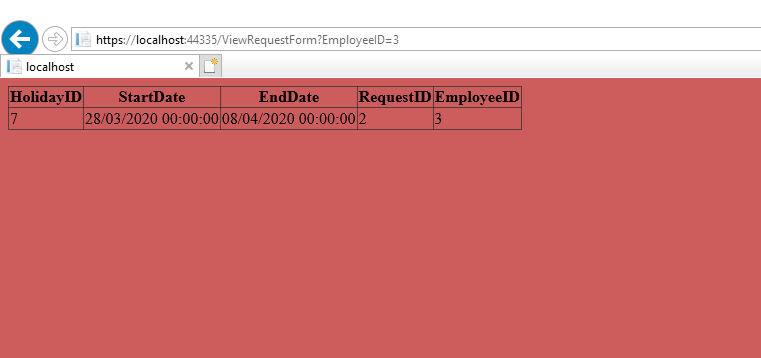
### Submit A Holiday Request:

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.



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.

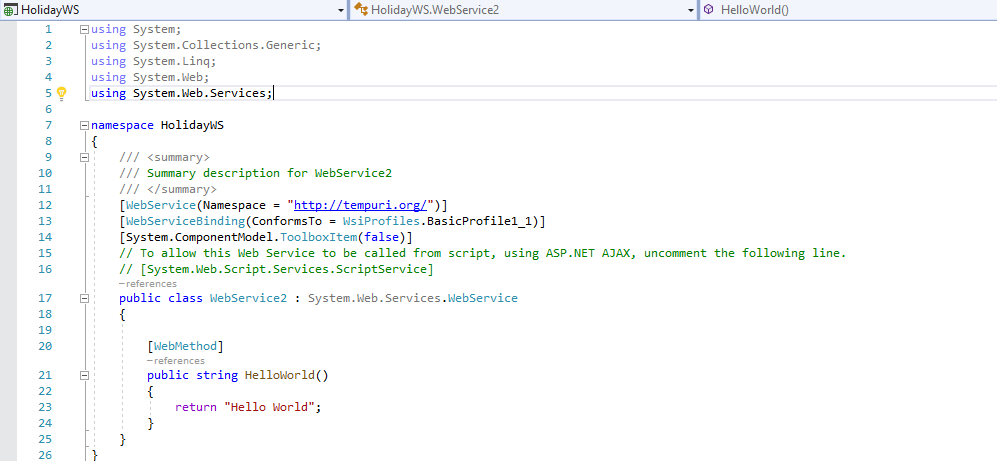


### Functionality D:

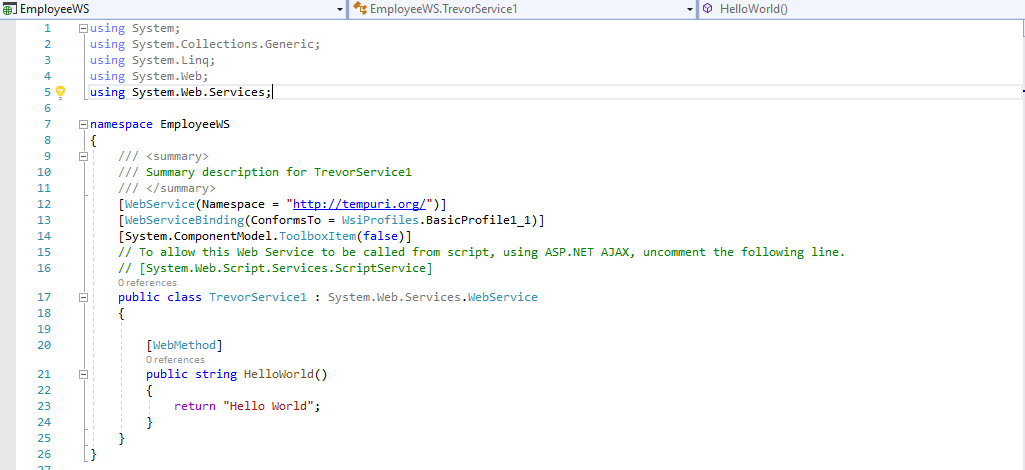
### Web Service:

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:



Employee Web Service:



### Functionality F:

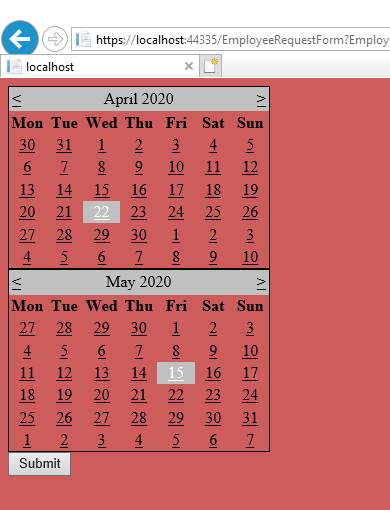
### Visualization Component:

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:



Web App:



## 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:

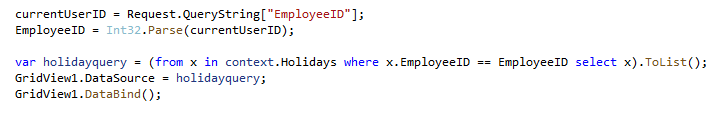
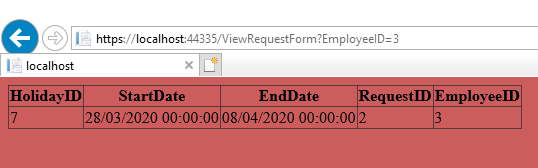


Diagram:



### 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:

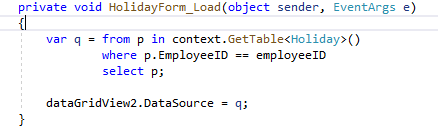
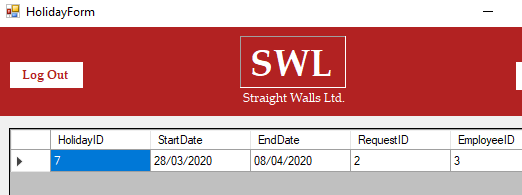


Diagram:



### 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:

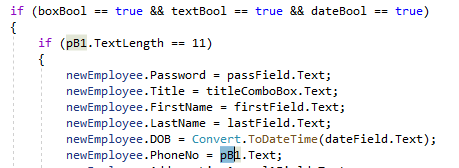
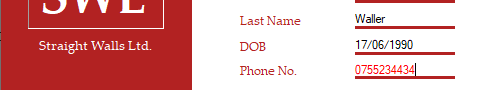
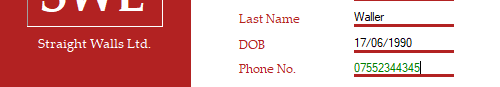


Diagram:





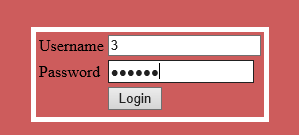
### ASP.NET Login

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:



Diagram:



### 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:



### Functionality F:

### Calendar Visualization:

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

Code Example:

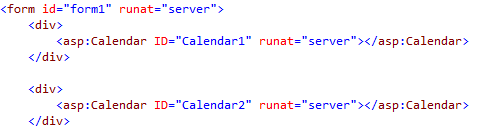
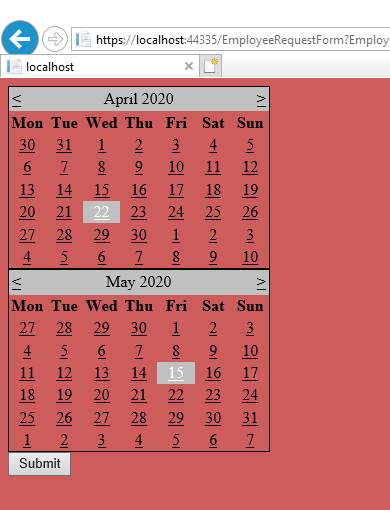


Diagram:



## 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.