Switch Up

Raheem Hussain

14 April 2019

A project report submitted in partial fulfilment for the degree of

**Bachelor of Science in Software Engineering**

**School of Physical Sciences and Computing**

**University of Central Lancashire**

Abstract

Having the ability to swap shift with someone could happen, however currently it is happening unprofessionally.

I want to be able to take the paper version and put in all online. Giving the employee to choice to swap shift online so other employees can see.

Employees can accept or decline and they can only swap shifts with employees that have the same amount of hours as you do

Attestation

I understand the nature of plagiarism, and I am aware of the University’s

policy on this.

I certify that this document reports original work by me during my University project.

**Signature**  **Date**

Acknowledgements

Li Guo – Supervisor Help throughout the project and weekly catch up meetings

Jonathan Edwards – Helping with the SQL Statements

Tesco – Helping with testing

Nick Mitchell – Helping with mental health issues

Table of Contents

[Abstract i](#_Toc6173969)

[Attestation ii](#_Toc6173970)

[Acknowledgements iii](#_Toc6173971)

[Table of Contents iv](#_Toc6173972)

[1 Introduction 8](#_Toc6173973)

[1.1 Background and Context 8](#_Toc6173974)

[1.2 Scope and Objectives 9](#_Toc6173975)

[1.3 Achievements 9](#_Toc6173976)

[1.4 Overview of Report 9](#_Toc6173977)

[2 Literature Review 11](#_Toc6173978)

[2.1 Introduction 11](#_Toc6173979)

[2.2 MVC Model 11](#_Toc6173980)

[2.2.1 Model 11](#_Toc6173981)

[2.2.2 View 12](#_Toc6173982)

[2.2.3 Controller 12](#_Toc6173983)

[2.2.4 View to Model Communication 13](#_Toc6173984)

[2.2.5 Model to View Communication 13](#_Toc6173985)

[2.3 ASP.Net 14](#_Toc6173986)

[2.3.1 Background 14](#_Toc6173987)

[2.3.2 Why use ASP.NET 14](#_Toc6173988)

[2.3.3 Visual Studio ASP.NET 14](#_Toc6173989)

[2.3.4 ASP.NET Vs ASP 15](#_Toc6173990)

[2.3.5 ASP.NET Web Forms 15](#_Toc6173991)

[2.4 Three-tier architecture 15](#_Toc6173992)

[2.4.1 Background 15](#_Toc6173993)

[2.4.2 Presentation Layer 16](#_Toc6173994)

[2.4.3 Application Layer 17](#_Toc6173995)

[2.4.4 Database Layer 17](#_Toc6173996)

[2.4.5 Pros of a three-tier architecture 17](#_Toc6173997)

[2.4.6 Cons of a Three-Tier Architecture 17](#_Toc6173998)

[2.5 Relational Database 18](#_Toc6173999)

[2.5.1 Background 18](#_Toc6174000)

[2.5.2 Advantages of using a relational database 18](#_Toc6174001)

[2.5.3 Disadvantages of using a relational database 19](#_Toc6174002)

[2.6 Commercial Factors 19](#_Toc6174003)

[2.7 Summary 20](#_Toc6174004)

[3 Project Planning 21](#_Toc6174005)

[3.1 Introduction 21](#_Toc6174006)

[3.2 Methodology 21](#_Toc6174007)

[3.3 Requirements 21](#_Toc6174008)

[3.3.1 Key Objectives 21](#_Toc6174009)

[3.3.2 User Stories 22](#_Toc6174010)

[3.4 Potential Solutions 24](#_Toc6174011)

[3.4.1 ASP.NET 24](#_Toc6174012)

[3.4.2 Web Forms 25](#_Toc6174013)

[3.4.3 MySql WorkBench 26](#_Toc6174014)

[3.5 Legal and Ethical Issues 26](#_Toc6174015)

[3.5.1 Ethical Issues 26](#_Toc6174016)

[3.6 Summary 27](#_Toc6174017)

[4 Design 28](#_Toc6174018)

[4.1 Introduction 28](#_Toc6174019)

[4.2 System Design 28](#_Toc6174020)

[4.2.1 Use Case Diagram 28](#_Toc6174021)

[4.2.2 Class Diagram 29](#_Toc6174022)

[4.2.3 Database Design 30](#_Toc6174023)

[4.3 User Interface Design 31](#_Toc6174024)

[4.3.1 Web Application Design 31](#_Toc6174025)

[4.4 Summary 32](#_Toc6174026)

[5 Implementation 33](#_Toc6174027)

[5.1 Introduction 33](#_Toc6174028)

[5.2 Login Page 33](#_Toc6174029)

[5.2.1 Login Page Functionality 33](#_Toc6174030)

[5.2.2 Login Page Design 34](#_Toc6174031)

[5.2.3 Complications 34](#_Toc6174032)

[5.3 Dashboard Page 34](#_Toc6174033)

[5.3.1 Dashboard Functionality 34](#_Toc6174034)

[5.3.2 Dashboard Design Implementation 35](#_Toc6174035)

[5.3.3 Complication with Dashboard 35](#_Toc6174036)

[5.4 Rota 36](#_Toc6174037)

[5.4.1 Rota Functionality 36](#_Toc6174038)

[5.4.2 Rota Design Implementation 36](#_Toc6174039)

[5.5 Shift Choice Page 37](#_Toc6174040)

[5.5.1 Shift Choice Functionality 37](#_Toc6174041)

[5.5.2 Shift Choices Design Implementation 37](#_Toc6174042)

[5.5.3 Shift Choices Complications 38](#_Toc6174043)

[5.6 Availability Page 38](#_Toc6174044)

[5.6.1 Availability Page Functionality 38](#_Toc6174045)

[5.6.2 Availability Page Design 38](#_Toc6174046)

[5.7 Request Page 39](#_Toc6174047)

[5.7.1 Request Page Functionality 39](#_Toc6174048)

[5.7.2 Requested Page Design 39](#_Toc6174049)

[5.8 Summary 39](#_Toc6174050)

[6 Test Strategy 40](#_Toc6174051)

[6.1 Introduction 40](#_Toc6174052)

[6.2 White Box Testing 40](#_Toc6174053)

[6.3 Black Box Testing 40](#_Toc6174054)

[6.4 Web Page Testing 41](#_Toc6174055)

[6.4.1 Login Page Test 41](#_Toc6174056)

[6.4.2 Dashboard Page Test 42](#_Toc6174057)

[6.4.3 Rota Page Test 43](#_Toc6174058)

[6.4.4 Shift Choices Test 43](#_Toc6174059)

[6.4.5 Request Page Test 44](#_Toc6174060)

[6.4.6 Availability Test 45](#_Toc6174061)

[6.5 Summary 45](#_Toc6174062)

[7 Evaluation, Conclusions and Future Work 46](#_Toc6174063)

[7.1 Project Objectives 46](#_Toc6174064)

[7.2 Self-Evaluation 46](#_Toc6174065)

[7.3 Project Evaluation 49](#_Toc6174066)

[7.4 Applicability of Findings to the Commercial World 50](#_Toc6174067)

[7.5 Conclusions 50](#_Toc6174068)

[7.6 Future Work 51](#_Toc6174069)

[References 53](#_Toc6174070)

[Appendix 1 – Project Proposal 56](#_Toc6174071)

[8 Computing Degree Project Proposal 57](#_Toc6174072)

[8.1 Current Modules (and previous modules if Computing or direct entrant) 57](#_Toc6174073)

[8.2 The Project Title 57](#_Toc6174074)

[8.3 Project Context 57](#_Toc6174075)

[8.4 Specific Objectives 57](#_Toc6174076)

[8.5 Potential Ethical or Legal Issues 58](#_Toc6174077)

[8.6 Resources 58](#_Toc6174078)

[8.7 Potential Commercial Considerations - Estimated costs and benefits 59](#_Toc6174079)

[8.8 Proposed Approach 59](#_Toc6174080)

[Appendix 2 – Technical Plan 60](#_Toc6174081)

[Appendix 3 – Title of Appendix 2](#_Toc6174082)

# Introduction

## Background and Context

There are still many small companies that still use paper rotas over online rotas. A study shows that over 73% of companies owners with fewer than 500 employees use paper rotas and are print off at least four times the rotas (Brandon, 2019). The reason behind so many print offs is the fact that the manager will need to sit down for at least a few hours to see which employees can work their shifts or to see if they are on holiday. The manager will also need to see whether any problem has occurred, for example, an employee could be sick for a period of time so, they would need to see other employees availability in order to cover the shift.

By still having a paper rota you would need to ensure that the employees have an up to date version of the rota and they would need to travel to work in order to see what days they are working. This is not very efficient for the employees that do not live in close proximity of the workplace, so they would need to go out of their way in order to know their shifts for the week. When I used to work for retail which was heavily based on a paper rota, I always need to make sure that I had an up to date copy of the rota with me at the beginning of the week. This became very stressful with other employees as sometimes they would not have an up to date copy of the rota which resulted in them not knowing their shifts for the week.

By having a paper rota each week over a long period of time could be bad on the business and could cost the company. The reason behind this claim is that if you are only using paper-based to record the rota and the hours this could lead to losing the track of the hours an employee has done (Fadilpašić, 2019) . It could also lead to not paying the employees correctly because you could lose the paper that holds that information or even misplace document if the payment to the employees is not correct and you only rely on paper this could happen more than once and it could lead to multiple employees being overpaid.

Currently, there are some employees that are swapping shifts, however, the way that they are going by this is not in a professional way. The way that some employees are swapping shift is either over text message or over Facebook. The best way to solve an issue like this could be to use emails as it is known as a professional platform for communication and the manager can also be informed. Email would be a great was it does not take a long time and it would increase colleagues interaction (Jackson and Hooff, 2012). This would be an alternative instead of using social media and text messages to swap shift, so by using emails the manager and shift leader can be included and know about the swap. This would also resolve the issue of some employee backing out last minute because something came up and if it was done over social media and text it’s not a full confirmation that they would cover the shift. Whereas, when you are using email its more professional and more than one person knows about it so they can not back down.

## Scope and Objectives

The objective of this project is to be able to swap shifts with another employee who has the same amount of hours as you do. There would be three users of the system the manager, the shift leader and the employee. The manager and shift leader will be able to see the employees available so if someone rings in sick they can see who is available to cover the shift. They would also be able to set over time, so instead of texting all staff all he would need to do is set a shift upon the system and the employee can accept the shift if they wanted overtime. The manager will also have the final say when it comes to permanently swapping shift between two employees. The employees will have the opportunity to either accept or decline a shift swap from another employee, once a swap has been accepted it would then be put onto the rota highlighted in red. If the shift is rejected then the rota will stay the same. The way the employees will be notified about a shift swap will be via an email that will be sent out to them. The employee will also be allowed to enter the amount of week they want that shift to be covered for.

## Achievements

The project achieved the objectives shown in section 1.2. My achievement would be to swap out a paper-based rota to an online rota, I am hoping that all employees can log on to the system to see which shifts they are doing that coming week. They should be able to log in on either a mobile device or a computer.

The rota will display all employees and they would be able to go to the shift swap page where it will show the employees with the same amount of hours as the employee. They will then have the option to request a swap with another employee and when the employee next logs in they can accept or decline the shift.

Manager and shift leader being able to see the availability the reason for this is so that if there is overtime available at the workplace the manager will be able to see who is available to do it.

## Overview of Report

Chapter one, an introduction to the report, outlining the background and the context giving a detailed explanation of the scope and objective.

Chapter two, a literature review going through research to see what makes a good web application and see the pros and cons to see why they're good to use.

Chapter three, describe the project planning, how it will be carried out over the next few months and the time that I have allocated for different stages of the project.

Chapter four, defines the design part of the project from the user interface to exampling the design process for the whole project, and show clear design criteria.

Chapter five, explain the implementation and the development stages of the project, show how the project was developed for the first deliverable to the second one, the methodology that I used and the result of the development process.

Chapter six, the testing scope, strategy and the test methods, as well as showing how well some aspect of the project performs.

Chapter seven, evaluate the project as a whole, the implementation, the approach, commercial factors and explore future potential features within the system.

# Literature Review

## Introduction

In this section, I will be investigating why the MVC model is used and look at the benefits of using it. We will also be looking at why ASP.NET is used and comparing it with ASP and to see which one should be used. The three-tier architecture was is also discussed to see why the architecture is useful for the project and how it can help the development and implementation process. The relational database will be going through how the data will be passed from one table to another.

## MVC Model

The Model View and Controller (MVC) was first seen by Trygve Reenskuaug back in 1970, he elaborated by saying “ the essential purpose of MVC is to bridge the gap between the human user's mental model and the digital model that exists in the computer” (Pop and Altar, 2014). The way in which MVC works is by splitting up all the roles into sperate roles which makes it which allows for efficient collaboration (Mcarthur, 2008). The reason why it is a good fit for web application development is that they all combine several technologies, these are usually split into layers. The MVC could have a specific behaviour to send a specific view to different types of users (Sendiang, Alelo and Ponggawa, 2017).

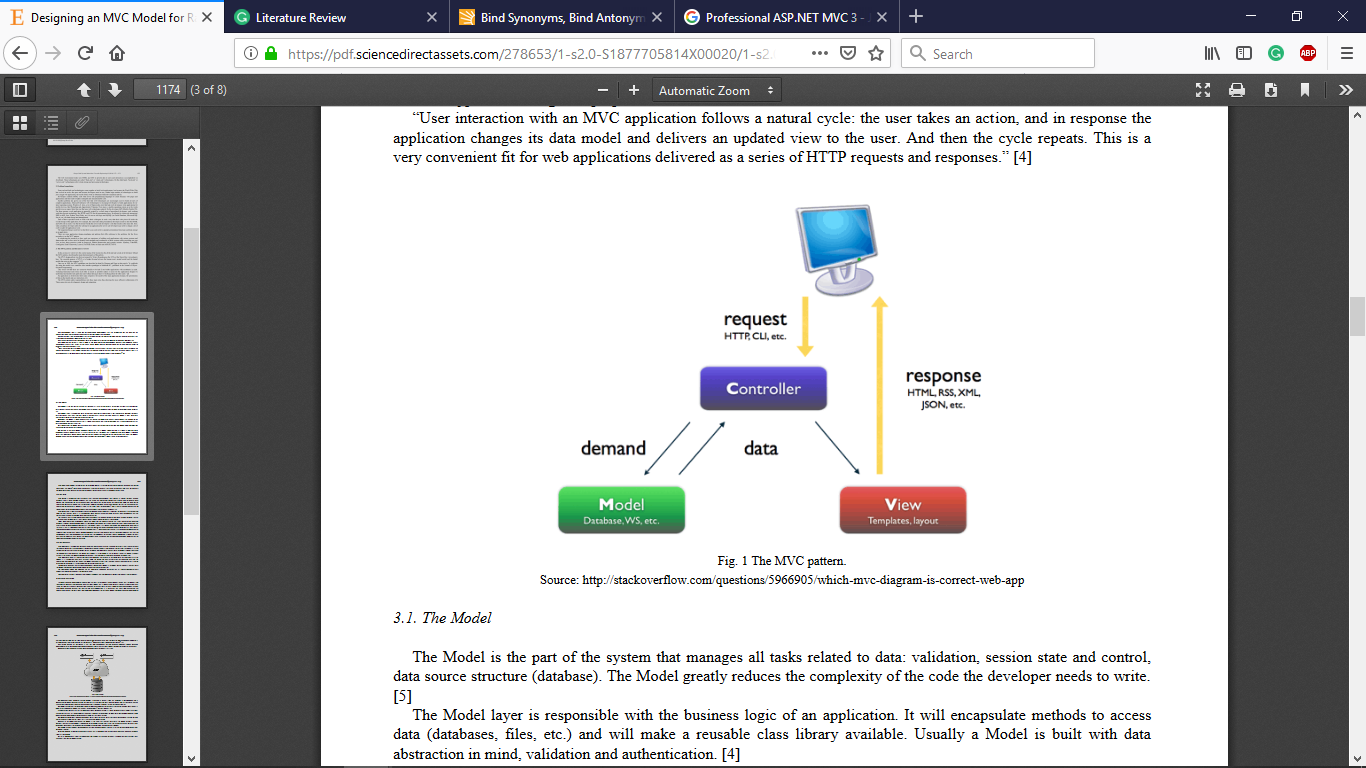


Figure 1

### Model

The way that the model is part of the system is that it manages the tasks that are associated with the data, validation, the sessions, control and the database. This would reduce the complication in the code that the developer will need to write (Gilmore, 2009).

One of the main responsibilities that the model layer has is with the business logic of the application that you are creating, it will compress the methods in order for it to have access to the data, for example, the database and any files that it requires, it also makes a reusable class library available. The way the model is built within the system with data in mind, validation and authentication (Sanderson, 2011).

The model layer is made up of classes that clarify the domain of interest, the objects that are built within the often at times will encapsulate the data and store it within the database, it will also involve code that will the data and enforce the business rules (Galloway et al., 2011).

To Conclude the model layer of the MVC model, it handles the data access validation and abstraction. It holds the method to allow interaction with all the different data sources.

### View

The role the view plays with the MVC Model is with the graphical user interface (GUI). This includes all the button, forms, graphic elements and any other HTML elements that are held inside of the application. The reason behind separating the design view from the logic aspect of the application is due to the reduction of risks that could show up from the designer of the application. The designer could decide to alter a table or the logo so by having the view different from the logic it would reduce the error. At the same time, the developer's job would not need to worry about the HTML element, design element and the graphical side of the application as it would be done on a separate layer (Gilmore, 2009).

The view layer is normally called by either web design or templates, the way that the layer works is the way the data is being displayed and how the users will interact with the design aspect of the application. The layer will provide a way of gathering all of the data from the user. Technologies that are normally used for the view aspect are HTML and CSS (Sanderson, 2011).

Most web application out there today use template engine, this will take the generated elements to keep the HTML code to a minimum to reduce the risk of any error occurring. We use these templates for a complexed web application that requires tables, menu or lists and so on (Pop and Altar, 2014). By using templates this would show a better design aspect and you could choose a design that would fit with your web application.

### Controller

The role of the controller is for event handling, the event would occur when the user interacts with the web application or via the system. The controller will accept the request and then send the data as a response, it will then be responsible for establishing the format for the response. It will interact with the model layer so that it can retrieve the data required and then it will generate the view (from the view layer), the process is known as an action (Gilmore, 2009). The request that has been sent going to the server, then the MVC Framework execute a method in a controller that is based on the URL (Stratmann, Ousterhout and Madan, 2011).

The controller puts together all of the application logic and combines the display from the view with the functionality that is inside of the model layer, it is responsible for retrieving the data from the view. The controller will have access to the model's functionality and it will retrieve the data so that it can be displayed by the view layer. It is also responsible for error handling(Sanderson, 2011).

The controller has a relationship with the model and the view. It mainly responds to all the user's requests and interacts with the model it then decides which view is going to be generated and displayed (Galloway et al., 2011).

### View to Model Communication

The view can communicate with and will know about the model, for example:

* If the user clicks on a button and message could be sent to an object model to do something
* A new value has been entered into an entry field to update a message and it might be sent to a model object so that it can get the new value that has been entered.
* If the value needs to be displayed the message might be sent to a model object in order to receive the value.

Above you can see these messages will be sent when an event occurs.

Once this is done it then goes to the job of the method inside of the application model which responds to that message in order to obtain the new value that is being entered and set the values (Deacon, 2011).

### Model to View Communication

The way in which the controller and the model communicate with the view is by having an event triggered. The event will provide a decoupled mechanism that would allow the communication with minimum dependencies.

The best way to explain this process to say that we are using graphical components, for example, a radio button, list box or entry field. The component that is listed can receive a notification such as being clicked. This event would normally come from the controller (Deacon, 2011).

## ASP.Net

### Background

Back in July 2000, Microsoft had officially released .Net, it was a business strategy that was mainly about providing software as a service and it was hosted on the Internet. For the users, it was simply accessing and storing the data in a platform independent way by using XML. However, for the developers, it was a whole new framework bring brought out which meant new languages and new programming tools (Mitchell et al., 2002).

Once this was released the .Net Framework SDK was also released, it was a collection of hundreds of classes. The classes would be used for the functionality that would be needed for .Net application development.

So going back to ASP.NET, when you are creating a web page using in ASP.NET what you are really doing is creating .Net application. This would then allow you to use all the rich functionality in the .Net Framework, which means from the ASP.NET pages you can only do things that are possible in ASP (Mitchell et al., 2002).

### Why use ASP.NET

Research shows that as of September of 2010, it was believed that around 25% of the internet’s web sites were being developed by using ASP.NET (Duong and Rizzo, 2011). The ASP.NET web pages are created by using compiled .NET compatible programming languages, these range from Visual basic .NET and C#. The ASP.NET web pages are created by event-driven and object-oriented techniques (Mitchell et al., 2002).

One of the main reason why developers use ASP.NET is that it is found to be easier to create a web application. These web applications are dynamic and data-driven, they work well across all different browsers, this is done without any custom coding being done by the developer. When you are using ASP.NET to have a wide range of choices developing a web application, you can code each web page by just using the bare bone text editor. This would only work if you were making a simple web page with complications, professional ASP.NET developers would rarely use this route (MacDonald and Szpuszta, 2006).

### Visual Studio ASP.NET

Instead of using the technique that I have discussed above, almost all complexed ASP.NET website is built by using Visual Studio. This is more of a professional development tool that would support a set of design tools, and this would include a set of debugging tools (this would catch any errors and also will give you suggestions as you type). Within Visual Studio it will give the developer the support a robust code-behind model and what this does is it separates the .Net code that you write for the web pages which are made up of mark-up tags. To conclude why people use Visual Studio has a built-in test web server this makes debugging hassle-free (MacDonald and Szpuszta, 2006) and can see the result of your code in any browser almost immediately.

### ASP.NET Vs ASP

The main difference between web services are :

. Asp.net is more event-driven with the event handler running on the server

. HTML is separated in Asp.net

. When you run ASP.NET it is compiled not interpreted (Liberty and Hurwitz, 2004).

The big difference is the separation of HTML, the reason for this is that the code complies rather than interpreted it will allow larger creation and easy to maintain a web site (Liberty and Hurwitz, 2004).

### ASP.NET Web Forms

These are a part of the .NET Framework and this is included within Visual Studio. The web forms are from what the user is requesting from their chosen browser, these pages are generally written by using a combination HTML and client script. When the user requests a page is it compiled and it is executed on the server by the .Net framework, the framework then will generate HTML mark-up that the browser will render.

One of the big benefits of using web forms is that it will allow the user to drag and drop the layout of your web form. It will give you the ability to set the properties, methods and the events for controls on the page, these will also define the behaviour of the web page. By using web forms it will allow the developer to separate the client and the server, they share a little information about one another and can communicate by simply exchange only a small chunk of data (Prosise, 2002).

## Three-tier architecture

### Background

The three-tier architecture is conceptual, when it is in practice there are different implementations of web applications that fit within the three-tier architecture. One of the most frequent implementations has the web server and database management system installed on the master machine. With this implementation in mind, it will be on modern hardware and your application will be able to handle thousands of requests every hour (Williams and Lane, 2004).

For many popular web sites, a common implementation is to have the web server and the database server on different machines, the reason behind this is so that the resources are dedicated to a permit which makes the application more scalable and faster. For the very advanced applications there can be many more machines used and on the machines, there will be a copy of the web server and database (Williams and Lane, 2004).

The modern web services now provide the use of a multi-tier architecture, this will provide the required services. before there was a three-tier architecture there was a two-tier one, only the database server and web server, they added an extra tier because to support the complex business logic. The three-tier architecture provides high-level scalability and reliability (Nagappan, Skoczyla and Sriganesh, 2004).

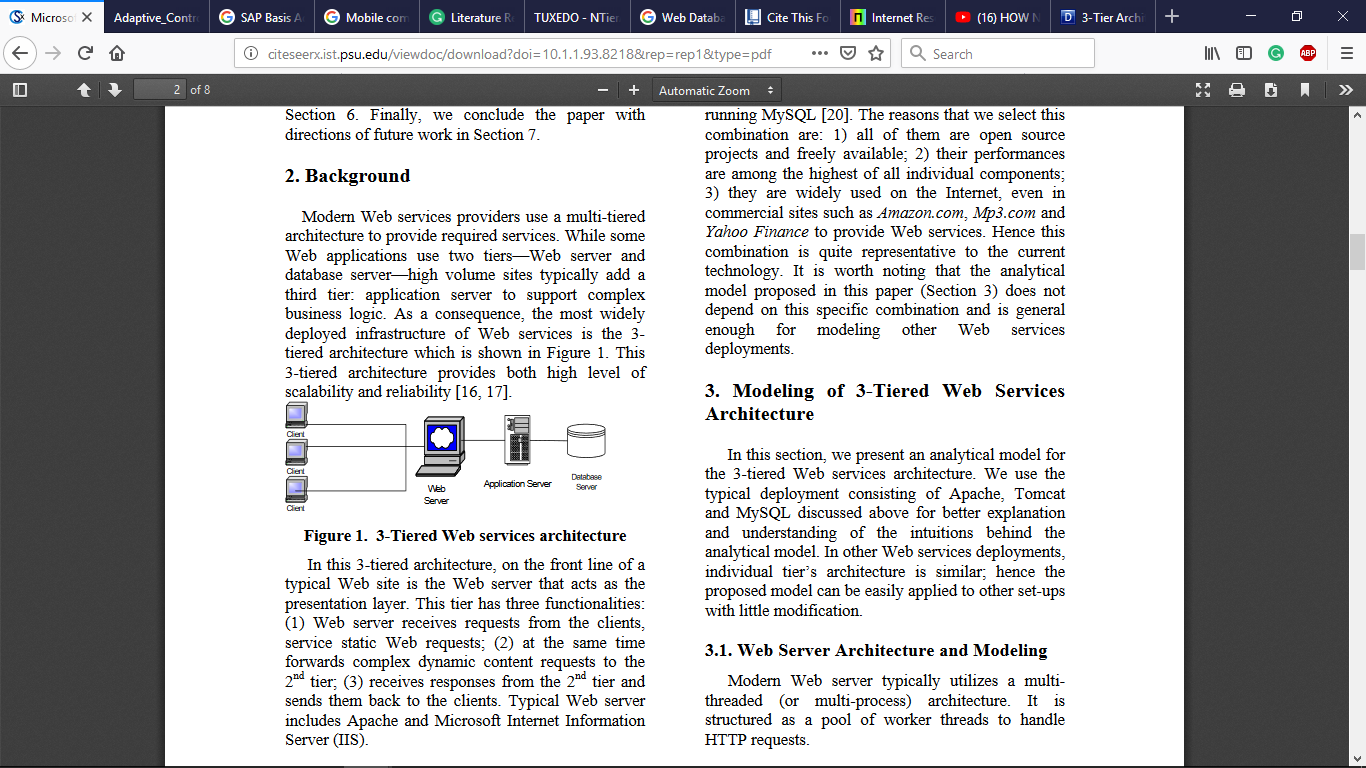


Figure 2

### Presentation Layer

In the three-tier architecture, at the front of the website is a web server and this acts as a presentation layer in the three-tier architecture. This layer has the responsibility for the user-facing device handling and rending the page for the user, this would also include the user system interface where user services (Talukder and Yavagal, 2005). This could be from sessions to display management and text input. there are typical web servers out there which include Apache and Microsoft Internet Information Server (MIIS). Apache has been one of the most popular web servers on the Internet (Foundation, 2019). In a web server survey back in February of 2005, it showed that 68.8% of websites that were on the internet used Apache (Mutton, 2019).

This tier has three functionalities:

* The web server will receive the request from the client.
* In parallel to the first one, there is a forwards dynamic content that requests from the second tier.
* Finally, it will receive a response from the second tier and sends them back to the client (Liu et al., 2008).

### Application Layer

The business logic of the websites sits on the second tier - the application layer. This layer is for the application program and managing the process, it is where the business logic and the rules are executed. It has the responsibility to control the transactions and asynchronous queueing, the reason behind this is to ensure reliable completion of transactions (Talukder and Yavagal, 2005). The application servers receive the request from the web server it will look up the information from the database (which is the third tier) and processes the information from the client. The processed information will then be passed back into the web server, it will then be formatted to be displayed on the client machine. There are some good examples of an application server; Apache Tomcat, IBM Websphere and JBOSS.

### Database Layer

The third tier which is the database server, this is what we would call a storehouse for the information of the website. This would be from the user's accounts to customers’ orders being stored inside of the database. The database server that is used within the web service architecture range from; Microsoft SQL Server to IBM (Liu et al., 2008).

### Pros of a three-tier architecture

One of the big advantages is with the development process, it can build complex applications rules which are easy to implement into the application server. With that in mind, the application server logic is, in fact, portable to other database server platforms by virtue of the application software. If you decided to make a change to the business logic what would happen is that it will automatically be enforced by the server, the changes will only need to require only on new application server software to then be installed. The performance is always great when it comes to the three-tier architecture, it has superior performance for low to high-end volume environment (Kambalyal, 2017).

### Cons of a Three-Tier Architecture

When building an application and are following the three-tier architecture it works really well if you are building a small simple application, however, if it gets too complexed application then the structure will also be complicated for anyone else to follow if they joined the project. The performance can also be a downfall at times, the physical separation of the application layer which contains the business logic functions, and inside of the database server, it contains databases which may affect the performance of the application/system (Kambalyal, 2017).

## Relational Database

### Background

It is a data model that stores data into tables, every table has a row (which can be referred to as a record) and has columns (known as attributes). Within a relational database, there can be many tables linked together. The model is originally based on mathematical principles, these principles were applied in 1960 by Dr Codd (Singh and Gupta, 2014). The relational model is one of the widely recognized as one of the great technological innovation of the twentieth century (Sumathi and Esakkirajan, 2010).

The mathematical basis is important in development and acceptance. The best way to look at the relational model (but not complete) is that they represent the data in a two-dimensional table. Before, the relational model has introduced the terminology that was used was files, fields and records, when the relational database came out the terminology changed to tables, row and column. However, if you were using a relational database for a mathematical reason then the term would be a relation, tuples and attributes (Ritchie, 2002).

Every row that is in the table has a unique identifier. Inside of the database model, there are relationships between tables that are defined using the unique identifier from one table and then joining them with the unique identifier from another table. This process is called normalization, the reason for this is because the data redundancy is reduced (Singh and Gupta, 2014).

### Advantages of using a relational database

A relational database is now widely used by many people and they are unaware that other forms of database exist (Whitehorn and Marklyn, 2007). So the question is why do people use a relational database. One of the biggest benefits of a relational database is well-formed and has a data independent model which means that it is easy to use for an application that fits within the model(Sumathi and Esakkirajan, 2010). By having more than one table what a relational database does is that it will retrieve the data either from one particular table or any tables that are related within the database (Singh and Gupta, 2014). The data that is being held within the database has to remain consistent and accurate because of the high level of integrity (Singh and Gupta, 2014).

One of the biggest reason why using a relational database is due to the fact it avoids duplication of any record and it will also detect if there are any primary values missing from the table. For the relationship part, it will ensure that the relationship between the pair of tables is valid before entering data inside of the table (Singh and Gupta, 2014).

### Disadvantages of using a relational database

A relational database can also have its downside to it if you do have more than one table they will need to be joined in many cases in order to retrieve the data (Sumathi and Esakkirajan, 2010). When you are using a relational database it heavily depends on the performance of the machine so, if you are on a small machine that does not run very well then using relational database would make the system run slow(Singh and Gupta, 2014).

## Commercial Factors

This is not the first shift swapping application that is out there. There are many out there and each of them have their own special feature. I will be looking at each of them and seeing how they did well and what worked for them.

Shyft (https://www.myshyft.com/ )

This was one of my biggest inspirations to crate this project. It has most of the features that I wanted to include inside my application. This has been quite a successful mobile application as it has received 4.5 stars out of 5.

When I was researching for good shift swapping application this was one of the top ones that came up. I had to receive a demo to see how the system works and I was shocked. It did everything I ever wanted to do, however what I did about this was take notes to see how it can be improved.

It uses your inbuilt calendar system that you have on your phone the problem that I can see with this is what if you already have something on that date? Would it clash with the other event? Which one will take priority? These are some of the question I would like to ask the users of the system.

I would need to make sure that in my system there are no issues that could occur and want to stay away from any clashes. This is why I used a simple grid view for the user to see and it will not be clashing with other applications. The feature that I would like to take from them is the notifying the manager feature as that is one of the problem them I am trying to fix.

This does come with a free trail for users to use however, if you would want to permanently there is an annual fee that will need to be paid.

Sling (<https://getsling.com/> )

This works again very similar to what I am trying to achieve. However, with this application it is more like a social media application. I feel like this from of web application would distract from the main objective of swapping shifts.

The reason why this works is because it is user friendly you see the look and it makes user want to interact with one another more. A feature that I think works well for them is the weekly report and they can then look back the rota and see how many hour the employees have done. I would like to take this idea and instead with the one I am creating have a button for report that the manager can see, it will show how many hours each employee did.

The similar factor is the price now for super small businesses that hold up to 50 employees that would be free if you are using this application. The more employees that you have the more expensive the application is. I would like to take a similar approach to this as if you have less then 50 employee the application is free however, if more then there will be an annual charge.

Branch ( <https://branchapp.com/#> )

This application is more about the communication with in the work place. It is mainly based on swapping the shift and only talking about swapping shift. The main idea that I got from this was the open shift.

One of my main objectives was to allow the manager to put overtime on the application and allow the employees to accept them, instead of texting them all the time. This application does just that and also write on the rota that it is an overtime shift. They mainly do this to sort out payment because in some organisation you get paid more for overtime.

## Summary

This shows how shift swapping application can be improved, the project that I am creating will need to ensure that goes with all the evidence that I have collecting to make a great shift swapping application.

# Project Planning

## Introduction

In this chapter, I will be looking at the steps that were taken to complete the project as a whole, and including the methodology, the requirements from the users and potential solutions. The chapter will also be looking at the tools and techniques that are used within my system and consider all the legal implications. I have attached the project proposal in Appendix 1, the technical plan in Appendix 2.

## Methodology

When it came to developing the project the methodology that I used was Agile. The Agile methodology will allow to continuous improvement to the system and this would happen throughout the development stage. It will be a more flexible approach because of any user requirements do change we can work around the new requirements. This whole project is designed, implemented and tested by on one person, the best what those Agile methodology works is when the project is part of a small team (Schuh, 2005).

The way that agile would be used for this project is by using the methodology of timeboxing, each feature can be timeboxes and given specific time to that feature (Stellman and Greene, 2014). By using timeboxing it will also show you how quick you can get specific features done and then see how long it would take you to do the next one.

You would look at the feature to ensure that it meets the users requirements, however, with agile approach the user requirements constantly change so the design and implementation also changes as well. An agile approach would be set the requirements as priority to either high to low depending on the users requirements, as you are asking new users you will be given new requirements. The way that agile works is by having all the software end done first over any documentation.

Agile for this project would be to start working to meet the ends user goal and constantly be in contact with the user about the system as the requirement would change as time will go on. I would need to ensure that I have set priority to tasks as software is an important part of the system.

## Requirements

### Key Objectives

The project key objectives:

• To be able to swap a shift with another employee with the same amount of hours

• The employee be able to either accept or decline a requested shift swap

• Manager and shift leader to see employees availability

• The manager would be able to set up overtime for the staff within the system

• Employee to request the amount of week for the shift to swap and can permanently shift swap with manager approval

### User Stories

The way that I had gathered the requirements from the users is by talking to them face to face and interviewing them about how they think the system would work.. Using user stories will describe the functionality that will be valuable to the user of the system (Cohn, 2004). User stories can be composed of three aspects, the one aspect that was used for this project was a written description of the user story which would be used for planning and looking back to make sure the user stories match the system that you have built (Cohn, 2004).

These are the user stories that I have created from my own experience at swapping a shift with another employee. This would also be considering the personas experience that is told inside the appendix. As stated in chapter one of there are three types of user for the system, manager, shift leader and an employee, these user stories will be coming from their aspect and what they would like from the system.

**Manager**

• As a manager, I want to be able to login/logout

• As a manager, I want to be able to see the rota

• As a manager, I want to be able to see the employee's availability times

• As a manager, I want to be able to confirm or decline permeant shift swap

• As a manager, I want to be able to set overtime for the employees

• As a manager, I want to be able to see conflicts in a shift if two shifts clashes and resolve them

**Shift leader**

• As a Shift Leader, I want to be able to login/logout

• As a Shift Leader, I want to be able to see the rota

• As a Shift Leader, I want to be able to see the employee's availability times

• As a Shift Leader, I want to be able to set overtime for the employees

• As a Shift Leader, I want to be able to see the employee allocated break time

**Employee**

• As an employee, I want to be able to login/logout

• As an employee, I want to be able to see the rota

• As an employee, I want to be able to swap a shift with another employee with the same amount of hours as me

• As an employee, I want to be able to select the number of weeks for a shift swap

• As an employee, I want to be able to request a permanent shift swap

3.3.5 Requirements

According to the user stories, the requirements have been gathered and set priority using MoSCoW analysis.

**The way that the manager will use the system:**

• Must be able to see the rota

• Must be able to allocate overtime

• Must be able to see employee availability

• Must be able to log in and log out of the system

• Must be able to see how many hours each customer assistant has done

• Should be able to have an administration right to edit within the application

• Could show conflicts for example if someone is working 2 shifts on the same day by mistake

• Could allow the customer assistant want to permanently swap their shifts with one another the manager would need to approve of it first

• Would be able to set up another employee account

**The way the shift leader would use the system:**

• Must be able to see the rota

• Must be able to allocate overtime

• Must be able to log in and log out of the system

• Must be able to see employee availability

• Must make sure each Customer Assistant has their allocated break time

• Should be able to see any shifts swap that does occur

• Could swap with another shift leader

• Would be able to set up another employee if the manager is away

**The way the Customer Assistant would use the system:**

• Must be able to see the rota

• Must be able to swap shifts

• Must be able to accept or decline requested shifts

• Must be able to see overtime page

• Must be able to swap shifts with someone with the same amount of hours as you

• Could be able to drop their shift if there cannot do the shift, for example, the timing of the shift or could be busy, but can only do this for overtime shifts, not there scheduled ones

• Should be able to see other people shifts

• Should be able to log in and log out of the system

• Could repeat the shift swap for a few weeks for example, if you wanted to swap shifts with someone you could request it to be up to 2 to 4 weeks

• Could permanently swap your shift with someone for whatever reason, if the 2 customer assistants agree with it (need to be the same amount of hours) then it would go to the manager to approve

## Potential Solutions

### ASP.NET

We had to decide on which language to use for the whole project, the decision was between ASP.net and PHP. We went with ASP.net over PHP because of these reasons.

When we was looking at the pros and cons of each language we needed to choose one suitable for web application. ASP.NET will supports all programming languages and it worked perfectly with C# and web applications (Bhagat, 2017), this would be a good because we can include different features and functionality together in order to make a good web application.

ASP.NET has been considered as one of the most secure, robust and flexible software framework (Shukla, 2017), as it support numerous libraries so that it can develop a robust web application for the user to use.

However, if we decided to use PHP it is an open source piece of software so it is free to use and it is being improved over the years. It has large community support of programmer, (Bhagat, 2017) which means that if you are struggling at what you are doing then you will know that there are people out there to help you.

One of the big advantages that PHP has over ASP.NET is that it is low cost and is easy to learn and pickup (Bhagat, 2017), as there are numerous support groups out there which will help with speeding up of the learning process.

### Web Forms

When it came to looking at the project and deciding which platform to use it was a hard decision to make. We had the choice between MVC and Web Forms, we decided to go with web forms.

The decision came when we was looking at what can be done on each platform and what resources we could use to make a good system. By using web forms it has shown that it has been around for many years and has been proven to be strong framework to use for web applications. As it is been around for years which could mean that it will be around for longer and will be getting better functionality with each release of the web form framework (Chadwick, Snyder and Panda, 2012).

The process of requesting a page is very important when making a web application, so when we looked into which one would load up a web page first it was web forms. The process to load a web page from web forms is simple, IIS will receive the request from the user, it will then find the physical file which in web forms is an .ASPX page and finally it would execute the web forms HTTP handler to execute the page (Chadwick, Snyder and Panda, 2012). However, with the MVC it is more of a process to execute a web page, it receives IIS but does not match with the physical file so it will query the route table which states that the request should be handled by a MVC HTTP handler and then finally execute the handler (Chadwick, Snyder and Panda, 2012).

When you are using web forms it is all within one project, so you can adjust to different part of a page, for example, you could be working on the functionality in C# about how the page will interact with the user and what would happen if a certain button was clicked. It also allows for two different design view, so if developers are comfortable with HTML they can use that aspect of web forms and if they are not as well experienced with HTML there is a drag and drop aspect of web forms MVC does that same however, they have the model view and controller, this works well but with a small team working on a system web forms would be the better option. When working in a small team to build a web form application every member of the team can be working on separate pages and then put them together would be easy to do.

To conclude, the evidence that we found for which framework we should use for the web application we decided to use web forms as working in small group would be easy and work efficiently.

### MySql WorkBench

To store all of the password, usernames, shift times and the rota it needed to be stored within a database. We could have gone with the built in one within Visual studio ASP.Net however we went with MySQL Workbench.

When using MySQL Workbench and built in database, you have a visual aspect of it so you will be able to see what data you are inputting into the database. However, if you wanted to execute a SQL query you will be able to do it directly on MySQL Workbench and see the results immediately.

Using the built in database within Visual Studio is nice and simple to connect to database and to have it connect with the project. Whereas, with MySQL Workbench you have to add the library externally in order for you to use it.

MySQL Workbench has a great user interface and is nice and easy to get your head around it and pick it up nice and easy. It works well with any errors you could have with any SQL statements, as it would give you a message if it was the wrong data type. The best error handling is when you delete data from the table it will ask you twice if you are sure about because deleting data would mean that you would not be getting that important data back.

## Legal and Ethical Issues

### Ethical Issues

I would need to consider the end user of the system, so it would be important that any data that is presented is accurate, especially if it contains personal information. One big ethical issue that will need to be discussed with the other employee if we can share their shifts with other employees. Some employee would like their shifts to be kept private and not shared with the rest of the employees. The way that we would try and get past this issue would be to let the employees know that you will be transferred online and that the shift times will also be shared, if any employee has an issue with this we would need to sit them down and discuss that private information such as availability and contract number will only be seen by managers or shift leader no one else.

Not everyone that works within an organisation will have a smartphone, so we would need to consider the age. The way that we would get past this issue is to have a tablet in the staff room so that people who do not have a smartphone at home can still use the system from the workplace. We will also need to consider training for the application (Cleland, Caldow and Ryan, 2007), this will take some time as the target audience is not specific.

## Summary

At this point, the project has a clear requirement and a plan of action to follow. The legal and ethical have to be identified and will be considered in the design and development process of the project.

# Design

## Introduction

The design of the system should give the user a good experience, however there is no agreement on what makes a good user experience (Kuniavsky, 2010). When it comes to the design process the user experiences is totally for the end user as they would be the user who will be using the system at the end of the project. The way that of thinking about it from the end user perspective, is by considering the user stories which us in the previous chapters, and looking at class/use case diagrams. The testing stage as the test plan will need to consider the end user. This will ensure that the system will give the user a good experience. An important key with the design is to balance the user experience with the cost this will ensure that the system that you are creating is valued for money.

## System Design

### Use Case Diagram

To assist with the designing of the project, a use case diagram has helped with the user requirement. The use case diagram captures the functionality of the system, it provides an implementation view of what the system is supposed to do and helps the developer to focus on the user needs rather than the under lying details (Pilone and Pitman, 2005).

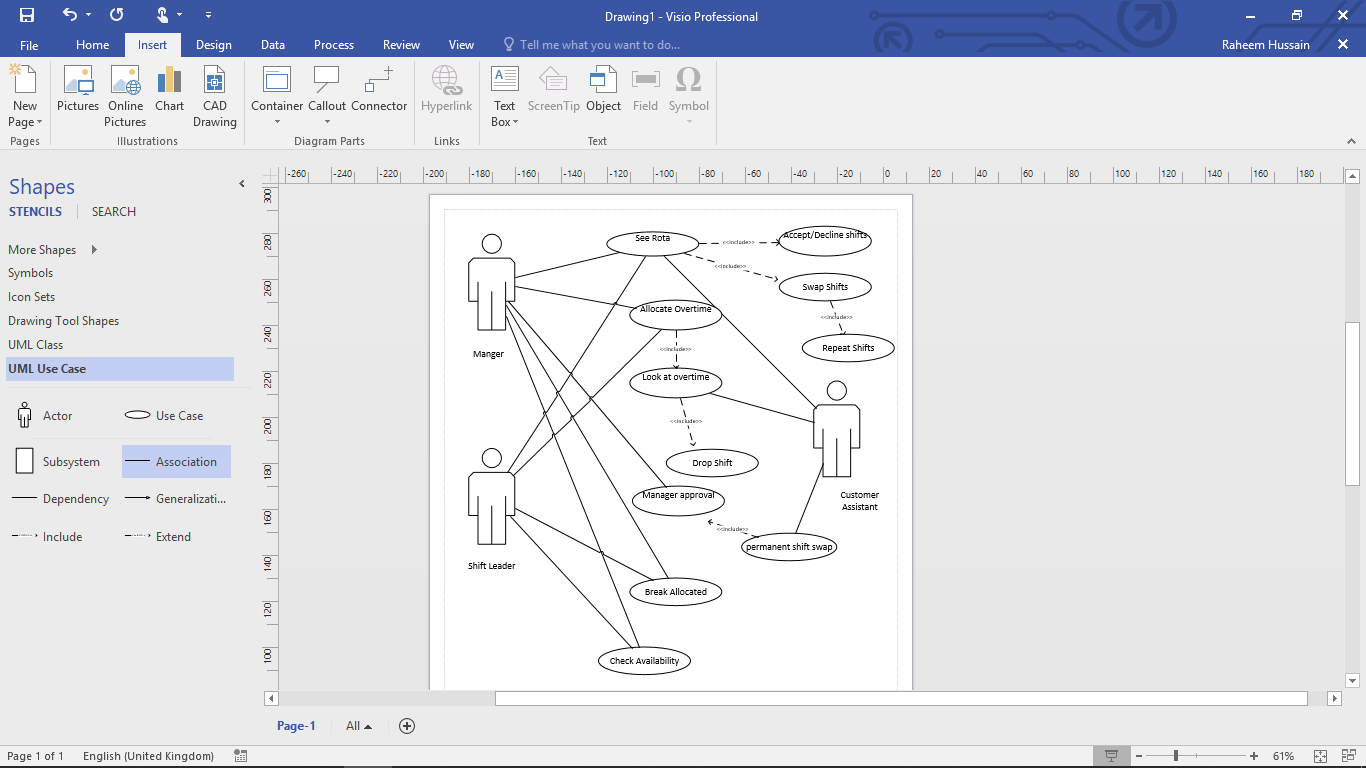


Figure 3

### Class Diagram

As well as using a use case diagram we also used a class diagram, a class diagram is split into three components, class name, class attribute and class operation. The main part of a class diagram is the attribute because this would refer to the properties(insert citation). Within the class diagram you would see the methods used within that class which would help with the implementation of the project. Inside the middle tier (attributes) will be the private and public methods that will be used and it is clearly identified if two classes share the same method.

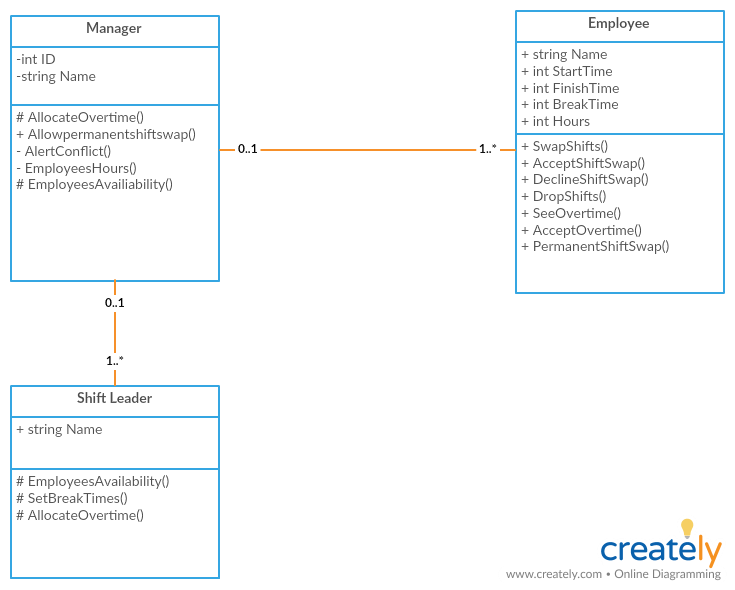


Figure 4

### Database Design

The database will have four tables, one table will store all the employees ID, Username, Password and the user type. Another table will hold the requested shift when the user wants to swap shift, the third table would be the rota. So the table would hold each employee shift, staff ID, start time, finish time, break time, hours and shift leader. This will be displayed when the user clicks on rota and show all the shifts and will display the data on this table. The final table, would be the staff availability, it will hold personal information that the shift leader and manager will be able to see. The database was in MySQL Workbench with all four tables included to the main database.

## User Interface Design

### Web Application Design

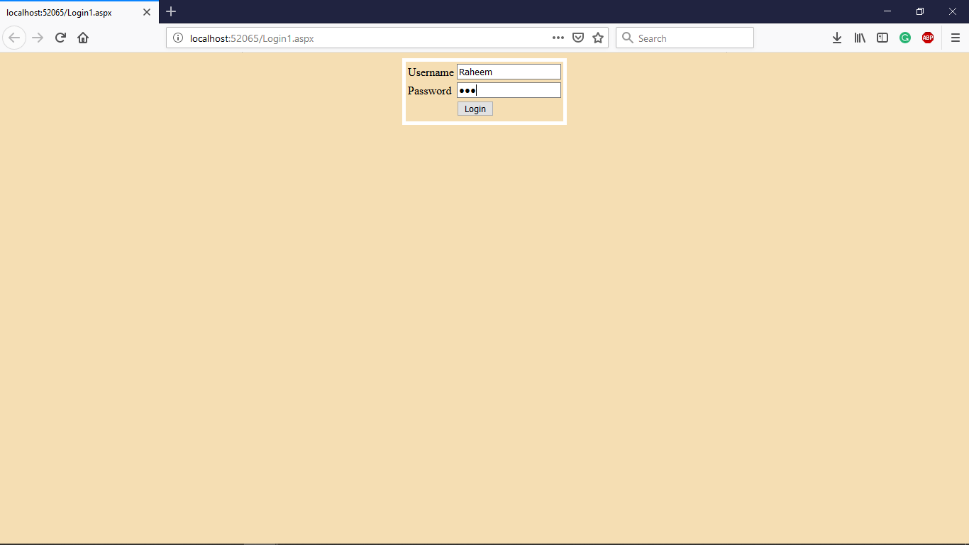
This would be user interface from the web application side, the image below is the interface for when the user will have to login to enter the site, if the user credentials are not correct it would not allow them to continue on to the next page.

Figure 5

Once the web application has been opened this page and the user has logged in with the correct credentials they would need to click on the login page in order for them to proceed to the next page. Once it has been accessed the web application will show a dashboard with pages that the user will be able to access (if shift leader and manager they can access an extra page). The user will be able to see your username with the hours that the employee is doing that week, the pages that an employee will be able to access are the rota page, shift swap and request page if they have any request to swap shifts.

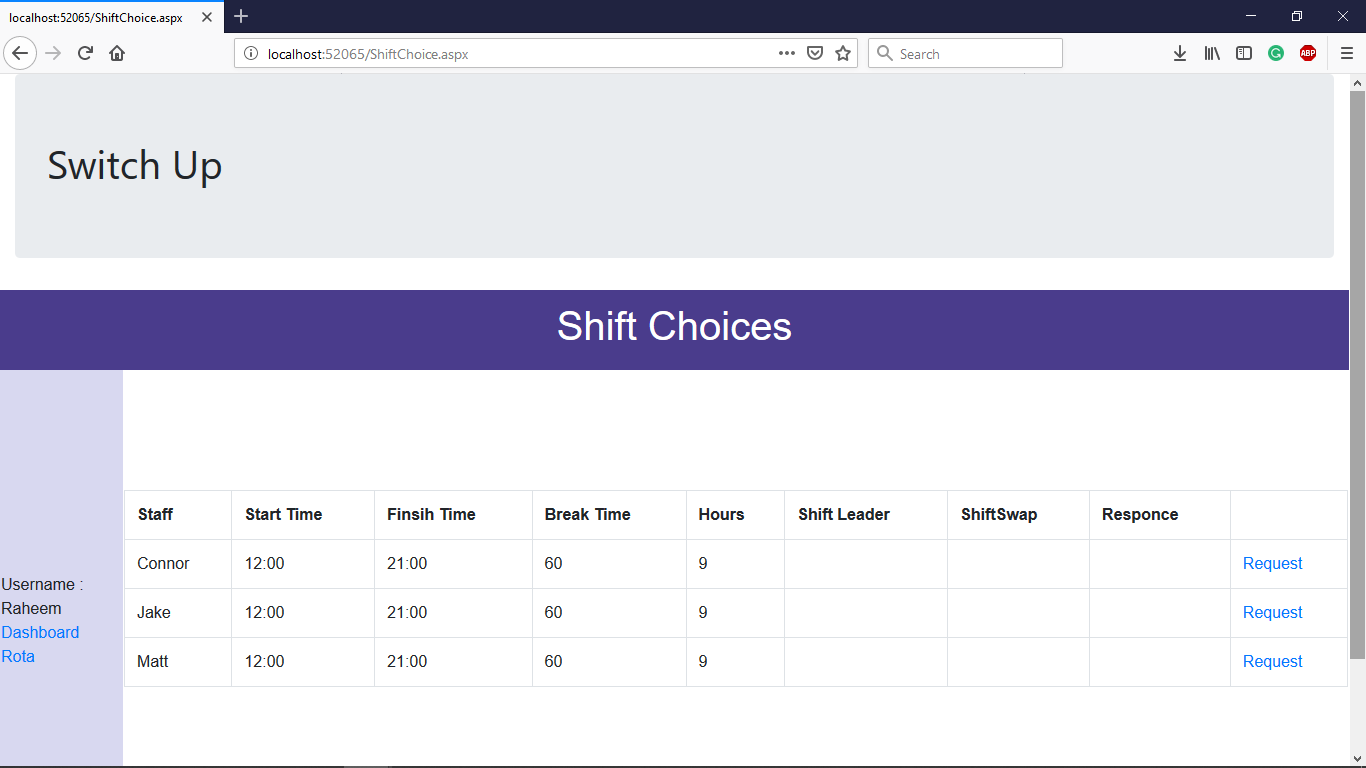


Figure 6

This would be shown to the employees who want to shift swap as you can see from the image above it shows every employee with a nine hour shift, so the employee who is logged in as a nine hour shift. This page will be different depending on the employee that has been logged in, it will give the user an option to request that shift from the employee and be taken to another page.

The design colours have been kept nice and simple to seem professional about the concept idea. White colour clearly shows up with the blue text and all the text can be seen clearly from a distance. The idea of the colours came from keeping a web application nice and simple in order to see the text and easily recognisable to the employees. Other colour concept were considered however other colours made the web application look to colourful and the colours clashed together, the best idea was to keep it nice and simple.

## Summary

There was a number of choices to be make at the design stage to make sure that the whole system will meet the user requirement and is suitable for the end user. This web application will be used to login and see the rota and have the ability to swap shift with another employee. The decision that was made was to ensure that the end user would have a good user experience as a whole.

# Implementation

## Introduction

The implementation and the testing process will go one after the other. Once each feature of the web application was complete a test was done straight after to ensure it was done correctly. When it came to developing the web application the Agile methodology was used, so once each feature was created and tested thoroughly, it then became a part of the final product. In this chapter, we will be looking at how the system functions and components are implemented within the system, it will cover any challenged that was encountered.

## Login Page

### Login Page Functionality

The way that this was implemented onto the system was by first having a Boolean data type which would hold the error message. To start with the Boolean would be false because no information inside username and password has been entered so there is no need to the error message. Once this was done then the database connection needed to be added, when using MySQL workbench and using ASP.NET you would need to include the library in order to do this you would have to add NuGet package to include MySQL library. This would need to be done to every page that uses MySQL.

When the library is included it would then allow you to make the connection to the database, it will require you to the administration right so you would need to make sure that you remember the username and password for that as it will come in handy. The SQL statement would come next, so you would select from the database table (whatever you have called it) where username/password in the database matches the username/password that the user has enter.

It will take the parameters from whatever the user has entered and remove all of the white spacings by using the text.trim(). When the white spacing has been removed then it would go through an if else statement.

if (Users > 0) //Depending on how many users

{

Session["username"] = txtUserName.Text.Trim();

Response.Redirect("Dashboard.aspx");

}

else { lblErrorMessage.Visible = true; }

The if else statements reads if the users are less than zero then allow them to process. The session username allows that username to appear on the next page and the next page is dashboard.aspx. As you can see from the code example the else part makes the Boolean true, so if the username/password does not match the database then the error message would appear and not allow them to proceed to the next page. These are all the functionality that have been used in the login page.

### Login Page Design

The design was kept very simple with a box which hold the username and the password in order to login and proceed to the next page. The GUI (Graphical User Interface) is very simple it will involve two asp label one for the password and one for the username. This will be followed by two asp textboxes which will hold the password and username. A login button will be required for the user to click on and a final label saying ‘incorrect credentials’. The way that the login page works is by making sure that the database held the employees username and password. The username and password that the user will be entering will need to correspond to the one on the database, if they do not match it would not allow them to proceed and a message would come up saying ‘incorrect credentials’.

### Complications

As simple as this sounds there was some complicated aspect of this page, a complicated aspect came from the logic that I was using. The if statement which was used I did not have it correct first time, when I first wrote the if statement it was:

if (Users < 13) //Depending on how many users

{

Session["username"] = txtUserName.Text.Trim();

Response.Redirect("Dashboard.aspx");

}

else { lblErrorMessage.Visible = true; }

The logic behind this if statement was that I had thirteen users inside of my database, and I was thinking if there are more than 13 users accept them and proceed to the next page. However, this would mean that any user can be accepted, when I was testing with this logic.

## Dashboard Page

### Dashboard Functionality

The dashboard would be the next page followed along from the login page, the functionality on this page is quite difficult as there are three types of users so each user will be seeing different buttons. The first part of this page is to ensure that a user has been logged in and if they are not logged in refer them back to the login page to make sure they login. This is then followed by a similar to the login page however the only thing that changes is the SQL statement, for this page there statement would read select all from login where username matches the session username who is logged in and the user type is equal to one and two.

string query = "SELECT count(\*) FROM login WHERE UserName = '" + Session["username"] + "' AND (UserType = 1 OR UserType = 2)";

This query checks if these user has the user type one or two (manager and shift leader), there would see the availability button as there they are user type one and two which is defined within the database. There is a Boolean which has been set to true if there is a user type equal to one and two, however, it is false for the rest of the users.

On the dashboard page, I wanted to display the amount of hours the user who has logged in as. In order to do this another query will need to be done, however with this one you would need to take the hours of the user who is logged in you would need to store the username into a session. The query would say: Select hours from the rota where StaffID is equal to the person who is logged in. I then needed to store them hours into a session so that I could use it for other pages.

### Dashboard Design Implementation

The design was much more complexed then the login page as this would be the first page the user sees when they login. I needed to make sure the colour are right and some CSS was done in order to make it look good. Inside the body of the design it was all split into tables so the top half was a table which should the company name.

Followed by that would be the name of the page which is dashboard, this is displayed in white text with a blue background so the text will stand out to the user. Along the left hand side of the web page comes the menu, there is a lot stored in the menu it will first say the username of the person logged in followed by the amount of hours they are doing. It will also state the pages that the user can access, one of the main ones is the rota page followed by shift choices that the user has. The next button is based on which type of user is logged in, so if a shift leader and manager is logged in they will see an availability button, however it will not be visible for other employees. The final button for everyone is the logout button which allow the user to logout of the system.

### Complication with Dashboard

When it came to this page one of the difficult challenges was only having the manager and shift leader see the availability button as I was not sure about the SQL Statement. The SQL statement before looked like this:

string query = "SELECT \* FROM login WHERE UserName = '"+UserDetails.Text+"' AND (UserType = 1 OR UserType = 2)"

As you can see from the statement about I did not store the username into a session so it did not know what it was referring to. The way that I got passed this mistake was by having a session to do this in order to store the username and refer to it for the statement.

## Rota

### Rota Functionality

The rota page was one of the most important pages as we needed to show the data from the database. Before we start we again need to make sure that the user is logged in first and if they are not logged in redirect them to the login page. In order to display the data you will need to connect to the database, so you will have a connection string that hold the connection to the database. A simple query will be executed in order to display the rota from the database, the SQL statement will be Select all from rota. Once the query is executed you will need to hold it the data and then use data bind to display it within a table.

### Rota Design Implementation

The table that I have used for the rota has been taken from a template online, I used a template bootstrap table in order to make the table look professional and not just reply on HTML to make the table. You would need to include the bootstrap library in order to use the downloaded template. These are the library that I included in the HTML to use the downloaded template:

<!-- Custom fonts for this template-->

<link href="vendor/fontawesome-free/css/all.min.css" rel="stylesheet" type="text/css"/>

<!-- Page level plugin CSS-->

<link href="vendor/datatables/dataTables.bootstrap4.css" rel="stylesheet"/>

<!-- Custom styles for this template-->

<link href="css/sb-admin.css" rel="stylesheet"/>

The menu portion of the page would be same however, this time in the menu there will be a hyperlink that links back to the dashboard and the shift choices the user has. In the main body of the web page comes the data grid view which will hold the rota information, in the data grid view there will columns from the database for example, start time and end time etc. These will be shown inside of the data grid.

<asp:GridView ID ="StaffID" class="table table-bordered" runat="server" AutoGenerateColumns="False" OnSelectedIndexChanged="StaffID\_SelectedIndexChanged">

<Columns>

<asp:BoundField DataField="StaffID" HeaderText="Staff"/>

<asp:BoundField DataField="StartTime" HeaderText="Start Time"/>

<asp:BoundField DataField="FinishTime" HeaderText="Finsih Time"/>

<asp:BoundField DataField="BreakTime" HeaderText="Break Time"/>

<asp:BoundField DataField="Hours" HeaderText="Hours"/>

<asp:BoundField DataField="ShiftLeader" HeaderText="Shift Leader"/>

<asp:BoundField DataField="ShiftSwap" HeaderText="ShiftSwap"/>

<asp:BoundField DataField="Responce" HeaderText="Responce"/>

## Shift Choice Page

### Shift Choice Functionality

Just like the rest of the pages I needed to make sure that user was logged in first in order to continue if not logged in then redirect to login page. I needed to refer to the database for this as I need ta grid view to show the employees with the same amount of hours as the user. The SQL statement to make this happen would be to select from the rota where the hours matches the session hours (your hours) and to know include yourself in that list as well, this is how the statement would look:

MySqlDataAdapter sqlDa = new MySqlDataAdapter("SELECT \* FROM rota WHERE Hours =" + Session["hours"] + " AND StaffID != '" + Session["username"] + "'", sqlcon);

This would just include every employee with the same hours as the employee and the employee does not match your ID. The next thing is to be able to select a specific row from a data grid.

In order to this you would need to have a function with the name of the grid and row command:

protected void StaffID\_RowCommand(object sender, GridViewCommandEventArgs e)

This will get the specific row, you will need the specific row in order to request a specific shift from an employee. Inside of this function it is a container and I used indexes to know what was at each position of the container. There would be a request button at the end of each row in order for them to click on and sent the request.

Once that button is clicked then a query will need to be put in to as when the request button is hit, insert that row to a requested table so it would move to another page. I needed to make sure that I passed the parameters as the ID, Start Time, Finish Time, Break time, Shift leader. As this is the information that the other employee will need to know before they accept or decline the shift.

### Shift Choices Design Implementation

The format of is still similar to the Rota page as it is showing the same information, however, on this page it will have the button to request the shift swap. The menu will hold the username and hours, followed by links to the dashboard and the rota. When you click on the request button there is a label at the end of the grid that will display that specific row to make sure that you have clicked on that row.

### Shift Choices Complications

The challenge for this page was to select a specific row from the grid as I had used a data grid view it did not directly allow me insert it is as it would just refer to the complete grid not a individual row that I needed. This did require a lot of my time and I had to go back and forth to find resources online in order to do it. The way that I got over this challenge was to go through a tutorial that was having the same problem and complete the tutorial on a sperate project to make sure the tutorial was correct for me. Once I had completed the tutorial I then included it within my situation and it worked perfectly for me to use. So it got me through that challenge and needed to include it as it was a vital for the project to be able to select a specific row from the grid.

string Request = "INSERT INTO request values ('" + ID + "','" + StartTime + " ','" + FinishTime + " ',' " + BreakTime + " ',' " + Hours + " ',' " + ShiftLeader + " ')";

## Availability Page

### Availability Page Functionality

The functionality of this page is very similar to the rota page as it will only display information from the database. This page will only be available for the shift leader and the manager. This page will only need one SQL statement which will just populate the data from the database onto the data grid. The data will fill the grid and will need to make sure that you bind it into that grid.

### Availability Page Design

The design of the web page is very similar to the Rota page, the downloaded template of the bootstrap will need to be included in every page that requires the grid view in order to be consistent throughout the web application. The table in this one will have different columns as this would hold personal information, the information that it would hold is; First Name, Surname, Contact Number, Email and the employees availability. As this information is personal to the employees only two types of users can see this information, the shift leader and the manager himself.

## Request Page

### Request Page Functionality

This page was the request that get sent from the shift choice page, so when they click on request it would refer to this page. The SQL statement will just be select all from the request table in order for the employee to see the request that could be sent to them.

In order for people to accept and decline I needed to get the specific row again in order to accept/decline that specific shift. It will require the same piece of code as the shift choice but instead of having a button with request it would be accept or decline. When one of the buttons are hit it will pop up a message saying either accept or decline, once you have accepted you cannot decline a shift.

### Requested Page Design

This page will be consistent with the rest of the web pages, as consistency is key when you are looking at a web page. However, in order to get to this page it will be on the dashboard so that when you log on you can easily access this page to see if you have any requests. As it is using a table I stayed consistent and used the same downloaded bootstrap template as it made it look professional for the user aspect. The username will also be consistent throughout the menu section as it will always have your name on the menu. Similar to the shift choices it will print out the accept below the grid to make sure you have accept the right shift.

## Summary

This implementation process was a long process to complete, I learned a lot along the way of the project. There was some complications with some parts of the implementation stage. Once I had my head around it became a lot easier to implement and each page design was the same to make sure it was consistent throughout all of the web pages. The functionality became more complicated as each feature was hard to implement as most of the implementation either was SQL statements or was something to do with the data grid that I had used. Overall, I feel the implementation was done at a disenable standard, more could have been done to import more features. The prototype shows the capability and the potential of being a full shift swapping application.

# Test Strategy

## Introduction

Each web page and its features need to be tested, as well as testing the whole project one it is complete. This chapter will go alongside the implementation, once a part of the project was tested it would then become part of the implementation.

## White Box Testing

White box testing is where you get a participant who has some knowledge about the program code and the language that you have chosen. The reason why we have white box testing is so that other programmers can have a look at the code and will spot some errors, other programmers will be able to pick up on the errors over the person who wrote the code. The person who wrote the code may not be most efficient in detecting errors with their own products (Gopalaswamy and Desikan, 2009). Another programmer will be able to see the error and could help you to improve the system that you currently have. The other programmer will take into account you program code, code structure and he design flow of your system (Gopalaswamy and Desikan, 2009).

The way that I used white box testing is by having a person who has been programming for a long time and has used web forms previously. Some complication was discussed in the implementation and the way that I got through them was with some white box testing. For example, the complication with the login page that I had with my logic behind the users being allowed on the system. I used white box testing to have another programmer to look at the code. The other programmer spotted the error within the code and helped me sort the logic out correctly.

I used white box testing to ensure that the code that I was writing was correct was logically correct and to ensure there was no error within the code. One of the biggest benefits I received from this was my logical errors within my code, having another programmer observe the code and see how each feature works they were able to see the logic could be wrong and how I would fix the error.

## Black Box Testing

Blackbox testing is where someone who has no idea about code or anything behind the scenes of the system. With Blackbox testing the user will only be looking at the specification, the way that this is done is by looking at the system from the customers viewpoint. The test engineer that is involved with black box testing will have a note of a set of inputs and the expected outputs, the test engineer is unaware about how the user is going to use the system. For example, the user could enter a whole load of number instead of logging and it could accept it, so you should be aware of what the user is inputting so that you are fix away around the user.

The way that I used Blackbox testing was by having the users of the system using the web application. So when the I completed the login page I should the end user what I had completed and told them to mess around with the login page. The user entered number instead of text, putting credentials inside the username and password and leaving it black to see if it will allow them to login.

I used Blackbox testing to make sure that the requirements was met by the end user, it would help me to identify any issues that can occur within the system, because Blackbox testing is based on the end user requirements (Gopalaswamy and Desikan, 2009). Since I wanted to test the behaviour of the product to see how the user will be using the system from their eyes. Blackbox testing helps with handling valid and invalid inputs (Gopalaswamy and Desikan, 2009), the user can be unexpected so as discussed above we don’t know what they could enter into the system. I do need to make sure is that every input that is enter comes back with correct error handling and does not just crash the system.

## Web Page Testing

For each web page it needed to be tested thoroughly, so I have created a test plan for each page one table for the test that I want the user to do and write down the results from that. The other table being what I have enter and what I expect the results to be from the input.

### Login Page Test

This is a table showing the test could be entered for the login page.

|  |  |  |  |
| --- | --- | --- | --- |
| **My Tests** | Expected Results | Actual Results | Pass/Fail |
| Username:  Password: | Incorrect Credentials | Allowed Access | Fail |
| Username: Raheem  Password: Correct | Incorrect Credentials | Incorrect Credentials | Pass |
| Username: 1234  Password: 1234 | Incorrect Credentials | Incorrect Credentials | Pass |
| Username: \*%£””  Password: \*&^%$ | Incorrect Credentials | Incorrect Credentials | Pass |
| Username: username  Password: password | Incorrect Credentials | Incorrect Credentials | Pass |
| **User Tests** |  |  |  |
| Username: Raheem  Password: Raheem | Incorrect Credentials | Allowed Access | Fail |
| Username: jfnsuhfdfs  Password: skfjksjdlkfj | Incorrect Credentials | Incorrect Credentials | Pass |
| Username:95494kfksf  Password: 947jfskf | Incorrect Credentials | Incorrect Credentials | Pass |
| Username: dmfd.464.342  Password: 545.3434 | Incorrect Credentials | Incorrect Credentials | Pass |
| Username: 9797492742  Password: 9385973975 | Incorrect Credentials | Incorrect Credentials | Pass |

Figure 7

As you can see from the table above some of the test that I have written up to be tested and what the user entered. Most of the tests pass however, two of them have failed, what I expected didn’t happen and it gave the user access to process to the next page.

After looking at these tests and seeing the ones that have failed I will need to go back and make sure they pass before the project is released. I needed to go back to the implementation section in order to make sure that these test pass and then go back and test again.

### Dashboard Page Test

This will be table that shows all the test from the dashboard page.

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected Results | Actual Results | Pass/Fail |
| Manger or shift leader login | Show availability button | Shows the availability button | Pass |
| Employee login | No availability button | Shows the availability button | Fail |
| Username displayed | Username displayed in the menu area | Displays ‘username’ not the actual username | Fail |
| Hours Displayed | Hours displayed in the menu area | Displays Hours in text and does not correspond to the hours on the database | Fail |
| Link to rota | Go to rota page | Goes to the rota page | Pass |
| Link to shift choices | Go to shift choice page | Goes to the shift choices page | Pass |
| Logout button clicked | Logout of the system | Logs the user out of the system | Pass |
| Request button clicked | Go to request page | Goes to the requested page | Pass |

Figure 8

With this page there was only limited amount of events that could occur and limited information on the page. So the user didn’t really have a lot to do other then go back to the login page.

The fail ones will need to be looked at closely and try to make them get to a pass and reshow the user so that the fails become a pass. One of the big fails was the availability button because no matter who logged in the availability button was visible, I would need to go back to the implementation stage and try and fix this error. This would be the same for the username and hours not being passed over from the database as it only showed the text ‘username’ and ‘hours’ but it was not the correct information that should have be there.

### Rota Page Test

These are the test from the rota page.

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected Result | Actual Result | Pass/Fail |
| Display the rota | Show rota | Show rota | Pass |
| Links to shift choices | See the shift choices page | See the shift choices page | Pass |
| Link to the dashboard | See the dashboard page | See the dashboard page | Pass |

Figure 9

All the test for this this page passed so no further implementation is needed.

### Shift Choices Test

This table shows the test on the shift choice page.

|  |  |  |  |
| --- | --- | --- | --- |
| Tests | Expected Results | Actual Result | Pass/fail |
| Should show hours matched to the user logged in | See employees with hours matching the user logged in | All employee was listed | Fail |
| When the request button is clicked request is sent | Request sent message | Nothing happened | Fail |
| Username displayed | Username displayed in the menu area | Displays ‘username’ not the actual username | Fail |
| Link back to dashboard page | Dashboard page displayed | Dashboard page displayed | Pass |
| Link to then rota page | Rota page is displayed | Rota page is displayed | Pass |

Figure 10

As you can see from the table above most of the test for the shift choices page failed, this page would require the most attention after this test plan.

I would need to go back and priorities this page as this is one of the important pages for the user to use. I would need to take this test plan and go back to the implementation stage and make sure that all the test that have come back as fail pass next time it is with a user.

### Request Page Test

This table will represent the events on the request page.

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected Result | Actual Result | Pass/Fail |
| Shows requested shifts in a grid | Grid of requested shift | Grid of requested shifts | Pass |
| Accept button | When accept button is hit, insert it back into the rota | Just a pop up message | Fail |
| Reject Button | When the reject button is hit, the rota stays the same | Rota stays the same | Pass |
| Link back dashboard | Dashboard page is displayed | Dashboard page is displayed | Pass |
| Link back to the rota | Rota page is displayed | Rota page is displayed | Pass |

Figure 11

The main feature that I wanted to pass unfortunately didn’t pass the test. I would now need to go back to this page and implement this feature again and look at the logic that I have used. I will need to check this feature again once I have implemented it differently to make sure I get the result that I need.

### Availability Test

The final test plan is for the availability page, as in previous test plan this page should have only been seen by shift leader and manager however, that failed so we know about one fail on this page already.

|  |  |  |  |
| --- | --- | --- | --- |
| Test | Expected Results | Actual Results | Pass/Fail |
| Grid shows availability | Correct grid showed | Correct grid showed | Pass |
| Links to dashboard | Dashboard page comes up | Dashboard page comes up | Pass |
| Link to the rota | Rota page comes up | Rota page comes up | Pass |

Figure 12

All the test from the availability page did pass, however I would need to go back to the implementation to make sure that only the manager and the shift leader can see this page.

## Summary

Testing stage had proven to be important and a helpful. Each page was tested prior to and during development, each page has its own test plan. Some pages have had some fail tests, however, by testing I found these fails which proves that this stage was important for the user to use a prototype of a system. I will be taken all the fail tests and going back to the implementation stage to make sure these fail tests come back as a pass next time I do them.

# Evaluation, Conclusions and Future Work

## Project Objectives

Looking back the project as a whole as seeing what I have been able to achieve compared to my objectives that was set in the first chapter. However, I have not been able to achieve everything that I wanted to achieve for this project.

There was some parts that I was able to achieve, the user is able to look at the shift for the week upcoming so I have been able to achieve the aspect of having a online rota inside of having a paper one.

The manager and the shift leader are the only two types of users who are able to see the employees availability, this will keep the personal information private from other users.

The end goal was to be able to swap shift with another employee, however I am only half way of doing that feature. You are able to request a shift swap with the web application and you will be able to see the request page, the problem occurs that other employees can see the request and accept and decline the shift from another employees behalf.

I have been able to achieve what I wanted to achieve to a minimum, it does the basic objectives as being able to see the rota and be able to request a shift swap. There are three different types of users within the system and every one of them does something different. I was able to have these users and for the user to have their own separate features.

## Self-Evaluation

At the beginning of this whole process my goal was to have a mobile application for the user to use on the go, so I thought it would be easier to make a web application first and then have the user be able to access the web application on their phone. I feel like I was a bit to ambitious about what I wanted to achieve at the beginning of the project, I felt like I would know how to do a lot of the stuff but when it came down to it I didn’t know a lot about the mobile/web application. I felt like I set myself challenges that knew was unable to achieve.

I found the whole experience difficult and faces a lot a learning curves. In the project there was tools and techniques that I have not used in the two years that I have been at university. I had never been used to making a web application with so many features with application, I knew how to use the C# and HTML at the basic level and I knew that I needed to go away and learn new things about C# and the web in order to make the web application. I did have some experience with database but only at the theory level not used database practically with an application, making the web page was nice and simple the complicated part was connecting it to make it work with my application.

The project as a while was a great idea theoretically in my head although I didn’t stick to that idea. Once I had fully started programming the web application, that it when it all got to me that I will not be able to complete all the objectives within the time that we have. Displaying the information on the screen from the database was the easy part, where it became difficult for me was the SQL Statements. I had never really done SQL before but I understand the theory behind it but had never done it practically. Where is all got difficult for me was the part where I wanted specific users to only see one particular page, as the query for that I needed a lot of help with because I didn’t know how to do it. I had to ask help for a lecturer who was a good programmer when it came to databases and knew what I am was trying to achieve. This got frustrating after a while as I started to realise that most of my code need complexed SQL statement and I could not get my head around it, so I was always either emailing the lecturer for help or booking an appointment once to week to see them.

Most of the time when I was struggling for help, I got the help that I needed or fixed it by looking online for help, I kept feeling like I was getting somewhere but was not getting anywhere. I felt as if I was not doing well which took an effect to my mental health, I didn’t feel like I was doing well. Talking to other peers and seeing how far they had got with their project make me feel bad. I know I was putting the work in but just felt like it was never enough at times. I wanted to finish the web application by Christmas and then start on the mobile application after Christmas, however at the Christmas period I was nowhere near finished with the web site. By Christmas I only had the users being able to login with the correct credentials and showing the rota correctly from the database.

The good experience about this whole project was to have the opportunity to see how difficult it is to make a project with little planning for a single person. This experience would help me in the real world as if I ever needed to do another project I would plan each step out and see if I am able to do it with the time I have. The experience has set me for real life future project. The idea kept me going. By having a good idea in my head made the whole experience worth it, as I felt like if this project was complete how it would help out small business to go from paper-based rota to an online one. I knew that I would be making a difference if all of my objectives was met.

I have has the ability to use web forms which in my head was very simple and easy. I didn’t set out to be using web forms at the beginning what I wanted to use was an MVC model by having them all on a separate project. When I started to realise how easy web forms was becoming I didn’t think it was worth a Software Engineering degree, the reason for this was because we had an assignment that involved web forms. This sparked a lightbulb in my head and was think is my project worth the degree I am doing? I never intended to use web forms but it made life so much easier and simple. So I wanted to use Angular JS to do all the view side of the application then use ASP.NET in order to do the rest. This fails as I could not get my head around Angular JS and didn’t know how to use it well, I wish that I didn’t give up on the idea because I feel like if I did use it the GUI would look incredible better than it does right now.

This was not all a good experience there was some bad moments within this process, one of biggest bad experience I had with this was when it came to requesting the shift swap, I could not get my head around how to get a specific row back to the database. When looking online I saw that people were using different grid views then I was using, the ones online was using a grid view but I was using a data grid view. So it came to the moment where I did not know how to do it and neither did some of the peers that I was asking, I went back and looked at the tutorials online and started using a grid view just like online but even that came with errors. At this point I did not know what to do but luckily I found a good tutorial that showed it in a data grid view and all worked perfectly, the bad thing about this was that I spent way too long trying to fix this and it approximately took around three to four weeks to fix.

Throughout this whole project experience and looking back at it now that it is done, I feel like I could have challenged myself a lot more. I felt like a gave up on a lot of ideas that could have worked if I put more time and effort into this. At the beginning what I should have done was to look at the option I can use and start experimenting with them to see what would give me the best result.

One big struggle was the errors what I did about the errors was try and fix them some of them I could do and was just logical/syntax errors that I was making. However, if I could not fix the error on my own I would need to go see a lecturer who was available to try and help me to over this error, some errors took longer to fix then other but these errors needed to be fixed before I could proceed on with the next stage.

Looking back the my MoSCoW analysis, I have been able to do most the ‘must have’ requirements and the ‘should have’ requirement. None of the ‘would have’ requirement mostly due to the time we had for the whole project.

The idea as a whole would have been a real good value for the small business who heavily have to rely on paper rota over a online rota. This will allow the employees to swap shift with their fellow employees who have the same amount of hours as they do, be able to have the choice to either accept or decline the shift that is requested.

Seeing the project now that it has been complete and to see that not all of the objectives were set out was complete. I would say only half of the objectives were met and complete to a good standard and that passed all the test cases that were carried out. When I look back at the project, I didn’t priorities the features that are important to the application if I would have been more strict with myself and set better timeframe about each feature maybe all the objectives could be met.

Currently the system is at the bare minimum, it will allow the user to be able to login to the system and it will identify if they are a manager or a shift leader. Once logged in the user will be able to see links to various other pages such as, rota, shift choice and a request page. If shift leader or a manager they will see all of the pages and another page which will be the availability page. The rota will show on a grid for the user to see, the shift choices will be shown with only the people with the same amount of hours as the user does. The employee will have the ability to request that shift swap and it will appear in the request page. You are able to accept it and decline it from that page, however when you accept it, it will then be added to the rota page which leads to duplication. I plan on fixing this in the future works section. The decline button so far does nothing only a pop up button will appear saying reject.

## Project Evaluation

Looking back the my MoSCoW analysis, I have been able to do most the ‘must have’ requirements and the ‘should have’ requirement. None of the ‘would have’ requirement mostly due to the time we had for the whole project.

The idea as a whole would have been a real good value for the small business who heavily have to rely on paper rota over a online rota. This will allow the employees to swap shift with their fellow employees who have the same amount of hours as they do, be able to have the choice to either accept or decline the shift that is requested.

Seeing the project now that it has been complete and to see that not all of the objectives were set out was complete. I would say only half of the objectives were met and complete to a good standard and that passed all the test cases that were carried out. When I look back at the project, I didn’t priorities the features that are important to the application if I would have been more strict with myself and set better timeframe about each feature maybe all the objectives could be met.

Currently the system is at the bare minimum, it will allow the user to be able to login to the system and it will identify if they are a manager or a shift leader. Once logged in the user will be able to see links to various other pages such as, rota, shift choice and a request page. If shift leader or a manager they will see all of the pages and another page which will be the availability page. The rota will show on a grid for the user to see, the shift choices will be shown with only the people with the same amount of hours as the user does. The employee will have the ability to request that shift swap and it will appear in the request page. You are able to accept it and decline it from that page, however when you accept it, it will then be added to the rota page which leads to duplication. I plan on fixing this in the future works section. The decline button so far does nothing only a pop up button will appear saying reject.

## Applicability of Findings to the Commercial World

One of the big factors that I wanted to make was to switch from a paper based rota to then go to an online rota. The way that my system is doing this is by showing the rota to the employee so it will hit the main objective of going from a paper based rota to an online one. The system will allow for shift swapping which was the reason for creating this system, it will allow the user to request a shift swap and take it another page to accept or decline. It will help employee to swap shifts efficiently and effectively by using an online system, manager and other employee will be able to see the shift swap happening.

The money factor would come into this quite a lot as when you are using paper based rota is does not really cost a lot to do other then the managers time. When they switch they would need to take into consideration the cost of system and training the employees to use the system correctly and affectively. This would cost a lot but I feel like it would save a lot of money in the long run for businesses and will only cost them a lot at the beginning of getting the system. It would save them a lot of time and would help the business to run well by having the system in place. It will help with employee requesting shift swap online so other people can see and other people can be informed about it not just the two employees.

The overall system is aimed to help the management process to stop them to keep writing up paper rotas week after week. Instead of just sitting down once a week and writing out the rota for the following week, this system would be work much better as you can just manual enter the rota for a number of weeks. By typing the rota it would be a lot easier as for the management because they can see their employees availability and would not make mistakes. It is easy to make mistakes when you are working with a paper rota so by having an online system it will the managers life a whole lot easier.

## Conclusions

The project that I currently have has shown a good shift swapping application prototype and it has the potential of being a good shift swapping application. However if it was given more time it would need to be much more robust, it could be possible to extend it to a mobile version.

There could be a big feature if we would consider other factors, for example what if we need extra help from another employee that works at a different store? If there was a way of getting around that could work because the employee that is helping out will not be on the rota and will not be getting paid due to not being on the rota.

Having a shift swapping application, which gives the employee the ability to look at their shifts and be able to swap online would mean that there is a potential to reduce the amount of paper wastage.

The database will hold all of the personal information that could be vulnerable information that could be leaked. I would need to ensure all the information is secure from any attacks from hackers.

## Future Work

If more time was given I would have loved to be able add the feature that didn’t make it in the final cut. These would have been permission from the employee to permanently swap a shift, the way I would do this would be similar to when you would request a shift swap. Instead, of requesting it there could be a button that would permanently swap the shift so when you clicked on that button it would then notify the manager about the permanent shift swap. Before you can click on the permanently swap button the two employees will need to agree on it first before requesting the permanent shift swap. When the manager gets that notification they have the choice to accept it which will then change to rota for the following week, however, if it is declined then the rota would stay the same. The employees will then be notified if it has been declined or accepted.

Another feature that unfortunately didn’t make the implementation due to poor time management skills is request to swap a shift for a number of weeks. The way that I intended to implement this was when you requested a shift swap, a drop down box would appear with the amount of week. The week would only go up to four weeks the reason why it would only go up to four weeks is because if it became more then that it would become a permanent shift swap. The request page would show the other employee how many weeks the shift swap is for. When the user wants to swap the shift if it is accepted or declined the manager will be notified about the event occurring.

As the accept and decline feature is almost there all that it needs is done more time on it. When the employees accepts the shift it should alter the rota and insert the new shift into the rota. Right now it just inserts it into the rota but then there is duplicates. The way that this would be fixed is to look into the rota and have a SQL statement which could read if the StaffID is already in the rota then change it if requested and switch it with the other employee.

For future, I would want to secure the system and make it robust. Currently the password and the username are just stored in plain text so what we would need to do is hash the passwords. By hashing the password it would secure the system and to make sure that we give the users strong password. So far, this is all within the database what I could do in the future is add a registration page, so that the user has the ability to choose their passwords and ensure they are strong and hashed. The reason to keep the system secure is the fact that we will be storing personal information about the employee and the employees would want to rely on us to make sure their information is safe.

What I originally would have liked to have done with the project is to once have the web application side up and running to then make a mobile web application. The reason why I would love to have had a mobile application version is because everyone now has a smartphone “the number of users accessing the Internet on their smartphones has more than doubled in the past 5 years, as has the number of users downloading and using mobile apps” (Agrimbau, 2019). This proves that mobile application are advancing rapidly and I would want my application to be relevant.

The business would not need to completely switch over to online immediately, what we could do is with our mobile application, we could have a feature inside the application that allows the manager to take a picture paper rotas. The picture will then be generated the rota and put in all online just by the picture. This has been done before because in the application call Shyft they use this feature and would be good to take a small step from paper to online. The application Shyft has been proven to be a success for such a small feature and has received a 4.5 star out of 5 (Myshyft.com, 2019).

References

Agrimbau, T. (2019). *Developing Mobile Web Applications: When, Why, and How*. [online] Toptal Engineering Blog. Available at: https://www.toptal.com/android/developing-mobile-web-apps-when-why-and-how [Accessed 14 Apr. 2019].

Bhagat, V. (2017). *ASP.Net vs PHP: Which is the Best?*. [online] PixelCrayons. Available at: https://www.pixelcrayons.com/blog/web/php-vs-asp-net-how-to-choose-the-right-one/ [Accessed 10 Apr. 2019].

Brandon, J. (2019). *Why paper still rules the enterprise*. [online] CIO. Available at: https://www.cio.com/article/3025928/why-paper-still-rules-the-enterprise.html [Accessed 3 Apr. 2019].

Cleland, J., Caldow, J. and Ryan, D. (2007). A qualitative study of the attitudes of patients and staff to the use of mobile phone technology for recording and gathering asthma data. *Journal of Telemedicine and Telecare*, 13(2), pp.85-89.

Cohn, M. (2004). *User stories applied*. Addison-Wesley Professional, pp.4-5.

Chadwick, J., Snyder, T. and Panda, H. (2012). *Programming ASP.NET MVC 4*. Beijing: O'Reilly, pp.416-417.

Deacon, J. (2011). Model-View-Controller (MVC) Architecture. *Computer Systems Development, Consulting & Training*.

Duong, T. and Rizzo, J. (2011). Cryptography in the Web: The Case of Cryptographic Design Flaws in ASP.NET. *2011 IEEE Symposium on Security and Privacy*.

Fadilpašić, S. (2019). *Some companies still use paper and it hurts*. [online] IT Pro Portal. Available at: https://www.itproportal.com/news/some-companies-still-use-paper-and-it-hurts/ [Accessed 3 Apr. 2019].

Foundation, A. (2019). *Welcome! - The Apache HTTP Server Project*. [online] Httpd.apache.org. Available at: https://httpd.apache.org/ [Accessed 27 Mar. 2019].

Galloway, J., Haack, P., Wilson, B. and Allen, S. (2011). *Professional ASP.NET MVC 3*. Indianapolis, Ind.: John Wiley & Sons.

Gilmore, W. (2009). *Easy PHP websites*. Columbus, Oh.: W. J. Gilmore.

Gopalaswamy, R. and Desikan, S. (2009). *Software Testing*. Don Mills: Pearson Education Canada, pp.48-67.

Jackson, D. and Hooff, D. (2012). Understanding the Factors that Effect Information Overload and Miscommunication within the Workplace. *Journal of Emerging Trends in Computing and Information Sciences*, 3(8), p.1240.

Kambalyal, C. (2017). 3-Tier Architecture. *Computing 2 and 3 Tier Architecture*, 2(1), pp.11-12.

Liberty, J. and Hurwitz, D. (2004). *Programming .NET Windows applications*. Beijing: O'Reilly, pp.3 - 5.

Liu, X., Heo, J., Sha, L. and Zhu, X. (2008). Queueing-Model-Based Adaptive Control of Multi-Tiered Web Applications. *IEEE Transactions on Network and Service Management*, 5(3), pp.157-167.

MacDonald, M. and Szpuszta, M. (2006). *Pro ASP.NET 2.0 in C♯ 2005*. Berkeley, CA: Apress, pp.23-25.

Mcarthur, K. (2008). *Pro PHP*. 1st ed. Dordrecht: Springer.

Mitchell, S., Walther, S., Seven, D., Mack, D., Payne, C., Anders, B., Nathan, A. and Wahlin, D. (2002). *ASP.NET*. 1st ed. Indianapolis, Ind.: Sams, pp.3-5.

Mutton, P. (2019). *Internet Research, Anti-Phishing and PCI Security Services | Netcraft*. [online] News.netcraft.com. Available at: https://news.netcraft.com/ [Accessed 27 Mar. 2019].

Myshyft.com. (2019). *Empower your teams and drive labor efficiency - Home - Shyft*. [online] Available at: https://www.myshyft.com/ [Accessed 14 Apr. 2019].

Nagappan, R., Skoczyla, R. and Sriganesh, R. (2004). *Developing Java Web Services: Architecting and Developing Secure Web Services Using Java*. Beijing: Qing hua da xue chu ban she, pp.69-74.

Pop, D. and Altar, A. (2014). Designing an MVC Model for Rapid Web Application Development. *Procedia Engineering*, 69, pp.1172-1179.

Pilone, D. and Pitman, N. (2005). *UML 2.0 in a nutshell*. Sebastopol, CA: O'Reilly Media, pp.4-6.

Prosise, J. (2002). *Programming Microsoft .NET*. Redmond, Wash: Microsoft Press, pp.25-29.

Ritchie, C. (2002). *Relational database principles*. 2nd ed. London: Cengage Learning EMEA, pp.8-9.

Sanderson, S. (2011). *Pro ASP.NET MVC Framework*. 3rd ed. Apress.

Schuh, P. (2005) *'Integrating agile development in the real world',* 1st edition, Hingham, Mass: Charles River Media.

Sendiang, M., Alelo, M. and Ponggawa, V. (2017). The Development of Accreditation App with MVC Architectural Pattern. *International Journal of Computer Applications*, 176(5), pp.28-32.

Shukla, D. (2017). *How ASP.NET is the Best Programming Language for the Internet of Things - DEV Community 👩‍💻👨‍💻*. [online] Dev.to. Available at: https://dev.to/dhrumitshukla/how-aspnet-is-the-best-programming-language-for-the-internet-of-things-4hi [Accessed 10 Apr. 2019].

Singh, N. and Gupta, C. (2014). *Relational Database Management Systems*. Abhishek Publications, pp.22-24.

Stellman, A. and Greene, J. (2014). *Learning Agile*. O'Reilly Media, Inc, p.355.

Stratmann, E., Ousterhout, J. and Madan, S. (2011). Integrating Long Polling with an MVC Web Framework. *USENIX Conference on Web Application Development*, 2.

Sumathi, S. and Esakkirajan, S. (2010). *Fundamentals of relational database management systems*. 11th ed. Berlin: Springer Science & Business Media, p.728.

Talukder, A. and Yavagal, R. (2005). *Mobile computing*. New Delhi: Tata McGraw-Hill, pp.37-39.

Whitehorn, M. and Marklyn, B. (2007). *Inside relational databases with examples in Access*. 2nd ed. London: Springer, p.202.

Williams, H. and Lane, D. (2004). *Web Database Applications with PHP and MySQL*. 2nd ed. O'Reilly Media, Inc.: O'Reilly Media, pp.2-4.

Appendix 1 – Project Proposal

# Computing Degree Project Proposal

**Name: Raheem Hussain Course: Software Engineering**

**Discussed with (lecturer): Nick Mitchell**

## Current Modules (and previous modules if Computing or direct entrant)

CO2401 – Software Development

CO2402 – Advanced Programming with C++

CO2403 – Professional Skills

CO2411 – Software Engineering Practices

CO2508 – Computer Security

CO2509 – Mobile Computing

CO2602 – Agile Systems Design

## The Project Title

Switch Up

## Project Context

I plan on making an application for my work place so that colleagues are able to view the rota and be able to shift swap with someone. The place that I would be doing this for would be Tesco Express as they only currently have a paper version.

One of the main reasons why I would like to do this idea is because we have been on paper versions for a really long time, I would like to use the skills that I have learned to make an application to have the rota on. It is important that we schedule our staff as we will need to get rid of the paper versions, “Optimised staff schedules can provide enormous benefits” (Ernst et al., 2004). Another reason for this project is the big problem that we have with the current paper version and that is people shift swapping, now we do not have a problem with people changing shifts as long as they are covered. Now the problem comes with people discuss it on platform such as Facebook to swap shift however, they cannot change the paper version. The way that my project solves this is by having the ability to swap shifts with someone within the application and it would update it on the main rota.

Another problem that occurs is overtime, now the manager will have to text everyone to see who can cover a shift and get some extra work in. The way that this application would help is that the manager could add in overtime on a separate database and the employees can take them if they are free to do that shift. Once they will confirm they can do that shift it will add it to the rota in bold so that the manager will know it is overtime.

Finally, I would like to have an application rota because smartphones are becoming the future, many people are now on their phones and by having this application I would be thinking ahead. Now this project would be on a smart phone and everyone would be able to access this as most people have a smart phone (Wilmer, Sherman and Chein, 2017). It will be future proof as they will not need to have a big update for a while as I would have done the main framework from Tesco.

## Specific Objectives

* The main objective that I would like to get out of the project is to scrap the paper base rota and have it online. This would help everyone as if they needed an update rota on the application and you would be able to view it at anytime
* The outcome of this project would be to have the ability to be able to swap shift with someone and for it to update on the main rota. This would help because the manager will be able to see who is in at what time instead of it being over Facebook and the manger not knowing about it until the last minute.
* The objective I would like achieve would be to give the manager the ability to put over time on the database and it give the employees a notification that there is overtime available for them to get. This would help the manager instead of texting every employee to see whether or not they can do the overtime
* The final objective that I would like to achieve is have the employees give availability and the application would be able to see when people are free to work. So if someone wanted to swap a shift with someone they will be able to see if they were available before swapping a shift .

## Potential Ethical or Legal Issues

* A big legal issue is the fact that I would need everyone’s rota information, not everyone would feel comfortable to share when they are working, the manager will be the only one with that personal information. So what I would need to do is to talk to all the employees and ask them if it was okay for me to see their working hours, I would tell them what I needed it for so that they feel comfortable giving me the information
* As well as this I would also need to consider the age. Not everyone will have a smart phone in Tesco, the way that I would get past this would be to have a tablet or an iPad in the staff room so that when the people who do not have a smart phone can see the rota on the iPad or tablet. We would also need to make sure that everyone is trained well on the application (Cleland, Caldow and Ryan, 2007), this will take some time as the target audience is not specific. Everyone will need to be trained on the application which might take some time from the project.
* The design aspect would also come into this, I would need to make sure that everything on the application is easy to see, I would want everyone to be able to see the buttons and the main reason would be the ability to reach the buttons (Jun Gong, 2004) As it would be on a hand held device I would like to ensure that everyone will be able to reach the buttons that are on the screen.

## Resources

After doing some research about this project idea there is some out there that are similar to the project. One of the big application that did pop up when I was doing my research was Rota Ville (Rotaville.com, 2018), this help the employees to sort their shift out and are able to adjust their shifts to the hours that suit them. It is on many platforms like IOS and Android two of the most popular operating systems. One of the good things about this is that it also has employees availability which shows when the employees are available to work, I could use this in my project as an idea. The way that I would use it is to have each employee to give their availability times to the manager so that they know when the employees can be free to work or do some overtime.

I have also seen that some application make it easy to shift swap with someone (Su, 2018), the application that they use is called shift smart which allow the employees to swap between them. Now this is something that I could get inspiration from, as it is something that I could use in my application.

I have researched into what software I would like to use for this project:

* Xamarin
* Code Over

## Potential Commercial Considerations - Estimated costs and benefits

If this was done commercially I would need to consider any costs that could come from then project. The way that I would put cost on this project would be to have it for a small business first as Tesco Express holds a max of 30 people. I would see if it works well in my workplace first and ensure that it is bug free. Once that is done I would then offer it to business who have 30 employees however, I would not stop their I would make it bigger and have it be up to 500 people. This will allow different business to see have the application and use it. The more employees the more money that the project would make.

The timing of this project will all depends on what business it would be for, the original plan would be to have it for 30 employees. This would at least take me a year or 8 months to complete then I would need to make sure that it is bug free and test it before releasing it out. The feature will include overtime added, swapping shifts and having availability within the application.

## Proposed Approach

The first step would be to decide on which language and which software that I will need to use, with research that should take a week to do and to come up with a decision.

Once that is done then I will need to start getting all the information together to make the application. For example, rota time and everyone’s availability. Then, I will need to start to write up a database to store all the information. A few months.

Have a prototype to be able to see if everything work. Once that is done then have the ability to swap shifts with someone and it work perfectly. Which should take another few months.

If that’s work perfectly then move on to allow overtime and get a notification when the manager puts it up. Another few weeks to do this.

Finally, have the availability written done on the application so that they can see who is busy on what days. A few more months.

Appendix 2 – Technical Plan

**Computing Project Technical Plan**

**Name:** Raheem Hussain **Mode**: Full Time

**Course:** Software Engineering **Supervisor:** Li Guo

**Title**

Switch Up

**Summary**

My project idea is for small business that currently has a paper rota. Small Business are relying heavily on having paper rotas each week so that the employees know when they are working. The reason why I would like to change this is that technology is advancing massively and I would like to make sure that even small businesses are keeping up to date with everyone else.

I have also seen that some application make it easy to shift swap with someone (Su, 2018), the application that they use is called shift smart which allow the employees to swap between them. Now this is something that I could get inspiration from, as it is something that I could use in my application.

Now this project would be on a smart phone and everyone would be able to access this as most people have a smart phone (Wilmer, Sherman and Chein, 2017). It will be future proof as they will not need to have a big update for a while as I would have done the main framework for small business.

The problem when working in a small business people sometimes like to swap shifts with one another, however, the way that they do this is through Facebook. I feel that this is an unprofessional way of swapping shift as some people tend to forget which shifts they have swapped, also the manager is never informed so that manager would expect the employee in but the manager does not know they have swapped shifts. Another issue with having a paper rota is allocating overtime, the current problem that we have with overtime is that we need to send a text to all the employees that are available to see who can do the shift.

My idea to resolve this issue is to have a web-based application that you will be able to have on your phone or on the web. On the application, you will be able to see the rota instead of having it paper-based, you will also be able to swap shifts through the application, the benefit of this is so that everyone can see who has swapper with who and see which employee is supposed to be in. The way that I would solve the overtime issue would be to have a subsection within the application where the manager would be able to add the overtime and the other employees will be able to see it and either accept the overtime or decline.

The methodology that I will be using for this project is Agile, the reason for my decision is because I will need to first get the requirements then design, implement and test and I would keep going until all the requirements are met. I will need to keep testing this to ensure that all the data is all there and up to date with the rest of the system.

The difficult aspect of this whole project it would be that everyone will not have the same dashboard, for example, the manager will be the only one to allocate overtime, the colleagues will only be able to accept and decline shift swaps. The employees will not be able to allocate overtime that will only be a feature for the manager only and the manager will also be able to see availability for the employees.

**Deliverables**

I will be delivering a well-tested web application that will be able to swap shifts through the application. This will be the core part of my project and can not be done without it, this will be delivered on the 10th of December. This is a "Must Have" feature that will need to be done by the first delivery. The "Could Have" can also be included but the main goal is to ensure that the "Must Have" are done. There will also be done documentation, feedback on the project and the source code. Any additional features will be added on after the first delivery.

**Constraints**

One of the biggest constraints that I will be facing is the deadlines that we have been set, the first big one is the first delivery which is on the 10th of December. So by then, I should have the core of my project done, it should have also been tested for bugs. I will also be receiving feedback after my first delivery so I will need to make sure that all feedback is positive and if it is negative to make it a positive by fixing any issues that the user has. This will all need to be done within the deadline dates.

Another big constraints that I will be facing is to learn a whole new software program as I have not used Angualr before. The way that I am going to be overcoming this will be to look at what Angular can do and try to see if I can try and start my project from the tutorials that I will be doing online.

**Key Problems**

The main obstacle that I will be facing is that this project is a totally new thing as there is nothing currently in place at my workplace. So I would definitely need to consider the user need and capability, there are three different types of user I will have using the system. One would be the manager who will be able to see the rota, the availability of the employees and be allowed to allocate overtime through the web application. Another user type would be the shift leader, they are very similar to the manager, however, they will not be able to see availability for the employees. The final user type that I will need to consider is the customer assistant, now the only feature that they will be able to see is the ability to swap shifts with the other customers assistant.

By having all these user types I would also need to consider the database and store multiple people on a database. The user should be able to communicate with one another in order to swap shifts. The database that I will be using is MySQL, this works very well with Angular, and I do a little bit of experience using MySQL.

There is another approach that can be taken when saving the data. The normal approach that I was going to take is to have a save button when the user makes changes to the rota. Another option that the user could have is when the user saves every input that the user is putting in.

**System and Work Outline**

The first approach that I will be taking in order to accomplish this project would be to gather the requirements for each of the user and see how they will be using the system. I will need to talk to each of the different types and users to see what they would like to see within the system. In order to this, I would need to have several meetings with the different users and look at their requirements to see if it is achievable.

Once I have met up with each of the users I would then need to go away and start making up some use case description, this would help with identifying how they will be using it and get the structure for the project. I will also need to do an entity relationship diagram the reason for doing this is to help me with the database, by doing the diagram I would be able to see where the relationship is and see what information is shared with the users.

When this is all done I can then start on the design aspect of the program, I will base this off the meeting with the users and entity diagram. I will draw up some prototypes and then again go and meet the users to see if all of the requirements were met correctly. The design aspect of this project is very important as this would be an original project within the company so it would need to suit everyone's need within the company.

Once I know that the users are happy with the designs then I would need to start the implementation process, I would first start by implementing one feature at a time and test them along the way to ensure that the program is bug-free. By doing one feature at a time it would help because when one is done and I know that there are no bugs within the program I would then be able to continue the process. The reason why this method would work well is that if something goes wrong I will then be able to go back to a feature that fully works. The feature is the most important is the shift swap feature as that is the core of the project, by doing this first it would help with my first delivery.

Once every feature for all the users is done and tested for bugs, I would then need to make sure that they can all feature can interact with one another to make a full program. Each user should be able to communicate with the other and swap shifts with one another.

One method to get through this would be the MoSCoW, this will help with the “Must Have” of the project and it will ensure me to have them done by the first delivery. This method would help with how many features I want to include in the project as I can think about other feature that "Could" be added to make the project much better.

**1. Swap-Shifts Feature**

- **Must-Have**

- Must be able to communicate with other users to be able to swap shifts

- Must be able to accept and decline shifts

- Must be able to see overtime page that has been set up by the manager

-**Could Have**

- You could drop a shift and that could go onto the manager side to put into the overtime section

**2. Overtime Feature**

**-Must-Have**

- The manager must be able to allocate overtime on the application

- Other users should have the option to accept the overtime

**-Could Have**

- Allow the customer assistant to drop their shifts and it is able to be on the overtime feature section

**3. Availability**

**-Must-Have**

- The manager would be able to see which employees are available

**- Could Have**

- Only be able to see availability for the employees that are not working the shift already

All the information about the different user types with be stored inside a database. The database software that I will be using is SQLite, the reason for me deciding to use this is that I have had some past experience with the software and it is compatible with Xamarin and Angular.

The employees' ID will be saved on to the database followed with their passwords. Each employee will have their own login for the application in order to see their shifts. The shifts will also be stored within the database it will give you a list of the employee's shifts everyone will have different shifts.

The communication between the database will also be a key factor when making the database. The communication will come when people request to change shifts, and when the employee accepts it should change for them on the database and show the swapper shift. The database will need to be updated once an employee accepts another shift.

**Risk Analysis**

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Severity** | **Likelihood** | **Action** |
| A huge bug that requires a lot of attention to fix | 8 – A critical bug that will consume a lot of my time | 5 | To make a plan and structure in case this does happen |
| Testing does not go the way I expected | 6 – Test goes wrong and having to go back to ensure bug free | 6 | Test along the way, so once it works test to ensure it does what it needs to do |
| Too ambious | 5 - Need to be able to stick with the must-haves before moving on to the other features | 4 | Follow the Moscow method and do the must have first |
| If the client changes the requirements last minute | 7 – It would make a big impact on the project if the requirement changed towards the end of the project | 2 | Need to ensure that a good communication between me and the client at all times to ensure no new requirements |
| Falling ill | 6 – Depending how ill I get it could lead to some serious time off | 2 | Plan with plenty of time |
| Database Failure | 6 – MySQL might lose the connection between the users and not be able to communicate | 4 | Test ahead of time to keep the database up to date and ensure it is working well with the application |
| User Interface not good | 6 – if the user interface screen does not suit the user needs and does not like the layout of the application | 5 | Ensure that I discuss design aspect of the application with the client before showing the final product |
| Loss of data | 10 – If there is a loss of data then the project would not be finished or delivered | 1 | Must make sure that I have kept a back up stored on the cloud. Every feature that is done will be saved in more then one place. |

**Options**

Following the plan, there are different ways in order to do this. The current options that I am debating between are whether or not to go with Xamarin or AngularJS.

The advantage of using Xamarin:

The way that Xamarin works with the Visual Studio IDE, I am very familiar with using Visual Studio IDE as I have used it for over 2 years of university. By using Xamarin it is also an Andriod application and I have previously made an Andriod Application however that was in Android Studio. By using Xamarin it would help to launch the application cross-platform between iPhone and Android and would be good to cover all the target audiences.

The disadvantages of using Xamarin

Having it cross-platform does help, however, I have an iPhone and to be able to develop for an iPhone you will need a Mac. I do not have a Mac as I am programming on a Windows Laptop, the only other option would be to go out and buy an android for the project.

The advantage of AngularJs

One of the big advantages of using AngularJs is that it is all web-based. So you would not need a smartphone in order to access the application, all you will need to do is have access to the internet.

This will also help when it comes to debugging as it is done instantly whereas with Xamarin I would need to wait a while for the emulator to load up. This would come in handy when testing comes and I would be able to see the result of the code immediately.

The disadvantage of AngularJs

The only big disadvantage of using AngularJs is the fact that it is not as familiar to me. So I would need to learn how it works in order to do the application, I would need to start looking at tutorials to see how it works and to make sure that the project can be done within AngularJs.

**Potential Ethical or Legal Issues**

A big legal issue is the fact that I would need everyone’s rota information, not everyone would feel comfortable to share when they are working, the manager will be the only one with that personal information. So what I would need to do is to talk to all the employees and ask them if it was okay for me to see their working hours, I would tell them what I needed it for so that they feel comfortable giving me the information

As well as this I would also need to consider the age. Not everyone will have a smart phone in the workplace, the way that I would get past this would be to have a tablet or an iPad in the staff room so that when the people who do not have a smart phone can see the rota on the iPad or tablet. We would also need to make sure that everyone is trained well on the application (Cleland, Caldow and Ryan, 2007), this will take some time as the target audience is not specific. Everyone will need to be trained on the application which might take some time from the project.

**Commercial Analysis**

If this was done commercially I would need to consider any costs that could come from the project. The way that I would put a cost on this project would be to have it for a small business first as Tesco Express holds a max of 20 people. I would see if it works well in my workplace first and ensures that it is bug-free. The time that it would take it 7 months to make the app for a small business and for that time the cost would be £15,000. The Software side of it would be a yearly licence which would be £1,000 for only a year to develop with.

The timing of this project will all depend on what business it would be for, the original plan would be to have it for 30 employees. This would at least take me a year or 8 months to complete then I would need to make sure that it is bug-free and test it before releasing it out. The feature will include overtime added, swapping shifts and having availability within the application.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Factor name** | **Description** | **Is this a cost or a benefit** | **Estimated Amount** | **Estimate of when paid** |
| AngularJS IDE | This will be the IDE to develop the application | Cost | £ 99.00p.a. \* | Before starting the implementation |
| Working Hours | Calculation would be based on £20 an hour and working time would be around 500 hours | Cost | £15,000 | After Delpoyment |
| Selling the application to other small branches | This would be to possible to sell the application to the branches that will need it | Benefit | N/A | N/A |

**Employability Contribution**

For my final year project, I decided to make a web application for small business that uses paper-based rota. The way that I did this was by using AngualrJS, I had usernames and password for all the user and they were able to see what shifts they were doing. The complicated aspect came when you were allowed to swap a shift with another employee, the way that this would work is you would select the shift that you wanted to swap and the other employee will need to accept or decline. Once accept it would go to the database and change it for the user.

I have gained a lot of skills while making the application, before this, I had never used AngularJS. So I needed to go do some research about the software, this showed that I was willing to learn a new software and a new language.

Appendix 3 – Title of Appendix

You may have one or more appendices containing detail, bulky or reference material that is relevant though supplementary to the main text: perhaps additional specifications, tables or diagrams that would distract the reader if placed in the main part of the report. Make sure that you place appropriate cross-references in the main text to direct the reader to the relevant appendices.

Do not blindly include all of your code in the appendix or the body. Only include the parts you refer to in the report. You can put those parts either in the appendix or in the body (e.g. in the “Implementation” part).