Developing mobile and web services for shift swapping, overtime, holiday and shift management at Sainsbury's

Submitted May 2013, in partial fulfilment of the conditions of the award of the degree BSc Honours Computer Science.

Naomi Shephard-Dodsley
nxs10u

Supervisor: Dario Landa-Silva

University of Nottingham
School of Computer Science and Information Technology

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature: Naomi Shephard-Dodsley

Date: 07/05/2013

Abstract

This project aims to produce a mobile and web system to aid workforce management at Sainsbury's by allowing managers and employees to manage schedules, shift swaps, holidays and overtime. The motivation for this has come from working at Sainsbury's for 3 years part-time, and so the shortfalls of the current system have become apparent enough to create a system to address these issues.

This project aimed to follow the structure of a software engineering project and focus on the completeness and thoroughness of the final deliverables; in terms of, not only producing a complete useable system, but also documenting processes such as research, design and testing etc in great detail.

The difficulty of this project lies within producing both desktop and mobile web technologies that can communicate with each other, and perform the same functionalities, whilst following a structured engineering process. The wide variety of devices that the system is aiming to be operable on also calls for a clear design process and method of effectively converting desktop, larger, layouts into concise, useable, mobile designs.

The project began by outlining a brief description of the problem, the motivation and aims for this project. This was important in order to gain a clear understanding of what the project was aiming to achieve. From this, the analysis of current systems was necessary in order to gain a more thorough understanding of what worked well in these kinds of systems, and what should be avoided. This also re-instated the need for the proposed system after seeing the disadvantages of how other systems worked and were presented to the users.

This project not only used research into existing systems but also carried out a literature review into motivation theories and how staff morale can be affected by the current system at Sainsbury's, emphasising the need to produce a solution. Along with this, effective interface designs for both mobile and desktop interfaces were studied, as an important part of this system is how it is presented to the end users.

From this, a requirement specification was created to address the problem. This allowed a structured, in detail, design process to be followed, as there was a clear set of requirements that the interfaces and functionalities must meet. The implementation then built on this design process, following the designs, however using an iterative approach to adjust and re-factor anything that didn't work quite as well as expected from the designs.

Finally, testing and evaluation were conducted in order to not only analyse personally what had been achieved, but also in terms of the functional and non-functional requirements, and the end user's thoughts on the system as a whole.

Acknowledgements

I would like to thank my supervisor Dario Landa-Silva for taking his time to supervise this project, and for offering support and guidance throughout the year.

Contents

List of Figures

Chapter 1 - Introduction

1.1 Description of the Problem

There are undeniably many systems already available to aid in workforce scheduling and management. However, from working part-time for almost 3 years at a supermarket I have gained an understanding of what some of these systems are lacking, and in particular how the current system at Sainsbury's could be improved.

During preliminary research it was found that there are many systems revolving around workforce management and helping the manager in doing so. However, something missing from the findings were systems that also fully focused on the needs of the employees, such as the ability to swap shifts and features involving holiday and overtime booking. This issue is also present in the current system at Sainsbury's and so therefore there is a need for a system which aids managers and employees alike, not just one of them.

Full-time staff can sort out any issues they may have relatively easily as they are at work up to 5 out of 7 days a week and will see their manager regularly. However, part-time staff find this much more difficult due to only being at the workplace on average 2 days a week, during a limited section of the working day; as shifts can range from 5am through to 10:30pm in most job roles. This means that it's very easy to miss your manager and not see them for up to weeks at a time, making it difficult to resolve any issues or queries. This results in the level of communication weakening, with employees becoming de-motivated, and in turn emphasising the need of a communication tool that can be accessed outside of work, especially for part-time staff.

Common scenarios that employees and managers may be involved in are:

- Holiday bookings
- Shift swapping
- Overtime allocation
- Queries regarding individual shift times

Although there are currently methods in place for all of these scenarios, the process is very informal and often relies on a verbal agreement, therefore making it very unreliable. For example, someone could agree to swap a shift, and then later on say that they never agreed to do it, leaving you in a difficult position. Taking this into account, the aim of the project will be to create a system that addresses the above issues and will now discuss further this will be achieved.

1.2 Approach to the problem

One of the most important parts of the proposed system is that it is available for employees and the manager, if they wish, outside of work. Most part-time employees tend to be students or young adults and so usually have access to Internet on their phones. This means that a website accessible on mobile devices is good for the target market, as they will easily be able to do things on the go. A

standard web layout will also be created so that it could be implemented inside the workplace on computers there, or for people from home who may not have access to the Internet on a mobile device.

Whilst using an algorithm can be efficient, the manager often knows best where they want to place employees in the schedule and at what time, dependant on varying factors each day, week, or month, and feel that this is an area where human input is important. This is partially coming from the experience gained from working at Sainsbury's, in knowing that managers are provided a template schedule generated from the HR department, but even after this they have to make multiple changes to this by hand. Therefore, a better system is one that provides a tool for the managers to create their own schedules or to edit a template one, rather than it being created for them, so the proposed system will aim to incorporate this.

In terms of the project as a whole, the focus will be on developing a complete, usable, system that is appealing to the intended users.

In order to achieve this, the main difficulties will be:
1. Performing a lot of testing to ensure the system is robust.
2. The design of the system must be carefully considered with a lot of thought going in to how the interfaces can be created to best suit mobile and desktop devices.
3. Creating a system that meets the needs of different types of users through different features and functions.

1.3 Aim and Objectives

The aim of the project is to develop a system for the management of employees at the Sainsbury's Kimberley store. Although the system is based around one department in one store, the same principles will apply throughout other departments and across different stores. The system should give managers and employees the ability to create workforce schedules, allocate overtime and holidays, and request shift swaps, respectively. The system will be accessible through a web browser on laptops/computers or with an optimised layout when accessed on mobile devices in order to improve accessibility.

Taking these points into account, the main objectives of the system will be the following:

1) Employees and managers can log in and view upcoming schedules.
2) Employees can see a version of the schedule comprised of just their shifts.
3) Managers should be able to log in and create schedules on the website, taking into account information they know about the employee, for example holidays, overtime etc.
4) Employees can request holiday on the system, with the manager able to see who else is on holiday and either accept or reject the request. If it is accepted, a flag will be put against the employee for that date, so that when the manager is making the schedules they can see who is on holiday and schedule around them.

5) Employees can view their schedule online and will let them request a shift swap with another employee by sending a notification to them. If the second employee accepts then the request will finally go to the manager to check that they are happy with it and accept the swap.
6) Overtime can be broadcast on the system, with employees able to request to do the shift.

1.4 Motivation

Having worked at Sainsbury's for almost 3 years, a good understanding has been gained of the current system and the way in which this can often affects staff morale in a negative way. The common scenarios of shift swapping and overtime allocation can happen on a daily basis, and the fact that there is no formal way of processing this can often lead to a decrease in staff satisfaction. This is because a lot of effort has to be spent arranging a satisfactory outcome between managers and employees, especially in terms of shift swapping. Alongside personal motivation, there has also been positive feedback from others, shown in Chapter 2.1 and Appendix 1. Therefore, this project will be very rewarding knowing that I, and others in the same position, would benefit from the completed system if it were actually to be implemented in the workplace. The challenge of creating a system that is appealing on both a desktop and also a mobile device is also part of the motivation.

Chapter 2 - Research

2.1 Informal Target Market Research

In the initial stages of this project, some informal target market research was conducted in order to gain an insight into other people's opinions on this topic and to get clarification that this would be a useful system to others.

The first part of this was to make an informal post to my blog, asking my followers to reply with their opinions on the system proposal (See Appendix 1 for the entry and replies). The answers received from this were very encouraging, with users who currently don't have a system like this at their workplace being very positive towards the idea. A number of users mentioned in their replies that their workplace currently has systems similar to the proposed idea, and so I messaged them separately asking for more information on their system. Some of the users replied with general comments on what their system does, ranging from verbal agreements, to some electronic organisation of holidays. However what became apparent is that, like the system at Sainsbury's there is a lack of proper process for scenarios that vary from the intended routine, but that happen often, such as swapping shifts between employees and overtime allocation. One user replied with details of an actual system that their company uses, called IT WERCS [10]. This system takes into account shift swaps, however, this is done by the employee themselves, meaning that it does not allow the second employee or the manager to be notified, and the schedule could be changed without the other employee and manager even knowing or being asked. This could obviously cause issues if the first employee assumes the second knows about the shift swap, and then neither one turns up for the shift.

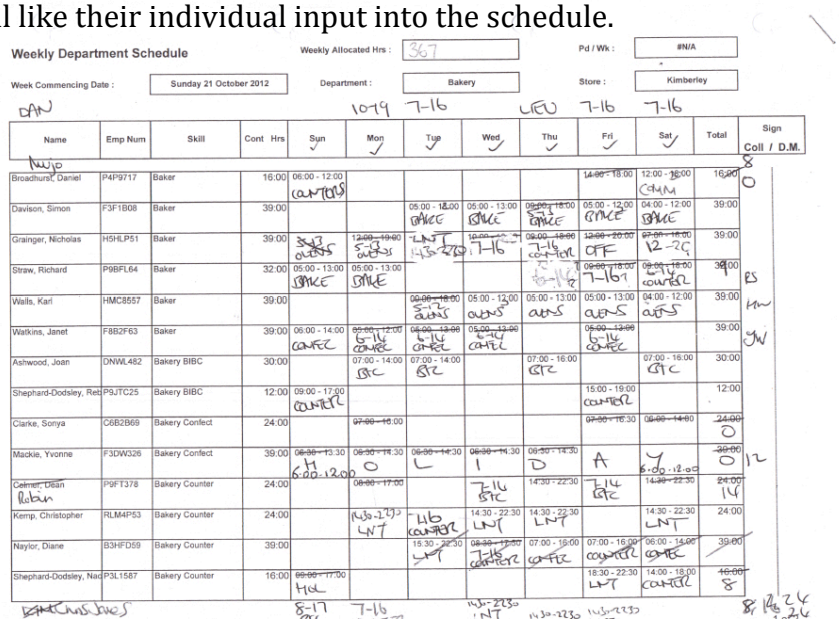
The next part of the target market research was to speak to people who currently work at Sainsbury's, as this will be the main focus of the proposed system. This was conducted by having casual, informal, conversations whilst at work about the topic with other employees. Here, it was found that employees agree that the system would be extremely useful and would eliminate the problems with the current system. One employee did mention that it might become ineffective if people don't check the system for requests, however if the system were to become the main tool within the company for communications, it is unlikely that this would be a big problem. Other observations and findings on the current system are described in the following sub-chapters.

2.2 Current System at Sainsbury's

As described previously, there are many flaws with the current system at Sainsbury's that have become apparent from working there and also from speaking to the manager and other employees.

At the moment schedules are located in inconvenient places, and they can only be viewed whilst at work as a paper hard copy. These schedules are a paper template copy printed out by the HR department and given to the manager to edit manually with a pen. Many changes tend to be made each week, meaning that the end result tends to be messy, crossed out and re-written on meaning

that they can therefore be hard to read. An example of the current schedule can be seen in Figure 1. This can lead to misunderstandings, and can mean that more clarification is needed from the manager, which can be difficult when you are only at work limited times in the week. This is a very real issue as, for example, employees have not turned up to work due to the schedule being unclear and therefore thinking they were booked off on holiday. However, from speaking to managers, it is apparent that automatic scheduling is not a desire, and that they would still like their individual input into the schedule.



Weekly Department Schedule

Weekly Allocated Hrs : 367 | Pd / Wk : #N/A

Week Commencing Date : Sunday 21 October 2012 | Department : Bakery | Store : Kimberley

Name	Emp Num	Skill	Cont Hrs	Sun	Mon	Tue	Wed	Thu	Fri	Sat	Total	Sign Coll / D.M.
Broadhurst, Daniel	P4P9717	Baker	16:00	06:00 - 12:00 COUNTERS					14:00 - 18:00	12:00 - 16:00 COUNTER	16:00	
Davison, Simon	F3F1B08	Baker	39:00			05:00 - 13:00 BAKE	05:00 - 13:00 BAKE	05:00 - 13:00 BAKE	05:00 - 12:00 BAKE	04:00 - 12:00 BAKE	39:00	
Grainger, Nicholas	H5HLP51	Baker	39:00		12:00 - 19:00 CLEANS		10:00 - 16:00	09:00 - 18:00 7-16 COUNTER	12:00 - 20:00 OFF	07:00 - 19:00 12-2C	39:00	
Straw, Richard	P9BFL64	Baker	32:00	05:00 - 13:00 BAKE	05:00 - 13:00 BAKE				09:00 - 18:00 7-16?	09:00 - 18:00 COUNTER	39:00	ES
Walls, Karl	HMC8557	Baker	39:00			00:00 - 18:00 5-17 CLEANS	05:00 - 12:00 CLEANS	05:00 - 13:00 CLEANS	05:00 - 13:00 CLEANS	04:00 - 12:00 CLEANS	39:00	
Watkins, Janet	F8B2F63	Baker	39:00	06:00 - 14:00 COUNTER	05:00 - 12:00 6-14 COUNTER	06:00 - 13:00 6-14 COUNTER	05:00 - 13:00 6-14 COUNTER		05:00 - 13:00 6-14 COUNTER		39:00	
Ashwood, Joan	DNWL482	Bakery BIBC	30:00		07:00 - 14:00 BTC	07:00 - 14:00 BTC		07:00 - 16:00 BTC		07:00 - 16:00 BTC	30:00	
Shephard-Dodsley, Ret	P9JTC25	Bakery BIBC	12:00	09:00 - 17:00 COUNTER					15:00 - 19:00 COUNTER		12:00	
Clarke, Sonya	C6B2B69	Bakery Confect	24:00		07:00 - 15:00				07:30 - 15:30	08:00 - 14:00	24:00	
Mackie, Yvonne	F3DW326	Bakery Confect	39:00	06:30 - 13:30 H 6.00-12.00	06:30 - 14:30 O	06:30 - 14:30 L	06:30 - 14:30 1	09:30 - 14:30 D	A	6.00-12.00	30:00	12
Celmer, Dean Robin	P9FT378	Bakery Counter	24:00		08:00 - 17:00		7-16 BTC	14:30 - 22:30	7-16 BTC	14:30 - 22:30	24:00	
Kemp, Christopher	RLM4P53	Bakery Counter	24:00		(W9-22?) LNT	16 COUNTER	14:30 - 22:30 LNT	14:30 - 22:30 LNT		14:30 - 22:30 LNT	24:00	
Naylor, Diane	B3HFD59	Bakery Counter	39:00			15:30 - 22:30 LNT	08:30 - 17:00 7-16 COUNTER	07:00 - 15:00 COFFEE	07:00 - 16:00 COUNTER	06:00 - 14:00 COFFEE	39:00	
Shephard-Dodsley, Nad	P3L1587	Bakery Counter	16:00	09:00 - 17:00 H&L					18:30 - 22:30 LNT	14:00 - 18:00 COUNTER	16:00	

Figure 1 Current Sainsbury's schedule

If employees want to book off holiday with the current system, a separate paper form is required. An immediate issue with this is obvious, in that these forms can get lost, resulting in more issues having to re-book holiday, or argue that you did indeed book it off in the first place. Secondly, a period of 4 weeks notice is needed in order to book holiday off, this means that if you are part-time you essentially lose a week of this and have to give near enough 5 weeks notice, in order for the holiday to be considered, which some times is not possible.

An employee wishing to swap a shift with another is very common with part-time workers and is an area that I have found no systems currently accommodating for successfully. It can be rare to be in work at the same time as the person you wish to swap shifts with, meaning that you have to have others mobile numbers, or a way of contacting them via social networks. This can be unprofessional, and even if you do have the ability to contact them, you can't view the schedule outside of work so you don't know whom exactly to contact and whom it is feasible to swap with, resulting in having to contact most members of staff to find out when they are working.

These issues also apply to the managers' task of offering overtime to employees. It is often that many of the team will receive a text or a phone call asking the employee if they think may be available to do a specific shift for overtime. This then becomes confusing, and requires extra communication to all of the employees asked, telling them whether they are actually doing the overtime or

not.

As evidenced above, there are numerous problems with the current system, and I feel that the main way to go about fixing these issues is to formalise and digitise the process. I will now go on to discuss existing systems that have taken this approach for these kinds of problems.

2.3 Existing Systems

By looking into existing systems an understanding has been gained of what is available in the current market, and what options would be available for Sainsbury's if they considered investing into an existing system. From this research it has been concluded that there are several categories for these existing systems, including off the shelf generalised systems, systems specifically for automated scheduling, and traditional methods such as what Sainsbury's currently use such as phoning and paper methods.

A system that I discovered was ITWERCS [10], which according to the blog responses obtained, is currently used in Hooters. This system is an off the shelf system for restaurants in particular and some of the features are out of the scope of this project such as customer ordering and inventory management, however some of the features are adaptable for my system. As described previously ITWERCS allows for shift swaps, however this is done by the particular employee him or herself editing the schedule, and still relies on communication by other means, to multiple people to check that this swap is acceptable. This means that if that communication fails then the schedule could be changed without the other individuals being notified, which will undoubtedly cause misunderstandings and confusion. This system also allows for managers to create schedules, which can be seen in figure 2.

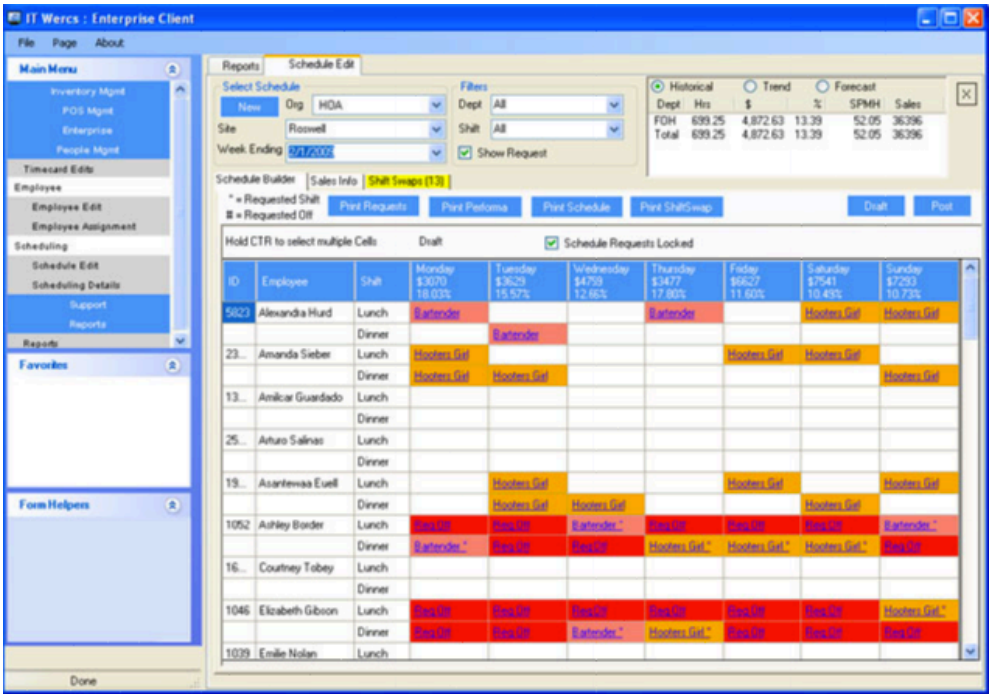


Figure 2 ITWERCS schedule creator interface

As you can see in the image, this interface is not aesthetically pleasing and is overly cluttered. In the design for the proposed system it will therefore be an aim to create easy to understand, well designed, interfaces.

Another existing system is OurLounge [11], which is used by McDonald's. This is a standard website, which allows employees to log in and view their schedule online. This is a feature that will be important to my system, as it provides the employee with the convenience and power to access the information they may need outside of the workplace, which is not available in the current system a Sainsbury's. OurLounge also allows for extra shifts to be advertised on the schedule, with employees able to book this overtime if they wish to do so. This again is another feature that which should be incorporated into the proposed system. However, a disadvantage with the OurLounge website is that, like many other systems, it does not cater for shift swapping. Also, it only has a desktop version, and it becomes very difficult to use if accessed on a mobile device, seen in figure 3. As said previously, a high percentage of workers at stores such as Sainsbury's are young adults who are more likely to access the system off of their phones whilst on the move. This emphasises the need for a desktop and mobile version of my system to ensure that it is accessible by the broadest range of users.

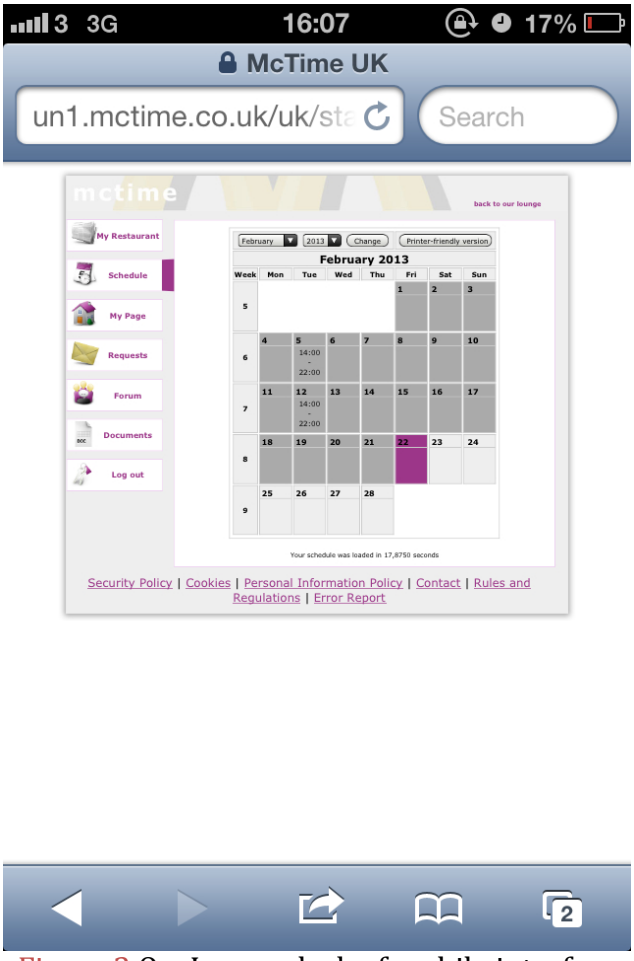


Figure 3 OurLounge lack of mobile interface

The systems described above were general off the shelf management systems with varying features and purposes. However, there are also many systems that

may fall into the category of management systems, which focus on automatic scheduling for the managers. An example of one of these systems is Workforce Scheduler™ [12] by Kronos. This type of system focuses on the auto-generation of workforce schedules based on rules the manager has input. However, on speaking to managers it has been found that this would not be preferred, as the environment in a company such as Sainsbury's is fast paced and situations can change and develop quickly. This means that they would have to spend time adjusting the constraints for the system as demand and situations change, meaning it would be more efficient to just make the schedule themselves as they already have a strong knowledge of the employees skills and preferences.

In conclusion, all of the above mentioned systems have their good and bad points. An objective of the project is to incorporate aspects of the good features into the system such as the ability to view schedules, whilst adapting and improving the bad ones, such as messy interfaces. Whilst at the same time, adding new important features that have not yet been found in other systems such as shift swapping.

2.4 Technical Research

The final part of the research is technical research. This was necessary in order to gain a good understanding of how the system would later be implemented, and included researching into tools and technologies that would be used.

Firstly, the decision of whether to create a mobile website, or a mobile application had to be made. A mobile website is essentially the same as any normal website you may view on your computer or laptop, however it has been optimised for viewing on mobile devices with much smaller screens [6]. This often takes a lot of thought and careful design to get features working on a mobile website at the same standard that they do on a desktop website due to limited screen space.

While mobile websites are viewed in a web browser on your device, a mobile application is one that is downloaded onto your device through 'stores' for your specific device, such as Android Market and Apple's App store. An initial downside of this is that you either have to choose a platform to develop for, or have to make different versions of your application for different mobile platforms.

For the context of the proposed system, considering the diversity of employees in a workplace, an application would not be very practical in that it may limit the amount of users who can access the system on a mobile device. This would defeat the object of trying to improve the problem in the current system of inaccessibility. Whereas a mobile website tends to be compatible across multiple devices. Another disadvantage of applications are that they are required to be downloaded initially, and may require further downloads by the user to update the app if changes are made to the system [6]. However, this would not be an issue with a mobile website, as changes to the system would be effective as soon as users accessed the site.

Mobile applications do have advantages however, such as accessibility to the users' camera or files on their device, but for this project this would not be necessary and so taking these points into account the decision has been made to create a mobile website for the mobile version of the system.

The next part of the technical research involved working out what tools and languages would be best suited to developing the mobile and desktop versions of the system.

Django is a web framework that uses the programming language Python for its server side scripting, as opposed to the common language PHP. Both Python and PHP work with CSS for the styling of the web pages, HTML for the layout/content of the web pages, and client side scripting languages such as JavaScript to create dynamic elements of the website. Taking this into account, and also having little experience in neither PHP nor Python more research was needed into what would be the best server side language to develop this project with [13].

Firstly, PHP has syntax similar to that of the programming language C, in terms of curly braces and semi-colons, which are familiar techniques and so this concept would be easy to pick up. On the other hand, Python uses white space to define, for example, the contents of a for loop and so this would not be so natural for someone with no experience with this type of syntax. However, because of this Python tends to read similar to pseudo-code and is therefore considered quite simple to understand.

Secondly, PHP gives the developer many different ways of solving a problem and so individual approaches to a problem can often be seen. This gives the developer a lot of control over how they wish to programme. On the other hand, Python tends to have set ways of approaching specific problems and if something is getting increasingly harder to achieve then the developer will know that it may have been approached in the wrong way. This controlled environment may benefit the development of a project as it limits the amount of error that can be made with the freedom of a language such as PHP.

In terms of web programming, although PHP can be used for other tasks its main purpose is for web development. However, if you wish to use Python for the web then a web framework is needed; a popular one of which, used with Python is Django. This may initially be seen as a hindrance as it involves learning how the framework works and then working within this framework. However, linking into the last point, this provides a controlled environment where there are specific ways of doing specific tasks, which, with a relatively short development time, could be useful.

For the reasons above it was chosen to use Python with the web framework Django for the server side scripting of the development, along with the use of standard JavaScript, CSS and HTML for other parts of the development as mentioned previously.

Chapter 3 - Literature Review

3.1 Management Theories

The management of employees plays an important part in retail. Sainsbury's alone employs 150,000 colleagues, 47,000 of which are under the age of 25. This could be due to a large amount of college and university students gaining a part time job during their studies. Due to this reason some may say it's unnecessary to focus on job satisfaction for these types of employees, as they expect to have relatively high staff turnover for this group of employees. However, Sainsbury's aim to focus on retention and long service, with one in three current store managers having started working at Sainsbury's before the age of 18 [14]. In turn, emphasising the need for a system that addresses some of the current issues that are inconvenient for employees, in order to improve staff satisfaction and retain their employment for as long as possible.

3.1.1 Herzberg's Two Factor Theory and Maslow's Hierarchy of Needs

Herzberg's two-factor theory takes the approach that there are two sets of factors, which affect employee motivation, these are hygiene and motivating factors. Hygiene factors, whilst they do not motivate employees as such, are important as workers may become dissatisfied if they aren't present [2]. Examples of hygiene factors are areas such as company policies, working conditions, pay, supervision and relations with other colleagues. The second set of factors is called motivation factors. These can be grouped as; personal achievement, recognition, interesting and rewarding work, responsibility, and personal development. These are said to positively motivate employees, which in turn leads to job satisfaction.

Maslow's hierarchy of needs follows a similar principle in that he said that people start by meeting the needs at the bottom of the pyramid, and once that need has been met, they can move onto the needs in the next level up. See Figure 4.

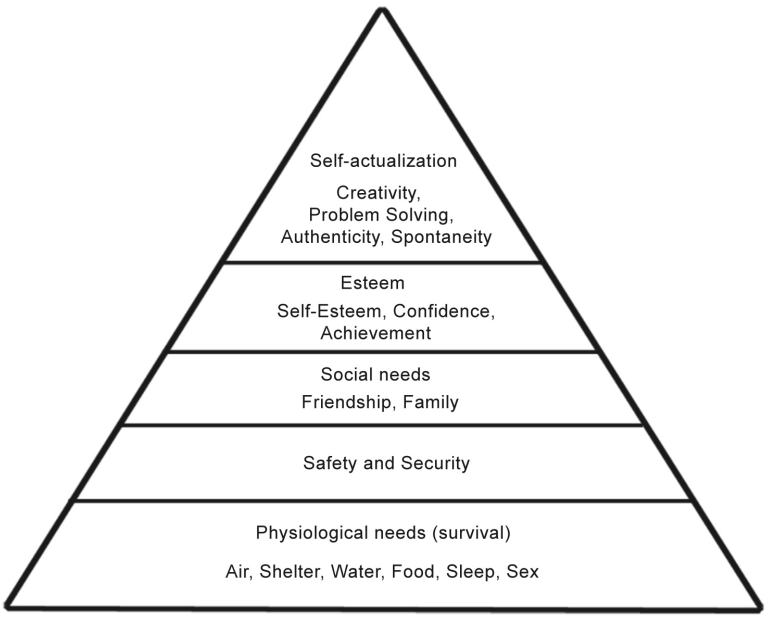


Figure 4 Maslow's hierarchy of needs

Maslow's hierarchy is similar to Herzberg's two-factor theory in the sense that it encourages some initial needs to be met in order to build on those and satisfy new needs. However, with the issues in part of the current system at Sainsbury's, it could be said that the areas regarding relations with employees/management and company policies in Herzberg's theory, and the social needs of Maslow's hierarchy are not being entirely met. This is because the current system, as described previously, creates flaws in communication between colleagues and managers, in turn hindering the efficiency of teamwork. More specifically, relations between employees/management are weakened as the communication level drops, due to misunderstandings occurring which can inevitably cause friction between colleagues.

3.1.2 Douglas McGregor - Theory X Y

Theory X Y is a management and motivation theory, which concludes that there are two approaches to managing people. Theory X takes an authoritarian management style, and assumes that the average employee dislikes work and will avoid as much as they can. Therefore, it leads to the opinion that people must be forced to work with the threat of punishment. It also assumes that employees prefer to be directed to avoid responsibility. On the other hand theory Y assumes that effort in work is natural and they will apply self-control and self-direction in order to complete a task correctly without external control. It has been said that many managers tend towards the approach of theory x, and generally get poor results. Whilst the few who use theory y, have found that it produces better performance and results [16].

In terms of the current system, the management style could be classed as theory Y. This is because the employees have the responsibility of arranging their own shift swaps and having the motivation to reply for overtime positions etc. Taking into account the proposed system, employees will still have this responsibility and control, however the process will be made easier and more formalised. This will therefore improve the manager's experience, as they will be more confident and able to keep track of what is happening between colleagues, but also make the whole process easier for employees, simplifying and improving communication as a whole.

3.2 Effective Interface Design

In order to create a system that people will want to engage with and use on a regular basis, it is important to carefully design effective interfaces. For this project it will be important to focus on interface design for a desktop computer or laptop, and also an interface for mobile devices.

3.2.1 Desktop Interfaces

This section will cover some of the rules in order to design an effective interface for the desktop part of the proposed system and these rules will aim to be incorporated into the design of this. According to Theo Mandel's paper on the golden rules for interface design [17] it is said that there are three main rules for effective interface design.

These are:
1. To place users in control of the interface.
2. Reduce users' memory load.
3. Make the user interface consistent.

Placing users in control of the interface involves thinking about different types of users. For example, a manager should be able to perform tasks on a system that an employee shouldn't, which, in the context of this system could be something like approving holiday whilst an employee would only be able to request it. This is an important feature for this system, as clearly an employee should not have the same rights and controls as a manager in terms of accepting shift swaps for example.

It is said that there are 10 principles that can be applied in order to place the users in control [17]. The principles that are believed to be most relevant to this project in terms of the features to be implemented will be discussed below:

1. Allow users to use either the keyboard or mouse (flexible)
2. Display descriptive messages and text (Helpful)
3. Provide immediate and reversible actions, and feedback (forgiving)
4. Provide meaningful paths and exits (navigable)

Allowing the user to use either the keyboard or mouse provides control on the users part. Just because a user has a mouse, doesn't mean that they would prefer to use it, and just because they can use a keyboard doesn't mean the user always will. However, the ability to use both if the user so wishes provides the flexibility to allow them to use whichever piece of hardware they feel is most suitable for them and for the task they are currently undertaking.

Displaying descriptive messages and texts is also an important part of interface design in order to let users fully interact with the system. For example, what use is an error message if it doesn't tell the user how to correct the problem? Communication with the user is important to allow smooth use of the system and avoid users becoming frustrated if something doesn't work and they don't know why.

Providing immediate and reversible actions with feedback is also considered a principle to allow users to be in control of an interface. The lack of feedback within systems can often force users to check that their original actions have actually been performed. Therefore to ensure users are provided with feedback, notifications and regular updates of the state of a process should be relayed to the users. This will put the users mind at ease knowing that their request has gone through the system, rather than if no feedback was provided. As well as feedback users should be able to reverse their actions, such as cancel a request etc. It is also important that users are provided feedback on these actions, otherwise it may cause worry with the user not knowing whether their actions have been fulfilled properly.

Providing meaningful paths and exits is involved with the ease of navigation of a

system. This is important in interface design in order to provide an easy, trouble free experience for users. For example, a user should be able to find whatever they are looking for with ease, and have an indication of where they currently are in the system. This can be achieved through a well-designed menu system, along with headings and sub-headings to inform the user of their current position.

The second of the three main rules for interface design is to reduce the users' memory load. This is related to the users ability to store and remember information, and a systems interface should help with this by refraining from making users have to remember information whilst using the system. An example of this is to relieve the users short-term memory. This can be implemented within tasks where the same information is needed on an interface from a different part of the system. This includes functions such as when a user my need to copy and paste information, however a better interface design would be where the system saves and recalls this information for the user. Another point linking in with this is to allow the users to rely on recognition rather than recalling information. This could mean, for example, when dealing with forms that the user has options to choose from, rather than relying on their own memory to enter information into a blank text field.

Providing visual cues also plays a big part in reducing the users' memory load. This can be achieved by letting the user know where they are, and what they can do next. This can be achieved by a navigation system that lets the user know which link they have currently clicked on, along with title bars that inform the user which page/part of the system they are on.

A final way to reduce users' memory loads is to provide default options. This means that once a user has input information before it is not necessary to do so again. This reduces the amount of information that the user has to remember and avoids repeating steps, therefore making the processes on the system more efficient.

The last golden rule for interface design is to make the user interface consistent, which can be linked to the last rule of reducing users' memory load. This is because if the interface is consistent, this will significantly reduce the amount of information that the user has to store about the system, as the layout will become natural to the user instead of having to adjust when switching to different pages or functions of the system. This can be explained further in terms of presentation and behaviour. In terms of the presentation of the system, if there is navigation on the top right hand side of the screen in a fixed position on the home page, then this should also be in that position on any other page the user visits. The same can be applied to text, in that if a header is in 20pt red font on one page, any other headings should also be in 20pt red font. In terms of the behaviour of the system, an object should work in the same way on different parts of the system. For example, if green buttons submit a form, then a blue button should not be used for the same function on a different page.

All of the above rules are important in order to create a sense of confidence

within the user when interacting with a systems interface. This could be especially true in this case where users may be resistant to a change made by a company such as Sainsbury's and so the system should be designed with the employees needs in mind. This will attract users to the system and allow them to embrace the change rather than resist it if the interface creates difficulties

3.2.2 Mobile Interfaces

All of the rules explained in the chapter for desktop interface design could be considered general rules that all interfaces should follow. However, when designing for mobile devices, the constrictive screen space calls for more in depth design of it's own.

There are a number of design patterns for mobile devices that can help in transforming a desktop website into one that functions correctly, whilst still looking aesthetically pleasing, on a mobile device [8].

To begin with, generalised patterns will be discussed before moving onto individual elements that can be adapted for mobile design.

Whilst on a mobile device it is not suitable to use a multiple column layout like many desktop websites use. Due to the narrowness of a mobile devices screen, a layout set out in this way may either become un-readable from being scaled down to fit the screen size, or cause horizontal scrolling which is not ideal for the viewing of information. To solve this problem a linear approach can be taken, where different blocks of information are stacked on top of each other using a fluid layout in order to fit the full width of the mobile device viewing it, as can be seen in figure 5.

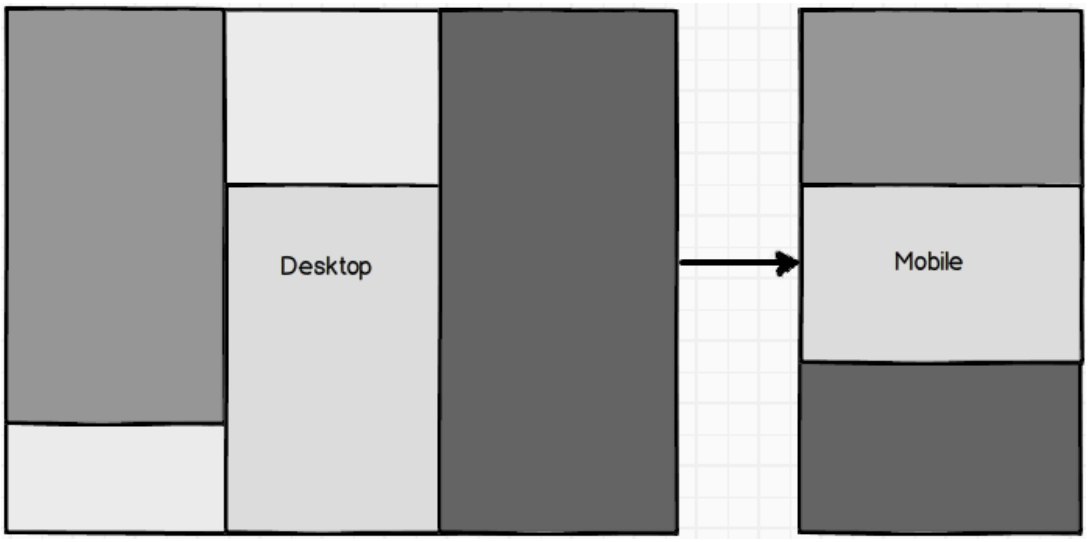


Figure 5 Converting to a linear mobile layout

Another adjustment method that can be used to convert desktop layouts to mobile layouts is the Grid method. This solves the problem where in some circumstances within the linear method where there may be information that is not intended to fill the full width of the screen. Therefore, the grid method allows the designer to place content side by side in rows. This can typically be used for

images or icons; as if this method is used with text it can become unreadable. This method of mobile interface design can be seen in figure 6.

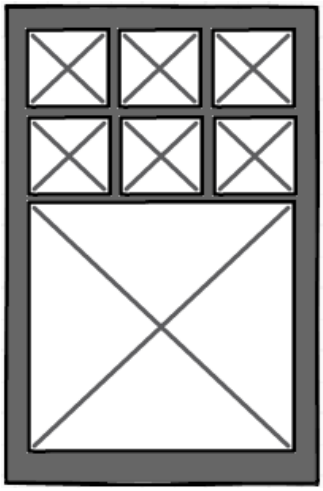


Figure 6 Grid layout on mobile

A third and final way to set out the content on a mobile device could be via a vertical list or an infinite list. This method splits the screen into a number of sections with a clear divide in between each one to form a list of content. This list could just be the content in itself, or it could work as a link. An infinite list works in the same way, however more posts are automatically loaded once the end of the list is reached, rather than a set amount of posts. This method is similar to that of Facebook or Twitter, and can be seen in figure 7.

Figure 7 Vertical list layout

With methods for converting the general content layout established, it is now important to consider how a user could navigate around a mobile site, as it would be impractical for all information to be displayed in one long page. Most desktop websites' navigation may take up a considerable width or length of an interface because there is the physical space to do so. However on a mobile device different approaches need to be considered in order to optimise the use of

a limited amount of space.

The first method is a linear menu. This type of menu is similar to the vertical list content layout, but is for links only rather than any form of content. This will use a fluid layout, fitting the navigation menu to the width of the devices screen. However, a problem with this type of menu is that it may still take up a large height of the screen, not allowing the user to see much content without scrolling down.

A second navigation method could be a toggle menu. This takes the same approach as a linear menu, however, allows the user to toggle the menu on or off. This means that it can be placed anywhere, usually the top, of the screen and keep the functionality of a linear menu, without taking up a large amount of space. This is demonstrated in figure 8.

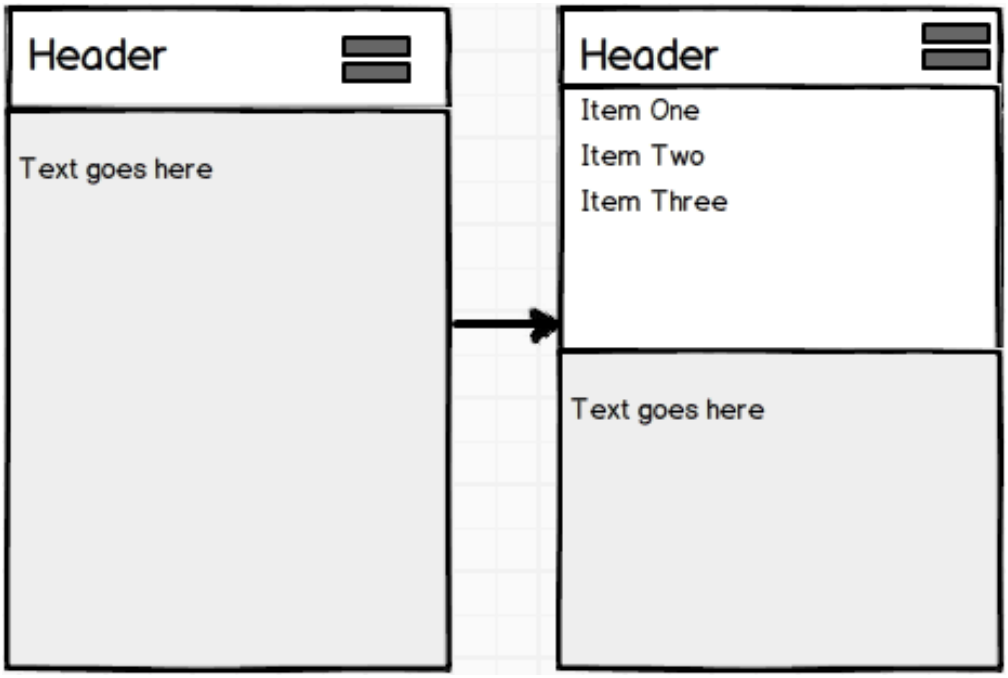


Figure 8 Toggle menu on mobile

The final navigation method to be discussed is a select menu. This is space effective as it is presented to the user as a single drop down box, which inherits the devices native select component. The fact that it uses the devices natural select component creates a sense of familiarity with the user and may make the user feel more comfortable when using the system, as seen in figure 9.

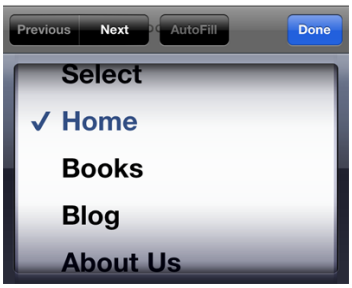


Figure 9 iPhone select menu

As well as navigation there are also individual elements that need to be considered within in a mobile web page. The use of tabs could be included in order to display related information that shouldn't be on different pages, but should be separated in some way.

Tables can take different approaches to display all of the information within them on limited screen space. Inverting the table so that the headings are vertically on the left of the table with the cells on the right can do this, as seen in figure 10. Another approach, which could also be combined with the first, is the use of toggles to hide and display different information in a table. This reduces the percentage of screen space taken up, however requires the designer to make informed decisions on which information should be shown together and which is appropriate to hide.

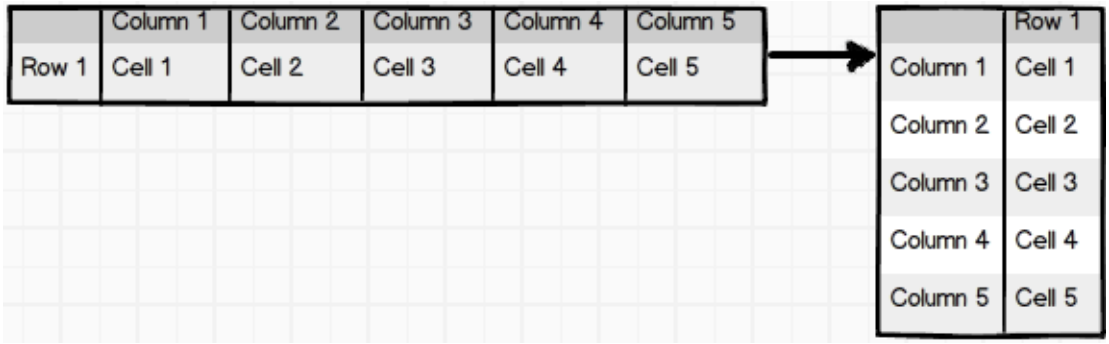


Figure 10 Inverting a table to fit on mobile devices

In conclusion, it will be important to ensure all of the rules and approaches described above for both desktop and mobile interfaces are taken into account when designing the proposed system. The skills and approaches learnt from the above analysis give a variety of different methods to consider when designing and implementing, in order to produce a high quality system.

Chapter 4 - System Requirements

4.1 User Classes

There will be two main user classes for the proposed system, employees and managers.

The amount of employees who use the system will be a lot higher than the number of managers due to the ratio between them, and so employees will therefore be classed as the main users of the system. Employees should have permissions to view schedules for the upcoming four weeks, request and monitor personal shift swaps, accept or decline overtime requests and request and monitor personal holiday.

The manager class has different permissions, however it does not as such extend the employee class. Managers should be able to create schedules, approve or deny shift swaps, advertise overtime to specified employees, and approve or deny holiday requests. However, they will not also have all of the features that the employees do such as requesting shift swaps, as this is very uncommon between managers.

A number of use cases will now be created in line with these user classes to demonstrate how the system may be interacted with.

Use case 1 – Create schedule

Actors	Manager
Description	Manager wants to create a new schedule.
Pre-condition	1. Manager must have a valid log in account.
Normal flow	1. Manager logs in to the system. 2. Manager clicks link to add schedule. 3. Manager deletes week 1 schedule to make room for a new one. 4. Manager types in times that employees are working. 5. Manager inputs week commencing date for the schedule. 6. Manager 'quick checks' holidays to see who is on holiday that week. 7. Manager adapts schedule for any holiday arranged for that week. 8. Manager clicks save schedule. 9. Schedule is saved.
Alternate flow	3a1. Week 1 schedule does not need to be deleted yet as it is not over. 3a2. Manager moves to step 4. 4a1. Manager uploads a default

	schedule. 4a2. Manager uses template schedule to fill times. 4b3. Continue to step 5. 4b1. Template schedule already uploaded. 4b2. Manager uses template schedule to fill times. 4b3. Continue to step 5. 6a1. An invalid week commencing date has been input meaning holidays can't be checked. 6a2. Error message is displayed. 6a3. Manager returns to step 5. 7a1. There are no holidays that week. 7a2. Manager continues to step 8. 8a1. Manager has not filled in all of the fields and error message appears. 8a2. Manager amends the schedule. 8a3. Manager returns to step 8.
Post-condition	New schedule has been successfully added/created.

Use case 2 – Advertise overtime

Actors	Manager Employee
Description	Manager wants to advertise overtime position.
Pre-condition	1. All actors must have a valid log in account. 2. Manager must have an overtime position available.
Normal flow	1. Manager logs in to the system. 2. Manager clicks link for overtime. 3. Manager fills in form to advertise overtime to specific employee. 4. Employee logs in to the system. 5. Employee sees notification for overtime. 6. Employee accepts overtime. 7. Manager receives overtime notification. 8. Manager final approves individual overtime.

	9. Overtime is automatically added to the schedule.
Alternate flow	3a1. Manager chooses specific employee to send request to. 3a1. Manager continues to step 4. 3b1. Manager fills in overtime form to send the request to all employees. 3b2. Manager continues to step 4. 6a1. Employee declines overtime. 8a1. Manager declines overtime. 8b1. Manager re-opens overtime to all employees.
Post-condition	Overtime has been accepted and approved, and is automatically added to the schedule.

Use case 3 – Shift swapping

Actors	Manager Employee
Description	Employee wants to swap a shift.
Pre-condition	1. All actors must have a valid log in account. 2. Employee must have viewed the schedule and worked out an appropriate shift to swap.
Normal flow	1. Employee logs in to the system. 2. Employee clicks on shift swap link. 3. Employee fills in shift swap details. 4. Employee sends shift swap. 5. Second employee logs in and receives shift swap notification. 6. Second employee accepts shift swap request. 7. Manager logs in and receives shift swap notification. 8. Manager approves shift swap between employees. 9. Employees are notified. 10. Shift swap changes are automatically reflected in the schedule.
Alternate flow	4a1. Employee has not filled in form correctly and receives error. 4a2. Employee returns to step 3.

	6a1. Second employee denies shift swap. 6a2. Whole shift swap request ends and employee receives a notification that the swap has been denied. 6a3. Use case ends. 8a1. Manager denies shift swap notification. 8a2. Employees receive notification to say shift swap has been denied. 8a3. Use case ends.
Post-condition	Shift swap request has been approved and automatically shown in the correct schedule.

4.2 Requirements Specification

Below are the functional and non-functional requirements for the proposed system. Functional requirements define a function or component of a system, whilst non-functional requirements are concerned with criteria that can be used to just the operation of a system, rather than specific behaviours [18]. The following functional and non-functional requirements have been assigned a number between (1) and (5) at the end of each one, to signify the importance of each requirement. 1 being the highest importance in order to solve the initial problem, and 5 being more of a desired requirement.

4.2.1 Functional Requirements

Accounts

1. There should be different account levels; one for managers and one for employees. (1)
2. Each employee should have access to his or her own account. (1)

Communication

3. The same features should be available on the desktop version and the mobile version. (2)

Viewing schedules

4. Employees should be able to view up to 4 weeks worth schedules in advance online. (1)
5. Employees should be able to view up to 4 weeks worth of their own shifts online. (2)
6. Managers should be able to view the same set of schedules in advance that employees can. (2)

Create schedule

7. Managers should be able to fill in a schedule with times that employees are

working and a week commencing date. (1)
8. Managers should be able to check who is booked off on holiday that week, overtime, and shift swaps whilst they are creating the schedule on the same page. (3)
9. Managers should be able to use a template to fill the schedule in with if they wish to do so. (5)
10. Managers should be able to upload a file for the template schedule. (5)
11. Managers should be able to edit schedules through the Django admin interface if something changes in the schedule that is not automatically done with shift swaps or overtime. (1)

Shift swapping
12. Employees should be able to request to swap a shift with another employee. (1)
13. Employees should not be able to swap shifts in different weeks. (2)
14. Employees should have some indication of available shifts to swap without having to switch between different pages. (3)
15. Employees should have an indication of the status of the shift swap. (2)
16. Employees should be able to accept or decline shift swaps sent to them. (1)
17. Managers should have a final say in whether the shift swap is approved or denied. (1)
18. The shift swap should be automatically updated in the schedule. (2)
19. Employees should have the ability to cancel shift swap requests up until the manager has approved them. (2)

Overtime
20. Managers should be able to advertise overtime to one employee. (1)
21. Managers should be able to advertise overtime to all employees. (3)
22. An employee should be able to accept to decline the overtime request. (1)
23. The manager should have final say in whether the overtime can go ahead. (1)
24. If accepted, the overtime should be automatically updated in the schedule. (2)
25. Managers should have the ability to cancel overtime advertisements up until they have been finally approved.
26. Managers and employees should have an indication of the status of the overtime request. (2)
27. If overtime has been sent to all employees and an employee accepts, this overtime should be temporarily blocked from other employees. (3)

Holidays
28. Employees should be able to request holiday through the system. (1)
29. Managers should have the ability to approve or deny holiday. (1)
30. Employees should have the ability to cancel requests up until the manager has finally approved it. (2)
31. Employees and managers should have an indication of the status of the holiday request.

4.2.2 Non-Functional Requirements
Security

32. Only employees or managers with an active account can log in to the system. (1)
33. The system should ensure that the correct username and password are entered before allowing entry. (1)
34. Passwords should be stored in an encrypted format. (2)

Capacity
35. The system should be able to handle at least 10 requests an hour for the specified department it is currently being created for. (1)
36. The system should be able to handle up to hundreds of requests per hour, should the system be used throughout a whole store or a number of stores at a time. (4)
37. Operations/requests should be possible to complete within a minute of logging into the system.

Availability
38. The system should be available to users 24/7. (1)
39. The system should be available in and out of the workplace. (1)

Usability
40. The system should have an easy to use interface on desktop computers. (1)
41. The system should have an optimised interface for use on mobile devices. (1)

Operating environment
42. The system should function on all three of the following web browsers; Google Chrome, Firefox and Safari (2)
43. The system functions on mobile web browsers, focusing mainly on iOS and Android. (1)

Documentation
44. The system should provide a help page online to aid in any issues that may occur. (5)

4.3 Assumptions and Dependencies

In order for users to be successful in using the system, there are some assumptions and dependencies that have to be satisfied. It is assumed that users will have an up to date version of one of the most popular web browsers out of Chrome, Firefox or Safari as older or different browsers may not display features correctly.

A second assumption is that the user has JavaScript enabled. This is because it will be used in development for client side scripting, and failure for the system make JavaScript calls will mean that pages may load incorrectly and functions of the system will not work how they should do. For example, failure for JavaScript to be present may result in the system not performing the correct validation checks on data, meaning that invalid data is submitted and the system does not function as intended.

The use of this system on a mobile device will assume that users have an Internet connection available outside of work and on their mobile devices. Although the system will be made as clear to use as possible, users should also have some knowledge of the use of computers or mobile devices in order to interact with the system with as little trouble as possible.

Finally, the success of the system depends on users' motivation to convert to the new system rather than reverting back to the old methods. Users should aim to use the system as their main communication tool and only use other methods when completely necessary.

4.4 Limitations

There are, however, aspects of this system that are out of the scope of this project. Firstly, this system will not directly link into the Human Resources department in terms of hours worked and pay given. It will be required that there is a separate system to track hours that employees have worked and connect this into the payroll department, as is done with the current system as Sainsbury's with the use of clocking in machines.

Also linked to HR is that this system will not allow managers to perform the same functions that employees can, such as request holiday or swap shifts with another manager. This is because in reality they would not be able to approve their own requests and would need a higher authority to do this, such as a member of HR or a store manager, which is out of the scope of this project.

Another limitation of the system is that it will not check whether shift swaps and overtime are feasible for the employee in question. It will be assumed that if an employee has accepted or requested a swap or overtime, and the manager has finally approved it, that the action in question is valid in terms of the schedule that it relates to.

A final limitation of this system is that it will not use automated scheduling. Some systems automatically create schedules based on rules that have been input by the user. However, from the analysis of research, it has been decided that this is not wanted in this system, and so rather, it should allow for the manager to make use of a default, editable, schedule if they so wish.

Chapter 5 - Design

5.1 Branding

To brand the proposed system, it was first necessary to come up with possible names. To do this, a brainstorming diagram, figure 11, was created.

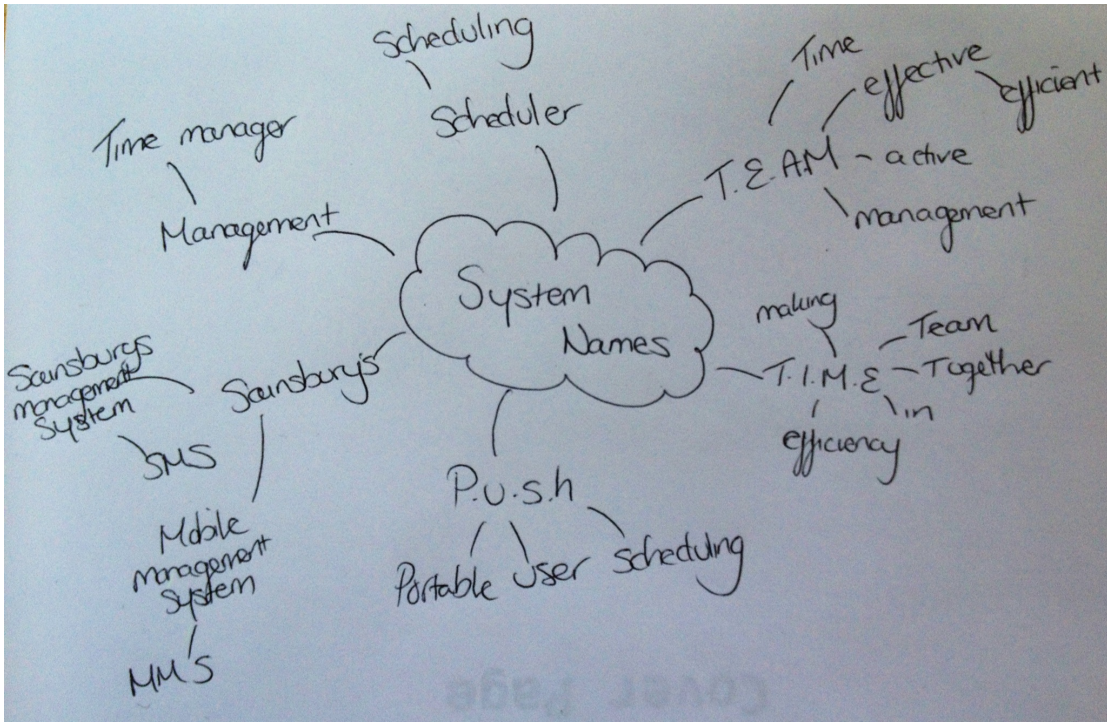


Figure 11 Brainstorm diagram for system names

From this, the three best names were chosen, from asking the general opinion of others and using personal opinion. The three chose names were T.I.M.E standing for "Together in managing effectively", Sainsbury's management system, and T.E.A.M, standing for "Time effective active management". These three names then needed to be narrowed down to just one. It was felt that Sainsbury's management system reflected what the system will do, however this abbreviates to SMS, which is an abbreviation for text messaging. However, there is no form of text messaging planned for this system and so it was felt that this name might be misleading to users. When asked about T.I.M.E – Together in managing effectively, it was felt that 'time' might not provide enough insight into the system and does not reflect accurately enough what the system does, without reading further into the acronym. T.E.A.M received the best response, with ideas that it could be used in a way of saying, "You have a new team request", referring to the name of the system but also working in practise that you have a request from a team member. The acronym for T.E.A.M, "Time effective active management", also captures what the system aims to do. The management part of the name could also refer to the manager him or herself, or the employees managing their own shift swaps and overtime etc.

After the name T.E.A.M was decided it was necessary to make a logo for this for the website. It was decided that the meaning of T.E.A.M, "Time effective active management" would not be incorporated into the logo, and would however just

be written in the introduction/welcome paragraph for the website. This is so that the logo is more prominent and easily readable, as having long strings of words would make it hard to read. The final logo is shown in Figure 12.

Figure 12 Logo for the system

This logo shows the main name T.E.A.M in a readable font, in the main Sainsbury's colour of orange. The dots, allow the user to know that team stands for something, whilst the little people icons at the side represent a team working together. This will be the final logo used throughout the website.

5.2 Database Design

In order to create a successful and dynamic web system, it is necessary to have a solid back end in the form of a database to store all the information needed to enable the system to function correctly. Django automatically creates a primary key for each table that exists, without the user having to include this in their code. Therefore, the following design does not include primary keys, however there is one in every table. Below is an entity relationship diagram for the database.

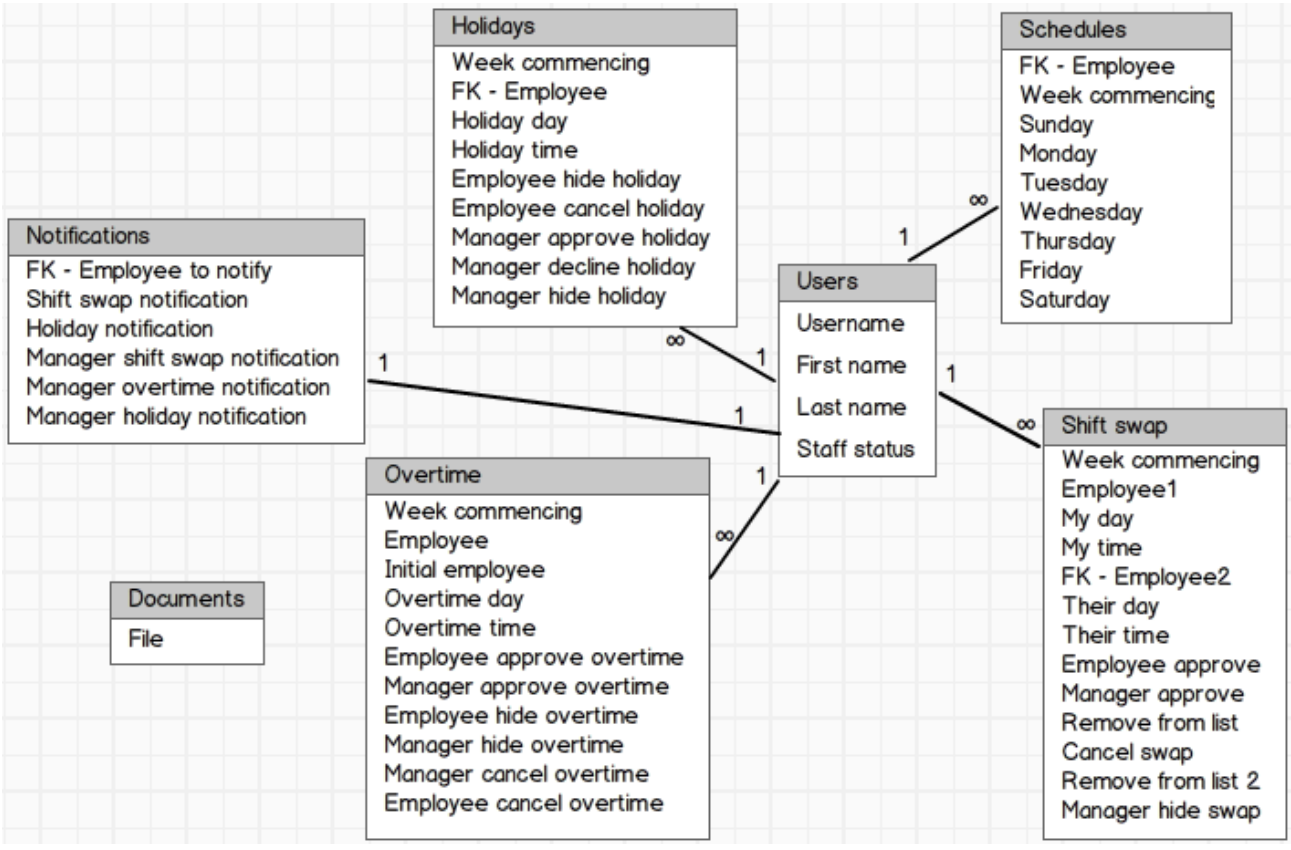


Figure 13 Entity relationship diagram

As mentioned above, Django's model system creates a primary key for all tables that exist, which is in the form of an automatically generated ID number, so although they are not part of the database design above, this field will be in each table. The users table links to almost every other part of the system. The users table holds information about all of the different users that can access the system. The schedules table works by having one user per schedule. This means that in implementation a 'schedule' will be classed as a row in the database for a single user. To make a full group schedule for a week, this will consist of multiple individual schedules put together. Shift swapping also has a one user to many shift swaps rule. This is because the employee that requested the shift swap will be identified by the current user logged in, whereas the employee sent the request will be identified through the user objects in order to send the request to the correct person. Holidays are fairly simple in that one user object can have many holidays booked off, and the same for overtime, where one user may be doing many overtime shifts. Notifications work on a one to one basis. This is because each user will only be in the notification table once, and the notification count is just updated and reset, rather than new rows being made each time. The document table is completely separate and has no relationship with the other tables. This is because it is simply a space for the storage for files containing template schedules and does not need a way of being identified except for the newest record.

5.3 Desktop User Interface Design

The next part of the design section is to design the user interfaces for the desktop system. To do this a wireframe creator called Balsamiq has been used, and in designing the interfaces an aim has been to follow the golden interface rules explained in previous chapters. This is an important section within this project in order to create a system that employees and managers alike will be happy to use regularly.

The login interface will be the first screen that the user see's. For the desktop system the logo will be displayed in the top left corner of the screen, and a welcome paragraph will be presented for the user to read, giving information about the system and what to do if they do not have a login, which will be helpful if it is the first time they have visited the site. A simple login section will then be provided, including the use of labels and text boxes to fill in to enter the website. This can be seen in figure 14.

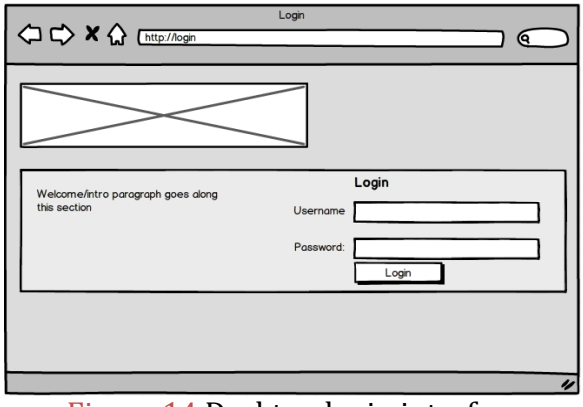
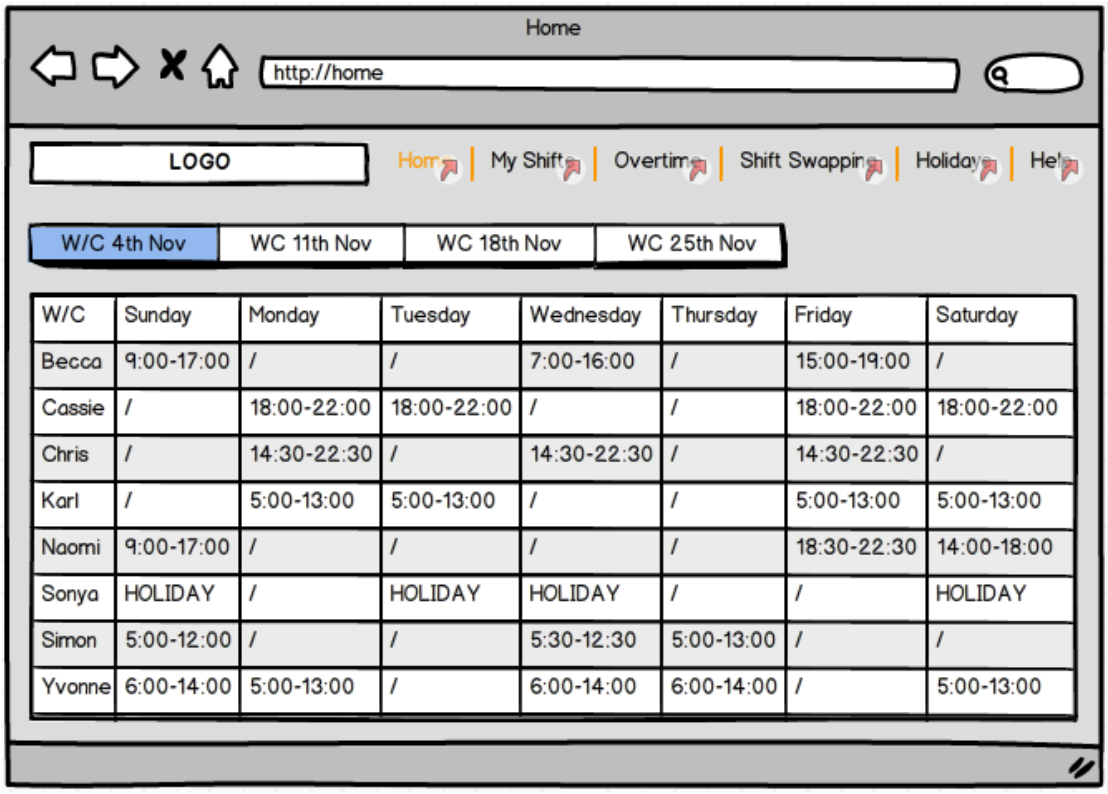


Figure 14 Desktop login interface

Once the user logs in they will be presented with the home screen. This will be the first instance where the user sees the standard layout for the desktop layout. This includes the logo in the top left hand corner and a menu stretching across the top right side. The home screen will have the same content in the interface for both the manager and employee. The content will consist of schedules up to 4 weeks in advance, where the user can flick through to see different ones via buttons. However they will be presented with slightly different options in the navigation menu to reflect different features available, in that managers can create schedules whilst employees can view their individual shifts. This can be seen in figures 15 and 16.



W/C	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Becca	9:00-17:00	/	/	7:00-16:00	/	15:00-19:00	/
Cassie	/	18:00-22:00	18:00-22:00	/	/	18:00-22:00	18:00-22:00
Chris	/	14:30-22:30	/	14:30-22:30	/	14:30-22:30	/
Karl	/	5:00-13:00	5:00-13:00	/	/	5:00-13:00	5:00-13:00
Naomi	9:00-17:00	/	/	/	/	18:30-22:30	14:00-18:00
Sonya	HOLIDAY	/	HOLIDAY	HOLIDAY	/	/	HOLIDAY
Simon	5:00-12:00	/	/	5:30-12:30	5:00-13:00	/	/
Yvonne	6:00-14:00	5:00-13:00	/	6:00-14:00	6:00-14:00	/	5:00-13:00

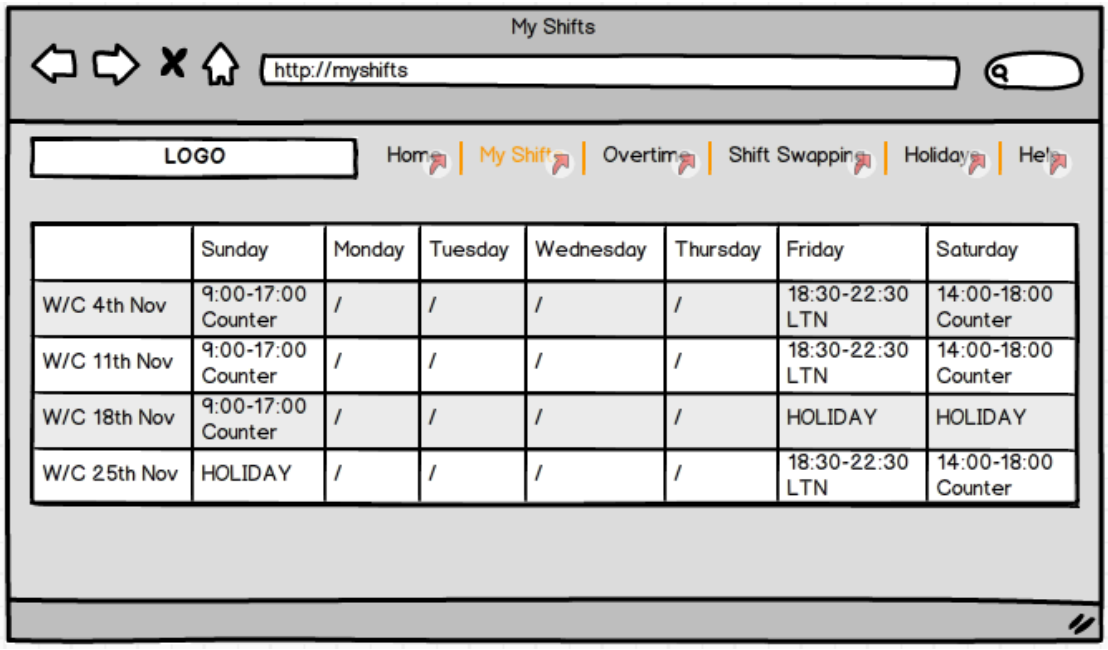
Figure 15 Desktop employee homepage

Figure 16 Alternative homepage menu for managers

This has been designed in a way so that the user knows which page they are currently on, by the link at the top highlighted orange, and also knowing which schedule they are currently viewing, with the button highlighted once it is clicked. This links in to good interface design, as the user will be are of where they currently are in the system, in turn reducing the users' memory load, which is a golden rule in interface design. This also links to the rule of providing meaningful paths.

The next interface is the 'My shifts' page for employees only. This interface is a simple one, which allows the logged in employees to view their own shifts for the upcoming four weeks, as opposed to the whole schedule as shown on the home

page. This allows for clearer viewing of shifts and should reduce any confusion and avoid accidently misreading other colleague's shifts. This interface can be seen in figure 17.



	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
W/C 4th Nov	9:00-17:00 Counter	/	/	/	/	18:30-22:30 LTN	14:00-18:00 Counter
W/C 11th Nov	9:00-17:00 Counter	/	/	/	/	18:30-22:30 LTN	14:00-18:00 Counter
W/C 18th Nov	9:00-17:00 Counter	/	/	/	/	HOLIDAY	HOLIDAY
W/C 25th Nov	HOLIDAY	/	/	/	/	18:30-22:30 LTN	14:00-18:00 Counter

Figure 17 Employee my shifts interface

The next interface is the one in which the manager will create a new schedule. This consists of a main blank schedule in the centre of the page, which can be filled in by the user. There is also an update box on the right hand side to inform the user of any changes they need to take into account for this week schedule, for example, any holidays that are happening that week. This aims to take into account the golden rule of reducing users memory load by having the information they need, where and when they need it, rather than the manager having to go to the holidays page, remember the holidays and return to create the schedule.

Along with this, the add schedule page has the ability to 'Delete week 1'. This is an aspect of the design that links in with the schedule database table. When a user clicks delete week 1, this will delete the oldest schedule set that the database contains, which means that on the home page week 2 will become week 1, week 3 will become week 2 etc. and the newest schedule that is being created will automatically become week 4. This puts the user in control of the interface, which is another golden rule, as it allows the manager to create schedules beyond the 4 week period, and only making a new schedule visible when an old schedule is over and it is time to display a new one.

The final part of this interface is the ability for the manager to upload a file for a template schedule and then use this file to fill in the schedule above. This will allow for the manager to use a template schedule every time and only change small parts if they wish, to save time, however it also allows them to still have the option to create a new one from scratch if they wish to do so, again placing the

user in control of the interface and providing default options for the user to reduce memory load. This interface can be seen in figure 18.

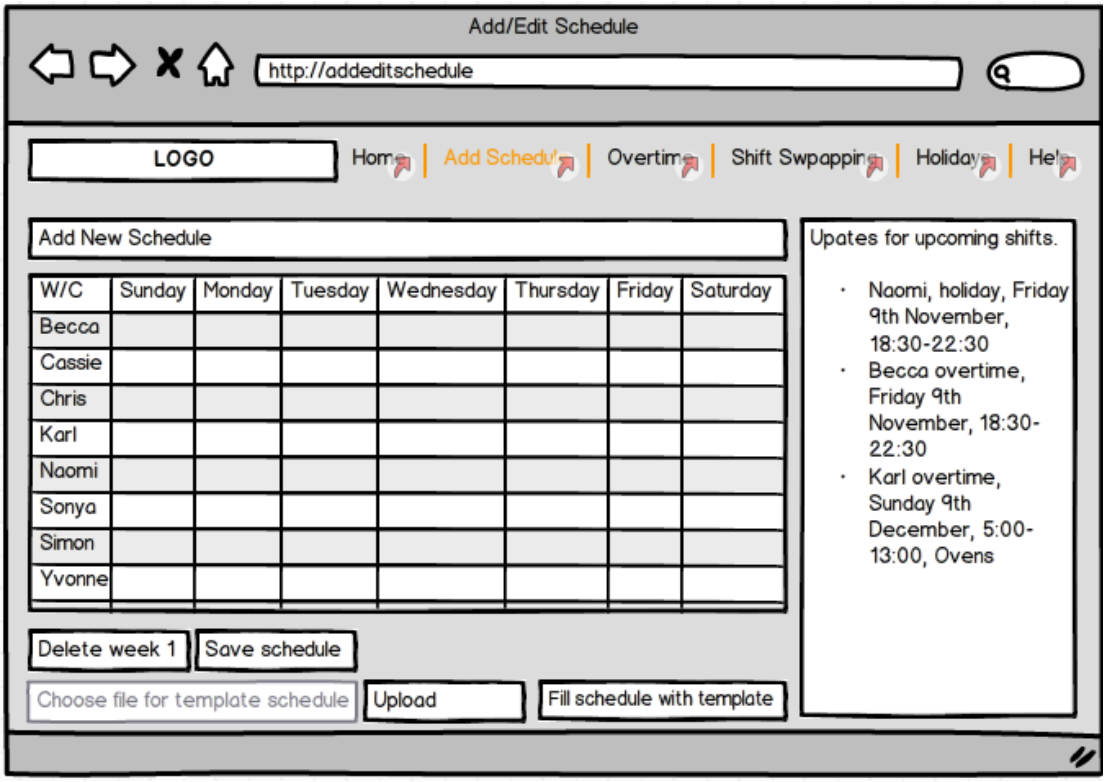


Figure 18 Manager create schedule interface

The next interfaces are the shift swapping pages for both the employee and manager. Firstly, the employee interface will have a form, which allows the employee to request a shift swap themselves. This will provide some automated features in that once a week commencing date is selected, it will automatically allow the user to see what shift they or another employee are currently doing that week. This will reduce the users memory load as they will not have to remember specific shifts for the request, they will be worked out for them. The second vital part to the user interface is a section that shows the user outstanding requests. This lets the user keep track of what is happening and also provides the option for an employee to cancel an outstanding request, providing reversible actions with immediate feedback so that the user feels comfortable that their actions have been completed successfully. The employee shift swap interface can be seen in figure 19.

The second half to this is the manager shift swap interface. This is used once two employees have accepted a shift swap between them. When this has happened, the request will finally get forwarded to the manager to either approve or deny. This is a simple interface that shows all of the shift swap requests and statuses, and can be seen in figure 20.

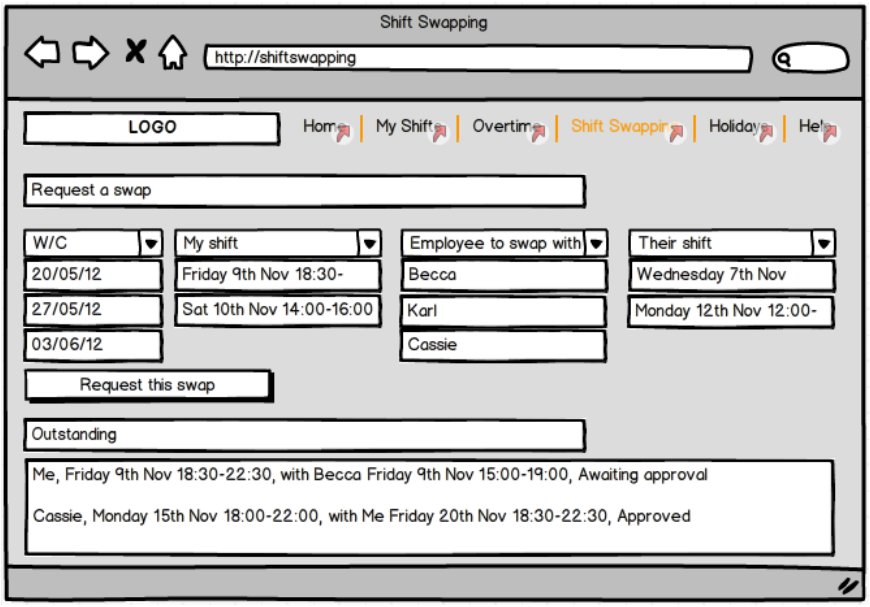


Figure 19 Employee shift swap interface

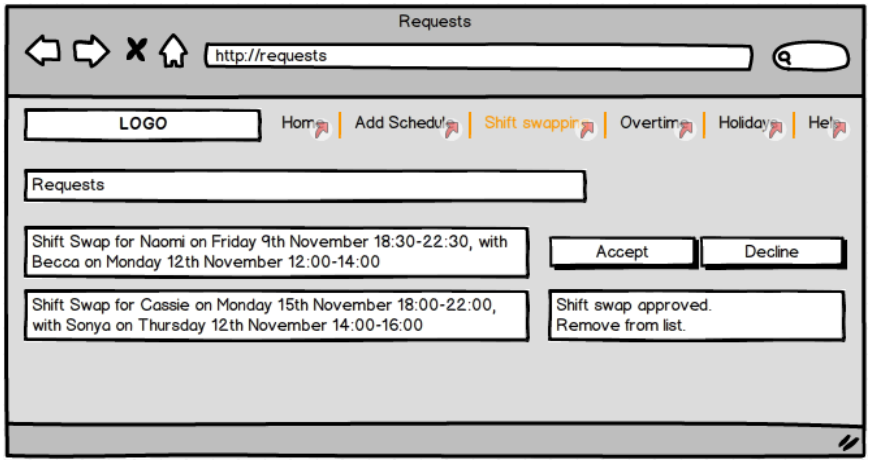


Figure 20 Manager shift swap interface

The holiday interfaces follow a similar principle in that there is one for the employee to request holiday and keep track of their requests, and one for the manager to accept or decline the holiday along with a list of all booked holiday for reference when approving or denying requests. These can be seen in figures 21 and 22.

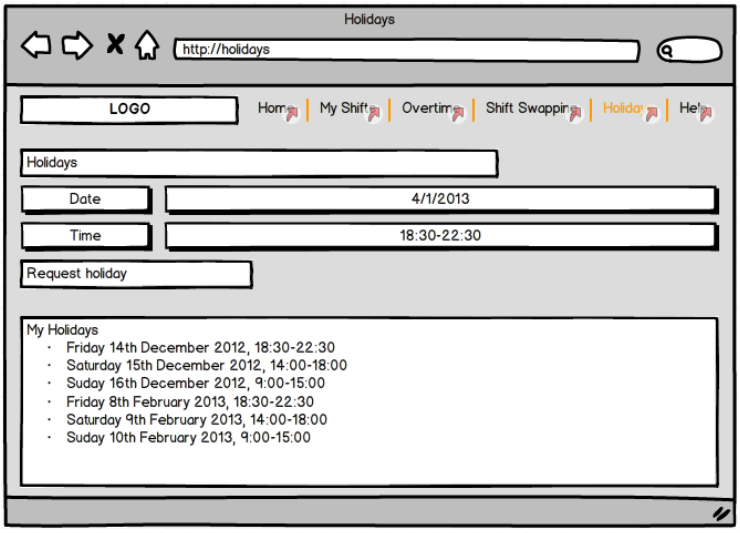


Figure 21 Employee holiday interface

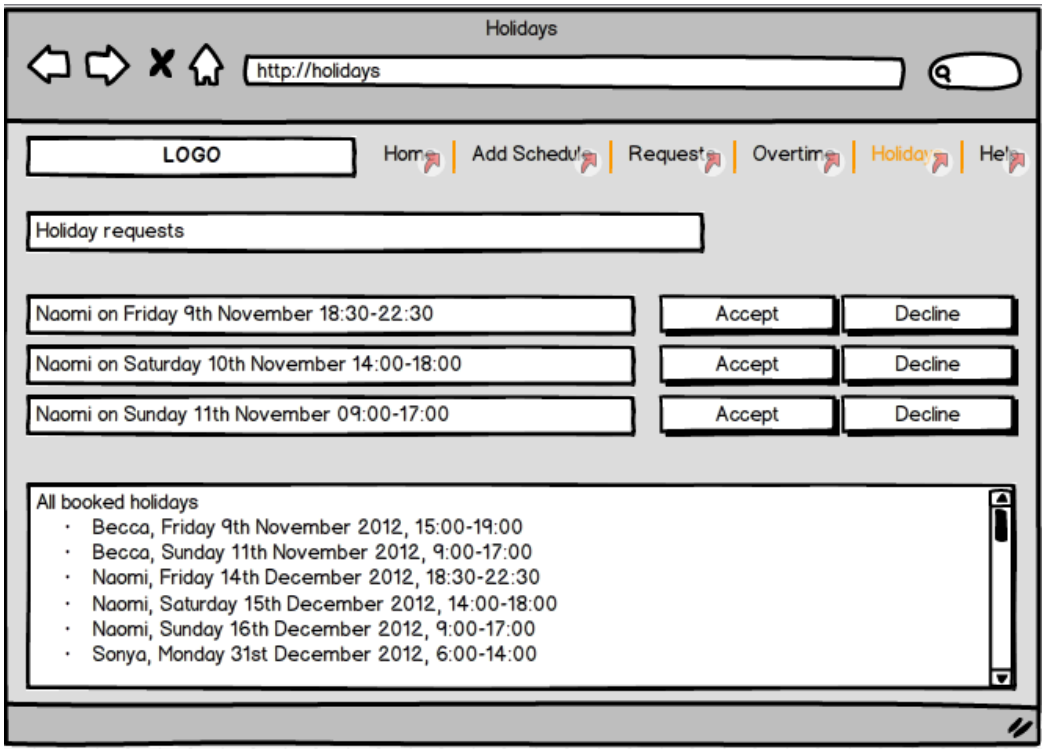


Figure 22 Manager holiday interface

The final set of interfaces is for overtime. This is in a way reversed from the other two sets, as now, the manager will send out requests and the employee will have the chance the accept or decline. The manager interface will consist of a form that allows them to choose the date and time for the overtime, and also allows the manager to choose who to send the overtime advertisement to; all employees, or a specific person. This page will also show the status of any overtime requests the manager has sent in order to keep track with ease and reduce the users' memory load, or stop the manager having to keep note elsewhere manually. The managers overtime interface can be seen in figure 23.

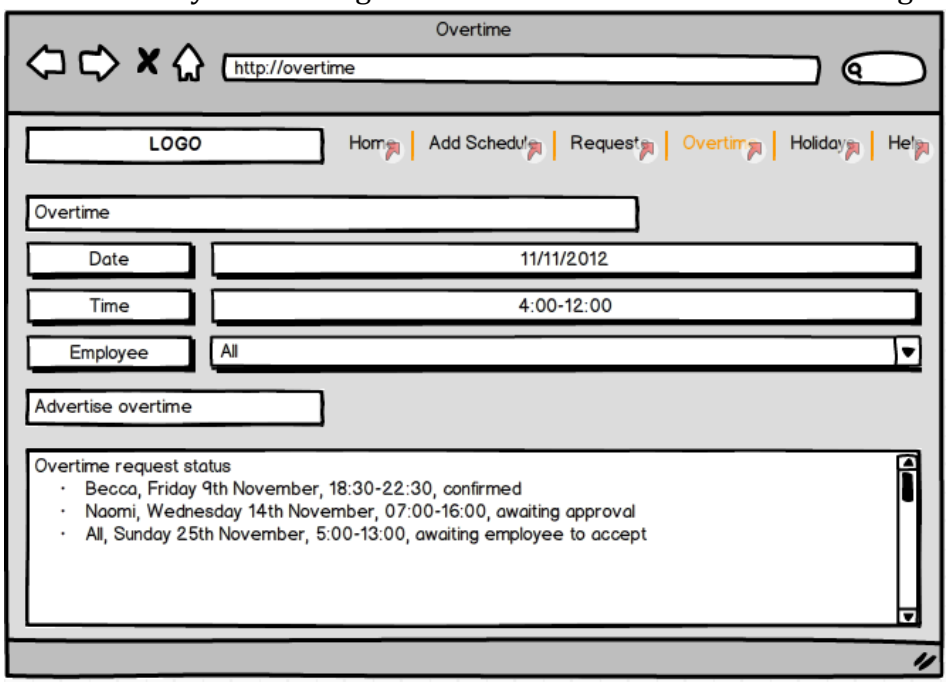


Figure 23 Manager overtime interface

The employees overtime interface is then very simple in that it is a list of all overtime requests that have been sent to either them or all employees, showing the status of the request next to it so that the user knows whether they are doing the overtime or if it has been declined etc. This can be seen in figure 24.

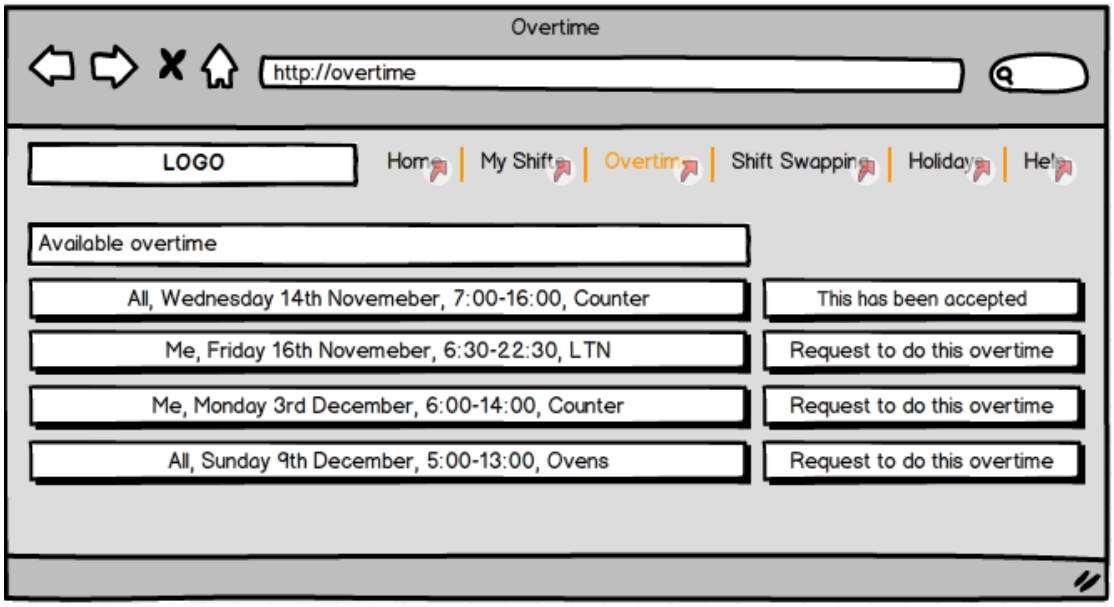


Figure 24 Employee overtime interface

5.4 Mobile User Interface Design

The next stage of design is to convert the current desktop interface designs into suitable designs for mobile devices. The aspects of the golden rules for interface design that were met in the desktop designs will continue to be met in the mobile designs, however other adjustments will have to be made in order to make the best use of small screen space. This will be done by following the rules that can be used to convert desktop elements into mobile elements as discussed in chapter 3.2.2.

To do this, it was felt that the best approach would be to choose a specific pattern and apply this to all desktop designs in order to generate the mobile designs. This will result in a consistent layout and design not only throughout the mobile site but also throughout the whole system. Looking back at the mobile design patterns discussed earlier, it is clear that this system will need to use a way to stack general information, a change of navigation will be needed, and adjustments to wide sets of information in the form of tables will be necessary.

As this system does not contain many images it will be necessary to convert the general layout of content to a linear one, which involves stacking blocks of information on top of each other, spanning the width of the device.

For the navigation, whilst a toggle menu can be customised nicely to fit the styling of the system, a select menu is much more practical. This is because it is clear to the user that this is a clickable menu, whilst also letting users be

comfortable in what they are used to because it uses the devices native select element.

Wide sets of information such as the forms to request shift swaps will take up the whole width of the desktop site. This is clearly not feasible for the mobile site, as it will become unreadable by the user. Therefore the table invert method, along with toggling between information will be used as a method to display all information on the screen.

Following these guidelines, an example of how the interface to view schedules will be converted to mobile is shown below in figure 25.

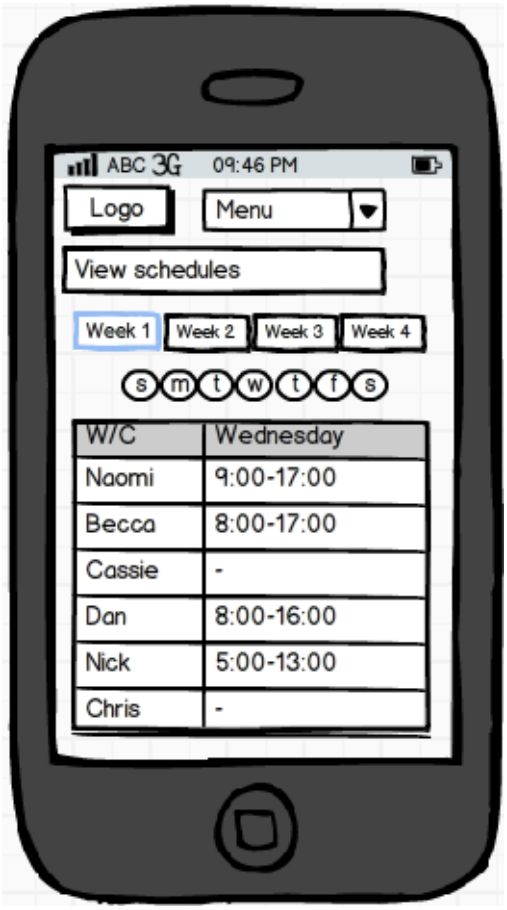


Figure 25 Home page for viewing schedules converted to mobile

For this particular instance, the table has not needed to be inverted. However, it has taken advantage of toggling parts of the table on and off. For this design, the user is able to choose week 1-4 as in the desktop version, however they can then also toggle through the days. This makes the contents width a lot thinner and therefore fits into the mobile device successfully.

An example of where the content is inverted will now be demonstrated with the employee shift swap interface, as can be seen in figure 26.

Figure 26 Example of inverted table for shift swapping mobile interface

Here you can see that, whilst in the desktop design headings such as W/C, my shift etc were along the top row, on the mobile device they are now down the left hand side of the table in order to make the content longer and thinner as opposed to short and wide as in the desktop version. For requests, this process would be exactly the same however there would be multiple 'tables' for each request stacked on top of each other to form a linear layout.

Sub-headers in the desktop site have also been converted to buttons at the top of the mobile page. This gives a solution to the problem where information belongs on the same page, but should not all be displayed in the same screen, as it would be an unreasonable length the scroll through.

The process for conversion of all desktop to mobile interfaces follows this exact same process, and so diagrams are not needed to depict each and every mobile interface.

Overall, all of these interfaces have been designed to attempt to meet the golden rules of interface design discussed in previous chapters. Another way that the interfaces will aim to follow these rules but cannot be depicted easily in initial designs is that they will provide lots of feedback to the user. Error messages will be displayed if, for example, a users login details are incorrect, or if the user chooses a date and time when they are not working to swap a shift. Confirmations will also be provided when requests have been sent so that the user knows and feels comfortable that their request has been processed. For example, if a manager sends an overtime request out, feedback will be given so

that they know this has been successfully done. In addition, it has been ensured that all interfaces follow the same form of design and layout in order to increase users' confidence within the system.

Chapter 6 - Implementation

6.1 Methodology

The methodology of a project determines the way in which it is developed. The methodology that was adapted for this project is an incremental build model, which uses aspects of the iterative methodology within it. This approach splits the project into key functionalities and components, and each is built separately in turn to create phases for implementation. Each component is shown to the client, or in this case the supervisor, once the component has been fully developed. The system is considered finished once all components have been created and the requirements have been met. If used in practice at Sainsbury's, this would reduce the effect on employees of having to deal with a new system all in one go, meaning that when done in phases, the users could get used to one component before another being introduced. This would gradually increase their confidence with the system, rather than having to adjust straight away [21].

The main components this system was split in line with the different pages/features of the website. These are as follow:

1. Login on mobile and desktop
2. Home page/view schedule desktop
3. Home page/view schedule mobile
4. Individual shifts desktop
5. Individual shifts mobile
6. Create schedule desktop
7. Create schedule mobile
8. Shift swapping desktop
9. Shift swapping mobile
10. Overtime desktop
11. Overtime mobile
12. Holiday's desktop
13. Holiday's mobile

These increments allow a working system to be released after each one, with the new increment adding to the functionality.

Each of these components was then also split into further, smaller, subcomponents to form iterations between the increments. These iterations within the components allowed for features or designs to be analysed and adjusted if it was thought not to be the best way to implement the component. Testing was also completed at each increment, and iteration of the increments, in order to ensure that functions that do not work correctly can be identified and fixed quickly. This can be done as few changes are made between iterations and so it was easier to find bugs or errors in the system. This allows for more extensive testing throughout the system, and so resulted in a higher quality product overall.

6.2 Tools and Technologies

In order to create this system, various tools and technologies were needed. As discussed in the technical research, the project would be developed using the Django web framework.

Django provides a development server in order to host the website and an SQLITE database. This can be accessed via the command 'python manage.py runserver 0.0.0.0:8000' on the terminal, which means the website can then be accessed locally or by anyone on the same network. This was a vital tool throughout implementation in order to not only test the system myself, but also have other users test the system on their own devices.

Django provides a template system where the use of HTML is required. A template determines how objects are presented to the user and is involved with the content of the webpage's, for example, a form would be included in a template. Django also provides functionality to define URL's for the web system all within one file. This tool uses pattern matching and once the URL is matched, it looks to the correct view for what code to execute next.

A view, in terms of Django is a group of views collected in a file named 'views.py'. This uses the programming language Python to perform server side scripting. When a URL has been requested this will link to the corresponding view, which in turn uses Python code to perform various operations. The most common operation in the implementation for this system is to call requests to the server/database in order to manipulate data, for example changing a record. At the end of each view a return call is made in order to tell the system which template to load for the specific URL that called the view. Therefore, it could be said that the template is concerned with how the user views the information, whereas a view decides which information is available for formatting in the template. Once the template has been loaded calls to client side scripting such as JavaScript are made. A graphical explanation of this is demonstrated in figure 27. This is said to follow an MVT process, which stands for model, view, template, which is a slight change on the widely used MVC in other frameworks.

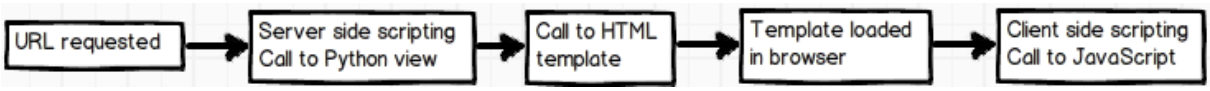


Figure 27 Flow of program calls

A simple example of this with the system being developed is that; the user visits the URL '/home/', which will then call the view named home. This view may have a request to the database to query all schedules for the user named 'Dan'. At the end of this view a return call will be made to the template named home.html. This call will then prompt the template to be loaded into the users' browser. This template file may contain code, which loads all of the schedules for Dan into a table and so this table would be displayed on screen. The call to JavaScript is then dependant on the users' interaction with the interface, however, for example, if a user clicked a button, JavaScript may be called for a confirmation message to pop up.

A final tool used in implementation is the Sublime text editor. This supports various different types of code including all of the ones used in this implementation: Python, HTML, CSS and JavaScript. This tool helped in the sense that it recognises standard code in each programming language and highlights it in the same colours. For example in Python code, variables are orange, and strings are yellow, in HTML tag beginning and ends are in pink. This greatly improves efficiency when writing code as it speeds up the process of looking for a particular piece of code, and gives a clear structured environment to work in.

6.3 Database Implementation

Django's database system works via the use of models and the programming language python. To do this, a file is a python file, models.py, is dedicated to the creation of the database. Each model in the file represents a table in the database and consists of variables specifying each field name and attributes for that field, such as maximum lengths, or whether the field is required or not. An example of the holiday model can be seen in figure 28. This creates a model named 'Holiday', along with fields named 'week_commencing_holiday', 'employee_holiday', 'holiday_day' etc. The blank attribute specifies that the field can be empty, therefore for fields such as holiday day this is set to false, whereas Boolean fields such as whether the manager has accepted the holiday allow blank to be true. This is because, for example, the manager may have not answered the request with either yes or no yet.

```
class Holiday(models.Model):
    week_commencing_holiday = models.CharField(max_length = 11, blank=False)
    employee_holiday = models.CharField(max_length = 11, blank=False)
    holiday_day = models.CharField(max_length = 11, blank=False)
    holiday_time = models.CharField(max_length = 11, blank=False)
    employee_hide_holiday = models.NullBooleanField(null=True, blank=True)
    employee_cancel_holiday = models.NullBooleanField(null=True, blank=True)
    manager_approve_holiday = models.NullBooleanField(null=True, blank=True)
    manager_decline_holiday = models.NullBooleanField(null=True, blank=True)
    manager_hide_holiday = models.NullBooleanField(null=True, blank=True)
```

Figure 28 Holiday model

This process of creating models was followed in line with the entity relationship diagram created in the database design chapter. The SQLITE database itself could then be accessed through the terminal, however Django provides an admin interface, which shows any records and tables in the database that you have enabled it to, shown in figure 29. In order to access this, it is necessary to make an admin file, which specifies the models, and fields in those models, to include in the interface. For this project, all fields were important in order to test that the right information had been sent through the system. Therefore, it was necessary to include all models and all fields for that model in the interface. Once this is complete the records can be directly accessed, added, edited or deleted from the admin interface located at 127.0.0.1:8000/admin/. As well as this, these records can be accessed in the views via the use of queries in order to manipulate the data and perform operations to create functionality to fulfil the requirements.

Figure 29 Admin interface

Another tool used, which is linked to the database implementation is model forms. A model form allows the developer to automatically create a form based on a model/database table. For example, using a model form on the Holiday model could create a form with all of the fields that the Holiday model contains and the form would only be considered valid if it meets all of the parameters specified for a field in the database such as length. This is useful when a form is required to create a new record in a database model, as when the form is saved this automatically links to the database. A variation of this used within the implementation is a model formset. This allows the developer to save a group of forms together in one go into an existing model, which will be useful in saving a set of schedules. In order to then access this form from the view, into the template it is possible to call the form name with the field, and the template will create the correct input for the user. For example if the model specified a field was Boolean, a check box would be shown when the template is loaded.

6.4 Desktop and Mobile Implementation

The next step was to implement the actual interfaces and functionality that would be present to a user interacting with the system.

6.4.1 Device detection

In order to create the different layouts for mobile and desktop, it was necessary to use the Django template system along with a special Python view [22]. This view did not link up to a template, however acted more as a function to determine whether the device accessing the system was a mobile one or not. This function sets mobile_browser to false, and checks whether the device being used has the same user agent as a list of user agents provided. If it does then mobile_browser is changed to be true. If it doesn't then it also checks a list of user agent hints such as iPhone and Android, and updates mobile_browser to true if any of these are true. If they are not then mobile_browser continues to be false. See figure 30. Then, at the start of each view an if statement determines

whether the base layout or m_base layout is sent to the template through the value of mobile_browser. These two templates, with the use of HTML and CSS, provide a default layout, which includes blocks where code can be added into, and then this layout is extended in the specific page layout. For example, both base layouts contain a content area with a block tag. This block tag has html before and after it, however when the specific page template extends the base template, this block tag can be filled in with content specific to that page. This means that there is one template for desktop and one for mobile, eliminating the need to repeat code, as only the actual content is necessary in each individual template document. In order to make the mobile template fit to a mobile screen the viewport HTML tag was used, as seen in figure 31. This changes the width of the layout to be the device width, and sets the scale to be 1.0 and user scalable to 0. This results in the mobile site not just being a shrunken version of the desktop one, however it fits to the device properly and keeps correct proportions. The actual code for the mobile and desktop sites is the same as it performs the same functions, the difference is that the information must be displayed differently; therefore the device detection above plays a big part in distinguishing between the different versions of the system. All code explained in further documentation refers to both the mobile and desktop sites.

```
mobile_uas = [
    'w3c ','acs-','alav','alca','amoi','audi','avan','benq','bird','blac',
    'blaz','brew','cell','cldc','cmd-','dang','doco','eric','hipt','inno',
    'ipaq','java','jigs','kddi','keji','leno','lg-c','lg-d','lg-g','lge-',
    'maui','maxo','midp','mits','mmef','mobi','mot-','moto','mwbp','nec-',
    'newt','noki','oper','palm','pana','pant','phil','play','port','prox',
    'qwap','sage','sams','sany','sch-','sec-','send','seri','sgh-','shar',
    'sie-','siem','smal','smar','sony','sph-','symb','t-mo','teli','tim-',
    'tosh','tsm-','upg1','upsi','vk-v','voda','wap-','wapa','wapi','wapp',
    'wapr','webc','winw','winw','xda','xda-'
    ]

mobile_ua_hints = [ 'SymbianOS', 'Opera Mini', 'iPhone', 'Blackberry', 'Android', 'Windows' ]


def mobileDetection(request):
    mobile_browser = False
    ua = request.META.get('HTTP_USER_AGENT', '').lower()[0:4]

    if (ua in mobile_uas):
        mobile_browser = True
    else:
        for hint in mobile_ua_hints:
            if request.META['HTTP_USER_AGENT'].find(hint) > 0:
                mobile_browser = True

    return mobile_browser
```

Figure 30 Mobile device detection

```
<meta name = "viewport" content = "width = device-width, initial-scale = 1.0, user-scalable = 0">
```

Figure 31 Viewport for mobile

6.5.2 Login

In terms of security and to avoid situation where an employee may use another employees account unauthorised, a security system is necessary in terms of each employee having his or her own log in. To do this, Django provides a user authentication system. The login view therefore utilises this system in order to

compare the values entered into the login form, with user accounts, as seen in figure 32. This checks that the password (automatically encrypted and decrypted by the system) and username entered match that of an account saved into the user database. This firstly assigns whether the username and passwords match into a variable 'user'. If they do match and the user is not empty it then checks that that user is flagged as active, i.e. is still an employee at the company. If it is then the user is logged in and redirected to the homepage. However if any of these are not true, then an error message is displayed and the user remains on the login page. A login decorator is added to every view, as seen in figure 33, which means that the page cannot be accessed unless the user requesting the page is logged in.

```
message = ''
username = password = ''
if request.POST:
    username = request.POST.get('username')
    password = request.POST.get('password')

    user = authenticate(username=username, password=password)
    if user is not None:
        if user.is_active:
            auth.login(request, user)
            return HttpResponseRedirect("/home/")
        else:
            message = "Your account is not active, please contact your manager."
    else:
        message = "Your username and/or password were incorrect."
```

Figure 32 Login authentication

```
@login_required(login_url='/login/')
```

Figure 33 Login decorator

6.4.3 Homepage

The homepage was implemented the same for both the manager and the employee. As the 'add schedule' functionality wasn't created yet, this started by inputting test schedule data through the admin interface described previously. To extract the data out of this database table it was then necessary to perform multiple queries in the view, as seen in figure 34. A requirement is to view 4 schedules in advance, and the number of employees there are determines a schedules length; therefore this requires five simple queries. The first gets the number of user accounts from the user table. The second then filters the Schedule object for the records from 0 to the user count and the third filters the Schedule objects for the records from user count to user count *2 etc. This means that one schedule consists of a weeks worth of shifts for each employee.

```
userCount = User.objects.all().count()
week1 = Schedule.objects.all()[0:userCount]
week2 = Schedule.objects.all()[userCount:userCount*2]
week3 = Schedule.objects.all()[userCount*2:userCount*3]
week4 = Schedule.objects.all()[userCount*3:userCount*4]
```

Figure 34 Homepage queries

These query objects then get passed through to the correct template and can be accessed with template tags as demonstrated in figure 35. This loops through the

query and allows you to access individual fields of it, for example {{column.wednesday}} within the for loop would output the times that each colleagues is working on Wednesday in week 1.

```
{% for column in week1 %}
    <tr>
        <td class="name">{{column.employee}}</td>
        <td>{{column.sunday}}</td>
        <td>{{column.monday}}</td>
        <td>{{column.tuesday}}</td>
        <td>{{column.wednesday}}</td>
        <td>{{column.thursday}}</td>
        <td>{{column.friday}}</td>
        <td>{{column.saturday}}</td>
    </tr>
{% endfor %}
```

Figure 35 Template tags from queries

As the template extends either the base or mobile base layout, the CSS styling applies automatically to all elements within the template. The final interfaces for the home page are as shown in figures 36.

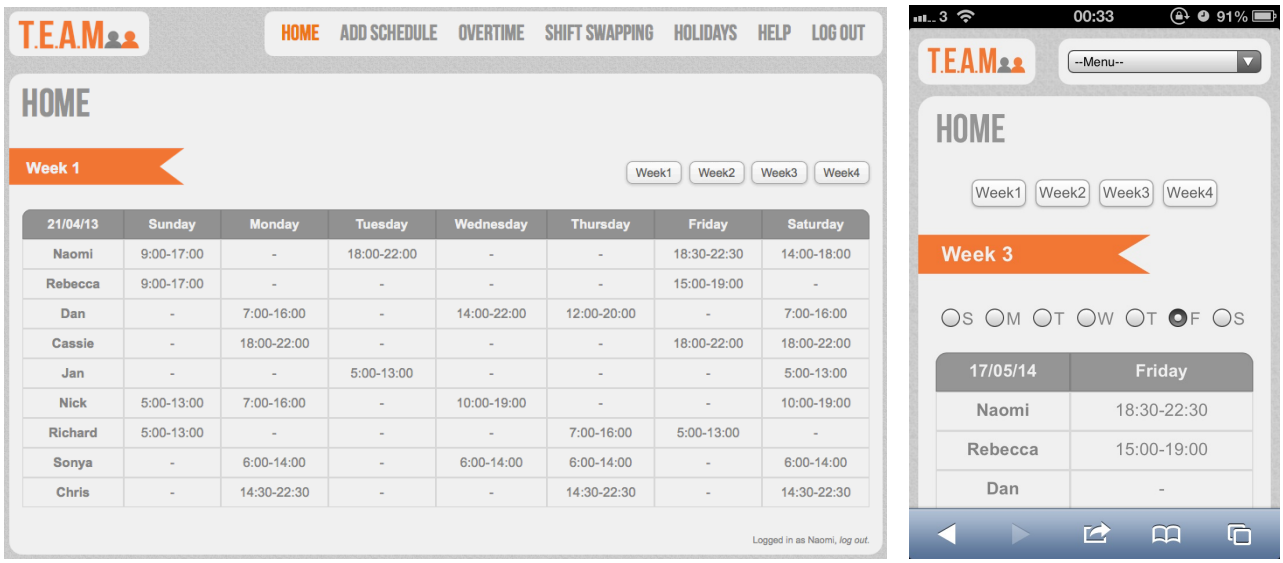


Figure 36 Desktop and mobile home page implementation

This followed the design principles explained in chapter 5.4 to convert the desktop layout to the mobile one.

The my shifts page is identical to the homepage, however only visible to the employees and only showing one employee's shifts.

6.4.4 Add schedule

The next section for implementation was the interface for the manager to create a schedule and was required that only the manager should be able to access this. To do this, a decorator was used above the view which says that the requesting user has to have permissions in order to access this page, and if they do not they are redirected to the login page. This is similar to the login decorator, however, specifies that permissions are required instead of login required. This implementation also needed the use of a model formset. All model forms go into a file named forms.py, as seen in figure 37, and is accessed by the view file. This

simply lets the server know which model it is basing the form on. In order to create a formset from this, it is needed to use formset_factory with groups together a set of forms the length of the amount of users, using extra=userCount, as can be seen in figure 37. The view then checks when a post request is made, and saves the data in that request to the model that it is linked to.

```
class ScheduleForm(forms.ModelForm):
    class Meta:
        model = Schedule
```

```
form = ScheduleForm

ScheduleFormSet = formset_factory(ScheduleForm, extra = userCount)
formset = ScheduleFormSet()
```

Figure 37 Creating a model form and a formset

The final main part of the create schedule implementation was using Djangos file system incorporated with csv.reader function from Python. The Django file system works similar to a form, in that it provides the correct input for the user when loaded in the template. Therefore, a browse files box is made available to a user. Once they have selected a file and the form is submitted, the view looks for a request post and uploads this document to the database. Cvs.reader is then used to locate the file in the directory it has just been uploaded to, and read the file, splitting each element on a comma. JavaScript is then used to fill the schedule formset with this data when the user clicks a button. The feature to view holidays for the week being added is discussed further on, as holiday features were not implemented until the end of the system. The final interfaces for the desktop and mobile system are shown in figures 38 and 39.

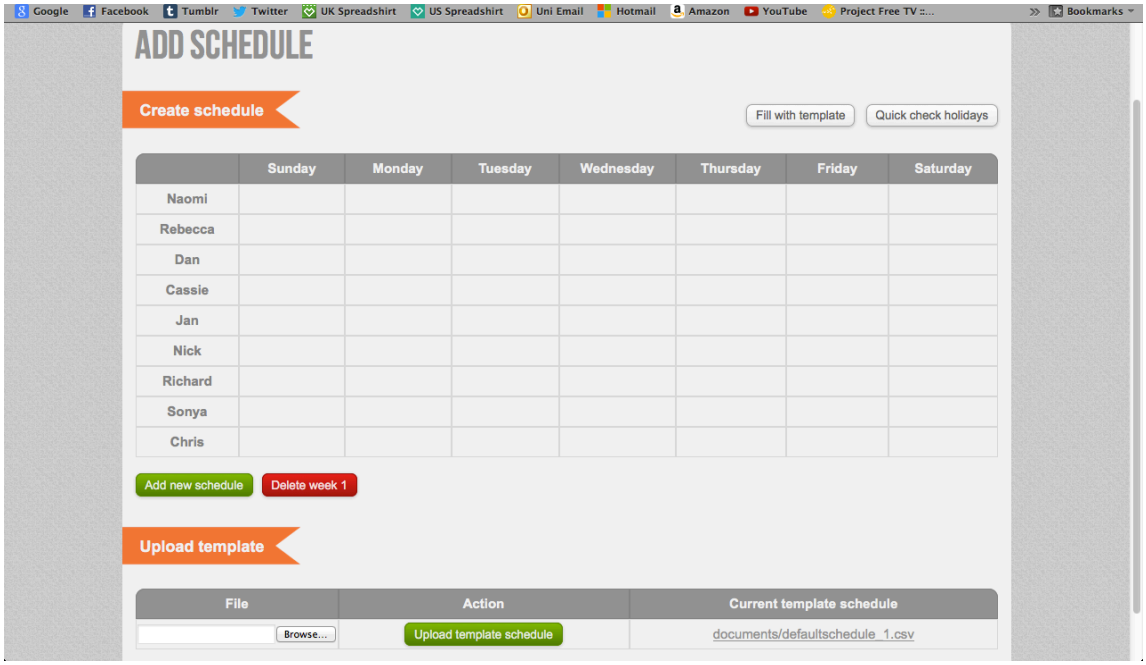


Figure 38 Add schedule desktop implementation

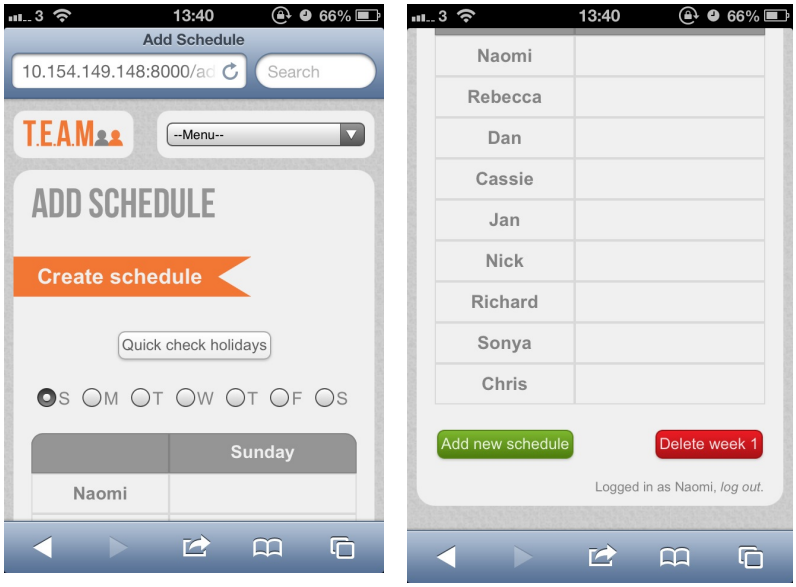


Figure 39 Add schedule mobile implementation

6.4.5 Shift swapping

The process of requesting a shift swap is again, via the use of a model form. However, when the user chooses a week commencing, employee, and day, the system automatically fills in the time of that specific shift for the employee. To do this standard HTML forms were created and eventually hidden from the end user. This automatically sends off the form when all 3 boxes have been filled in (determined by JavaScript), where a query is performed in the view. This query uses the data from the form to filter the correct time from the database. Once a request has been sent, the outstanding requests are shown to the user underneath, which provides a range of options. If the user has sent the request then there are buttons to cancel the swap, and then to hide the swap from the list. If you have received the request then you are able to approve, deny, and then hide the swap. If you are a manager the requests give you the option to approve, deny, and then hide the swap. Each of these buttons makes a call to the database in the view, which updates the relevant values in the database to true or false. For example, if the second employee accepts the swap, then employee approve overtime, is updated to be true. Different combinations of these values determine the message displayed to the user. If all of the options are true, meaning that the manager has approved the swap, then this swap needs to be reflected in the schedule. To do this the database was queried in Python to make the value for the user who requested the swap on the day requested become empty to signify they are not working anymore, and then the time that they have swapped to is updated to be the time. The same is done for the user who agreed to the swap but the other way around. For example if Naomi has swapped her 9:00-17:00 shift on Sunday, with Chris's 14:30-22:30 on Saturday, then Those two days and times would become empty indicating no shift, and Naomi on Saturday would become 14:30-22:30 whilst Chris would be shown as working 9:00-17:00 on Sunday.

The final desktop and mobile implementations of shift swapping for both manager and employee are shown in the following figures.

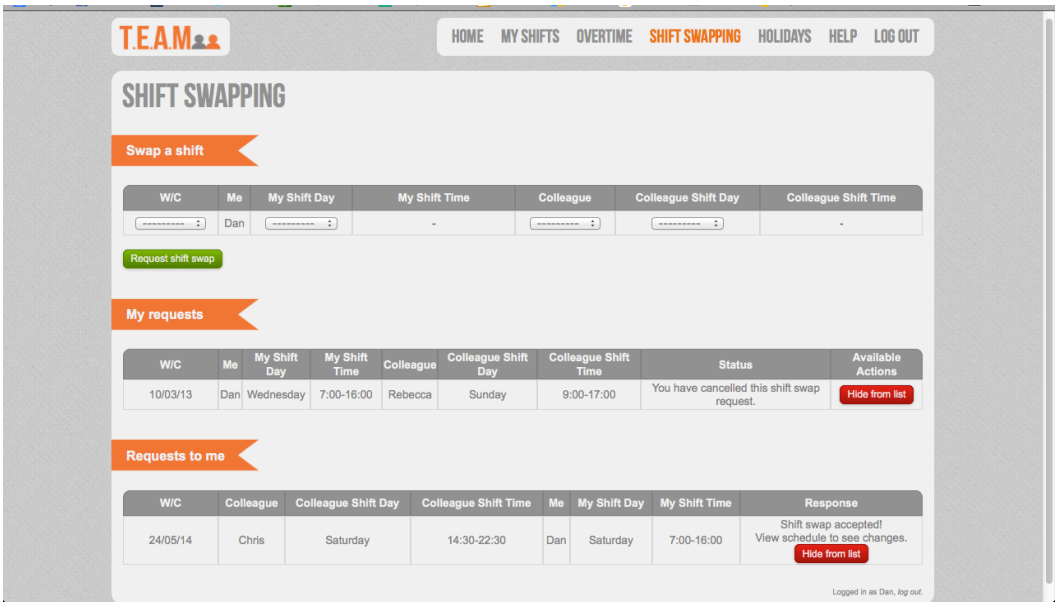


Figure 40 Employee shift swap desktop implementation

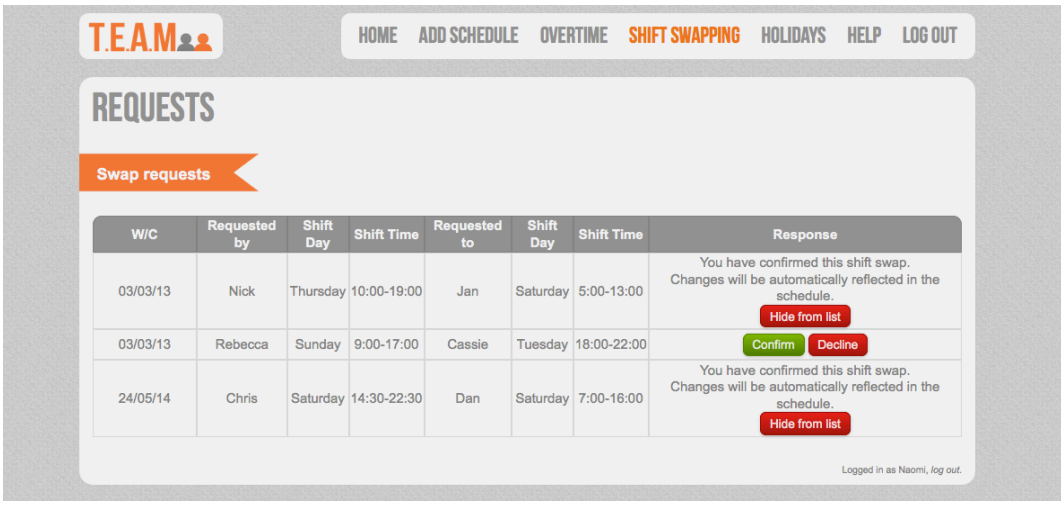


Figure 41 Manager shift swap desktop implementation

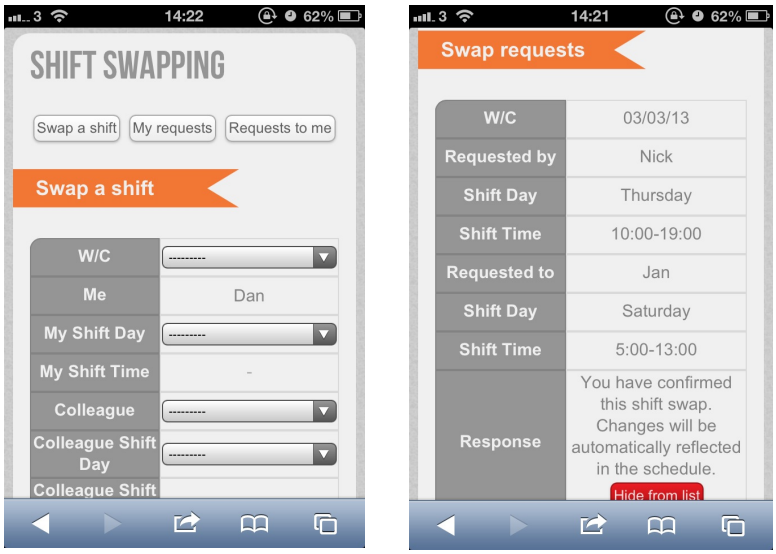


Figure 42 Employee and manager shift swap desktop implementation

6.4.6 Overtime

The overtime functionality allows the manager to broadcast overtime within the current 4 weeks of schedules available, to all employees or a specific one. To send the overtime to one employee works in the same way that shift swapping did, in terms of the request only showing to the user who it was sent to. However to send the overtime to all employees the user was saved as 'All' and if the user equalled 'All' it would be displayed, regardless of which use was logged in. Once a user accepts this swap, then the user value in the database is updated to the user that accepted it. This means that the overtime can no longer be seen by any other employee, as demonstrated in figure 43.

```
.update(employee=answer_overtime_employee, employee_approve_overtime=True)
```

Figure 43 Query updating employee value

This then allows the manager to review the swap with different available options, as explained with shift swapping. If the request was originally sent to a single user then the manager has the chance to finally approve the overtime or cancel it. However, if the overtime was originally sent to all employees, then the manager has the ability to re-open the request to employees if the user that accepted it is not suitable for the overtime, i.e. they are already working that day. Throughout the whole process, both the manager and employee have helpful messages letting them know the status of the requests worked out by the values in the database, providing meaningful and immediate feedback to the user, which is necessary in good interface design as described in chapter 3.2.1. Once the manager has finally approved the overtime, this overtime is automatically updated in the schedule, again using the '.update' function in python. The final interfaces for desktop and mobile, for both managers and employees are shown in the figures below.

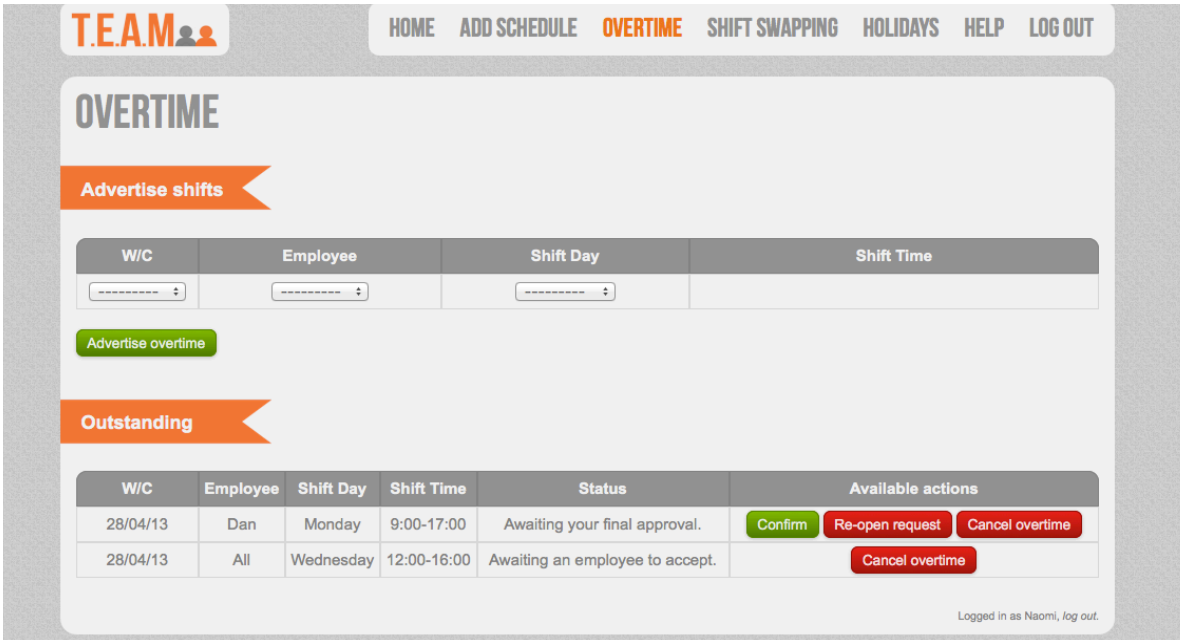


Figure 44 Manager overtime desktop implementation

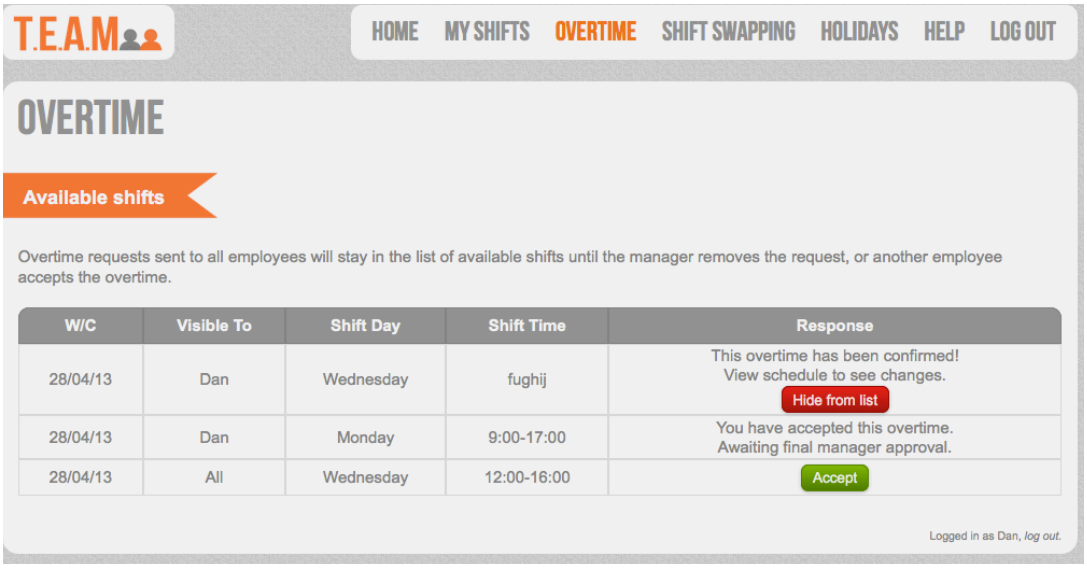


Figure 45 Employee overtime desktop implementation

The mobile versions of overtime are the same as the shift swapping interfaces, however, with tabs on the manager interface to switch between advertising overtime and viewing requests.

6.4.7 Holiday

The final main feature to implement is Holidays. The requests in this work in the same way that shift swapping and overtime do; via model forms and python queries. However when an employee has requested holiday, if the manager accepts it, this holiday is not only saved as a holiday, but it is also saved to the overtime records. As overtime is only shown to users when it is within the next four weeks of schedules, this overtime stays hidden from requests until that week commencing matches one of the first four week commencing dates available as a schedule. This means that holiday could be booked months in advance, but the manager will not need to worry about remembering to advertise this as overtime.

Another aspect involving holidays is the ability to view holidays whilst the manager is creating the schedule. This has been implemented on the add schedule page via a pop-up window. The use of JavaScript is utilised here in order to open a pop-up window in a specific place on the page, so as to not obstruct the creating of the schedule as much as possible. This pop up window displays to the user any holidays in the current week that they are making the schedule for. If the user tries to view the holidays before a week commencing date has been entered, then an error message is displayed.

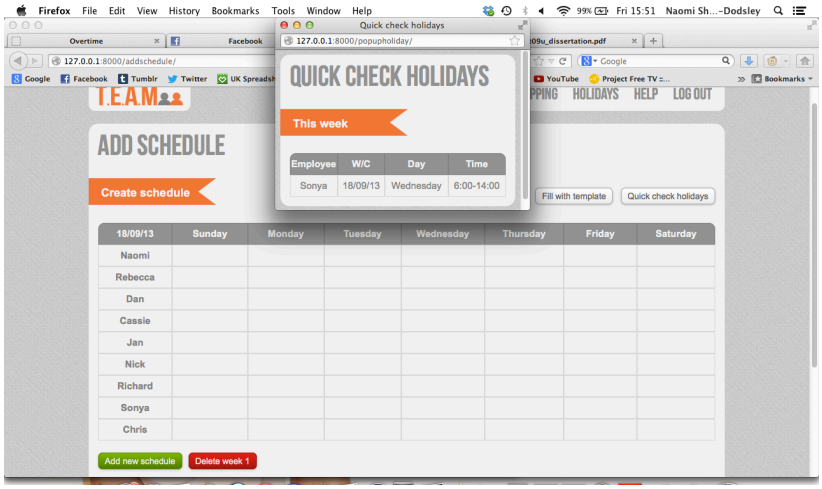


Figure 46 Popup holidays desktop manager implementation

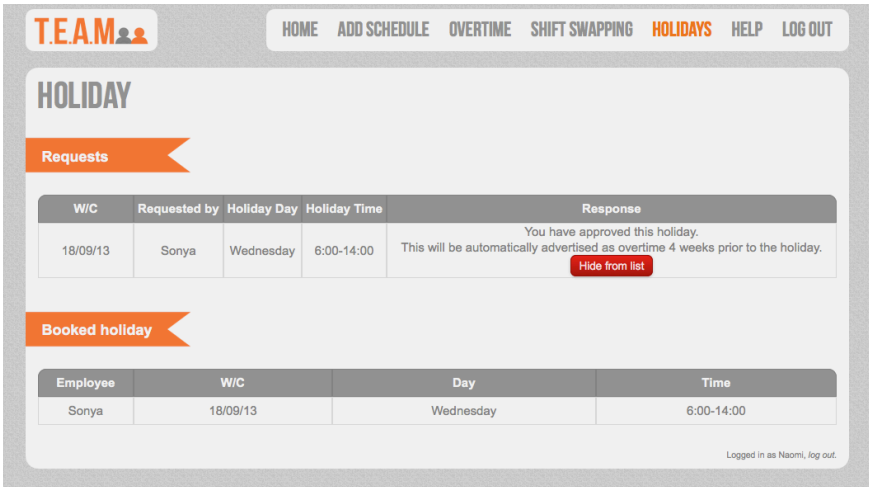


Figure 47 Manager holiday desktop implementation

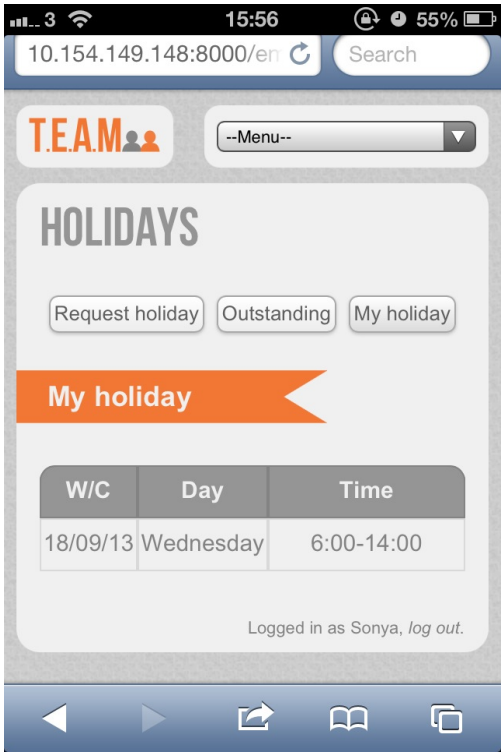


Figure 48 Employee holiday mobile implementation

6.4.8 Data record processing

Each of the main functionalities involves the use of table rows as an ordered way to show requests. However, it was found that when a button was clicked to process a request, the POST function did not know which record to submit. Therefore, when a query was made and accessed in the template view, the row id of each row would be set, in a for loop, to the id of the row outputting from the query set. The same was then also done for the button ids, please see figure 49. As a result of this JavaScript could then be used to access the specific table row that had been clicked, and use the ids assigned to get the data from the standard html form and fill in a hidden Django form with the values from that row. This meant that the forms correct row data could then be sent off to use in a view.

Figure 49 Row id configurations

6.5 Differences from original designs and requirements

The main focus and difficulty in this project is creating usable interfaces that work on both desktop and mobile devices along with creating a system to meet requirements from a real life scenario. This also includes the difficulty of designing for different users and permission types. Therefore a lot of time was spent considering the original designs, how they worked in practice, and how they could be improved throughout the phases of development and the iterations within those phases. The figures shown in chapter 6.4 are the final designs, however these vary slightly from the original designs shown in chapters 5.3 and 5.4.

Firstly, the functionality to allow managers to view holidays whilst they create schedules has been adapted. In the original design this showed a list on the main page, however there is already a lot of content on this interface, and so in implementation it was noticed that this made the screen even more, unnecessarily, very cluttered. To overcome this a popup screen was created with this information in. This means that the page becomes less cluttered, as all that was needed is a button, which takes up a lot less space. In relation to this, the 'fill schedule with template' button was moved to the top of the screen just above the schedule along with the popup holiday button. This is because they are functions which have been implemented to makes the managers experience with the system easier, and having them at the bottom of the page became a bit of an inconvenience having to scroll up and down to use them. This ties in with the golden rule of providing meaningful paths and exits by providing the user with what they need, when they need it, in an accessible and clear position.

Another aspect that has slightly changed, more in terms of functionality, is that the add schedule page only shows holidays instead of holidays, overtime, and shift swaps for that current week. This is because confirmed shift swaps and overtime are automatically changed in the schedule, and can only be made once the schedule is already online. This means that even if the decision had been made to keep them, there would never be any anyway.

A slight addition to the overtime functionality is that the manager has the ability to re-open the request if it was sent to all employees in the first place. For example, if the request was originally sent to all employees but the manager does not think that the employee who accepted is suitable for the overtime, then the manager can easily just re-open the request instead of having to make a new one.

The next major difference is that in the designs, any requests were shown in an area on the bottom half of the pages. For example, on the shift swapping page the request a swap form would be at the top, and any outstanding requests at the bottom. Once implemented it was noted that this got very unorganised once requests starting building up. Therefore this section was split into two; 'my requests', and 'requests to me', each with a sub heading banner above them. This was also repeated for overtime, and holiday included an 'outstanding holidays' section and a booked holiday section, rather than all booked holidays/requests being shown together. This provides consistent interfaces with visual clues to the location and sections available throughout the system, helping to fulfil the golden rule of interface design in reducing users' memory load.

Lastly, as an extension of the previous point, it has been ensured that all sections of the page have been split with a small banner including a heading. This change was especially important on the homepage, in order to indicate which week schedule was selected, and on the mobile site in order to let the user know which section they are on when using the buttons to change between, for example, requesting a swap and viewing outstanding swaps.

Chapter 7 – Testing

7.1 Functional Testing

Functional testing is used to ensure that the system performs how it should do in terms of inputs and outputs and is concerned with what the system does rather than how it does it. This type of testing is called black box testing, and knowledge of the systems code or structure is not necessary, in comparison to white box testing where the underlying code and structure is tested.

To perform this functional black box testing, three different types of tests were performed and each of these types of testing builds on the last. These were:
1. Unit testing
2. Integration testing
3. System testing

All test plans and results can be seen in appendix 2.1. The majority of tests are relevant to both the desktop and mobile implementations. Any tests specific only to mobile have been specified in the test case.

Unit testing begins by testing the smallest pieces of testable code. Within web applications this could include the test of form validation, submit buttons working correctly to save in the database or links redirecting through to the right places etc. This type of testing ensures that every little part of the system works as it should do. The test plans and results for this set of unit tests can be seen in appendix 2.1 under 'unit testing'.

These unit tests proved to be successful, in that all of the tests passed with the exception of one; number 12. Users only had visible links to pages that they were supposed to be able to visit, however this test aimed to check that an unauthorised user couldn't reach a page that they weren't supposed to just by putting the URL in. Once logged in as an employee the /addschedule/ URL was entered and it was found that it was possible for an employee to access this page even though they definitely should not have access to these functions. Therefore, the code was updated to include a permissions decorator on all of the views which only one type of user should be able to access. This therefore resulted in this test being fixed, passing once being retested, and any subsequent tests of the same nature passing.

Integration testing is involved in ensuring that different pieces of the system work together and communicate with each other correctly [23]. This could include, for example, passing data around correctly to different parts of a system. Integration testing combines unit tests to form larger groups of tests throughout the system. This allows the tester to know that if a unit test passed, but an integration test regarding the same piece of code fails, then the error has occurred somewhere in between the two pieces of code; either in transportation of data or by one piece of new code wrongly affecting another existing piece of code. An example of a test could be in terms of shared data areas, in that the request for one users page of shift swaps, should not affect how another users

swaps is displayed. The integration test plans and results can be seen in appendix 2.1 under 'integration testing'.

As can be seen in the test plans, this integration testing was performed in a bottom up fashion, where smaller component and modules are tested together, before testing larger sections. This goes from lower level integration testing where the system is tested between, for example, two interfaces, and then leads on to higher-level integration testing, where the system is checked that whole functionalities when complete do not affect the current functionalities and communications. This higher level of integration testing can be seen as combining the lower level tests, implying that if all of the lower level tests passed, then these tests should also all pass.

Within integration testing there was one error. This was involved with the data that the default schedule file stores and how it retrieves when uploaded. These tests worked correctly and passed as two separate unit tests (test numbers 14 and 21), therefore meaning that something was happening once these functionalities were combined. On inspecting the code, it was found that the problem was that the database was retrieving the wrong file to fill the schedule with. This means that when tested individually, the file was uploading, and the schedule was being filled with the contents of a file, however it was the wrong file. The system was accessing the previous file and so therefore the contents that the schedule got filled with was always one behind the actual intended default schedule. Once this was fixed, it was retested, and ensured that all other tests passed.

System testing then builds on unit and integration testing and assumes that both of these types of testing have already been conducted [24]. This type of testing ensures that the system works as a whole, and is focused around the system requirements and the way in which a user will interact with the final system in terms of use cases and scenarios. An example of this could include the user logging on, performing an action, and logging out, reflecting a full process from start to finish. The system test plans and results can be seen in appendix 2.1 under 'system testing'.

As stated previously, system tests are done inline with the requirements. However, as explained in chapter 6.5 the final implementation of the system changed slightly from the original requirements and design, therefore resulting in two of these tests failing. The first of these was that all of the features available on desktop were also available on mobile. However, the ability to upload a default schedule was not implemented on mobile. This is because it is unlikely to have the file on a mobile anyway and was decided that it was unlikely this would ever get used. The second is that the manager should be able to see who is on holiday, overtime and shift swaps for that current week while creating a schedule. However, this was changed previously to only be holiday that needs to be viewed, meaning that this test failed. After this the test was re-conducted with the correct description and steps, and passed.

7.2 Non-Functional Testing

Non-functional testing is involved in testing the non-functional requirements for the system. Below I described how of each set of these were tested.

7.2.1 Security

Here, it was important to check that security aspects of the system worked as they intents, to prevent unauthorised access by different people. Only employees or managers with an active account can log into the system. Once the employee has left the company, for example, but Sainsbury's may still want to keep their user record, the user can be set to inactive. If this happens, the account login should no longer work to prevent access to them. Therefore an account was made inactive via the Django interface, and tested whether it look log them in, and it wouldn't. Therefore, this works, as it should do. Following on from this, using an account that is active should allow the user to log in should the username and password combination be correct, however it should turn them way with an error message if they aren't. This all works, as it should. The final part of security testing was to ensure that the passwords were stored as an encrypted format. When the user inputs their passwords it is covered up with dots instead of the letters, meaning that users cannot see what they are inputting. Within the database Django also encrypts the password, as seen in Figure 50. Therefore meaning that all of these security requirements pass testing.

Figure 50 Django password encryption

7.2.2 Capacity

The next area to test was capacity testing. This testing used a test plan, as the functional tests did, and so can be viewed in appendix 2.2 under 'capacity'. All of these tests ensured that the database could store the needed amount of requests per hour, and that the system could handle the access. These were performed throughout an hour period by making multiple requests on iPhone and desktop. Another part of this testing was to check multiple requests of the same information, for example an employee booked the exact same holiday twice. It was found that the way the system has been designed and developed handles this quite well. If there are two requests exactly the same and an action is performed, e.g. accepting the holiday, then both requests will be accepted. If hide from list is then clicked, it will hide both requests from the list. However, if the requests are in slightly different states, it keeps them as separate requests. For example, if an employee requested holiday and then cancelled it, but realised they did want it and re-requested it, then any actions performed on the new one would not affect the old one, unless it reached the same state of being cancelled.

7.2.3 Availability

Availability was a big part of the original problem to be solved and so it was important to tackle this issue. The two requirements here are that the system should be accessible 24/7 and the system should be accessible outside of the workplace. These link together in that the system is accessible on mobile devices

as well as desktop, through the Internet. This makes the system not only accessible outside of work 24/7, but also inside of work.

7.2.4 Usability

This set non-functional requirements was hard to measure quantitatively and so the main results of this are in gathered qualitatively in the user evaluation chapter 7.3. However it is involved wit the usability of the interfaces and great care has been taken when designing these to ensure that both the desktop and mobile interfaces are easy to use, in terms of golden rules for interface design.

7.2.5 Operating environment

The operating environment is involved with what browsers the system is able to function correctly with. To test this, Safari, Chrome, Firefox, Safari on iPhone and an Android phone browser were tested, and this can be seen in appendix 2.2 under 'operating environment'. The results of these tests were positive in that all the main functionality worked correctly with each browser. However, some of the way information was presented to the user varied and so a few changes were made. For example on Chrome the hidden attribute for an option in a drop down menu did not work correctly, whilst it did on Firefox. Therefore they were changed to disabled options instead of hidden.

7.2.6 Documentation

The final system includes a help page in an easy to access location, which helps the user with any queries they may have.

7.3 User Evaluation

In order to conduct user evaluation, 5 managers, and 5 employees of Sainsbury's were asked to fill out a short questionnaire on their experiences after testing the system. The questionnaire and results can be seen in appendix 3. The first 5 users are managers, whilst the second 5 users are employees. The results of this questionnaire were of a generally positive nature, however there was some patterns noticed, and ideas for further improvement.

On analysing the results, it was noted that the managers all gave lower responses than the employees in terms of the functionality available. Upon looking at the feedback for anything they would like to see added to the system, as opposed to the employees responses for the same question, it was noted that the managers had more to say. The managers believed that the system, as an improvement, should be linked up to HR and also provide features revolving around contracted shifts and hours worked. For the scope of this project that was too big, however that would indicate why their scores were lower for that specific question. Whereas, on the other hand, as the system was designed and built mainly around the shortfalls of the current system for employees, the employees found the functionality a lot better, scoring an average of 9.

Another comment made, was that the system could be made a bit livelier in terms of including some pictures, however no other comments were made on this, so it can be seen as a minor improvement that could be made.

A final set of answers that stood out were the errors and difficulties were all 1's. This implies that no major errors occurred or any issues encountered. This demonstrates that the testing from the previous sections has worked well, and all bugs and errors so far have been found and fixed.

The final part of the user evaluation was to ask 10 other employees who work in various other retail stores to fill out the user evaluation questionnaire. The results of this can also be seen in appendix 3. Here, it can be seen that the general results are slightly lower then the first set, however, they are all relatively similar. For example the errors are low, whilst the ease of use is still higher.

This general shift in results to slightly worse reviews could be due to the fact that these employees all work in different retail stores. This emphasises the fact that the system has been built well in relation to Sainsbury's specific needs and problems with their current system. However, this also shows that the system could also work well in other retail stores, with slight tweaks to make it perfect for that specific store. For example, one comment was related to the colour scheme of orange, which has been chosen to match Sainsbury's, rather than any other store.

Chapter 8 - Evaluation

8.1 Achievements referring to the requirements

In terms of the requirements a lot has been achieved, in the sense that all of the requirements have been implemented successfully on both mobile and desktop devices. By following a structured process of, researching into the current problems, existing systems, then basing designs on these problems, and creating a requirements specification from these designs, it has ensured that throughout the whole project there has been a good understanding of what needs to be achieved and clear goals to work towards.

The requirements could be split into six main sections; add schedules, view schedules, my shifts, shift swapping, overtime, and holidays. When talking about these it is important to note that most points refer to both the mobile and desktop versions.

The add schedule section of the requirements has been implemented successfully, even including some of the features which were considered desirable at the start of the project rather than necessary. For example, the ability to use a template to fill in the schedule was rated a 5, and was deemed not necessary to meet the original system needs, however would make the system more desirable if it was implemented. This shows that the system not only meets the necessary requirements but a lot of effort has been made to add extra features in to also make the system desirable.

In terms of viewing schedules an easy to understand interface has been created for both mobile and desktop that comprises of the schedule being laid out in the same way it is currently at Sainsbury's, with four buttons to switch between the four different schedules available, similar to the way in which there would be four paper schedules to flick through in the current system.

The my shifts section is a simplified version of the viewing schedules section, however only displays the logged in users' shifts, in case they only wanted to make note of their own, easily, for the next four weeks.

Shift swapping was another main section of the requirements of the system. It is felt that the final system successfully meets these requirements and has produced a good interaction between employees and the manager, which will inevitably save time and effort when trying to arrange things like shift swaps. This has been implemented with features such as, shift times automatically being retrieved after week and days have been put in. This allows the user to play around with the swap and try different combinations, without having to switch between different pages. This also applies to the shift swaps automatically changing in the correct schedule, as the manager will not have to remember to do it, or run the risk of changing the schedule wrongly.

Overtime has been implemented with the function to not only choose one person to advertise overtime to but all employees if necessary. This is a good feature, as it does not limit the manager, and gives them options and keeps them in control

of the system. The overtime functionality also automatically places the overtime in the schedule once finally approves, again removing the need for the manager to remember to do it.

Finally, the holiday features in terms of the requirements specification, in a way exceed it. This is because holidays also links with overtime in that once a holiday has been confirmed, it automatically sets overtime to be advertised for that holiday when the holiday is 4 weeks away. As explained previously this avoids the need for the manager to remember to advertise the overtime when the holiday comes up. Because if it was left to be advertised until the schedule was made with the holiday, then there may only be 1 days notice to get overtime covered, whereas now, it will automatically advertise 4 weeks in advance which gives plenty of time.

In terms of the title, "Developing mobile and web services for shift swapping, overtime, holiday and shift management at Sainsbury's." I feel that this system has achieved this successfully and can be demonstrated throughout the points shown above, the implementation, the testing and as a system as a whole.

8.2 Personal Reflection

8.2.1 Technical

In terms of personal technical reflection, I feel that this project has been challenging. The web framework Django was something I had never used before and so learning it for a whole project under time constraints was difficult. However, I managed to do so successfully and have now learnt a new skill that I would definitely like to use in future projects and develop my knowledge further in.

This also applies for python in the sense that I had never programmed in python before, and so could have just gone with the simple option of PHP. However, I chose to challenge myself and have now found a new programming language, which I really like the syntax and simplicity of.

This project has also helped me understand data flows throughout a system a lot more, through the use of the MVT (model, view, template) pattern to separate the way in which the user interacts with the system and the way in which this information is presented to the user. This was something I had very little knowledge of before the project and so will now understand the structure of developing applications like this in the future.

In terms of the platforms developed for, whilst I have had experience in web design, I had never developed mobile system, nor had I created a desktop website with this much functionality. Any that had been created in the past were more focused on the layout and design.

Overall, I feel that this project has challenged me, and a lot of work has had to be done from a technical aspect in order to achieve the functionalities that have been achieved.

8.2.2 Management

Time management and organisation are inevitably key parts in projects, which should not be neglected. I realised this from the very beginning of the project, even when choosing a topic, and so feel that I have ensured my work has been completed consistently throughout the project's duration rather than all towards the end.

In order to have good time management, time plans were needed. Please see appendix 4.1 for the original time plan. This time plan was quite strict, in order to give myself plenty of contingency time if something did go wrong, or if I had difficulty implementing a certain feature. Clearly though, as time went on the plan and timescale of components did change, and so the final time plan can be seen in appendix 4.2.

The original time plan aimed to have all research and design completed before the Christmas break, in order for implementation to start over the break. This was successful and everything had been completed in time for the plan to go ahead as proposed, however as exam time came around, the project took a lower priority and so 2 weeks over Christmas were not utilised for this project.

This called for the plan to be re-ordered and so, as shown in the Gantt charts, the second half of the project has been adjusted. This meant that there was less implementation time than originally expected, but because the first plan was so strict, there was still plenty of time to implement the features to a high standard.

Ordering the work and backing up the project was also essential and a large part of organisation. The contents of this project were in organised folders, such as "Research", "Documents", "Design" in order to ensure finding documents or ideas to refer back to would be as easy as possible. Regular backups of work were made using Apples time machine function which stores backups of all folders in date order so you can revert to previous versions very easily if something goes wrong, in terms of code or files corrupting, for example.

In conclusion of my personal reflection, I feel that I have managed the project and my time well. This meant that enough time was present to evaluate and analyse as I was working and ensure that the work was of a high quality.

8.3 Future Improvements and Extensions

Although a solid system has been produced there is inevitably, as with any system, room for extensions and improvements. As a result of user testing and evaluation, along with the limitations of the developed system identified in chapter 4.4, a number of ways in which to improve the system have been identified below.

The first improvement would be to allow the user more functionality with the holiday requests; for example, the ability to cancel holiday after the manager has finally accepted it. It is unfeasible to let the employees cancel overtime or shift swaps after the manager have approved due to the short period in which they

are arranged in, however as holiday can be booked up to a year in advance usually, plans can change, and the employees functionality should be able to reflect this.

In terms of adding functionality, a big step would be to link the system with a HR system. This could include the use of contracted hours and hours worked in order to calculate pay etc. This would be a very large step, which was too big for the scope of this project, however in the future would make the system usable by a wider range of users, rather than a separate one for HR. This would create a fully integrated system that would be easier to maintain as cross compatibilities with another system would be avoided.

Currently, the managers are not able to perform functions themselves such as swapping shifts and requesting holiday. This is because they would need another user with higher up permissions to authorise this for them. However, in the future this would be necessary, which also ties in with linking the system to HR, as they could be the user responsible for authorising those kinds of requests from managers.

A large extension to this project would be the use of automatically checking the feasibility of swaps and overtime, rather than relying on the employees being sensible, or the manager having to check it themselves. Although the manager would probably want to double check it in the schedule, the system could be improved to check the schedule where the request or swap is looking to take place, and report the status of it back to the manager. This would maintain the managers control, whilst also making this easier for them by not having the leave the page to check. Another improvement with overtime could be that the system does not allow the employee to accept the overtime if they are already working that day.

Finally, a native app could be created for iPhone, Android and Blackberry users, as well as keeping the current web app. This would give the advantages discussed at the start of the project, such as the app being downloaded to your device. This means that just clicking a button can access it, rather than having to go to the web address every time.

Chapter 9 - Bibliography and References

[1] Herzberg, F., Mausner, B. and Snyderman, B. (1959) The motivation to work, 2nd edition, New York: John Wiley.

[2] Herzberg's Motivators and Hygiene Factors [Online] Available at: http://www.mindtools.com/pages/article/newTMM_74.htm

[3] Frederick Herzberg - Theory of Motivation [Online] Available at: http://www.trainanddevelop.co.uk/article/frederick-herzberg-theory-of-motivation-a78

[4] Steinberg, S. (2007) An introduction to communication studies, Cape Town, South Africa: Juta & Co, Ltd, pp.22-23.

[5] Marcus, A. Principles of Effective Visual Communication for Graphical User Interface Design [Online] Available at: http://www.diliaranasirova.com/assets/PSYC579/pdfs/05.2-Marcus.pdf.pagespeed.ce.V_7c8MiNPB.pdf

[6] Mobile Website vs. Mobile App: Which is Best for Your Organisation? [Online] Available at: http://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/

[7] Hoober, S. and Berkman, E. (2011) Designing Mobile Interfaces, 1st edition, Canada: O'Reilly Media.

[8] Web Design Patterns for Mobile Devices [Online] Available at: http://www.hillside.net/plop/2012/papers/Group%203%20-%20Coyote/Web%20Design%20Patterns%20for%20Mobile%20Devices.pdf

[9] Django Documentation [Online] Available at: https://docs.djangoproject.com/en/1.4/

[10] ITWERCS [Online] Available at: http://www.itwercs.com/

[11] OurLounge [Online] Available at: http://www.ourlounge.co.uk

[12] Workforce Scheduler™ [Online] Available at: http://www.kronos.co.uk/scheduling/employee-rostering-software.aspx

[13] Choosing your next project's language [Online] Available at: http://www.techworld.com.au/article/398573/python_vs_php_choosing_your_next_project_language/

[14] Sainsbury's factsheet – A great place to work [Online] Available at: http://www.j-sainsbury.co.uk/media/1375856/csr_factsheet_great_place_to_work.pdf

[15] Business Studies for AQA, Newcastle upon Tyne: Coordiation Group Publications Ltd, pp.44-45

[16] Douglas Mcgregor - Theory X Y [Online] Available at: http://www.businessballs.com/mcgregor.htm

[17] Mandel, T. The golden rules for interface design [Online] Available at: http://theomandel.com/wp-content/uploads/2012/07/Mandel-GoldenRules.pdf

[18] Non-functional requirements checklist [Online] Available at: http://leadinganswers.typepad.com/leading_answers/2009/03/nonfunctional-requirements-minimal-checklist.html

[19] How to create use cases [Online] Available at: http://www.usability.gov/methods/design_site/usecasesresource.html

[20] Balsamiq [Online] Available at: http://www.balsamiq.com/

[21] Incremental methodology [Online] Available at: http://en.wikipedia.org/wiki/Incremental_build_model

[22] Build a mobile and desktop friendly application in Django http://mobiforge.com/developing/story/build-a-mobile-and-desktop-friendly-application-django-15-minutes

[23] Integration testing [Online] Available at: http://www.cs.bham.ac.uk/~exc/Teaching/STesting/Lectures/13.%20Integration%20testing.pdf

[24] System testing [Online] Available at: http://www.cs.bham.ac.uk/~exc/Teaching/STesting/Lectures/14.%20System%20testing.pdf

Appendices

Appendix 1 - Blog Entry and Replies

The informal blog entry I made is shown below:
“Anyone who has a part time job please answer below?
Do you think a system where you could:
- The manager could upload the schedule.
- You could view your schedule/working hours for your work/department.
- Request to swap shifts with other employees over it.
- Overtime could be advertised/offered and you could accept on the system.
- The schedule would automatically update with any changes made from swaps, overtime, holiday etc.

would be a good idea?

It would be online as a website, so the employee could get on it on their laptop at home, or through the internet on their phone etc. Rather than having to look at the schedule at work, or waiting until you go in to arrange something, it could be done over the system at any time.”

The replies I got were as follows:
- User 1 answered: “Yeah, this sounds like a solid idea. If you want to go above and beyond, you could make the app/site functional with Outlook, etc.”

- User 2 answered: “My job already kind of started doing something like that. We can request vacation from home or submit leave requests for doctor visits.”

- User 3 answered: “As a Manager I would say that's a brilliant idea. Except if the site or program went down. But still very good thinking!”

- User 4 answered: “Yes it’s a good idea!”

- User 5 answered: “It’s a great idea! My job has this system but you can definitely make improvement. Ex. make the system aesthetically pleasing.”

- User 6 answered: “A lot of companies have something similar, but yes, the parts you added like being able to request swaps and stuff is a very good idea.”

- User 7 answered: “That would save so many lives at work.”

- User 8 answered: “Good idea!!”

- User 9 answered: “That sounds like an awesome idea!”

- User 10 answered: “Yes! Definitely! Evaluation point could be that it keeps business and personal lives more separate than having mobile nos etc! G’luck!”

- User 11 answered: “YES! it would be sooo convenient!”

I then messaged a few of these users to ask them to expand on what they had said:

- User 2
Q) Thank you. But nothing which incorporates the ability to shift swap and accept overtime etc?
A) I work in collections for a bank and our schedules are pretty much the same every day for part time. If you need to come in earlier or later, you have to speak with a supervisor and then it goes to our general manager. Overtime is rarely offered because of the budget, but when it is, we have to e-mail our manager and she'll put us on the schedule. Good luck with your project!

- User 5
Q) Thanks for answering. Could you give me any details on what the system your company uses actually does, and what kind of company it is, i.e. retail, a lot of part-time workers? Etc. would appreciate it a lot!
A) The system that my job (Hooters) uses is called IT WERCS. It's an all-in-one restaurant solution that includes all of the features that you mentioned along with reporting, inventory management, POS terminals (digital cash registers) and online ordering. This system manages about 50 employees; some employees are full time, while many are part time. This system allows employees to request days off and swap shifts but it's ultimately up to the managers to actually make the schedule.

Q) Thank you! Does the system allow you to actually request shift swaps with a certain person so they accept and then it gets forwarded to the manager, or does the system just let you say 'I want to swap this shift' and the manager has to sort the rest out?
A) The system doesn't request shift swaps. It's up to the employee to look at the schedule and look for a coworker who isn't working on that day and notify the employee themselves before a swap. After that, they can go into the system and make changes to the schedule. Since this process is somewhat inefficient, we often pencil in the changes on the master schedule so that the manager on the shift knows. What I would like to see on my job is a digital display in the back that only employees can see that allowed you to view the schedule. This way, with the manager's clearance, touch and drag shifts onto the schedule. Also, when swaps are requested the person that you wish to swap shifts with will get a notification (maybe via SMS) on which they can approve/deny the schedule change. This takes the manager out of the equation, pending that all of the employees have the same work ethic (manager doesn't want a slow worker on a day where business is predicted to be busy).

-User 6
Q) Thanks for answering. Could you give me any details on what the system your company uses actually does, and what kind of company it is, i.e. retail, a lot of part-time workers? Etc. would appreciate it a lot!
A) I work for JC Penney, so retail. And yes, there are a lot of part time workers (though the fact that it's the summertime and a lot of teens are out of school may have something to do with that). The basics of the system are that each associate

has an account. On that account, their general info given through the application process is listed. The associate puts in their general availability, and they also have a feature where you can request days/time off.

Q) Thank you. So it is similar but nothing to do with overtime/shift swaps? Just holiday requests?
A) Yeah the overtime availability and shift swap options make your proposal unique. And it was no problem. Good luck with your assignment.

Appendix 2 - Test Plans and Results

2.1 Functional test plans

Unit testing

Test number	Description of test	Data used	Expected result	Result
Login				
1	User logs in to system with valid details	Dan Sainsburys1	User is logged in and redirected to home page	Pass
2	User logs in to system with invalid details	Dan Sainsburys2	User is presented with error message and redirect to login page	Pass
Homepage				
3	Homepage link	Homepage navigation button clicked	User is redirected to the homepage and correct content is shown	Pass
4	User clicks a week selection button	Week 1 button clicked	Week 1 is shown in the sub header and schedule is updated to represent week 1	Pass
5	User clicks a week selection button	Week 2 button clicked	Week 2 is shown in the sub header and schedule is updated to represent week 2	Pass

6	User clicks a week selection button	Week 3 button clicked	Week 3 is shown in the sub header and schedule is updated to represent week 3	Pass
7	User clicks a week selection button	Week 4 button clicked	Week 4 is shown in the sub header and schedule is updated to represent week 4	Pass
8	User clicks day radio button on mobile	"M" button clicked on mobile	Schedules changes to display Mondays shifts	Pass
My shifts				
9	My shifts link	My shifts navigation link is clicked	User is redirected to my shifts page and correct content is shown	Pass
10	Employee clicks week selection button on mobile	Week 2 button is clicked on mobile	Content is updated to show only that week 2s shifts	Pass
Add schedule				
11	Add schedule link	Add schedule navigation link is clicked	User is redirected to the add schedule page and correct content is shown	Pass
12	Add schedule URL	Add schedule URL is input by an employee without permissions to access the page	User is redirected to their home page and access is not allowed.	Fail
13	Add schedule	Add schedule	User is	Pass

	URL	URL is input by an employee without permissions to access the page	redirected to their home page and access is not allowed	
14	Fill schedule with default template	Fill with template button clicked when a default file is uploaded	Schedule fills with the contents of the specified default template file	Pass
15	Fill schedule with default template	Fill with template button clicked when no default file is in the database	Schedule remains blank	Pass
16	Quick check holidays with valid date	19/05/13	Quick check holidays pop up is opened with holidays in that week	Pass
17	Quick check holidays with invalid date	199/05/13	Pop up is not opened and error message is displayed asking for valid date dd/mm/yy	Pass
17	Add schedule with full data	All fields full including date in the form dd/mm/yy	Confirmation that schedule has been created is alerted and schedule is saved to database	Pass
18	Add schedule with missing data	Any one or more field empty	Error message is displayed	Pass
19	Add schedule with invalid date	199/05/13	Error message is displayed	Pass
20	Delete week 1 schedule confirm	Delete week 1 button pressed then	First week of schedules is deleted from	Pass

		ok pressed	the database	
21	Upload file, with valid data	Upload file button is pressed	Selected file is uploaded and displayed as current default schedule file	Pass
22	Upload file, with invalid data	Upload default file with no file selected	Error message is shown	Pass
23	Upload file with invalid file type	Upload default file with .txt extension	Error message shown. Asks for .csv file	Pass
Overtime				
24	Overtime navigation link as manager	Manager clicks overtime link	User is redirected to overtime page and correct contents is shown for manager permissions	Pass
25	Overtime navigation link as employee	Employee clicks overtime link	User is redirected to overtime page and correct contents is shown for employee permissions	Pass
26	Advertise overtime with valid input	Week commencing chosen from dropdown, employee chosen from dropdown, shift day chosen from dropdown and shift time entered	Overtime request is saved in database	Pass
27	Advertise overtime with invalid input	No week commencing chosen	Error message is shown	Pass
28	Manager cancel	Cancel overtime	That specific record is	Pass

	overtime	button clicked	updated to be cancelled in the database	
29	Manager confirm overtime	Confirm overtime button clicked	That specific record is updated to be confirmed in the database	Pass
30	Manager re-open overtime to all employees	Re-open overtime button clicked	That employee for that specific record is updated back to "All"	Pass
31	Manager hide overtime from list	Hide from list button clicked	Overtime record is hidden from the interface	Pass
32	Manager outstanding overtime on mobile	Manager clicks outstanding overtime button on mobile	Content changes to display outstanding requests only	Pass
33	Manager advertise overtime on mobile	Manager clicks advertise overtime button on mobile	Content changes to display advertise overtime content only	Pass
34	Employee accept overtime	Accept overtime button clicked	Overtime is accepted for that specific employee	Pass
35	Employee decline overtime	Decline overtime button clicked	Overtime is declined by that specific employee in the database	Pass
36	Employee hide overtime from list	Hide from list button clicked	Overtime record is hidden from the interface	Pass
Shift Swapping				
37	Shift swapping navigation link as employee	Employee clicks shift swapping link	Employee redirected to shift swapping page with	Pass

			correct content showing	
38	Shift swapping navigation link as manager	Manager clicks shift swapping link	Manager redirected to shift swapping page with correct content showing	Pass
39	Request shift swap with valid input	Option chosen from each drop down box and times displayed	Shift swap request sent, saved in database and confirmation message pops up.	Pass
40	Request shift swap with invalid input	Week commencing missing, second employee not chosen, shift days not chosen or employees do not have a shift to swap with	Error comes up for each issue that is present	Pass
41	Employee cancel swap request	Click cancel swap button	That specific swap request is marked as cancelled by employee in the database	Pass
42	Employee hide request from list	Click hide from list	That specific swap request is hidden from list	Pass
43	Employee accept request from another	Accept request button clicked	That specific swap request is marked as accepted by second employee in the database	Pass
44	Employee decline	Decline request	That specific swap is	Pass

	request from another	button clicked	marked as declined by second employee in database	
45	Employee swap a shift on mobile	Employee clicks swap a shift button on mobile	Content is updated to only show the request swap content	Pass
46	Employee my requests on mobile	Employee clicks my requests button on mobile	Content is updated to only show shift swap requests from that user	Pass
47	Employee requests to me on mobile	Employee clicks requests to me button on mobile	Content is updated to only show requests sent to that employee	Pass
48	Manager confirm shift swap	Confirm request button clicked	That specific shift swap is confirmed in the database	Pass
49	Manager decline shift swap	Decline request button clicked	That specific shift swap is declined	Pass
50	Manager hides shift swap	Hide shift swap button clicked	That specified record is hidden from the interface	Pass
Holidays				
51	Holiday form sent with valid data	Week commencing filled in dd/mm/yy, shift day chosen, and time filled in	Holiday is sent, and record added to the database	Pass
52	Holiday form sent with empty data	Shift day not chosen	Error presented to the user and request not sent	Pass
53	Holiday form sent with invalid data	Week commencing not in for	Error presented to the user and	Pass

		dd/mm/yy	request not sent	
54	Employee cancel holiday	Cancel holiday button clicked	That specific holiday record is updated to be cancelled in the database	Pass
55	Employee hide holiday request	Hide from list button clicked	That specific holiday request	Pass
56	Employee request holiday on mobile	Employee clicks request holiday button on mobile	Content is updated to show only request holiday content	Pass
57	Employee outstanding holiday on mobile	Employee clicks outstanding button on mobile	Content updates to show only outstanding holiday requests	Pass
58	Employee booked holiday on mobile	Employee clicks booked holiday button on mobile	Content updates to show only booked holiday content	Pass
59	Manager confirm holiday	Confirm holiday button clicked	That specific holiday record is accepted	Pass
60	Manager decline holiday	Decline holiday button is clicked	That specific holiday record is declined	Pass
61	Manager hide holiday request from list	Hide from list button is clicked	That specific record is hidden from the list	Pass
62	Manager booked holiday on mobile	Manager clicks booked holiday button on mobile	Content changes to only show booked holiday content	Pass
63	Manager holiday	Manager clicks holiday	Content changes to	Pass

	requests on mobile	requests button on mobile	show only holiday requests content	
Logout				
64	Logout redirect	Logout navigation button is clicked	User is redirected to the login page	Pass

Integration testing

Test number	**Objective**	**Description of test**	**Expected result**	**Actual result**
Lower integrated tests				
1	Check the link between add schedule and homepage	Delete week 1 is clicked	Week 1 schedule is removed from homepage	Pass
2	Check the link between add schedule and homepage	Schedule is created, and add schedule is clicked	New schedule should appear on homepage	Pass
3	Check the data transfer in file upload	File is chosen, upload button is clicked and then fill schedule with template is clicked	Same data from the original file is displayed in the schedule	Fail
4	Check the data transfer in file upload	File is chosen, upload button is clicked and then fill schedule with template is clicked	Same data from the original file is displayed in the schedule	Pass
5	Check the correct date is being sent to the popup holiday window	Fill in week commencing, click quick check holidays button	Correct holiday is shown for the week commencing date entered	Pass
6	Check the data transfer of overtime to correct accounts	Advertise overtime to specific employee	Only that employee receives a request in their overtime, no	Pass

			other employee can see the request	
7	Check the data transfer in overtime responses	Respond to overtime using any of the options	Correct overtime is updated to the other user with the correct outcome. E.g. accept is clicked, correct record is changed to accepted	Pass
8	Check link between automatic shift times in shift swaps and the schedule	Select week commencing, employee and day for a swap	Correct time from the schedule is transferred to the users interface	Pass
9	Check the data transfer of shift swaps to the correct account	Submit shift swap request	Second employee can view request with all the correct data	Pass
10	Check the data transfer when all employees have accepted	Accept shift swap request	Manager can view request with all the original, correct, data	Pass
11	Check that holiday request data is sent to the manager correctly	Send holiday request	Manager can view request with the original data sent	Pass
12	Check that approved holiday correctly converts request to overtime	View overtime advertisements once the schedule of a week of a holiday has been created	Overtime for that holiday appears with the correct date/time etc for when the employee is on holiday	Pass
Higher integrated tests				
13	Check that	Check existing	All works	Pass

	add schedule functionality as a whole does not break any other part of the existing homepage	features work as originally did before add schedule was created	correctly	
14	Check that overtime functionality does not break any other part of the existing homepage, or add schedule	Check that existing features work as originally did before overtime was created	All works correctly	Pass
15	Check that shift swapping functionality does not break any other part of the existing homepage, add schedule or overtime	Check that existing features work as originally did before shift swapping was created	All works correctly	Pass
16	Check that holidays functionality does not break any other part of the existing homepage, add schedule, overtime or shift swapping	Check that existing features work as originally did before holidays was created	All works correctly	Pass

System testing

Test number	**Related requirement(s)**	**Description of test**	**Steps taken**	**Expected result**	**Actual result**
1	1	Log in with an account which has manager	Log in using, Naomi Abcd	User is logged in and is shown the	Pass

		permissions		relevant manager functionalities i.e. they have “add schedule” as a menu choice	
2	1	Log in with an account which has employee persmissions	Log in using Rebecca Sainsburys1	User is logged in and is shown the relevant manager functionalities i.e. they have “my shifts” as a menu choice	Pass
3	2	Log in as any employee	Log in using Nick Sainsburys1	Message at the bottom of the screen says “logged in as Nick”	Pass
4	3	Check that features on desktop are all available on mobile	Log in, click on shift swapping, and perform request. Repeat on mobile device	Functionality works the same on both devices.	
5	3	Check that features on desktop are all available on mobile	Log in, click add schedule, upload default schedule and repeat on mobile device	Functionality works the same on both devices	Fail
6	4	View 4 weeks schedules in advance	Log in, get redirected to homepage, click the week 1-4 buttons one after each other.	All four schedules are viewable.	Pass
7	5	View up to 4 weeks of personal shifts	Log in, click on ‘my shifts’ page, and four weeks should be viewable, provided	There is a full 4 weeks worth of individual shifts available for	Pass

			there are 4 schedules available	viewing	
8	6	Managers view up to 4 weeks schedules in advance	Log in, get redirected to homepage	Be able to see and click between 4 weeks worth of upcoming schedules	Pass
9	7	Create schedule, including times and a week commencing date	Log in, click add schedule page, fill in schedule so that all fields are full, click create schedule	Schedule is filled in and made with all of the information input	Pass
10	8	Check who is booked off on holiday that week, overtime and shift swaps whilst creating a schedule	Log in, click add schedule, fill in date commencing	An area is available showing all the holidays shift swaps and overtime currently set for that week commencing	Fail
11	8	Check who is booked off on holiday that week, whilst creating a schedule	Log in, click add schedule, fill in date commencing, click quick check holiday button	A popup area is available showing all the holidays currently approved for that week commencing	Pass
12	9	Fill schedule in with a default template	Log in, click add schedule, click fill with template button	If default schedule is available, fill with the contents of the current default schedule	Pass
13	10	Upload new file for template schedule	Log in, click add schedule, browse for file, click upload file. Click fill with template	Schedule should reflect the file just uploaded	Pass

14	11	Editing schedules through Django admin	Visit /admin/, click on Schedules, select record and click edit	Manager can make changes and save	Pass
15	12, 13, 14	Request to swap a shift with another employee in the same week	Log in, go to shift swapping page, fill in form to request swap	Swaps should only be available in the same week and there should be automatic indication of shift times available	Pass
16	15	Check the status of the shift swap request	Log in, go to shift swapping and view swaps both sent and received	Each request should have its own status	Pass
17	17	Managers have final approval of the swap	Log in, go to shift swapping	Check that they have the final approval of requests after thee employees have agreed between each other	Pass
18	16	Employees should be able to accept or decline swaps sent to them	Log in, go to shift swapping, click accept or decline on a swap	System allows them to perform either option	Pass
19	18	Shift swap automatically updates in the schedule	Log in, go to shift swapping, accept request, go to homepage, check the schedule week/day/time	Schedule should have been automatically updated with correct changed for both employees	Pass
20	19	Employee has the ability to	Log in, go to shift	Employee has the	Pass

		cancel shift swap requests up until the manager has approved them	swapping, click cancel request	opportunity to cancel if the status of the shift swap is not approved	
21	20	Manager has the ability to advertise overtime to one employee	Log in, go to overtime, fill in form and set employee to one specific name	Only that employee gets the overtime advertisement	Pass
22	21	Manager has the ability to advertise overtime to all employees	Log in, go to overtime, fill in form and se employee to one specific name	All employees can see the overtime request	Pass
23	22	Employee should be able to accept or decline overtime request	Log in, go to overtime, choose either accept or decline for request	Correct option is executed on click	Pass
24	23	Manager has the final approval of overtime	Log in, go to overtime, and finally approve or deny any requests that employees have accepted	All requests originally sent by the manager are re-confirmed	Pass
25	24	Overtime automatically updated in the schedule	Log in, go to overtime, final accept overtime	Overtime should be in the correct schedule for that week commencing / day / time	Pass
26	25	Managers should be able to can overtime advertisements, up until they have been finally approved	Log in, go to overtime, cancel request	Overtime status should reflect the cancellation in the employees requests	Pass
27	26	Managers and	Log in, go to	All should	Pass

		employees should both have an indication of the status of the overtime request	overtime, look at outstanding requests	have a suitable status message or available actions	
28	27	If an employee accepts a request that has been sent to all, then the request should be hidden from other employees	Log in, go to overtime, accept a request sent to all employees	Request should now be hidden from all other employees	Pass
29	28	Employee request holiday	Log in, go to holiday page, fill in holiday form	Holiday request is sent off	Pass
30	29	Managers should be able to approve or deny holiday requests	Log in, go to holiday page, approve or deny a request	Correct action is performed depends on response	Pass
31	30	Employees cancel request up until the manager has finally approved it	Log in, go to holiday page, cancel request	This function is allowed if the status of the holiday isn't already approved	Pass
32	31	Employees and managers have an indication of the status of the requests	Log in, go to holiday page	Each request should have the status it is currently in	Pass

Non-functional test plans

Capacity

Test number	Related requirement(s)	Description of test	Data used & type of data	Expected result	Actual result
1	35	Perform requests on the system,	10 requests within	System functions fine	Pass

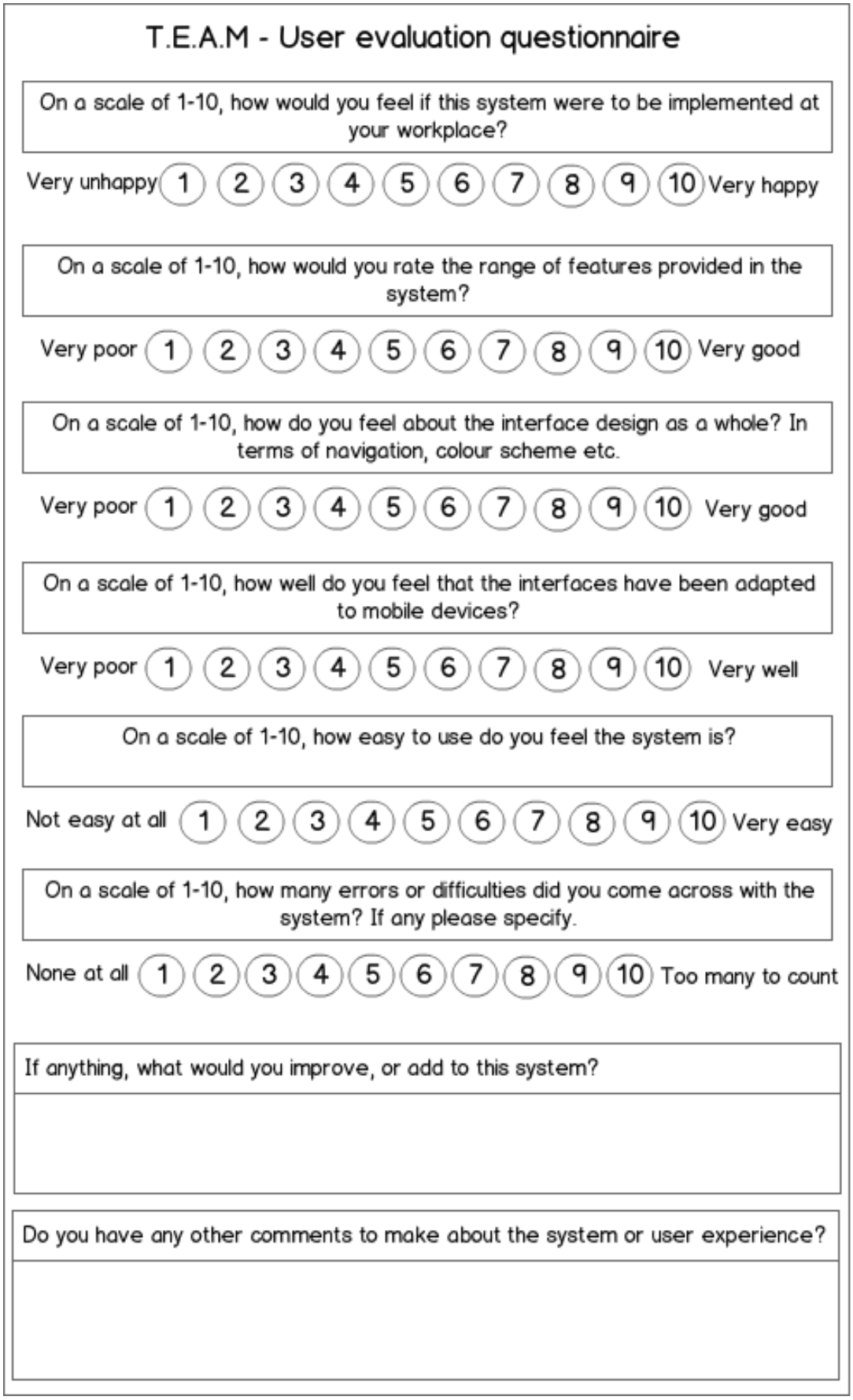
		such as request holiday, accept overtime etc	the hour – Typical data		
2	36	Perform requests on the system, such as request holiday, accept overtime etc	100+ requests per hour – Typical data	System functions fine	Pass
3	37	Test the speed the user is able to complete a request in	Log in, go to overtime page, fill in form, click advertise overtime	The user can compete these tasks within a minute.	Pass

Operating environment

Test number/related requirement	**Safari**	**Chrome**	**Firefox**	**IE9**	**iPhone (safari)**	**Android browser**
1	Pass	Pass	Pass	Pass	Pass	Pass
2	Pass	Pass	Pass	Pass	Pass	Pass
3	Pass	Pass	Pass	Pass	Pass	Pass
4	Pass	Pass	Pass	Pass	Pass	Pass
5	Pass	Pass	Pass	Pass	Pass	Pass
6	Pass	Pass	Pass	Pass	Pass	Pass
7	Pass	Pass	Pass	Pass	Pass	Pass
8	Pass	Pass	Pass	Pass	Pass	Pass
9	Pass	Pass	Pass	Pass	Pass	Pass
10	Pass	Pass	Pass	Pass	Pass	Pass
11	Pass	Pass	Pass	Pass	Pass	Pass
12	Pass	Pass	Pass	Pass	Pass	Pass
13	Pass	Pass	Pass	Pass	Pass	Pass
14	Pass	Pass	Pass	Pass	Pass	Pass
15	Pass	Pass	Pass	Pass	Pass	Pass
16	Pass	Pass	Pass	Pass	Pass	Pass
17	Pass	Pass	Pass	Pass	Pass	Pass
18	Pass	Pass	Pass	Pass	Pass	Pass
19	Pass	Pass	Pass	Pass	Pass	Pass
20	Pass	Pass	Pass	Pass	Pass	Pass
21	Pass	Pass	Pass	Pass	Pass	Pass
22	Pass	Pass	Pass	Pass	Pass	Pass

23	Pass	Pass	Pass	Pass	Pass	Pass
24	Pass	Pass	Pass	Pass	Pass	Pass
25	Pass	Pass	Pass	Pass	Pass	Pass
26	Pass	Pass	Pass	Pass	Pass	Pass
27	Pass	Pass	Pass	Pass	Pass	Pass
28	Pass	Pass	Pass	Pass	Pass	Pass
29	Pass	Pass	Pass	Pass	Pass	Pass
30	Pass	Pass	Pass	Pass	Pass	Pass
31	Pass	Pass	Pass	Pass	Pass	Pass

Appendix 3 - User Evaluation Questionnaire and Responses



T.E.A.M - User evaluation questionnaire

On a scale of 1-10, how would you feel if this system were to be implemented at your workplace?

Very unhappy 1 2 3 4 5 6 7 8 9 10 Very happy

On a scale of 1-10, how would you rate the range of features provided in the system?

Very poor 1 2 3 4 5 6 7 8 9 10 Very good

On a scale of 1-10, how do you feel about the interface design as a whole? In terms of navigation, colour scheme etc.

Very poor 1 2 3 4 5 6 7 8 9 10 Very good

On a scale of 1-10, how well do you feel that the interfaces have been adapted to mobile devices?

Very poor 1 2 3 4 5 6 7 8 9 10 Very well

On a scale of 1-10, how easy to use do you feel the system is?

Not easy at all 1 2 3 4 5 6 7 8 9 10 Very easy

On a scale of 1-10, how many errors or difficulties did you come across with the system? If any please specify.

None at all 1 2 3 4 5 6 7 8 9 10 Too many to count

If anything, what would you improve, or add to this system?

Do you have any other comments to make about the system or user experience?

Sainsbury's colleagues	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
User 1	7	7	9	10	9	1	-	-
User 2	9	8	9	10	10	1	Functionality for manager to perform same actions i.e. book holiday	-
User 3	8	6	10	9	10	1	Link with HR department – Hours worked, contracted shifts	-
User 4	9	7	8	10	10	1	-	-
User 5	10	6	9	10	10	1	From a managers point of view, link with HR to provide even more functionality	-
User 6	9	9	8	10	9	1	-	System is very easy to navigate and learn to use, however could include some pictures to liven it up
User 7	9	8	8	10	10	1	-	-
User 8	8	10	10	9	9	1	-	-
User 9	10	8	9	9	10	1	Ability to cancel holidays, after they have been approved	-
User 10	10	9	8	10	10	1	-	-

Colleagues at various other retail stores	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
User 1	7	7	9	9	8	1	-	-
User 2	7	7	9	9	9	1	-	-
User 3	8	8	9	10	8	1	-	-
User 4	8	7	10	8	9	1	Could make an app as well as mobile website	-
User 5	9	7	8	9	8	1	-	-
User 6	8	8	9	8	8	1	-	-
User 7	8	8	8	8	8	1	-	Don't like the orange colour
User 8	9	7	9	9	9	1	-	-
User 9	7	8	8	8	9	1	-	-
User 10	8	8	8	9	7	1	-	-