Venues Assistant Staff Portal (VASP)

By Karl Emlyn James Latham

Student Number: C1335031

CM3203 – One Semester Individual Project

Module Credits: 40

Supervisor: Dr Andrew C Jones

Moderator: Dr Paul L Rosin

School of Computer Science and Informatics

Cardiff University

May 2016

Abstract

The aim of the Venues Assistant Staff Portal (VASP) project is to design, develop, and implement a web based employee shift booking, and time clock system for Cardiff University Student's Union Venues Team. The web based system will manage employees booking shifts at a time convenient to them, and recording the time at which they begin and end a shift. It will also allow the Management Team to place available shifts online to be booked, and manage the payment to employees for work which they have done.

Acknowledgements

I would like to express my thanks to my project supervisor Dr Andrew C Jones for his continued help and support throughout my work on this project. I would also like to thank the Venues Management Team at Cardiff University Students' Union for allowing me to undertake this project as part of my studies. Also to my friends and family who have supported me throughout my time at Cardiff University, especially during my final year.

Contents

1 Introduction

Venues Assistant Staff Portal; otherwise referred to as VASP, is a system designed to ease and improve the shift and payroll management of Venues Staff who are employed by Cardiff University's Student's Union. VASP is designed to replace the Students' Unions current system Central Services.

Based entirely within a web format, it is accessible across a wide range of devices and operating systems, ensuring it meets the needs of both the business, and the staff who use it on a daily basis.

Personal Involvement

Having worked as a Team Leader at the Students' Union since September 2014 I had personally experienced the limitations of the Central Services system, and the frustration and extra work load it seemed to place upon everyone.

A conversation in work with a member of the Management Team inspired me to take on this project as part of my final year studies.

Aims and Goals

VASP aims to provide a solution that fits with the needs of the venues department at Cardiff University's Student's Union, the way the business functions, and the flexibility afforded to the staff who work for them, makes an off the shelf product an inappropriate business choice. The requirements of the business are; as far as can be reasonably determined, unique to a Student's Union environment. This is mainly governed by the company's decision to allow staff to choose the shifts they wish to work from a pool of ones which are available, rather than the accepted norm of giving staff fixed shifts, or asking for staffs availability and then giving shifts out based on that.

Currently the Venues Department use a web based system (Central Services), constructed by a member of their IT support staff around 10 years ago, to allow staff to login and pick the shifts they wish to work. This system has many limitations, meaning it no longer meets the needs of those who use it. Central Services has never been kept up to date with advancements in the framework it is based on (PHP), or with the needs of the business, though it has continued to be used. Although it does provide the basic functionality that the Venues Department requires, it can be at times resource intensive requiring salaried staff to spend time intervening, and manually inputting information that could be automated, or streamlined.

With the above in mind, this project will aim to assess the current system, and procedures that are in place, from the perspective of the Venues Department Management, the Venues Department Staff, and from a technical point of view. Then formulate a design for a new system which will meet the essential needs of the business, and look at how optional needs can be accommodated, based on their importance, or ease of implementation. This design will then be approved by the Venues Department Management before it is implemented.

The implementation of the system will be subject to an evaluation to determine how well it has met with the end users requirements, this will involve meeting with the different groups

of end users, and conducting evaluation exercises with them in various formats. The evaluation exercises will also form a part of the end user testing of the system.

Evaluating the system will lead onto a conclusion of how well the system met with the end users requirements, and a plan for future implementations of bug fixes, changes to features brought up during evaluations, and also additional features which could be implemented.

From a personal perspective I wish to improve my knowledge, and practical experience of dealing with an end user, determining the requirements they have, and following this through into creating a working system, which meets these requirements. This will require me to do personal research in order to determine the best way, and method, to use to approach this project.

Audience

The audience for this project is split broadly into two groups, the Venues Department Management; who act effectively as the systems administrator group, and the Venues Department Staff, who act as the users of the system, this is not to say that a person must be specifically in either group, they could be in both.

The Venues Department Management need to be able to access the system and perform the day to day management of the system, adding shifts to the available pool, running payroll, and absence reports, and correcting mistakes made by employees during the clocking in and out process. They also have full access to the system as a member of Venues Department Staff, and can be assigned to, and book shifts using the system.

Venues Department Staff need to be able to access the system, have access to a list of the shifts which are available to book, and the basic information about each shift; what time it starts, what time it ends, and the date. They need to be able to drop shifts they have booked, or exchange them with a colleague. They need to be able to clock in and out for a shift, and to view this information after the event, and amend it if it is wrong.

Higher level management above those in the Venues Department may be considered as an end user of the system in so far as they will be exposed to the reports generated by the system about payroll, lateness, and absence.

Though the system is being designed for use by Cardiff University's Student's Union, it is important to consider that it could be easily redeployed to another organisation if they find it meets the needs of their business, and therefore the scope of the audience is not just confined to Cardiff University's Student's Union, and the implementation of the system needs to keep this in mind.

Approach

A modular approach to system design allows for a more structure approach both to developing, and to maintaining a system.

From a high level point of view, the system is split into a user interface, a back end database, and an interface between these two parts.

In order to have a more focussed approach to coding the system, each of these three parts would be further split into separate modules, with each module doing a specific task in the system. When it comes to debugging the system it means all of the code is well structured, and it is clear what each part of the code is supposed to do, and each part of the code can be compiled and run separately with test data values in order to ensure it is giving sensible outputs. When it comes to adding additional features to the program, it would be a case of building a new module, with some slight modifications to existing ones, for example making part of the UI module call a new part of the back end module Rather than heavily modifying an existing part of the code.

As a web based project the choice of languages is naturally limited to one which can be deployed in a web based environment, it is also limited in ensuring cross browser and platform compatibility. In order to take advantage in the latest developments in web based technologies and languages, I approach the language selection with a view to not supporting legacy web browsers. This decision was made through a survey of devices that people who worked at Cardiff University's Student's Union had access to in their daily life, and what browser these devices would support.

Previous experience in the PHP language motivated the author to select a PHP based approach to developing the website. PHP is a widely supported web based language, with support for a variety of database architectures, libraries, and code snippets are also widely available.

As mentioned above it is clear that a database management system will be needed to manage the back end data, and organise it in a structured format which can be accessed through the interface layer of the system. Due to the nature of the data a relational database management system was chosen over a none relational one. Due to the authors past experience, and In order to keep the system flexible; using an open source language, rather than one which needs a paid license, as well as one which is cross platform compatible; the server side of the system could be hosted on wither a Windows or Linux based server, the open source MySQL language was chosen.

To drive the functionality of the User Interface something more dynamic than HTML and CSS was required. Manipulating the Document Object Model on a large scale was not something the author had had previous experience in, to counteract this research was carried out in order to determine exactly what would be needed. After research into the available technologies, HTML5 mixed with Javascript was chosen.

Infrastructure required by the system is limited to a web server, running the appropriate versions of PHP and MySQL. Due to the nature of the system, and the critical nature of the business it would need to be regularly backed up, both on site and off site. This would require the web server to be fitted with the appropriate software in order to perform these backups.

The client side of the system is provided by each person that is accessing the system, either by using their own mobile device, or by using a computer at home. They also access the

system through computers within the workplace in order to record the time they start and end a shift.

In theory the system is expandable far in excess of the current needs of Cardiff University Student's Union, the amount of users who would currently be registered to use the system would be around 500, though it is not anticipated that all 500 registered users would attempt to access the system at the same time. Increasing the amount of users registered on the system to even ten times the current amount would only require the amount of server storage space available being sufficient to cope with the extra demand. There would be a point where the number of users attempting to access the system in parallel would begin to put a strain on the processing resources of the server. It would be worth exploring the options at that point to separate out parts of the database across multiple servers in order to faster parallel access.

Assumptions

In order to construct the system, a number of assumptions where made in order to streamline the design and implementation process.

It was assumed that users would be split broadly into two groups, managers, and "normal staff". Management would have full access to the administrative functions of the system, whilst at the same time they maintain access to booking their own shifts, shift swaps, and clocking in and out functions. Though this could lead to a conflict of interest, due to the trusted nature of the management position it is not felt that this would cause a problem. Normal staff have access only to their own shift booking, shift swaps, clocking in and out, and their own personal information, they have no access to administrative functions. Although this implies that the system is split only into two groups, each member of staff still has their own personal set of rank flags, which determine what sort of shifts they are allowed to have access to.

From an early point in the research and design into the project, it was seen that staff would be more comfortable moving to a new system if parts of it where similar to that of the existing system. As such it was decided early on to remain with a calendar style approach to booking the shift systems, with shifts already booked highlighted in some form, with shifts available to book highlighted in a different style.

The way in which the Venues Department Is structured relies on each area being supervised by a Team Leader, the role of Team Leader is considered a supervisory role that sits between management and the Venues Assistants who work day to day. Whereas a Venues Assistant may only work in one or two areas, a Team Leader is expected to be able to work in any area, and are trained to this standard. Though the Team Leaders form a smaller pool of staff, and have a higher number of shifts available per person, they tend to work a lot more shifts than Venues Assistants, and are used to fill vacant Venues Assistants shifts where needed. As such the system assumes that a Team Leader can "work down" and fill a Venues Assistant shift if it doesn't overlap with an available Team Leader shift.

Though Team Leaders hold a position of trust, it is felt that them being able to authorise pay roll adjustments may bring them into a conflict of interest, especially as they are only

employed on a casual, rather than a contractual basis. As such it is assumed that management are required to authorise any adjustments to clock in and out times, or payroll adjustments.

Summary of Key Outcomes

- Implement the ability for staff to book shifts from an available pool, based on their rank within the venues department; a staff member must be a Team Leader in order to book a Team Leader shift for example, as well as limiting them to booking a defined number of shifts within a defined time window.
    - Allow the person booking shifts to reserve the shifts for a defined period of time, and either confirm the booking before the time window expires, or release them back to the general pool for another person to book.
    - Allow management staff to create new shifts, and set the definition of the time windows given above.
    - Estimate the persons pay for that pay cycle based on the shifts they have booked.
- Implement the ability for staff to record the time at which they clock in, and clock out for a shift on the system, the method of entry of this information is to be decided based on the parameters given above.
    - Allow a staff member to select if it's a clock in, or a clock out they wish to record.
    - Alert the member of staff if they haven't clocked in or out correctly for a shift and allow them to correct this with management intervention.
- Generate pay roll reports based on the information obtained from staff recording there clock in, and clock outs. In a format which is compatible with the pay roll departments systems.
- Calculate on request the optimum positions for staff within the venue for a given shift. Based on a location being given an input of an optimum amount of staff to be on it at any one time, and load balancing this across all the locations in use at that time.

2 Background

Research Methodology

In order to better understand what exactly was required from the system, both for the Management Team and the needs of the business, but also to the student staff who use the system, I used a number of different methods, both formal and informal to gather information.

Formal meetings where arranged with the Venues Department Management Team, this allowed a formal setting to discuss and record what they felt they needed from the system, and a platform to formally agree specifications for the system.

Informal meetings happened incidentally with members of the Management Team; for example part way through a shift a problem relating to Central Services or Payroll arose and I commented on how VASP could be used to overcome the issue, leading to a discussion on what could be included within the system.

Informal conversations with my colleagues amongst the Student Staff happened regularly both inside and outside of work. These mainly related to the accessibility of the system and the layout, rather than functional requirements.

A more detailed analysis is provided within the Design and Specification section.

Software Development Methodology

Before commencing on further work into the project I felt it was important to establish what Software Development Model I would follow throughout the project, as this would give a guide as to how I would carry out each stage of the development.

As I had personal experience with using the Waterfall model approach to software development this was the method which I first considered, and I examined whether it would be practical to use for the deployment of this project. The Waterfall model itself has spawned several modified versions which attempt to counteract the perceived downsides of the traditional model, the traditional Waterfall model has five stages; Requirements, Design, Implementation, Verification, and Maintenance. The model strictly relies on starting at the Requirements stage, and moving through each stage in turn; 20% - 40% of the time is dedicated to the Requirements and Design stage, 30% - 40% to the Implementation and the rest of the time given to Verification and Maintenance. This model gives a clear outlook on achievable milestones in the development process, and aims to deliver a fixed set of goals to the client as decided on in the initial phase of the process. The downside of the Waterfall model is that the client, or end user does not have the opportunity to give input into the project during the development process other than right at the beginning, and right at the end during the acceptance testing. This can lead the client to changing his or her mind as what they envisioned during the requirements stage may not come to fruition as they thought, this leads to problems with needing to backtrack on what has been coded to make adjustments to suit the user's needs.

The second method which I considered was the Modified Waterfall model, born from the Iterative and Incremental development methodology. The model itself is similar to the

traditional Waterfall model. You start with an initial planning phase, this then leads into the Requirements, Analysis & Design, Testing, and Evaluation stages, which then lead back to another iteration of the same four stages. A deployment stage leads off from the Implementation stage, and can be taken from any of the iterations. This allows working and useful software to be deployed out to the client at any stage of the development process, and can be prove to be more useful than meeting with the client and only being able to present them with paper based material. It is possible to demonstrate part of the software to them at any phase in the design process, or as more and more iterations have taken place deploy a working version of the software out into the business, the feedback from its use can then be fed back into the next iteration of the development process. It overcomes the problem with the Waterfall Model where the client gets little or no input during the middle part of the development process, as there feedback is sought out at any point by forking off a deployment of the system. This in itself can be a disadvantage, if the client is given too much input into the system at every stage, it can lead to them making a large amount of pedantic and perhaps unnecessary changes which slows down the development time of the system, it can also dishearten the development team if the client wishes to make a change after every iteration.

As I plan to have regular client based meetings with the Venues Management Team, and in some cases with the Student Staff who worked at the Students' Union, I feel that the Waterfall model would be an inappropriate one to follow, as strictly following it would not allow these regular meetings to show what has been achieved or to discuss improvements that could be made. I feel the Modified Waterfall Model would be a better approach to take, as each time there is a client meeting I can branch off from the last iteration of the Implementation section and demonstrate the system so far, and gain feedback from the user having done so. I will have to carefully manage the amount of influence the client is allowed to have over the design model, allowing them to make a large amount of small design changes because they think it will make the system look better will slow the development process, and risks a late delivery of the system. Choosing the Modified Waterfall Model also fits in with the project being continued on after this report is finished.

Wider Context

Cardiff University's Student's Union Venues Department operate the Taf pub, Y Plas night club, the Great Hall Live Music Venue, and various catering outlets throughout the building. Each of these places may contain one or more bars, kitchens, or outlets that are manned by the Venues Assistant Staff. Venues Assistant Staff are all University Students studying at a University in Cardiff, though in general a Venues Assistant can work in any location, some rolls are specialised, and only certain people can undertake these roles; a chef position is one such example. Venues Assistant Staff are led by a Team Leader, a Team Leader is cross trained and can work in any position within the Student's Union. All Venues Staff are employed on a casual zero hour basis.

The overall running of the Venues Department is down to a six person management team, they are full time salaried staff, who report to the Operations Director of the Student's Union. They are supported by a two person admin team, who are also salaried staff. The

management team are responsible for ensuring that the Venues Department is staffed at the appropriate level, that all shifts are filled, and that staff are paid correctly for the work which they do.

Problem

The current system (Central Services) employed to manage staff booking shifts has been outgrown by the needs of the business. It doesn't support features which the Venues Department management team feel are essential to the day to day running of the business.

A new system needs to be designed from scratch that handles the needs of the Venues Department management, it needs to handle all of the features which they are essential to the business, and also leave room for future expansion of features.

The key parts of this are:

1. Allowing management to post a list of shifts which are available online.
    a. Allow the appropriate members of staff to be able to book these shifts.
2. Allow Staff to record the times at which they start and end a shift digitally.
    a. Allow management and the admin team to generate a pay roll report based on this data.
3. Allow staff the flexibility of swapping, and droppings shifts which they have booked.
    a. Without the need for management to intervene on each and every case.

Stakeholders

The stakeholders in this project are split into three groups.

The Operations Director; and the board of directors at large, require a system that meets the need of their business, and reduces the amount of contact time the Venues Department management team need to spend organising staff, processing and correcting payroll information, and authorising and arranging shift swaps, and shift drops. This will improve the business from their point of view as the time freed up can be used more productively by the business in order to grow and expand it.

The Venues Department Management Team, require a system that meets their needs, and not just those of the business. Tasks which they perform on a day to day basis need to be easy to perform, and not seem arduous. Where possible parts of the system need to be automated to reduce the amount of time it takes for manual data entry. For example the Wednesday club night even "YOLO" generally requires the same amount of staff each week, rather than enter each shift individually for every week, a feature to copy a previous event would be useful, and would also free up more time for other work to be carried out.

Venues Assistant staff require a system that is accessible from wherever they are, on whichever device they are using. One of the biggest complaints from Venues Assistant is that the functionality of Central Services suffers when used on a mobile platform, this is something essential to overcome in a new system. The Venues Assistant staff also need to be able to easily carry out their tasks on the new system, and see at a glance when they are working.

Although Team Leaders are generally grouped in with Venues Assistants for most of the system's needs, they do have some slightly different needs. Being able to view a list of the people who are working in a particular location would be extremely useful.

Constraints

The most considerable constraint with this project is that it must be web based, and cross device and platform compatible.

For web development, there is only really PHP which was built as a dedicated language for web applications. Although several programming languages come with both officially supported, and third party supported frameworks; such as Django for Python, for web application development. The mainstay of database based projects are still carried out in PHP due to the wide nature of the support available for integrating with a variety of database systems. This naturally steers the project towards using PHP as a language, but research needs to be carried out in order to determine if this is the best approach to take.

What also constrains this project is the need to keep the user interface similar to that of the current system; although this is done to help staff transition between the systems, it does mean that the basic user interface of the system is already fixed.

Existing Solutions

To understand if an existing software solution would be more appropriate than designing and implementing a proprietary solution research was conducted into two software based solutions. One of these; MakeShift was mentioned to me in an early meeting the with the Students' Union Management Team as something they had looked at implementing, but there was some uncertainty as to whether it would fulfil their requirements. The other Homebase was found while looking at other existing solutions.

MakeShift

MakeShift is an online employee scheduling software "It's more than just staff scheduling software; it's a complete HR software solution with advanced online employee management features designed in a way that puts people first." (Makeshift.CA accessed January 2016).

Though it is advertised as being an online solution, MakeShift is an entirely app based solution, and requires that all users have access to an Android or iOS device. A limited amount of output, mainly what availability the employee has set and what shits they have been allocated can be viewed through an online portal, but no interaction with the system is supported through this portal.

Rather than allowing staff to directly select the shifts they wish to work MakeShift requires each employee to set availability for each day they are available to work, specifying a time window within which they can work a shift. A manager than has to create each of the shifts required within a day, and choose a member of staff who is available to fill that shift.

Despite this MakeShift does support shift swaps being handled by the employees, albeit with the requirement for management to approve each swap before it is actioned in full. One useful part of this is it allows an employee to both drop (completely get rid of the shift)

or exchange a shift with another colleague all in the same interaction, in the latter case this means the manager only has to approve one shift swap and not two.

It also supports clocking in and out of a shift on the app. This was the feature I was most interested in, to see what solution had been found. From the marketing information available on the website I was not able to determine how the clocking in and out worked, other than “employees use the app to record their time clocks”. I downloaded the trial version of MakeShift and went through the configuration in order to better see this feature in use. It became apparent that the clocking in and out was done based on Geolocation, when configuring the company settings it requires you to pick a location, and set a radius for where clocking in and out can take place, the help documentation recommends “The radius should be between 50-100 metres surrounding the edge of your building”. Tests were carried out on an iPhone 6S Plus to determine if this would work within the SU building. The phone struggled to locate itself within this recommended geofence until the radius was increased to 500m. This had a downside of allowing a clock in to be recorded from a long distance away from the building.

MakeShift also supports a range of notifications being sent out to employees, but like most of its features only within the app itself. It doesn’t support sending out text messages or emails.

Although it does have a range of features that fit in with what Cardiff University Students’ Union needs from a shift booking system, it does not have the full set (A full list can be found in Section 3), or like in the case of the time clock feature, the feature is there but the implementation is simply not appropriate for the environment it is going to be used in; allowing people to clock in 500m away from the building is open to abuse by staff as it doesn’t actually confirm they were in the building at the time. For these reasons it would seem that MakeShift would not be an appropriate solution to the problem presented.

Homebase

Homebase is an online employee and time sheet management system marketed as “The easiest way to schedule your team” (joinhomebase.com accessed January 2016).

In contrast to MakeShift the web portal that is available when subscribing to Homebase is fully featured, and allows the employee or manager to carry out all of the tasks they can do on the app which is also available to access the system. The app that is produced is available for iOS and Android, the version for employees is free, the manager version costs £1.99 per manager account. The portal option being fully featured make this a more attractive choice as this is a key requirement for the Venues Team.

Again rather than allowing an employee to choose the shift they wish to work, it relies on them setting a day by day availability and a manager then allocating an available employee into the shift. It is more featured than Makeshift and allows an employee to duplicate a day’s availability, or for that matter an entire week, which saves time. Similarly a manager can duplicate all of the shifts that have been created for a day, or an entire week which saves times inputting each one manually.

Managing shift swaps is an available feature, and there are various configuration options available to change exactly how these are managed by the company; for example it can be set so that a manager has to approve every swap, or it can be set so that swaps only need to be approved if an employee has swapped away X number of shifts, or it can be set so that the swap only needs to be approved if it will take the employee picking up the shift into an overtime bracket. These are all useful features that are worthy of consideration.

Clocking in and out, and automatic generation of timesheets is a large part of what Homebase has to offer, and a large amount of their product marketing is aimed at these features. Clocking in and out is done through a web browser, or through the app. A web browser must have been approved by a manager before it can be used to clock in and out of the system, this prevents an employee from logging in at home and clocking in from there. Clocking in through the app works in two ways, a device can be put in a fixed location, and setup as an approved device, then all employees can use it to clock in and out. If there is an internet failure employees can use their own device to clock in and out, this information is stored and flagged as coming from an employee's device, management can then approve these time clocks. Clocking in is achieved by entering an employee ID and pin number between 4 and 6 digits long, then selecting if it is a punch in, or punch out.

Exporting to main stream pay roll packages is supported out of the box with Homebase, Sage Pay is fully supported with direct integration, though it does require the payroll report to be run manually, and there are no notifications produced if this has been missed.

Notifications are supported by Homebase, but only by allowing a broadcast message to all employees via the app. It doesn't support email or text notifications.

Again, although it has features that are required by Cardiff University Students' Union, it doesn't have the full set. Mainly in relation to the fact employees can't select the shifts they want to work, but a manager or other paid member of staff must sit down and allocate shifts manually. This takes time which is what they want to cut down on.

Methods and Tools

In order to determine the best way to carry out this project it was important to carry out research on what would be the best solution to building parts of the system.

Gathering a specification on what data needs to be stored within the system is important, and this will give a base idea of the relationship between the data entities. This will then need to be taken and transformed into a normalised entity relationship diagram, detailing all of the tables which make up the backend database of the system, the fields stored in each table, including the data types, and how each table is related together.

Researching frameworks that are available for rolling out a web based interface is also important to look at, as this may save time, and duplication of effort when developing this project. Although at the same time it is important to not just consider frameworks as an easy way out.

Research Questions

The aim of this project is to construct a shift booking and payroll system which fits the specific needs of Cardiff Universitys Student's Union Venues department. This will build upon the current system that is used, and bring parts of the system that are currently done manually into the system.

In order to carry this out it will be necessary to identify the users of the system, and what the needs of each user are. An analysis of the current system will also be needed to understand what information is currently stored and how this information is used. Once this is done it will be possible to carry out a detailed specification on what is required by the new system. A full testing plan will need to be designed, and implemented to ensure that the system meets the specifications, the testing plan will be considered the acceptance criteria for the system.

3 Design and Specification

Design Introduction

The client is at the heart of any system, and although the client commissioning the project; Cardiff University Student's Union Venues Management Team, would be making most of the specifications, it was acknowledged both by them, and myself, that the Student Staff who also used the system on a day to day basis would need to be involved, and consulted throughout the project, in the same way the Venues Management Team would be.

The first step into the design process was to arrange a meeting with the Venues Management Team, and have an open ended discussion about what they wanted the new system to achieve. This would also be an opportunity for myself to ask any questions, to gain a greater understanding of the existing system and procedures, and how they might be incorporated into a new system. It was agreed that I would take the outcome of this discussion, and formulate that into a formal system specification, which would be reviewed at another meeting by both parties, and changes could be made to these before they were carried forward into the final version of the system specification.

Due to the large nature of the Student Staff base; approximately 350 staff, it was felt that involving them in an open ended discussion at this early stage would be impractical, and would add little benefit to the project outcome. It was agreed that the Team Leaders could be consulted, either individually or as a group, if any decisions at this stage would have a larger impact on the student staff user base, rather than the Venues Management user base. This would provide a much smaller sample; approximately 25 people, who are all familiar with the system, and are considered a more intensive user of the system, due to the number of shifts they work.

Research into the Current System

The current system; and the system under design as will become clear later, is split into three main parts, these being the shift booking, time records, and payroll. Each of these areas was researched separately in order to gain an understanding of the current system, and how these might be improved upon.

Shift Booking

Central Services, is the name used by Venues Management and Student Staff alike to refer to the current online portal for booking shifts. Though it is referred to by this name, the rota is only a small part of the Central Services system, which actually handles other parts of the Student Unions IT needs.

For a member of Student Staff, they login to this system using there Cardiff University username and password, and are presented with a monthly calendar view. Buttons are provided to navigate between the different months. Shifts which are available to book are shown as a hyperlink, and shifts which have previously been booked are shown highlighted with a dark blue background. This is all the information the member of staff is provided with, they don't know where in the building they are working, just there start and end times.

For a member of Venues Management, they login to the system in exactly the same way as a member of Student Staff, and they too are presented with the same monthly calendar view. For each day on the calendar they are shown two pieces of information, the number of Team Leader shifts that have been filled, and the number that are available, the number of Venues Assistant shifts that have been filled, and the number that are available. Clicking on either of these opens up a popup menu where they can add new shifts, or modify, or delete existing ones. There is also an option on this screen to print the day's staff list.

It was pointed out that one of the most irritating things about the current system was the need to enter each shift individually, an option to clone selected shifts to another day would be extremely useful.

Time Clock

The current system for recorded the time spent working is done entirely on paper. The option mentioned above to print off a list of the entire days shift is used each morning, to produce a paper copy which is left at the staff signing in point.

Staff fill in their start time when they enter the building, and then when they leave at the end, fill in their end time.

It is the admin staff which are involved mostly in this part of the system, as it is them who has to take the time sheet each day and enter it into a spreadsheet in order to have a digital copy of the clocking in and out times. They pointed out a number of problems with the current system. Some staff don't round the time they start or leave to the nearest 15 minutes as they are asked to do, this slows down the data entry process as they have to work it out as they enter the data. Some staff use the 24 hour clock, whereas some use the 12 hour clock, which is especially confusing when shifts start and end around the clock. Staff also forget to fill in one, or both times, and there is no easy mechanism with which to contact that particular member of staff to alert them to the problem, and the fact they won't be paid for that shift.

The management team also expressed some concerns about the current system, namely that it is very easy for a member of staff to turn up late for work, or to go home early, and still fill in the times as though they worked correctly. Or for another person to sign someone else in and out.

Payroll

The spreadsheet that is created by manually inputting the information off of the time sheets generates a report, person by person, of the number of hours they have worked in a pay cycle. This spreadsheet also includes the member of staff's bank details.

This report is passed onto the payroll department and is used to input the information into Sage Pay. Sage Pay then generates the payment requests into each person's bank account, and also generates the payslip for each person.

Once the report is passed from the Venues Management admin team to the payroll department, it is outside the scope of this system, the design just needs to factor in the information, and the format in which it needs to be passed onto the payroll department.

Specifications

A formal meeting was held with the Venues Department Management Team, and draft copies of this specification; which had been devised through a series of informal meetings and conversations, where distributed for final discussion and agreement.

The specifications where agreed upon verbally, but where formally written into the project specifications, and would form the basis for the acceptance criteria of the system. All parties agreed this was the best way forward, the Venues Department consisting of a large number of Student Staff generally trends towards making agreements on a verbal rather than written basis.

Mandatory specifications

1. Staff must be able to book shifts from an available pool of shifts
    a. The ability to book a shift must look at the member of staffs rank, and the rank required by the shift before allowing them to book it. For example none Team Leaders should not be able to view or book Team Leader shifts.
2. When staff are booking a shift they must be able to reserve a shift they are looking at for a time period defined in the system settings. They must confirm the shift booking before this time window expires, or the shifts will be released back into the general pool of available shifts.
3. When a member of staff books a shift, or previews a shift before confirming the booking, the system should display an estimate of the amount of pay the member of staff will receive for that shift.
    a. It is only an estimate as the system does not take into account any Tax or National Insurance, or other compulsory deductions that need to be taken from the member of staffs pay.
4. Allow members of staff to record the time at which they begin and end a shift on the system.
    a. This will be done by the member of staff logging into the system with their username and password and pressing the appropriate option to record the current time as a clock in, or a clock out.
    b. If the member of staff attempts to record a clock in or out but does not have a shift defined the system should prompt them to also enter where in the building they are working and which member of management asked them to come in for the shift.
5. Alert members of staff through the calendar view if they have not clocked into, and / or out of a shift, and allow them to submit a report to correct this.
    a. A shift which misses just one of the two time clocks should be shown differently to one which had both time clocks missed.

    b. The report submitted that is submitted to management should include both the start and end times of the shift, and a short text explanation of why the time clock wasn't recorded.
6. Allow management to add shifts to the pool of available ones which are able to be booked.
    a. They must be able to define all the relevant information about a shift, such as the start date, start time, end time, location the shift will take place, and the rank of staff which must take this shift.
    b. They must be able to edit the details about a shift at a later date if a mistake has been made.
    c. They must be able to delete a shift if required.
7. Allow management to adjust system settings, add and amend ranks, and add and amend locations.
    a. They should be able to add new locations to the system at any time.
    b. They should be able to add new ranks, and stop old ranks from being used at any time.
    c. They should be able to adjust system settings, such as the amount of time a shift is held as reserved for before it is released back into the general pool.
8. Allow members of staff to swap shifts without management intervention.
    a. A member of staff should be able to select a shift they wish to drop.
    b. The shift should then be made visible back in the general shift pool, while at the same time still showing on the member of staff's calendar.
    c. Once another person has taken the shift it should be removed from the original members of staff calendar.
    d. Management should be able to bar shifts swaps on a particular day, or time window.
9. Generate a pay roll report for a particular pay cycle.
    a. The report should be produced line by line with the member of staff's employee ID, First Name, Last Name, and number of hours worked.
    b. It should be produced in standard CSV format (There is no formal standard for CSV files, but for the purposes of this specification it should be compatible with the RFC 4180 standard)
10. Run an operation to move staff between locations when not all shifts have been filled for a particular time.
    a. Examine the number of staff that are available for that particular shift.
    b. Work out how many locations require staff at that time.
    c. Try and get each location as close as possible to 100% filled based on the optimal number of staff that that location needs.
    d. Move the staffs assigned locations based on this.
    e. Update the staff that there locations have moved by way of a notification.

Optional Specifications

The following optional specifications where decided upon, as being features which would be nice to have, but weren't a core part of the system, and could be added at a later date is so desired.

1. Implement a method for notifications to be pushed out to members of staff via the web portal.
    a. Managers should be able to add these in at any time, and select whether the notification should go out to all staff, or just a selected group of staff, also an option just to send to a group of staff working a particular shift.
    b. Staff should be able to acknowledge that they have seen the notification, and remove it from there pending notifications.
2. Implement a method for sending out text message notifications to staffs mobile phones.
    a. Managers should be able to send these out at any time, and select whether the notification should go out to all staff, or just a selected group of staff, also an option just to send to a group of staff working a particular shift.
    b. The system should be intelligent enough to not text a none mobile phone number, and not to text a none UK mobile number.
3. Implement a method for sending out email notifications to staffs email addresses.
    a. Managers should be able to send these out at any time, and select whether the notification should go out to all staff, or just a selected group of staff, also an option just to send to a group of staff working a particular shift.
    b. Managers should be able to set whether they want the messages to be sent with message read requests.
4. Implement integration with Sage Pay
    a. Integrate the pay roll report directly with sage pay. This would eliminate the need for the pay roll report to be manually passed to the payroll department, and then input into Sage Pay manually.
    b. Define each member of staff as a Sage Pay object, and directly input the number of hours they have worked into the object.
    c. Include the ability to read back the information that has been passed into Sage Pay to ensure that the data was correctly passed into it and received.
    d. Define each pay cycle as a Sage Pay object and include checks on whether that pay cycle has been actioned into Sage Pay or not. Produce a warning on the manager's accounts if this has not been done 48 hours before the pay day is due.
5. Research and implement an app to allow the features of the web portal to be accessed from outside of a web browser.
    a. Decide on whether a write once, deploy to all approach, or a fresh coding for each platform would be most appropriate. This would require researching whether speed would be effected by not writing the application for its native environment.

b. Decide on which platforms the app would be deployed on. iOS and Android are likely to be definite choices, but other platforms may also need to be considered.
c. Redesign the User Interface of the main web portal to be compatible with an app style layout. Some IDEs for app development allow HTML and CSS code to be used to design the app interface. This may save time by being able to redeploy the web based interface.
d. Carry out the design, implementation, and testing of the app.

Test Plan

A full test plan to confirm that the system has met the specifications as agreed with the client is included within the Results and Evaluation section.

User Interface

In the initial discussions with the Venues Department Management team, they expressed some concern that a major change to the user interface in the new system, compared that in Central Services, could cause more problems than it would solve in the short term. Mainly due to the large number of staff that use the system, and the lack of time that could be given to training each member of staff individually on the new system.

Due to the above points, it was clear that the new systems interface would need to follow a similar format to that of Central Services, and this was made into a mandatory system specification. This gave a starting point for the design of the new user interface.

From the workflow of the system it was clear that the system would be accessed through a login screen, and that this would be the screen that the user would land on most of the time that they were accessing the system. As this system is for a specialised use, and in the expansive terms of the internet used by an extremely small user base, the homepage of the web portal, which includes the feature to login would need to have a little more information on it, than instructions on how to login in, or who to contact in the event there is an issue with logging in.

From the login screen, on successfully authenticating as a user, then site would take you to the “Rota” view. This would be the calendar format used to display shifts that had been booked, and also shifts that where available to book. Most of the use of the system would be on this one page, and although simple in function a lot of different information is conveyed to the user on this screen. How best to display this information was a challenging concept to overcome, not only did a distinction need to be made between an available shift, and a shit that had been booked, but also between a shift where there was an issue with the time clocks, a shift where the person was late, a shift that was pending being swapped, and a shift where the person was absent.

A colour scheme was devised based on the heat map concept (http://searchbusinessanalytics.techtarget.com/: accessed January 2016); the times of a shift would be highlighted with a coloured background depending on their status. Colours at

the green end of the scale would be used to highlight shifts where the user didn't need to take any action, for example a shift where the user has worked, and has clocked in and out correctly. Colours at the red end of the scale would be used to highlight shifts where the user needs to take some action, or needs to be drawn to that shift, for example a shift where the user was absent, they may need to email a report to management or similar.

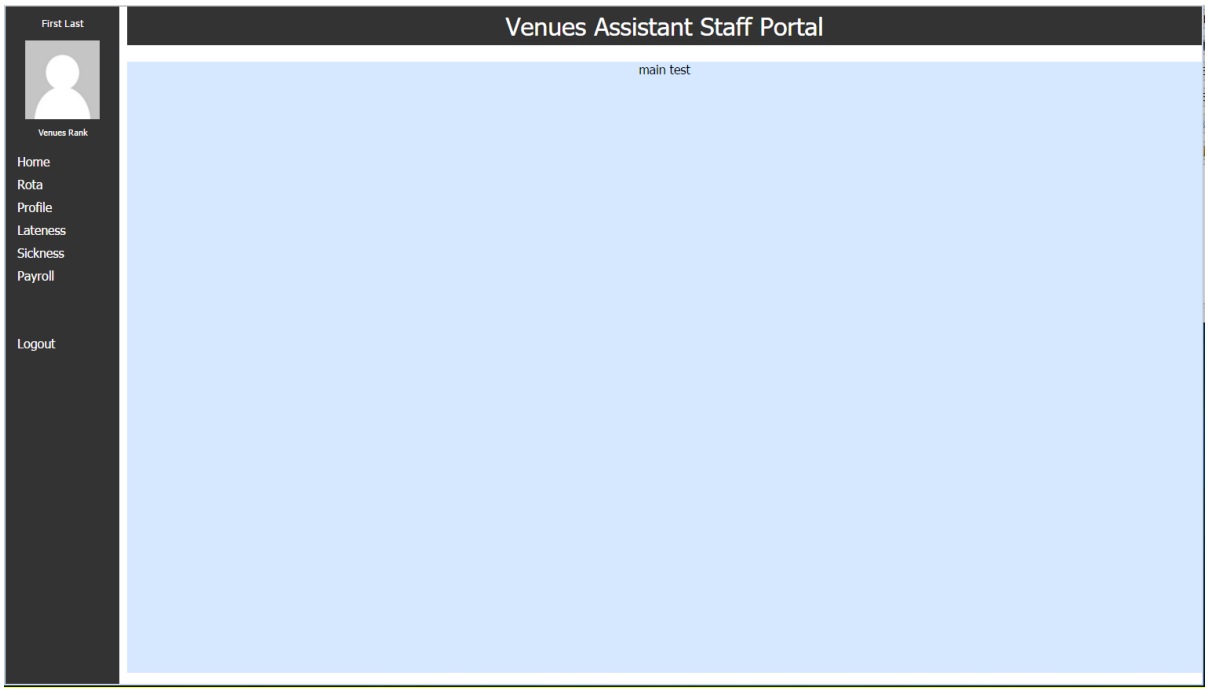
One of the key User Interface design questions to overcome was what to do in terms of handling the user clicking on a shift to bring up more detailed information. For a shift which needs to be booked the information that needs to be conveyed to the user is quite large; what date is the shift, what time does the shift start, what time does the shift end, where in the building is the shift, what type of shift is it, how much money could the shift potentially pay. What would be the best way to display this information, whilst at the same time making the UI quick, and simple to navigate?

One option would be to setup a different page on the site for each of the different shift types, with a template layout for the information, with this being filled by the database each time the page is loaded. Using this method leads to natural modularity of the system, as each page needs to be within its own file. It is clear to the user what information they are expecting to see based on the page that has loaded. However it does mean the user has a lot of back and forth browsing to do, if they wanted to check a variety of shifts for more detailed information, the page would have to load each time for each shift, and they'd have to navigate back to the rota screen to select the next shift. Not only would this make the system less responsive from the users point of view, it would also add to the load on the system.

Another option would be to have the information requested "popup" on top of the already loaded webpage, the information on the popup changing based on what exactly the user has clicked on. This approach may detract from the modularity of the system, as all of the code for this could be placed in the one file, however this could be overcome through good coding practice. What this approach does relieve is the need for the user to navigate back and forth between different pages, and reduces the amount of unnecessary database calls caused by the rota page being reloaded each time.

Although having a different page for each of the shift types would lead to more modular coding naturally, it does mean that a lot of extra calls would be made to the backend database, as the user would have to move between pages causing a reload of the information each time. Using the "Popup" method means that AJAX can be used to reload the relevant sections of the page only, which will lead to less strain on the backend of the system. As such I feel this is the most appropriate choice to make for the system.

In order to better understand how the user interface would look, and to gauge the opinions of the end user on how the user interface looks, a series of mock ups of the User Interface where created, and the feedback at each stage sought from the end user. These mock ups went through several different iterations, I have picked out two of the most important iterations to show how the User Interface design progressed.

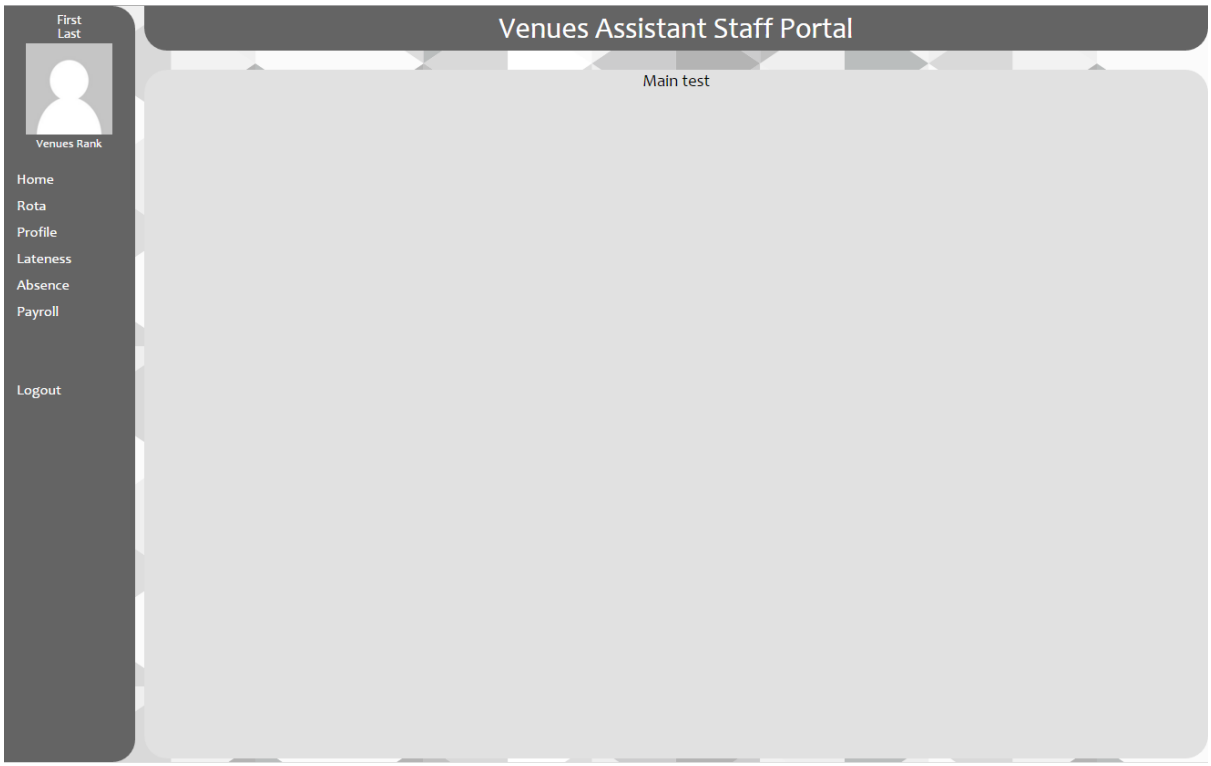


The image above is the first mock up that was developed initially to give a feel for how the web portal would be laid out. I went for a design where the header of the page, and the navigation bar would remain the same throughout, and allow the user consistency throughout the system. The body of the page (The section with the blue background) would be where the contents of each page would be displayed to the user.

User feedback was very mixed for this initial design, more so than I had perhaps anticipated. Positives from the feedback showed that the end users liked the idea of a navigation bar, and all of the users felt that the options presented where simple, and self-explanatory of their functions. They also liked the idea that the layout would be consistent on each page; something which Central Services lacked depending on how or what device you used to sign in on.

Negatives from the feedback where mainly reflective on the mock up looking like “A website from the past”, and the colour scheme not quite matching between the background and the navigations bar and headers.

Comments on the profile picture location and the need for it fell into both camps. Some people liked the idea that a profile picture could be displayed, others felt it was a pointless feature that wasted space on the screen.

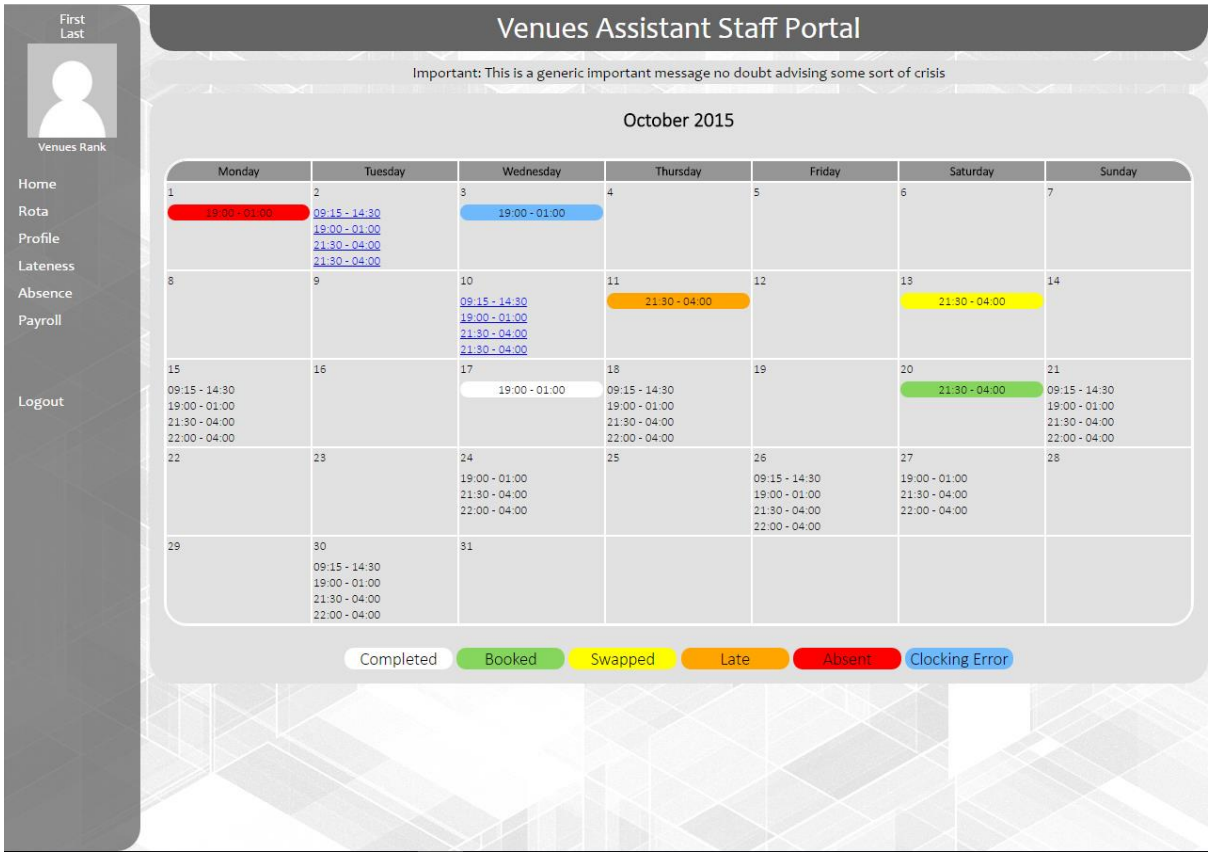


The second mock up shown here attempted to address the feedback received from the first mock ups user feedback. It was not designed to be the final design of the User Interface, but something which would give a good grounding to develop it further.

User feedback was much more positive for this version than the first iteration. They felt that the colour scheme was a much better match than in the previous version, and that the addition of curves, and a background image made the site feel more modern the then previous incarnation.

A small group of users felt that the colour of the navigation bar and header shouldn't be the same as it tended to draw the eyes into both at the same time, rather than one or the other. Some people also felt the background image was pointless as most of it was hidden by the rest of the page content.

Comments on the profile picture where still mixed, though on explaining to people that it would work similarly to Facebook and Twitter, most people seemed to accept it as a feature. It would also make it obvious at a glance if you used a shared work computer and where logged into someone else's account; given that the picture would be much larger than the users name.



This is the final incarnation of the User Interface showing the Rota screen and some sample data taken from October 2015.

The navigation bar was adjusted in colour to take into account the user feedback from the second major iteration, a transparency was also added to allow more of the background image to be viewed. The main content was also set to only wrap to the size needed, and thus display part of the background image below it.

At the request of the Management Team the small “important message” box was added to allow a message to be shown to all members of staff immediately on logging into the system.

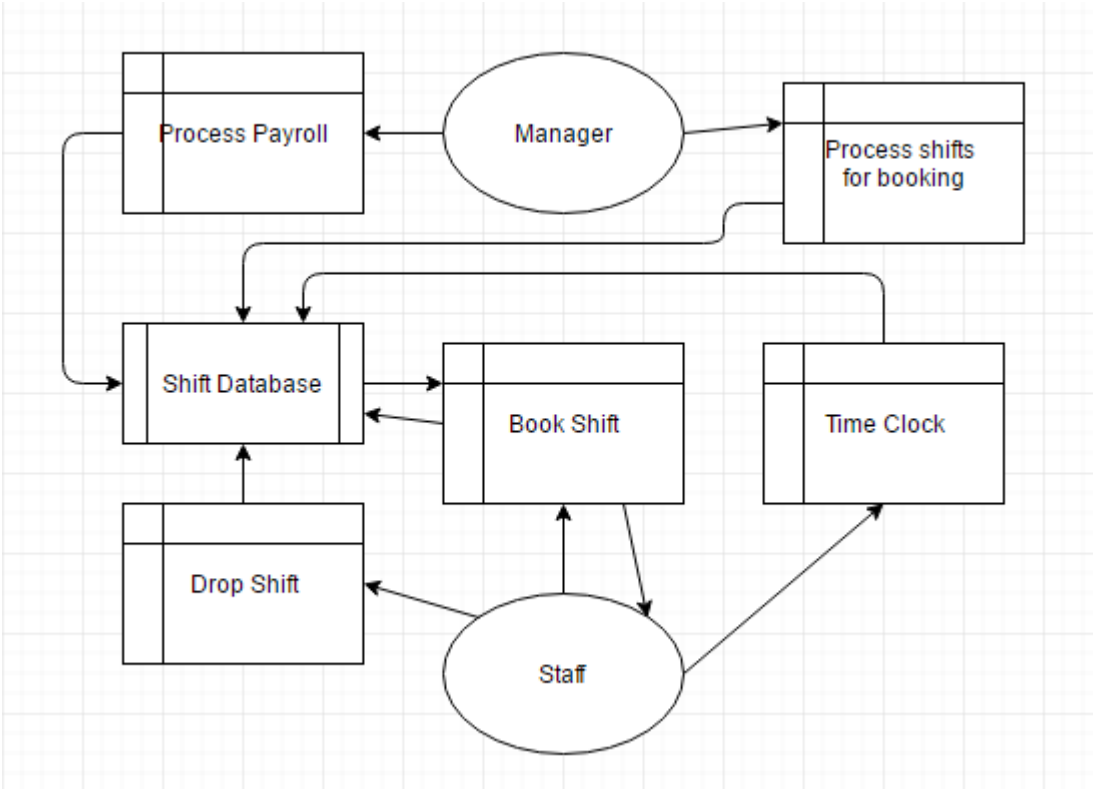
Data

Data stored, and generated by the system is split into various groups. In order to better understand exactly what data would need to be stored, a list of the data needed for each part of the system was listed.

- Shift
    - Date
    - Start Time
    - End Time
    - Location
    - Shift Type
    - Pay Rate
- Staff

    - First Name
    - Last Name
    - Rank
    - Pay Rate
- Ranks
    - Rank Name
    - Pay Rate
- Location
    - Location Name
    - Minimum Rank

Data Flow



Data from the system originates from various different points:

Managers set up each shift that is available to be booked, they do this by inputting the date of the shift, the start time, the end time, and what rank person is required for the shift, they then input the location that the shift is for.

Members of staff look at the available shifts, and make a decision for themselves as to whether they are able to work that shift or not. They then select the shifts they wish to work and mark that they are going to work that particular shift.

Members of staff may find they are no longer able to work a shift for whatever reason. They mark that they wish to drop a shift and this releases it back into the shift booking pool. Another member of staff can then mark that they wish to work that shift.

When a member of staff enters the building, they record the time that they start there shift into the system. At the end of their shift they record the time at which they have finished. Sometimes a member of staff may have been called in at short notice and may not have an assigned shift. In this case when they record the time at which they start the shift they also need to record what location in the building they are working, and which manager asked them to come in at short notice.

A member of staff may forget to clock in and or out from there shift; if they forget to do both then the system will flag them as being absent, if they only forget to do one then the system will flag a time clock inconsistency. The member of staff then needs to manually input the time of the missing time punch. A manager then has to authorise this to prevent fraudulent use.

At the end of pay cycle the admin team need to run a payroll report, this will reconcile all of the shifts that have been worked in that pay period, and flag up any that still require attention. The number of hours that each member of staff has worked is then recorded on the report against their name and employee ID.

When the number of staff available for a shift doesn't meet the required number, the system is designed to load balance the number of staff against all locations so that each location has the maximum number of staff it has out of the available staff. The data to do this is already stored within the system, but a manager must trigger the system to attempt a load balance. The system then looks at the staff who are working within that time frame, and what the most efficient number is to have per bar, it then tries to evenly distribute the staff, and reallocated shift locations based on this output.

Management may need to delete, or amend details about a particular shift. This is done by marking the shift as delete, or by manually editing the shift details.

Languages

The choice of languages was briefly discussed in the Background section of the report, which concluded on the languages which were selected to carry out the project.

As the project is clearly divided into three parts, the User Interface, the backend database, and the interface between the two, it was logical to break down the choice of languages to these three components, and find the most suited language for each one.

User Interface

The user interface is restricted the parts of the system which display the content to the user, and also receive input from the user. The processing behind this data, either what is to be displayed, or validating the input data is handled by the middleware section of the system.

Hyper Text Markup language (HTML), is the only choice available for displaying a web based site; this is distinct from the choice of means of producing the HTML, what must be decided upon is which version of HTML to use, and how this will potentially affect what can be done by the system.

Although there are four version of HTML, I have found it only relevant to discuss HTML4 versus HTML5, due to age of HTML3 (1997).

HTML5 was released in late 2014 and replaced HTML4. It was designed to supersede not only HTML4 but also XHTML 1; which is a version of HTML that complies with the stricter XML format, and can therefore be parsed using XML rules, something which cannot be done with HTML, and DOM Level 2 HTML; which is the organisation of elements within a HTML structure into the Document Object Model tree, it provides methods and operators which can be used to manipulate parts of the document without reloading the entirety of the page.

HTML5 was designed not only to be a mark-up language, but to also provide APIs for complex tasks that in HTML4 or lower would have required specific, proprietary plugins that not only required to be compatible with the users chosen browser, but also with the underlying operating system.

HTML5 was designed from the outset to be available across all platforms and device types, meaning it was built with low powered devices in mind, and has optimisations for battery life of these devices. Previous versions of HTML where not designed with this in mind as low power devices where not in mainstream use (HTML 4.01 was published in late 1999).

Between HTML4 and HTML5 only a small amount of the document tags where depreciated; in other words no longer supported, all but one of these where related to formatting of the document, something which was handled through HTML before Cascading Style Sheets (CSS) came into use. The removal of tag attributes was also mainly related to loss of formatting, which again can now be done through CSS.

HTML5 provides a greater support for formatting the document on the fly, without reloading entire sections of the page; although this was possible in HTML4 using Asynchronous JavaScript and XML (AJAX), it can be done through the various APIs provided, as well as through AJAX.

One of the most important benefits of HTML5 for this system is the Canvas API, this allows for scriptable rendering of 2D graphics and bitmaps, with full support for AJAX to manipulate it. Prior to HTML5 the only globally available product for doing this was Flash, a technology which is prone to attack and security vulnerabilities, and one which big name manufactures; such as Apple, do not support on their devices.

None of the features present in HTML4 which where depreciated in HTML5 are essential to the needs of the project; all of the removed features are still accessible by using CSS, and the additions of APIs into the HTML5 specification mean that third party plugins aren't required; for example the canvas API replaces the need to use Adobe Flash Player. For this reason the system will use the HTML5 standard throughout.

Cascading Style Sheets (CSS) are used to style not only mark-up language documents such as HTML, but can also be used to format other mark-up languages such as XML. There is no other choice to use, and therefore CSS is essential to the system. Unlike HTML, CSS does not have strict versions, it is an ever evolving standard with little or no depreciation over time.

Middleware

The middleware section of the system, sits between the User Interface, and the backend database, it feeds the data between the two sections, and ensures the data is suitable for insertion into the database, and transforms and processes the data outputted by the database.

PHP (Originally Personal Home Page, now known as PHP Hypertext Preprocessor) was originally designed as a programming language specifically aimed at web development, but is now also used as a general purpose programming language. It is an interpreted language, and doesn't require compilation before it is executed. It has inbuilt support for a number of SQL based database languages, and is fully capable of generating HTML output. As it is a server side language it does not require the client to install anything on their device other than a compatible web browser capable of viewing the HTML output. PHP has a simple syntax, and is widely used in web development; as per its original design intention. What PHP lacks is a framework structure for integrating with a website front end. This needs to be hand written by the developer, or an appropriate third party framework found.

Ruby on Rails is a web application framework written in Ruby, it designed to be deployed on a webserver running an SQL based database or similar, and Apache. The principle of Ruby on Rails is to concentrate the developers time on parts of the system which the framework doesn't have inbuilt support for. Although Ruby as a language is supportive of both Object and Imperative programming, Ruby on Rails is built around the concept of Object Orientated Programming.

Node.JS is an open source cross platform runtime environment for developing web based applications. Although it is not strictly a JavaScript framework the majority of its libraries are written in JavaScript, and users can add to it by writing their own modules in JavaScript. It is an event driven language, and was designed to handle real time web based applications, such as online games, and other asynchronous applications. It is not generally used for database based system, but it can be used for such applications as there is some support for the most common SQL based languages.

ASP.NET is a framework created by Microsoft that allows for web applications to be built using any .NET language; C#, Visual Basic etc, that is then compiled into a web based application. Because of this it can be an extremely useful language, as the mainstream .NET languages are well supported, both by the creators and the third party developers of libraries and plugins. One of the biggest advantages of ASP.NET would be the ability to later reuse large parts of the code to roll out a desktop based .NET application, as well as a mobile device based .NET application; although one limitation of this is lack of support from ARM based processor devices.

Django is a Python based framework for deploying web based applications quickly and easily with a principle of no code repetition. It is specifically designed to ease the deployment of database driver websites, with wide support for a number of database types, not just SQL based ones which most frameworks look at.

Java Applets are cross platform compatible apps; despite the name given to them, they don't have to be written in Java, but can also be written in languages such as Pascal, they are delivered to the clients machine via web browser, and are executed using the Java Virtual Machine. Unlike JavaScript Java Applets have full access to the client machines 3D hardware, this means that processing can be done much faster than JavaScript. Because they are precompiled they also tend to run faster than the equivalent functions in JavaScript. Although they are executed on the Client machine, they do not have full access to the client machine and this prevents them from installing malware or similar on the client machine. Generally Java Applets are used to develop online games, or simulations but can also be used for other tasks such as this one.

As I have personal little experience with deploying web based frameworks, but have previous experience in deploying web based database applications using PHP, I have chosen PHP from the above options.

As discussed above in the User Interface section, HTML5 was chosen because of its increased support for the DOM, and being able to refresh parts of this without refreshing the entire HTML document.

Outside of the inbuilt HTML5 APIs for DOM manipulation, javascript is the only standalone language which supports access and manipulation of the DOM tree. As such javascript will be used by the Middleware section of the system to refresh relevant parts of the webpage as the user interacts through the User Interface.

Backend Database

SQL; Structured Query Language, is a special purpose programming language specifically designed for managing relational data held in a series of database tables, with the ability to perform a series of queries and manipulations on this data. It has been the international standard RDBMS since 1987, but was adopted as the American standard in 1986. Almost all RDBMS are spawned off of SQL in some way shape or form. SQL is an open format. Although it is a language in its own right, very few people actually use SQL on its own, due to the complex nature of its implementation, generally one of its off shoots is used instead.

MySQL is an off shoot of SQL, it is completely open source, but is developed by the Oracle Foundation. It is an expansive language that has huge support from both the Oracle Foundation, and other third party developers. One of the biggest benefits of MySQL is the cost free aspect, but also that it has a dedicated GUI application, MySQL Workbench which can be used to administer any MySQL database. MySQL has also been incorporated into "AMP" stacks, (Apache, MySQL, PHP) these are software bundles that can be installed on a webserver and provide almost all the functionality that is needed.

OracleSQL is a commercial implementation of SQL wrapped in Oracles commercial command line and GUI interface for managing SQL data. It doesn't just implement the base SQL92 standard, but also several other features such as SpatialSQL; though this would not be relevant to this system. Although the interface they have designed for SQL is extremely functional and useful, the cost is prohibitive for this project. Especially when open source version are available with the required feature set.

SQLite is an embedded version of the SQL92 standard. It is completely different from the other SQL packages discussed here in that it doesn't implement a client server model, but is embedded within the application itself. Although this means it generally isn't used with a web based application, it can be used standalone from the main database, and the data later uploaded for future use in a main central database. It can also be used within applications to implement a local database of user's settings etc. This could have some potential use in the future of the application, and it is worthy of considering data, and method compliance with the SQL92 standard because of this.

PostgreSQL is another open source implementation of SQL, similar to MySQL, though it is based around the concept of stricter enforcement of the SQL principles, and is described as "more picky" than most other SQL implementations. This is a benefit in that it naturally enforces strict programming principles which ensures code compliance. A disadvantage compared to MySQL is the lack of an officially supported GUI package with which to manage the database. Any management has to be done through a command line interface.

Another brief consideration for a back end database would be some form of implementation of a none relational database management software. Due to the nature of the data that is being stored in the system, and its close relational links, I believe this would not be a suitable implementation for a back end database.

From the choices that have been outlined above, I feel that MySQL would be the most suitable choice, it has a client server model which is what this system needs due to the concurrent access by multiple clients, it is open source and therefore doesn't require a license fee, it has an officially supported GUI interface that can be used to maintain and backup the database, and it is also widely supported in AMP packages that would install everything the system needs on a webserver.

Infrastructure

The infrastructure required by this system is fairly small in relative terms. The backend database, and middleware part of the system are designed to run on a dedicated webserver, hosting the appropriate services required by the languages selected above.

This means the server will need to host a webserver, with PHP installed, and be running a version of MySQL in order to host the database backend.

The webserver would need to have an appropriate amount of storage capacity to support the system. For best practice and to ensure an onsite backup is available with which to restore the system, this storage should be built as a RAID 1 array, in the event of a single disk failure the RAID array could be reconstructed to allow the system to resume without needing to use the offsite backups.

As the server would be a central point of failure for the system, the system would need to be deployed with a mechanism in place to perform off site backups, which in the event of a failure of the main server could be used to restore the system to a point in time close to that of the failure. Although the size of the entire system would not be prohibitive to making a

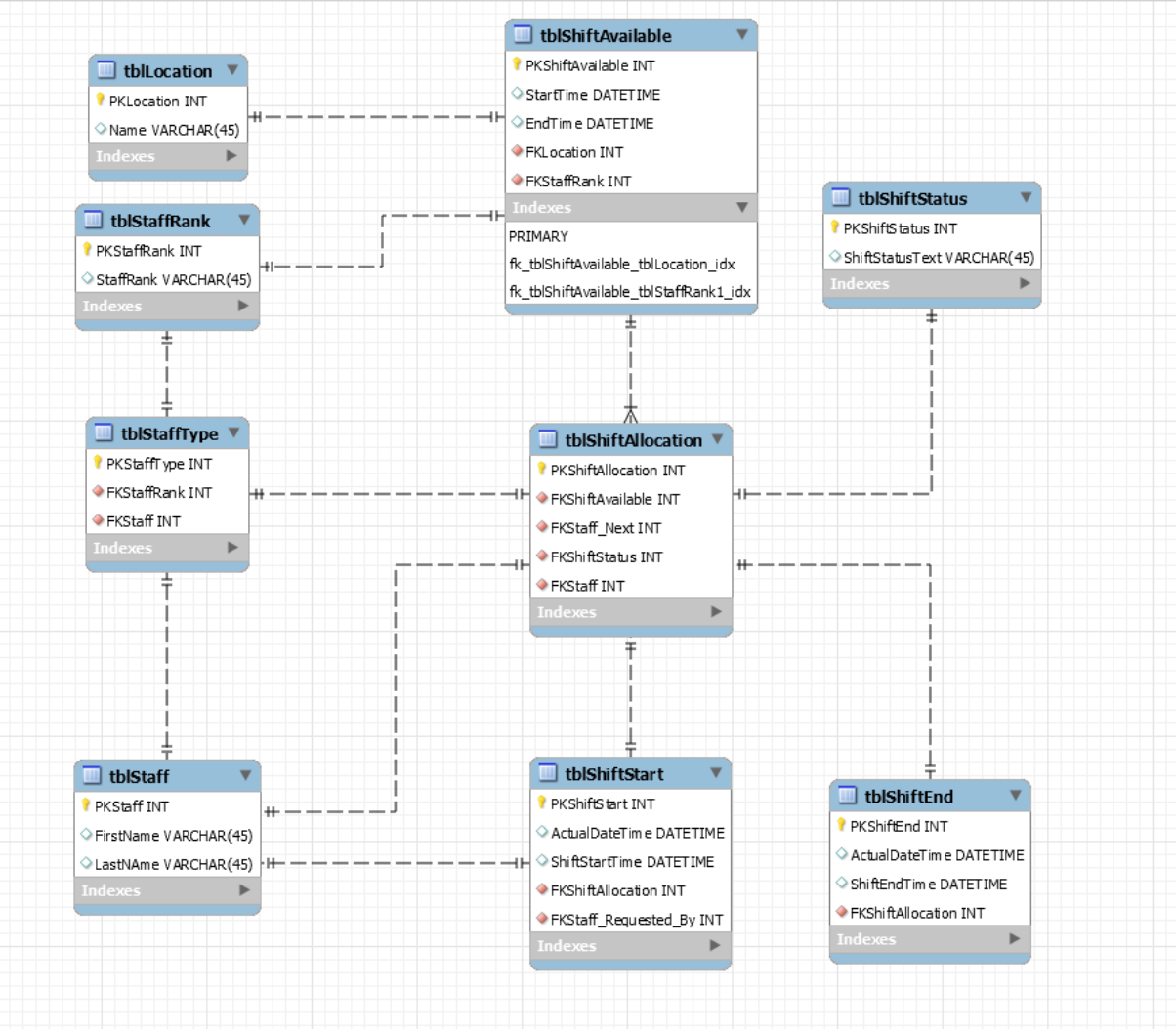
full backup of the system, it would only be essential to back up the backend database, the rest of the system could be restored from the source code.

4 Implementation

Database

The database for the system went through a number of iterations in the early parts of the implementation to ensure that it could store all of the required information, and that the relation links where correct and appropriate.

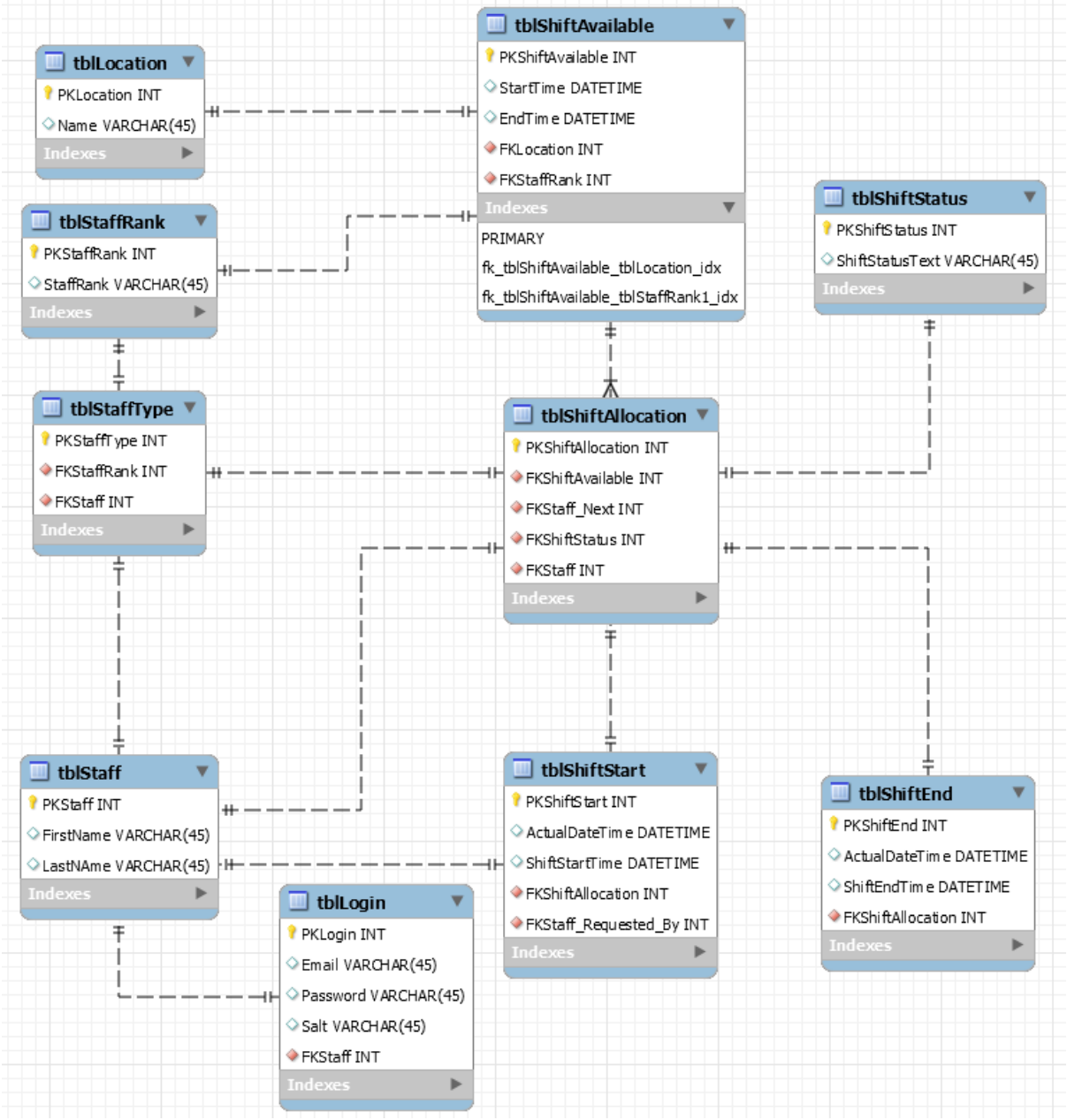
To give a better demonstration of how this was achieved I have picked out two of the Entity Relation Diagrams to show how the process evolved.



This was the final version of the initial iteration of the Entity Relationship diagram, each table is in pace, and the relationships between them, not all of the data that will be stored in each table is in place.

I wanted to ensure that the relationships that had been implemented between the tables would allow the data to be queried in a logical manner, and keep the amount of data being queried from the database to a minimum. Another aim was to ensure that no records needed to be deleted from the database, but could be adjusted with flags, not only would these lead to a better audit trail on the system, but it reduces the load on the database as a delete operation takes more time than an update.

For example tblShiftAllocation is used to store that a particular member of staff has been allocated a shift, FKStaff has a relational link with tblStaff, and that field allows us to see what member of staff has been allocated the shift. If they carry out a shift swap, rather than deleting the entry in tblShiftAllocation, the original entries FKStaff_Next field is populated; again this is a relational link to tblStaff, advising who has picked up the shift after a swap.



This was the improved version of the entity relationship diagram, again not all fields stored in each table are shown.

The aim to reduce the load the database was put under due to queries was compromised in the original iteration as it was intended that the login information for users would be stored in tblStaff, this would mean that each time the tblStaff staff table was queried the login information would also be queried, on top of this it would be bad security practice to query the login information more than absolutely required.

The login information was moved out into the new tblLogin, with a relational link back to tblStaff.

Code

To demonstrate the most complex parts of the system I have picked out parts of the code which relate to the sections outlined below. Because of how the system is designed there is more than one section of code for some of the parts, and some of the code blocks are in different languages as indicated. The explanation of the code shows how these interact with each other to drive the system.

Login System

```
function validateForQuotes(stringToValidate) {
	if((str.stringToValidate('\'') > -1) || (str.stringToValidate('"') > -1)) {
	return false;
}

	$username = $_GET['username'];
	$password = $_GET['password'];
	$errorMessage = "An unknown error occured";

	if(!empty($username) and !empty($password)) {
		$usernameExistsQuery = mysql_query($mysqli, "SELECT * FROM tblLogin
WHERE email LIKE $username");

		if($usernameExistsQuery->num_rows === 0) {
			$errorMessage "Incorrect username or password";
		}
		else {
			while($usernameExistsQueryResult = $usernameExistsQuery-
>fetch_object()){
				$databasePassword = $usernameExistsQueryResult-
>password;
				$databaseSalt = $usernameExistsQueryResult->salt;
			}

			if (md5($password + $databaseSalt) == $databasePassword) {
				grantSessionCookie();
			}
			else {
				$errorMessage "Incorrect username or password";
			}

		}

	}
	else {
		$errorMessage = "Username or Password empty"
```

```
        }
```

- The JavaScript within the login page validates that no illegal characters have been entered in either the username or password field
    - If there is a problem an error message is displayed and the user is not allowed to proceed until the error is corrected
- The PHP login script queries the user table of the database for a single record which matches the given username
    - If no such record exists an error is displayed to the user
- If the username is found, the password given by the user is appended with the salt from the user table, the whole string is then hashed using the MD5 algorithm and compared against the encrypted password from the user table.
    - If the password does not match an error is displayed to the user
- A session cookie is then created which proves that the user has logged in. Each time a page is loaded this cookie is checked and if it has expired, or doesn't exist the user is taken back to the login screen.

Shift Swaps

```
function ShiftSwapInitiate(){
                $staffID = $_GET['staffID'];
                $shiftID = $_GET['shiftID'];
                $timeWindow = getShiftSwapTimeWindow();
                $errorMessage = "An unknown error occured";

                if(!empty($staffID) and !empty($shiftID)) {
                        $currentShiftQuery = mysql_query($mysqli, "SELECT * FROM
tblShiftAvailable WHERE PKShiftAvailable = $shiftID")

                        while($currentShiftQueryResult = $currentShiftQuery-
>fetch_object()){
                                $originalStart = $currentShiftQueryResult->StarTime;
                                $originaEnd = $currentShiftQueryResult->EndTime;
                        }

                        $conflictStart = $originalStart - $timeWindow;
                        $conflictEnd = $originaEnd + $timeWindow;

                        $conflictingShiftExistsQuery = mysql_query($mysqli, "SELECT *
FROM tblShiftAvailable WHERE StartTime > $conflictStart AND EndTime < $conflictEnd"){
                                if($usernameExistsQuery->num_rows === 0) {
                                        $markShiftForSwapQuery = mysql_query($mysqli,
"UPDATE tblShiftAvailable SET reallocate=1")
                                }
                                else {
```

```
                                        $errorMessage = "A conflicting exists - Please
contact a manager for exceptional needs"
                                    }
                                }


                            }
}
```

- The member of staff wishing to drop the shift picks the shift they wish to drop
- A query is run that check if any other shifts exist that haven't been booked within a system defined time around the start and end times of the shift being dropped.
    - If this query returns true an error is displayed and the drop doesn't take place
- If the query returns false, another query is run which makes the shift available to be booked by another member of staff.
- When another member of staff books the shift that has been put up for a drop, part of the booking query amends a flag on the entry in the shift swap table, this removes the shift from being displayed on the original users rota, but also allows an audit trail on what shifts have been dropped.

Implementation Difficulties

The implementation phase of the systems development is where the largest amount of time was lost versus the time that had been allotted in the initial project plan. This was due to a mixture of myself underestimating the time it would take for certain parts of the project to be implemented, and unexpected complications arising in the way I had implemented certain parts of the project.

Below I have outlined the problems that I found implementing a particular part of the system, and how I have overcame them.

Login System

Discussed in the Design Section of the report was the decision not to use Cardiff University's central login server to allow users to access the system, but this would be a later optional addition after an initial login system had been established; brought about by the complications of not all staff having a Cardiff University login.

The first mistake I had made in regards to planning the time given to implementing the login system was not to give a dedicated amount of time to it as an objective, but to group in the work plan alongside staff being able to book a shift from the pool of available shifts. At the time I wrote the work plan I believed both could be sufficiently implemented within the time available, as I had written a login system for a project previously, and didn't envision the issues which I faced.

My previous experience with implementing a login system was in an offline, Python based system. The login system was basic, and stored the list of usernames and passwords; passwords encrypted using an MD5 hash, in a CSV file. The amount of users using the system was limited to 10 or less, and looping through the entire list of usernames, and

checking if the supplied password matched was feasible. Naively I thought this previous experience would stand me in good stead for developing an implementation for this system.

Due to the nature of the data being stored a secure login system was essential, but I was insistent that I should code it from scratch rather than use a prewritten framework to handle this part of the system. I set about researching what exactly was required from a "good" login system. Salt, and user friendly seemed to be the consensus. After two days of research I had a fair idea of what I needed to create the system, but I had already taken two days from the allotted seven, and I still needed to implement the staff being able to book shifts as well.

Implementing the login system in itself wasn't too difficult, I have outlined in the Code section above exactly how it works; in short, the given username is queried from the database, and the given password checked for compliance. A range of error messages are then generated, or the user is granted access to the system.

Testing the login system initially threw up some strange occurrences when using passwords which contained certain symbol characters within them. To save time I used a password generator (http://passwordsgenerator.net/ : accessed January 2016) to generate a wide range of passwords to test the system with. It was at first difficult to track down exactly what symbols in the passwords where causing the issue. Through the process of elimination it was discovered that the ' and " symbols where the problem. A solution to this wasn't forthcoming, and rather than spend more time attempting to find a solution, I used the JavaScript validation methods to bar these symbols from being used in any passwords.

The JavaScript validation methods for checking if a username, or password where valid entries proved to be troublesome with cross browser compatibility. This was due to me attempting to use them to change the background colour of the form boxes using RGB values, rather than the colour name in the JavaScript code.

Overall though it was a challenge, and personally satisfying to overcome, I would use a prebuilt framework that is well known and used in the industry. One of those I have found during research was php-login (http://www.php-login.net/: accessed January 2016) which has since been superseded by HUGE (https://github.com/panique/huge: accessed January 2016).

Detecting JavaScript

As most of the website would be driven using JavaScript it was essential to warn the user in the event that the browser they were using to access the site didn't support JavaScript, or didn't have JavaScript enabled; third party JavaScript blockers such as NoScript (https://addons.mozilla.org/en-gb/firefox/addon/noscript/ : accessed January 2016) are becoming more popular in recent years.

A small amount of time was given to how to solve this problem, and I devised a solution on paper where a message would be displayed on the screen by default advising the user that JavaScript wasn't enabled in there browser, and this message would be suppressed using a short piece of JavaScript code rendering it invisible if JavaScript was enabled.

By not devoting enough time to researching how this would best be achieved, and again by not having planned for this in the work plan; as I'd assumed it would be a trivial issue to solve, I ended up spending more time than was required.

The suppression of the default message interfered with the popup windows, and how they are displayed by the browser, this was especially prevalent in mobile browsers if the connection was interrupted during page load.

A conversation with a fellow student about the problem lead me to the HTML tag <noscript> which is not rendered if the browser detects that HTML is supported. Time had been wasted because I had attempted to engineer a solution to a problem, when an adequate solution already existed!

Popups

The use of popups within the system wasn't planned as a standalone component, but was planned under the parts of the system that would require to use them. I envisioned that different parts of the system would use a different popup from other parts, and didn't think to plan time to work on a template module.

If the system had been able to use completely separate popup modules for each different component, this approach would have worked fine, and this is what I was aiming for at the beginning of the implementation phase.

I am not able to fully comprehend the reasons behind the problem, but using CSS and JavaScript to supress more than two or three popup windows out of view began to cause serious issues with the CSS overlapping, or more than one popup overlapping when the relevant method of JavaScript was called to show it.

This required a rethink of how the project would be structured. I prototyped a method where a template popup was used, and a JavaScript method was used to call this. This method in turn called another JavaScript method which laid out the content of what would be displayed on the popup screen at that moment.

At first I thought this was moving away from the modularity of the system, but I felt that keeping each type of popup within a separate JavaScript method; which could in theory be stored in its own class file, was in keeping with the modularity. The styling of each type of popup was contained within its own code block within the CSS file, at a later date this could be split into separate CSS files, and perhaps this would have been a better choice from the offset.

Colour Blindness

Closer examination is given to this problem within the Results and Evaluation section of the project, however I felt it appropriate to discuss the issue here as initially the problem wasn't raised during the Testing phase of the project, but during a conversation with a colleague in the Student's Union.

The mock ups of the User Interface, and the Design section of the project lay out that the shifts on the rota will be highlighted using a range of colours to differentiate between. I

hadn't during the design stage considered the impact this would have on a user who was colour blind. Showing an early iteration of the system to a colleague lead to a comment that they struggled to tell the difference between booked shifts, and absent shifts on the rota as they were colour blind to red and green.

I set about trying to work out a solution to this problem as I wanted to be as inclusive of all users as I could be. Because I hadn't factored in encountering this type of problem, I had not allocated anytime to fixing it.

I attempted to implement a feature where the user could change the theme used on the rota page to be more accessible if they were colour blind. The design I had in mind was inspired by a copy of the London Underground map I had come across on my travels (http://content.tfl.gov.uk/bw-large-print-map.pdf: accessed February 2016), the map replaces the usual tube line colours with a series of different shading patterns allowing differentiation without the colours.

The feature was not a success, and rather than spend more time on it I made the decision to defer it to a future date.

Clock Changes

Central Services; the current system used by the Students' Union, was not capable of handling the change in time zone the UK goes through twice a year (The clocks advance an hour between late March, and late October). Management had to remember to set shifts after the October change an hour behind what they needed to be in reality, if the shifts where put up before the clocks changed, and vice versa in the March.

I had looked into why this was for my own personal interest before I had thought of developing this system, so I was aware that using PHP to fetch local time zone data could lead to anomalies in time data being input into the system.

In order to overcome this problem I had opted to store all of the times in the database in the GMT time zone, and use the user interface of the system, and the middleware to display these as the correct local times.

Focussing on the conversion from GMT to BST (GMT + 1) I did not factor in the code taking into account the user accessing the system from outside of the UK, and in doing so being in neither time zone.

Better planning in the initial stages of this would have helped me to see that this could be an issues, and save wasting time correcting code that had already been written.

Mobile Devices Rendering

Mobile devices do not usually use the same HTML rendering engine as desktop based browsers do, they use either an entirely different, or a cut down mobile version; such as in the case of iOS's Safari Browser. This can cause issues with rendering parts of a website on a mobile device even though it works perfectly fine on a desktop version.

I had given consideration to this during the Design of the project, and it was a specification that the system should be cross platform, and cross device compatible.

The mine problem encountered with ensuring that the system rendered correctly on a mobile device was the popups section of the site, a careful amount of CSS was needed to make the buttons on the popups more user friendly when rendered on a mobile device screen.

Using CSS to allow the system to render correctly on mobile devices may not have been the correct choice to have made. A better option, and one which would allow the system to be expanded better at a later point in time would have been a dedicated mobile version of the portal.

Unfortunately I didn't allow time in the project to create what are effectively two different websites to drive the portal on the different types of devices, but instead used a shortcut to meet the system specification.

Database Race Conditions

A race condition is defined as "A race condition is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly." (http://searchstorage.techtarget.com/definition/race-condition: accessed April 2016).

As there is only one database backend to the system, but it is accessed by multiple clients the issue of race conditions occurring was a known factor in the implementation of the system, and steps were taken to mitigate against circumstances where it was known they could occur.

What wasn't factored in was the sheer amount of race conditions that could occur within the system, and the effects these could have on the back end data, and therefore the output that the system was giving.

Though it is referenced more in depth in the Results and Evaluation section I feel it is appropriate to highlight it here as an example. *User A opens up there rota and sees a shift on a particular time starting at 21:00. Meanwhile User B opens up there rota and sees the same shift starting at 21:00. User A clicks and books the shift. User B still has the option on their screen to click and book this shift, they do so, and as far as they can see the shift is confirmed as booked. User B later returns to the rota page and finds no evidence of the shift they have previously booked.* This scenario was caused by a particular race condition because no validation was made to check if the shift had been booked already when the book button was pressed by User B. Later when User B returns to the rota and it queries the shifts they have booked it fails to display the shift that was double booked, as the query returns only one record of the shift being booked, and that Is against user A who's record was inserted into the database first (of course the scenario could be reversed and User B's record could have been inserted first).

This was a difficult race condition to overcome, as I wasn't able to understand at first why the shift wasn't displaying for both users. A lot of time was spent attempting to debug this

issue. The cause of it was the query on booked shifts only returning one record, where it was actually possible in the database to have two; or in theory more, records.

Hindsight is a wonderful thing, but in future projects I would encourage the use of a state diagram to understand exactly how the system is entering into these conditions, and planning around them before the implementation begins. In this project rather than do this, I simply coded out race conditions as I found them, which is not a consistent approach as some may not be noticed, even in testing.

5 Results and Evaluation

Test Plan

The following outlines the testing that will be done on the system to confirm if the system meets the required functionality of the specifications.

Test 1	Test whether a user can book a shift
Environment	Windows 10 – Google Chrome
Specification Number	1
Pre-Conditions The user is logged into the system	
Process 1. Select the rota view 2. Select an available shift 3. Select the book option 4. Return to the rota view 5. Confirm the shift is now highlighted as booked	
Expected Outcome The shift should change from being bookable, to highlighted as booked on the rota view	

Test 2	Test whether a user can book a shift for which they do not have the correct rank
Environment	Windows 10 – Google Chrome
Specification Number	1
Pre-Conditions The user is logged into the system with the rank of venues assistant At least one Team Leader shift is available to be booked	
Process 1. Select the rota view 2. View the day where the Team Leader shift is available for 3. Confirm if the shift is available to view 4. If it is available to view attempt to book it	
Expected Outcome The Team Leader shift should not be viewable when logged in as a Venues Assistant. In the unlikely event that is is viewable it should not be possible to book the shift.	

Test 3	Test whether the timeout on reserving a shift before booking it works
Environment	Windows 10 – Google Chrome
Specification Number	2
Pre-Conditions The user is logged into the system At least one shift of an appropriate rank is available to book The second user is logged into the system	
Process 1. Select the rota view	

2. Click on a shift that is available to book 3. Remain on this page 4. Log into the system as another user on a second device 5. Select the rota view 6. Confirm whether the reserved shift can be seen or not 7. Wait for the reservation timeout to expire 8. Confirm whether the reserved shift can be seen or not
Expected Outcome When the second user attempts to view the shift before the reservation timer has expired it should not be visible. After the time has expired and they attempt to view the shift it should be visible

Test 4	Test whether the estimate for a member of staffs pay for the shift displays when viewing a shift
Environment	Windows 10 – Google Chrome
Specification Number	3
Pre-Conditions The user is logged into the system At least one shift of the appropriate rank is available to book	
Process 1. Select the rota view 2. Select a shift that is available to book 3. Confirm whether the estimated pay is shown 4. Confirm if the amount of pay is correct	
Expected Outcome The amount of pay should be correctly shown as the number of hours the shift is multiplied by that member of staffs pay rate	

Test 5	Test whether a member of staff is able to record the time at which they have started and ended a shift which they already have booked on the system
Environment	Windows 10 – Google Chrome
Specification Number	4
Pre-Conditions The user has a booked shift	
Process 1. The user should login to the system 2. Select Time Clock 3. Select Clock In 4. Wait for the shift to finish 5. Select Time Clock 6. Select Clock Out 7. Confirm the shift has changed to completed on the rota view	
Expected Outcome	

The shift on the rota view should change colour to confirm that there are no time clock issues with this shift

Test 6	Test whether a member of staff is able to the record the time at which they start and end a shift which they didn't have booked previously, and that they can record the member of management who asked them to come in.
Environment	Windows 10 – Google Chrome
Specification Number	4
Pre-Conditions The user does not have a booked shift	
Process 1. The user should login to the system 2. Select Time Clock 3. Select Clock in 4. The system should prompt for the user to select a manger name, and a location 5. Wait for the shift to finish 6. Select Clock Out 7. Confirm the shift has appeared on the rota view	
Expected Outcome The shift should appear on the rota as a completed shift. Clicking on it should show the manager who authorised it, and the location which was worked in the venue	

Test 7	Test whether the user is alerted to missing one time clock for a shift which they had booked, and that they can submit a correction report
Environment	Windows 10 – Google Chrome
Specification Number	5
Pre-Conditions The user has a booked shift The user only recorded the time at which they started the shift	
Process 1. The user should login to the system 2. Select the rota View 3. Look at the shift where they missed the time clock 4. Confirm it is displayed with the correct colour for missing a single time clock 5. Click on the shift 6. Click submit time clock correction 7. Fill in the time and submit 8. Confirm that the shift changes to a standard completed shift	
Expected Outcome The shit should initially be displayed on the rota view as having missed one time clock. Once the correction report is submitted it should change to being a completed shift	

Test 8	Test whether the user is alerted to a missing both time clocks for a shift which they had booked
Environment	Windows 10 – Google Chrome
Specification Number	5
Pre-Conditions The user has a booked shift The user failed to record the time at which they started and ended the shift	
Process 1. The user should login to the system 2. Select the rota view 3. Look at the shift where they failed to clock in and out 4. Confirm it is displayed in the correct colour for an absent shift	
Expected Outcome The shift should be correctly shown as the member of staff having been absent	

Test 9	Test whether a manager is able to add a shift to the pool of available shifts to be booked
Environment	Windows 10 – Google Chrome
Specification Number	6
Pre-Conditions The user is logged in as a manager	
Process 1. Select the rota view 2. Select add new shift 3. Fill in the details as appropriate 4. Click submit 5. Log in as a Venues Assistant on another device 6. Confirm that the shift is visible	
Expected Outcome The shift should be available to be booked by the Venues Assistant User	

Test 10	Test whether a manager is able to edit the details of a shift
Environment	Windows 10 – Google Chrome
Specification Number	6
Pre-Conditions The user is logged in as a manager At least one shift has been added to the rota	
Process 1. Select the rota view 2. Select any available shift on the rota 3. Edit some of the details about the shift 4. Click submit 5. Return to the rota view and confirm the details have correctly changed	
Expected Outcome The details about the shift should correctly change to the new values	

Test 11	Test whether a manager is able to delete a shift
Environment	Windows 10 – Google Chrome
Specification Number	6
Pre-Conditions The user is logged in as a manager At least one shift has been added to the rota	
Process 1. Select the rota view 2. Select any available shift on the rota 3. Click delete 4. Return to the rota view and confirm the shift has been deleted	
Expected Outcome The shift should be removed from the rota view	

Test 12	Test whether a manager is able to add a new location to the system
Environment	Windows 10 – Google Chrome
Specification Number	7
Pre-Conditions The user is logged in as a manager	
Process 1. Select admin 2. Select add new location 3. Fill in the location name 4. Fill in the default rank for this location 5. Select rota view 6. Select add new shift 7. Confirm that the new location is available in the list of locations	
Expected Outcome The new location should be available for use in the add new shift screen	

Test 13	Test whether a manager is able to add a new rank to the system
Environment	Windows 10 – Google Chrome
Specification Number	7
Pre-Conditions The user is logged in as a manager	
Process 1. Select admin 2. Select add new rank 3. Fill in the rank information 4. Select the rota view 5. Select add new shift 6. Confirm that the new rank is available from the list of ranks	
Expected Outcome	

The new rank should be available to select from the list of ranks needed to be able to book that particular shift

Test 14	Test whether a manager is able to stop a particular rank from being used
Environment	Windows 10 – Google Chrome
Specification Number	7
Pre-Conditions The user should be logged in as a manger	
Process 1. Select admin 2. Select edit ranks 3. Tick the box on a rank to mark it as dormant 4. Select the rota view 5. Select add new shift 6. Confirm that the rank that has been marked as dormant is not available to choose from the list of ranks	
Expected Outcome The rank that has been marked as dormant should not be available for selection	

Test 15	Test whether a user is able to put a shift up to be dropped, and that another user is able to pick up that shift from the original user
Environment	Windows 10 – Google Chrome
Specification Number	8
Pre-Conditions The user is logged in They have booked at least one shift The booked shift is at a time when no other shifts are available to book	
Process 1. Select rota view 2. Select the shift they wish to drop 3. Click Drop Shift 4. Return to the rota view 5. Confirm that the shift has changed to the pending dropped colour 6. Login as a second user on another device 7. Confirm the dropped shift is visible 8. Book the dropped shift 9. Confirm that it shows as booked on the second users rota view 10. Return to the first users rota view 11. Confirm that the shift is no longer visible on their rota view	
Expected Outcome The shift that has been dropped should initially change to the dropped colour on the first users rota view. The shift should then be visible as a shift to be booked on the second users account.	

When the second user books the shift it should change to the booked colour on their rota view. When the first user checks their rota view the shift should have been removed from it

Test 16	Generate a payroll report under a controlled set of conditions to confirm that the output format and information format is correct
Environment	Windows 10 – Google Chrome
Specification Number	9
Pre-Conditions The user is logged in as a manager Four different users have booked shifts as follows: User 1 (TL) – 1 shift 6 hours long User 2 (VA) – 1 shift 6 hours long User 3 (VA) – 2 shifts both 6 hours long User 4 (VA) – 2 shits the first 4 hours long, the second 6 hours long The four users have correctly recorded the time clocks for these shifts	
Process 1. Select admin 2. Select run payroll report 3. Select the payroll cycle where the above shifts have been booked 4. Select to run the report 5. Open the report in an appropriate CSV file viewer (Notepad, or Microsoft Excel) 6. Confirm that the payroll report has generated correctly	
Expected Outcome The payroll report should have been output in CSV format, with each different member of staff on a line, with their first name, last name, pay rate, amount of hours, and total payment shown. These should all be correct in line with the Pre-Conditions laid out above	

Test 17	Test that the load balancing feature correctly balances the staff across two locations as set out in the controlled conditions for test
Environment	Windows 10 – Google Chrome
Specification Number	10
Pre-Conditions The user is logged in as a manager Two locations are in use as follows: Location 1 – Optimal number of staff 6 Location 2 – Optimal number of staff 3 Shifts booked my members of staff as follows: 8 shifts all allocated to Location 1	
Process 1. Select admin 2. Select load balancing 3. Select the time period to run the load balancing for 4. Select to run the load balance	

5. Wait for confirmation it has been executed 6. Select the rota view 7. Confirm that the staff have been moved into the two locations
Expected Outcome After the load balance has been executed Location 1 should have 5 staff allocated to it, Location 2 should have 3 staff allocated to it.

Summary of Test Results

During the implementation stage of the project a number of tests sessions where carried out with a small amount of users, this did not form part of the formal testing plan as they were not part of the formal system acceptance testing. The purpose of those tests was to get direct user feedback on the layout of the system, and whether they felt the system was self-explanatory to use.

After the implementation stage, formal testing of the system took place in accordance with the test plan outlined above. The results from this are summarised in the table below.

Test No	Test	Outcome	Notes
1	Test whether a user can book a shift	Pass	Expected Outcome
2	Test whether a user can book a shift for which they do not have the correct rank	Pass	Expected Outcome
3	Test whether the timeout on reserving a shift before booking it works	Pass	Expected Outcome
4	Test whether the estimate for a member of staffs pay for the shift displays when viewing a shift	Fail	This test was only partially successful. Spurious results were observed when a Team Leader was attempting to book a Venues Assistant shift, on occasions the correct rate of pay was shown, on other occasions incorrectly the Venues Assistant rate was being displayed to the Team Leader.
5	Test whether a member of staff is able to record the time at which they have started and ended a shift which they already have booked on the system	Pass	Expected Outcome
6	Test whether a member of staff is able to the record the time at which they start and end a shift which they didn't have booked previously, and that they can record the member of	Pass	Expected Outcome

	management who asked them to come in		
7	Test whether the user is alerted to missing one time clock for a shift which they had booked, and that they can submit a correction report	Pass	Expected Outcome
8	Test whether the user is alerted to a missing both time clocks for a shift which they had booked	Pass	Expected Outcome
9	Test whether a manager is able to add a shift to the pool of available shifts to be booked	Pass	Expected Outcome
10	Test whether a manager is able to edit the details of a shift	Pass	Expected Outcome
11	Test whether a manager is able to delete a shift	Pass	Expected Outcome
12	Test whether a manager is able to add a new location to the system	Pass	Expected Outcome
13	Test whether a manager is able to add a new rank to the system	Pass	Expected Outcome
14	Test whether a manager is able to stop a particular rank from being used	Fail	The rank that had been marked as dormant was not being removed from the list of available ranks for a new shift. Though when submitting a shift with a rank that had been marked as dormant selected the rank field was incorrectly being stored as Null rather than with the dormant value
15	Test whether a user is able to put a shift up to be dropped, and that another user is able to pick up that shift from the original user	Pass	Expected Outcome
16	Generate a payroll report under a controlled set of conditions to confirm that the output format and information format is correct	Pass	Expected Outcome
17	Test that the load balancing feature correctly balances the staff across two locations as set out in the controlled conditions for test	Fail	The load balancing feature was designed with the intention to have each location as close as possible to 100% staffing. Incorrectly during the test Location 1 was given 6 staff, and Location 2 was given 2 staff. Location 1 should have

			been given 5 staff, and Location 2 3 staff

Failed Test Results

The tests which were not fully successful are discussed in more detail here outlining exactly why the test had failed, and what actions have been taken to debug the problem, and the solutions that are proposed.

Test 4

This test was successful all of the time if a Team Leader looked only at shifts with a minimum rank of Team Leader, and a Venues Assistant looked only at shifts with a minimum rank of Venues Assistant. The issue arose when a Team Leader looked at a shift with a minimum rank of Venues Assistant.

At first it was not clear why the error did not arise all of the time when a Team Leader was looking at a Venues Assistant shift. I asked the user to attempt to recreate the issue a number of times before I was able to ascertain the problem.

If he first shift that was looked at when a session started was a Venues Assistant shift by a user with Team Leader rank the pay rate was being incorrectly set globally for the session at the Venues Assistant level. If the first shift looked at was of Team Leader rank the rate was being correctly set.

A solution to this problem is to refactor the code that queries the users pay rate to being executed as soon as the rota view loads, and to query the individual user rank, rather than being executed when a shift is queried, and the pay rate being based on the rank of the shift rather than the user.

Test 14

This test was never successful, the rank which had been marked as dormant was never being removed from the list of ranks that where available to use. However if a shift had been created with a dormant rank, no rank was displayed when the shift was viewed by a manager account, and this also had the side effect that it was not possible for anyone to book the shift.

Time was needed to examine the back end database to see that the Null value was being assigned to a shift when a dormant rank was picked, this implied that somewhere along the line the system was coded to intercept a dormant rank, but it wasn't catching the error fully.

Analysis of the code showed that the method which was called to create a new shift was validating that the rank that had been selected wasn't already marked as dormant within the database at the time of submission; this allowed for the scenario where another manager was editing the ranks, whilst one was setting up shifts. What wasn't being checked was if a rank was marked dormant when it was queried to populate the form fields. This was in itself not incorrect as if a manager was editing a shift created before a rank was marked dormant it would still allow the rank field to be shown correctly.

A solution to this bug is to refactor the code that runs when the shift window is loaded, and to mark any ranks which are flagged as dormant in the database with "(Dormant)" on the

end of their string value. A small amount of JavaScript can then be used to validate that a dormant rank has not been selected by the user when they go to submit the shift form.

Test 17

This test was not successful in the controlled conditions as set out in the Test plan. To further investigate the issue and to understand exactly why the test results did not match the expected results a series of three additional tests where setup.

Two out of the three of these additional tests passed, and gave the expected results, the third did not give the expected result.

The link between these tests was that they only failed if the last location in the load balancing had a small number of optimal staff, and therefore its load balance percentage was greatly influenced by removing one member of staff.

I examined the algorithm used in the load balancing part of the system, and determined that it wasn't checking all possible outcomes of its actions. The algorithm works through each location in turn, and attempts to get each as close to the optimal figure as possible. If it left with an odd number of staff on the final pass through, the last location is left one member of staff short in comparison to the others, leading to this type of result.

A way to factor this out will require a redesign of the algorithm so that on the final pass through it compares the different combinations of load factor percentages before making a decision on where the final members of staff should be placed.

6 Future Work

Hofstadter's law states that – "Everything takes longer than you think, even when you take into account Hofstadter's Law." This certainly held true throughout various parts of this project, more noticeably on issues that were initially expected to take a short amount of time to implement or resolve, perhaps the expected simplicity of implementation caused me to underestimate the exact amount of time which I would need to complete the tasks.

This section is split into three parts, future work which directly impacts upon the work that has already been implemented, and either improves it or expands it, work which could be implemented in the future, which either would bring entirely new functionality, or a new feature which relies on the basic functionality already contained within the system, and future expansion, which looks at how the system could be modified for use by other organisations.

Existing Features

The Results and Evaluation section highlights the bugs in the software which were found during the testing which took place. Some of these where resolved during the development iterations which took place after the Results and Evaluation stages where finished. Some of these weren't fixed during those iterations, and are still outstanding for resolution. It is important that as the first stage of any future work that these are resolved to ensure that the software is robust, and working as intended. As part of this I will investigate and implement a bug tracking platform to ensure that any issues found with the software during its deployment and future development is correctly recorded and tracked to ensure they are not forgotten about or overlooked.

Currently the system uses its own login system in order for members of staff to gain access, this was done primarily because although Cardiff University Student's Union is supported by Cardiff University, it actually employs student staff from any University within Cardiff, though the vast majority of staff are Cardiff University Students. Cardiff University provide accredited services with access to their central login database, which can be used to allow people to login to services using there standard University login credentials. The existing login system could be maintained for none Cardiff University students and staff, and integration obtained from the central Cardiff University. This would only require a small change to the backend database, and would have the benefit of allowing fall back to the systems in built login system if the Cardiff login system failed.

The payroll reports need to be run manually by a manager, or member of the admin team, though a warning is provided by the system if the report hasn't been run X amount of time before the pay day is due. When the system was being designed and specified I considered automating the running of the payroll report, either in its entirety, or automating the execution of it, with a report of anomalies being produced to be manually checked before the report is considered fully completed. Automating the report in any way adds risk to the business if it fails to run. Staff may not be paid on time which could cause a range of emotional responses such as anger or panic. The system would need to be carefully coded to ensure that if the automated execution of the report failed, there would be prompts

given to fall back to a manual process in time for the situation to appear to the staff as normal. The first stage of implementing this feature would be to partially automate the execution of the report, with the second stage being complete automation.

Shifts swaps are currently possible at any time within some constraints; the person who is taking the shift can't pick up a shift that has been swapped away if there is another shift of their rank available within the same time frame, and a shift cannot be put up to be swapped away if management have set a blackout on shift swaps for that particular time period. Although this removes the need for management to manually intervene when a shift swap needs to occur, it only cuts down on the abuse of shift swapping where management have set a black out period, this then puts an absolute ban on shift swaps, where in fact one or two may be permissible. Expanding the shift swapping function to allow management to authorise an individual swap would be an advantage, it would be much quicker than the old method of dealing with these over emails, but would still allow careful control. The functionality to generate a report on shift swaps would also be a useful expansion to this part of the system, this would need to print out each member of staff, with a line by line account of the shifts they have swapped, with a summary per member of staff of how many shifts they have swapped away, and a total at the end of the report of how many shifts have been swapped away overall.

Additional Features

With the system configured as it is at the moment a member of the management team, or the admin team must sit and set up each individual user account when a new member of staff is hired. This is a time consuming process as the recruitment intake is usually done in two large batches 6 months apart. These intakes can be up to 300 people at a time. A way to ease this process would be to allow staff to pre fill in the information that is required from them before they attend an induction day, on attending an induction day a member of the Management Team or Admin Team could then just quickly verify the information, rather than having to wait for it to be input. Information not directly pertinent to the business, such as the person's name or address wouldn't require to be verified, only information such as the rank of the person would need to by physically verified, this would greatly speed up the process of enrolling new staff onto the system. This would require expansion to the staff registration part of the system, with an extra table needed in the database to confirm that management have verified the information that was prefilled in.

Team Leaders have little more access on the shift booking system when compared with Venues Assistant Staff, for the moment this is fine, and meets the need of the specifications of the system. A large change to the system would be the ability for Team Leaders to have a greater level of control over the staff who are working on a particular shift. At the moment staff are only load balanced across the bars when it is specifically requested of the system, and this only works on the basis of moving a member of staff for the entire shift, not for a short period of time. Implementing a way for Team Leader on a particular location to either request extra staff, or to say they have a surplus of staff and alerting other Team Leaders in the venue via a push notification would be a useful addition. This would allow staff to be balanced across locations where a particular surge in sales has been noted. This would

require either an additional database table which would store the notifications that would need to be sent out, or inclusion within the database tables used for the push notifications section as discussed in this section.

Records of what training a member of staff has received are currently stored in paper format, and are difficult to search through, or correlate exactly what members of staff have completed what training. An additional part of the system could be setup to store a record of the dates that training was attended and completed. Alongside this a method of querying this information into a structured set of reports would be useful. The only link between this part of the system, and the existing parts of the system and database would be a relational link back to the member of staff's employee ID. This means that this part of the system could be built, and tested independently of the rest of the system. Though it would be accessed through the same web based portal as the rest of the system. Individual members of staff would also be able to see the information that is related to them on hat training they have completed and when.

It is not possible within the current system for a member of staff to edit there personal details, these are nominally fixed when they enrol on the system, and can only be adjusted by a member of the management team, or admin team. Opening up the ability for a member of staff to edit there personal details does open up risks to the integrity of both the system, and of the data within the system. For example if a member of staff changes there bank details between the payroll report being run, and the pay being received, they may not receive the money into the bank account they wished too. Careful consideration would need to be given as to how to control these risks, and mitigate against them causing a problem for the system. It may be that a lock needs to be put on certain parts of the data when other reports are being run, or within a certain time frame in relation to payroll reports being generated.

The current help documentation from the system is not interactive, it is a static set of documents which talk the user through how to complete various tasks within the system. The documentation is split into two parts, one part deals with the tasks that student staff need to carry out, the other parts deals with the tasks and operations that the Management Team, and Admin Team carry out. Whilst the documentation for the Management Team tasks is adequate for them, as they carry out the various tasks regularly on a day to day basis, the documentation for the Student Staff could be improved upon by providing an interactive tutorial that can be launched to demonstrate how to carry out booking a shift, swapping a shift, and correcting a time clock error.

Future Expansion

Though the system was written for use by Cardiff University Students' Union, none of the User Interface, nor the middleware, or backend database make any formal reference to the organisation.

If an organisation had a similar operational structure to the Students' Union, it would be a relatively trivial matter to deploy the system onto a webserver for that business, and to

configure it with the staff rank structure, and location structure used by the new organisation.

In the case where a second organisation takes on the VASP system, and serious consideration is given to more organisations coming on board it would be worth investigating whether a cloud based deployment option would be better in the long run than running each organisations system on their own dedicated web server. Deploying each organisations copy to a new webserver would reduce issues with server load if a large number of users was attempting to access the one cloud server from different organisations, on the other hand deployment of software updates to fix bugs or add new features would be more difficult over multiple servers.

If an organisation wished to use the system but had a different operational structure to the Student's Union, an analysis would have to be carried out as to exactly what would need to be changed, and whether a modified version of VASP would be appropriate, or whether it would be better to start with a new system; though almost certainly modules from VASP could be reused to achieve this where appropriate.

7 Conclusions

The aim of this project in short was to design, specify, and implement a shift booking and payroll management system for Cardiff University's Students Union. A personal aspect of the project was to improve my own knowledge on implementing a system for a real world client, following a specified software development model.

Did the project achieve its specified aims? The project certainly implemented a shift booking and payroll management system, in line with the vast majority of the mandatory specifications which were laid down in the Design stage of the project. The system did not at the time this report was written achieve all of the mandatory specifications, and some minor bugs where present in the rest of the system, these where highlighted in the Results and Evaluation Section, as well as the Future Work section.

Did the project allow me to improve my own knowledge of implementing a system for a real world client, whilst at the same time following a specified software development model? It did, to a much greater degree than I originally anticipated. Having taken part in group projects during Year 1 and Year 2 of my degree course I felt I had an understanding of how dealing with a client for system implementation worked. Though these had giving an understanding of how dealing with a client worked, it did not prepare me for how particular a none technical savvy group of clients could be, especially when it came to the design of the User Interface. When I selected the Modified Waterfall Development model I stated that I would need to carefully manage the amount of input the user had into the development process to ensure any constant tinkering in none essential parts didn't lead to the development process being slowed down. Having an awareness of this allowed me to intervene when the end client tried to make this type of decision, and prevented time being wasted on none essential issues.

Did I handle the time keeping of the project well? As alluded to in other sections of the report, the time keeping I displayed in parts of the project left some room for improvement, and this is something I think I've learnt the most from overall. Going back to the original project plan, I feel the amount of time which I allocated to certain parts of the project was underestimated; such as the implementation of the database structure, whilst other parts where overestimated; such as the design of the user interface. Overall I have managed to deliver a system which meets the objectives of the Students' Union, however I would have preferred to have implemented some of the optional specifications by this point, in time for the system to be fully rolled out by September 2016. The delay in some parts of the system may mean the system will need to go live without all of its features.

Was moving away from frameworks the correct decision? As I had no previous experience in using frameworks to deploy a system, I felt that the learning curve would be too great in order to deliver the project on time, and I wanted to gain a better experience in hand coding a project rather than using existing frame works. In hindsight the time taken to solve some of the relatively small parts of the project; looking at the project as a whole, could have been easily resolved by using a framework, and the lost time in the learning curve would have been outweighed by the time taken to hand code a solution. If I was to go back and do

a similar project I would give a more serious consideration to using frameworks to save time on sections that could be done without coding something that a framework could handle. However in spite of this, I do feel that coding some of the more mundane parts of the system such as the login part, gave me a greater understanding of how these parts work, and the security implications of them.

In summary I believe I delivered what I essentially set out to do. Though I missed the opportunity to deliver any of the optional specifications within the time frame of the project report. I have gained a wealth of experience in dealing with a real world client, and the pit falls, and strategies that can be used to mitigate against them. Time keeping was also highlighted as an area that I struggle with, the project has taught me to greater assess the time I need to complete a particular in the planning stages, through further research to allow a more accurate estimate, and also to build in extra time to account for a reasonable amount of delay in the project delivery for any unaccepted problems that arise. Frameworks may have allowed me to deliver parts of the project quicker, and I would investigate there use more in future projects.

8 Reflection on Learning

One of the things that strikes me most about this project was the insistence by the Students' Union on sticking with a completely web based approach to the problem, with the web portal being the primary means of accessing the system, with consideration being given to an app at a later stage. With the prevalence of smart phones in society especially in the age bracket of staff which they employ I would have expected an app to be a more suitable choice. This is backed by the fact the other two solutions I found where primarily app focussed; although Homebase did allow we portal interaction as a secondary method. Persuading the Students' Union that an app would be a better choice may have led to a more functioning and practical solution to the problem, and also allowed myself a personal exploration into gaining experience in app development, rather than a field which I had already had some experience in.

The other insistence of the client was that the web based portal remained focused on a calendar view as the primary means of interacting with the system. They justified that this would be an easier transition for staff from their current system, and that it would prevent them needing to carry out a large amount of training with the staff. I can understand this view, but by making this a system specification they limited the amount of creativity I had to design the user interface, and it could almost be considered that this has stifled the ability of the system to grow and adapt to later trends.

Both of these things have shown me that I need to be more assertive when dealing with a client. They believed as outlined above that these where the right decisions to be made in respect to their point of view, if I had put my point of view across more strongly, it may have led to the system going in a different direction which would allow for a greater range of features, and also bring a fresh look to the business rather than sticking with what is known.

One of the problems faced by the system was the need for anyone to be able to login, as the Students' Union employs people who don't always attend Cardiff University. I felt it was easier to have everyone use the same login system, rather than attempt to integrate the Cardiff University login portal into the system, alongside a second login system. As I identified in the future work section it would not actually take much to integrate the Cardiff University login system to act as the login method for the vast majority of people who use the system. On one hand the decision to ensure all users could at least access the system in the first instance seems like the correct one, but a greater level of functionality could have been obtained by starting with integrating the University login system first.

The decision not to use frameworks in the system in hindsight does seem to be the wrong choice, and not just on a programming level. In order to give myself an easier time I stuck with what I knew rather than exploring new options, this in itself is fine, and could have led to a system which was implemented much faster than learning a new way to implement it. However in this case most of the problems which slowed my progress where items which would have been handled by the frameworks, and the amount of time spent learning how to use a framework, would have more than made up for the time lost. It would also have

given me a better personal experience, and given me confidence in using frameworks in other projects that I undertake later on in my career.

Both of these points show that I should perhaps spend more time researching exactly what something can achieve, or how something can be achieved in various different ways, rather than sticking with the first solution I find which seems to achieve what I need it to. That is not to say that I don't carry out research, but that I tend to stop when I find a solution which will meet the requirements, even If another solution might suit the requirements more.

9 Glossary

Venues Assistant – The workhorse of the Students' Union, they are responsible for staffing all of the locations within the Student's Union.

Team Leader – The Team Leaders are employed to provide supervision to a location within the Students' Union, they act as a supervisory layer between the Venues Assistants and the Management Team.

Manager – A Manager is a salaried employee of the Students' Union, they have day to day responsibility for running of the Venues Department.

Admin Staff – A member of admin staff is a salaried employee of the Students' Union, they are employed to provide administrative support to the Management Team, and the Venues Department.

Management Team – The Management Team is made up of seven managers, they are jointly responsible for running the Venues Department

Admin Team – The Admin Team is made up of the Admin Staff.

Student's Union – Cardiff University Students' Union.

Venues Department – The Venues Department is made up of the Taf pub, the Y Plas night club, the Great Hall music venue, and various catering outlets throughout the building.

Central Services – The shift booking system used by the Venues Department at present. It is intended for this to be replaced by VASP.

10 References

[Online] www.Makeshift.CA [January 2016]

[Online] www.joinhomebase.com [January 2016]

[Online] http://searchbusinessanalytics.techtarget.com/ [January 2016]

[Online] http://www.php-login.net/ [January 2016]

[Online] https://github.com/panique/huge [January 2016]

[Online] https://addons.mozilla.org/en-gb/firefox/addon/noscript/ [January 2016]

[Online] http://content.tfl.gov.uk/bw-large-print-map.pdf [February 2016]

[Online] http://searchstorage.techtarget.com/definition/race-condition: [April 2016]