The Recreation and Evaluation of a Project Management System

BTech 451 Final Year Report

Dannii Brown
University of Auckland

1 Abstract

This report contains the progression throughout the final year BTech 451 project. The project involved the redesign of an existing system, which needed to be fitter for use and have greater usability, in relation to the company specifications. User experience is of great concern as this will lead to the new system being accepted by users and more widely incorporated into daily business activities across the company.

This report first outlines the project and provides background information about the company who supported this project. From there, a framework is presented for the development stages consistent with the agile approach used to manage this full stack project. Evaluations of the current system, existing project management systems and feedback from users of the current system, allow initial designs to be presented for the new system. As usability is a goal of this system, related work is reviewed regarding the topic of usability, to gain an in-depth understanding of this objective. Using this information, I make it my goal to create a new system for these users that is more user-friendly and fit for purpose, compared to the existing system.

Once I have understood these areas, the available technologies for the creation of a solution are compared to determine which one will best meet the requirements. After the evaluations, ASP.NET MVC is selected as the development framework and MS SQL Server is chosen to maintain the database. The database for this system is also redesigned, and an ERD is designed, and the database schema created.

Furthermore, the process of implementing the new system is provided. The requirements of the system have been grouped my priority, beginning with the core functionality. Progress through these functional requirements are documented.

Finally, the completion of the full stack project is evaluated, and achievements are assessed. Due to time constraints of the project, future work is suggested for the full deployability of the new system across the company.

2 Introduction

Opus International Consultants Ltd in an infrastructure consultancy company that employs over 3,000 employees in New Zealand, Australia, Canada, United States and the United Kingdom. A recent local project that Opus completed was providing building and architectural services for the redevelopment of Newmarket train station. Opus also specializes in seven other services including transport, water and energy.

Opus employees currently use an OpusCOMMS system that allows communications to be created for employees or contractors to complete. The communications are related to projects that the employees are assigned. The communication is assigned to a particular employee or contractor, and that person can update the assigned communication with activities. Activities are the individual tasks or progress updates that cause the progression of that communication towards completion. Once completed, the status of the communication can be changed to 'Closed'. When a status is created, the status is Open.

The current OpusCOMMS system has been implemented as two separate interfaces. These interfaces are specific for internal use within Opus and external use outside of Opus. The external site is used by contractors who are not Opus employees. Both these systems run on different servers and use different databases. The current external interface has a very basic interface and needs a new design. The current internal interface has many issues that has led to staff becoming unhappy with its use. The system does not work as it should and contains many bugs that have not been attended to.

The goal of my project is to design, create and implement a new OpusCOMMS system that can be used both internally and externally to Opus. This new OpusCOMMS system must be user-friendly and enhance user experience. A new database is also to be created for the new system, so that data can be contained in a centralized area.

3 Project Approach

I have decided to take an agile development approach to this project as, throughout the project, I will be given tasks that I am expected to complete in a timely manner. Once those tasks are complete, I will be moving on to the next tasks that need to be completed. As I am taking an agile approach, I may not necessarily work through these phases in order. Throughout the process, I gained feedback from my supervisor within Opus and applied this advice into the project.

The agile approach, shown in figure 1, will implement the traditional system development lifecycle and allow me to work through the following phases, focusing on the core tasks:

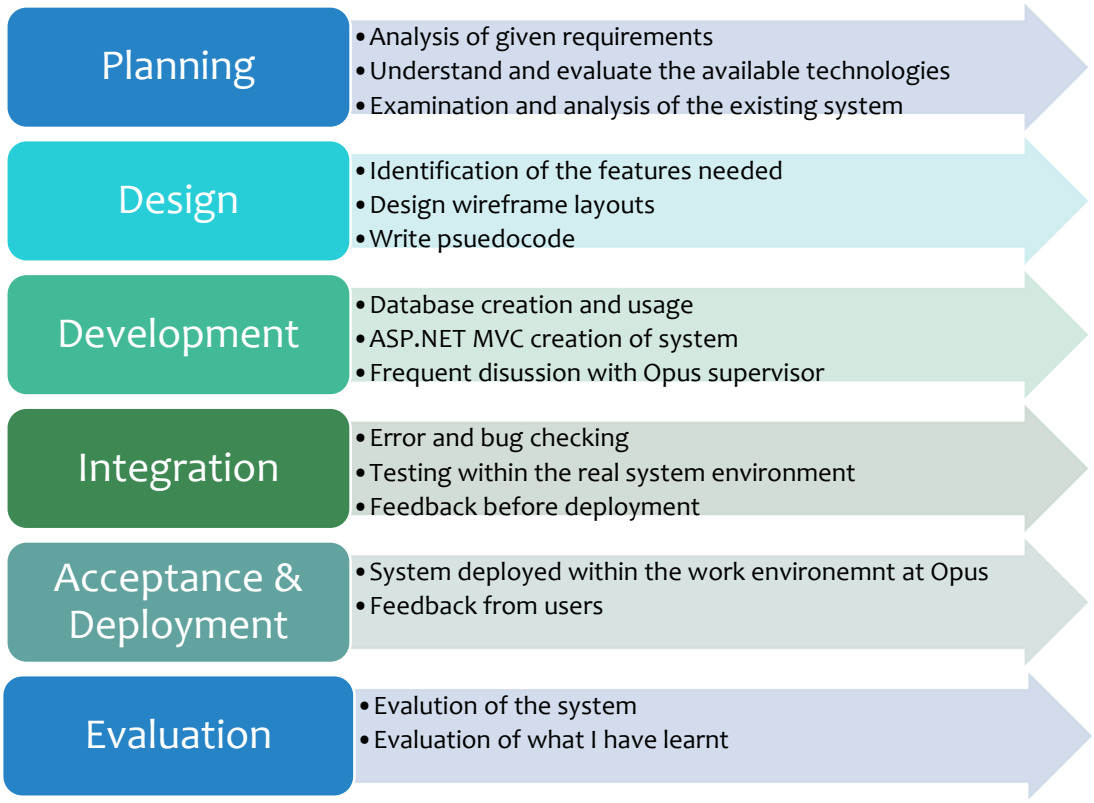

Figure 1. Agile approach that will be used for this project, and identification of the key tasks involved.

The requirements of the project were discussed at length during a meeting with my Opus supervisor. My initial assigned project did not have the necessary support and organization from Opus, therefore I was assigned a more appropriate project that will be known as OpusCOMMS. The tasks involved with the creation of OpusCOMMS were prioritized based

on their importance in relation to the functionality in the project. Some tasks are vital to the functionality of the project, such as the ability to add a communication. This is a task of great importance is creating and assigning communications as these are core functionalities and necessary for the use and deployment of OpusCOMMS.

In order to create a working OpusCOMMS site that could be accepted by users, these are some of the functional requirements:

- Ability to create a communication
- Ability to view all projects and communications
- Ability to add an action that relates to a communication
- Ability to create a contact
- Ability to log in
- Ability to see all your projects and communications
- User/Contact management

4 Related Work

The new OpusCOMMS site must have the level of user experience that is intended to capture user enjoyment and ease of use. For this to be attainable, I must explore this area and seek the relevant research publication available. Another area that will be explored is the creation of a user-friendly system, as this is also the goal of the new system. Finally, the interfaces of available sites that have be designed to cater for similar needs to OpusCOMMS will be evaluated. The areas of strengths and weaknesses of these sites will prove invaluable for the design of OpusCOMMS and the features that enables the users to use.

One of my goals for this project is to embrace the image that Opus expresses by incorporating this into my designs and implementation of the new OpusCOMMS site. I was not given any direction or restrictions in terms of the design of the site. However, it must gain approval from my supervisor within Opus and the employees who would potential use this system.

4.1 User Experience

A focus of this project is to create a new OpusCOMMS system that will be user-friendly and have a high level of user experience. Both user friendliness and user experience are related to usability (Latimer, 2014), which is defined as the amount of effort a user must put in to gain what they want. Often, the phrase 'user experience' is misused as many use it rather than usability (ebit.). User experience is the understanding of how users interact with a system, through their feelings, mentality and motivations. User friendliness is a solution that is often gained by a system that is easy to learn, understandable or easy to deal with. The term is usually used in regards to computer software or systems. Although user experience refers to the interaction a person has when using a product, often these two term influence each other.

Haberbusch (2014) suggests the six user experience strategies to make a site more user-friendly:

- Focus should not be purely on the homepage, other pages on the site must not be rejected and attention must be put into the experience of these pages also.
- Navigation must be readily understandable
- The F-Pattern, as shown in figure 2, should be considered. The F-pattern is a method that has arisen from various eye tracking research. It is the theory that users tend to

browse through sites in an F shape, focusing mainly on the headers and skimming through the body. With this in mind, bodies of content should be displayed in this F-shape where most relevant content is in the arms of the F.

- The site should be readable. A stimulating site can be achieved by avoiding long paragraphs and incorporating line breaks and headers within the body of the content.
- Forms should be implemented in a way that enables them to be easy to complete.
- Error handling should be taken into consideration. For example, 404 errors should direct the user to a page that is not frustrating and displays help information relevant to how they can resolve this issue.

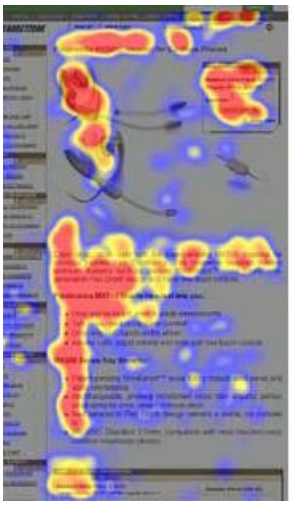

Figure 2. Haberbusch's user-experience strategy suggestion of the use of the F-pattern implemented on a webpage, showing the heat map gathered from an eye-tracking experiment. (Nielsen, 2006).

According to Latimer (2014), an essential component of user experience is knowing the users. Identifying is vital as it enables the system to cater to their needs and allow these users to complete their tasks and enrich their lives.

Jakob Nielson is very well respected in the field of usability, he suggested the ten heuristics of usability design, and these are very widely known within the human computer interaction industry. The ten heuristics of usability design are:

- Visibility of system status
- Match between system and the real world
- User control and freedom

- Consistency and standards
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help users to recognize, diagnose and recover from errors
- Help and documentation

A heuristic evaluation can be conducted by inspecting the interface in attempt to identify usability problems (Nielsen, 1995). However, Macefield (2014) strongly suggests that a heuristic evaluation is not sufficient enough to act as a usability study. A usability study often includes observing a user interact with the design of a system (Chisnell, 2010). The study reveals how the users interact with the system and their overall feelings and motivations as to why they interact with this system a particular way (Macefield, 2014). A usability test is a useful tool for developers to identify areas of improvement for their systems and better accommodate the functionality and experience for users (The University of Waikato, 2015).

The Webby Awards, established in 1996, has over 1000 judges that strive to award excellence in the web environments, during the Webby Awards 2015, five websites were awarded Best User Experience.

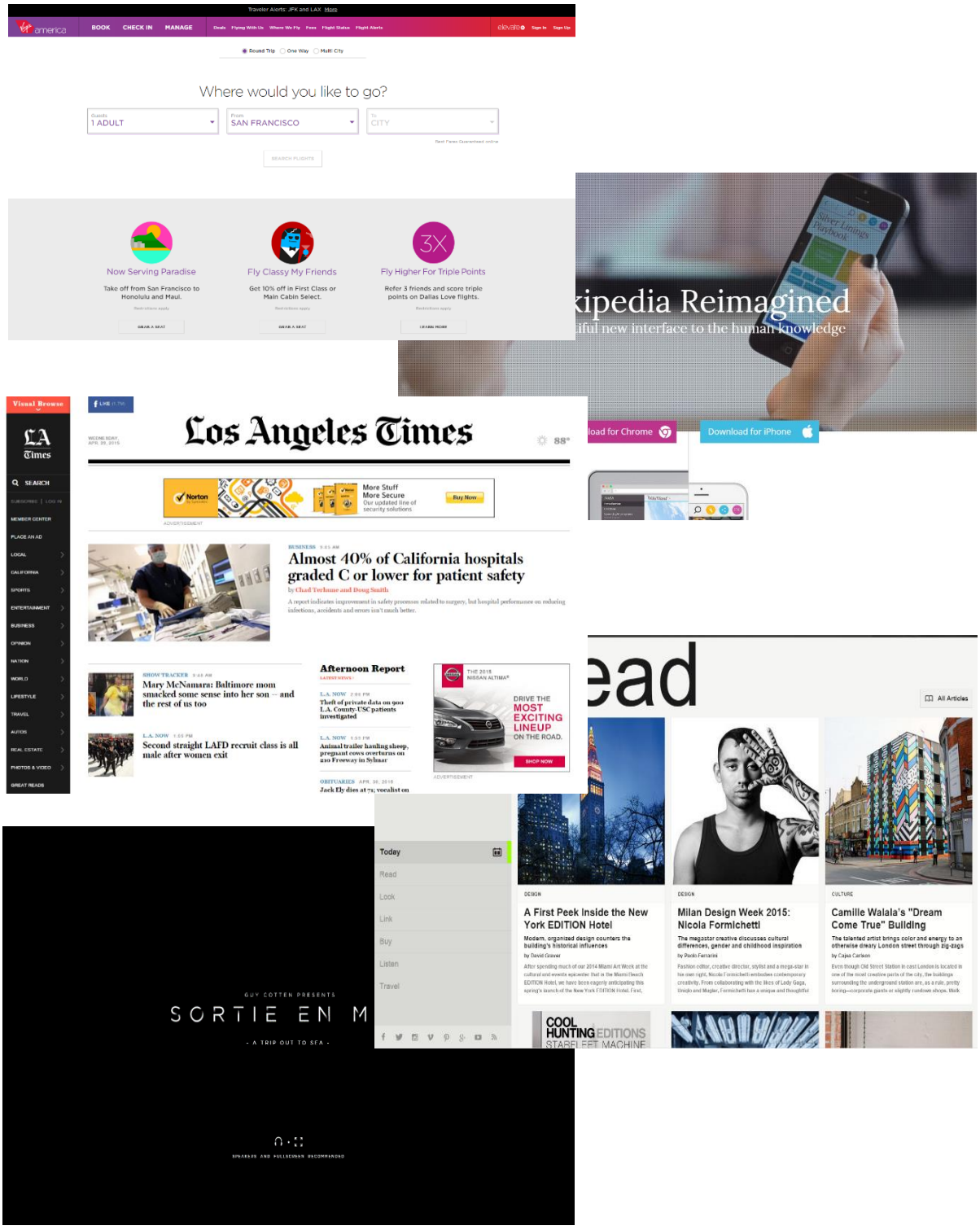

Figure 3. The five websites awarded Best User Experience from the Webby Awards. (Webby Awards, 2015).

After analyzing the style of these sites, I was able to determine these similarities and common themes:

- Clear navigation bars that have descriptive page titles. These navigation bars are fixed to the pages and consistent throughout all pages of the site
- These pages incorporate a fun feature, through bold color schemes
- The sites have different layouts, creating personality for that site
- Use of colored, rectangular panels or greyscale panels to separate areas
- Consistent styling throughout webpages and main page
- All the pages have been considered and have received the necessary attention
- All sites had functionality relevant to the website intention

In terms of usability, I noticed the following characteristics:

- Navigation was descriptive
- Themes were consistent throughout all pages of design
- When input was required, site restricted user input to reduce errors
- Implementation of dropdowns and auto-complete when inputs are required
- Designed for the purpose of the site
- Sites were aesthetically pleasing, and the designs are simplistic
- Ability to contact staff members if help is required

Overall, these websites have created their designs that make them unique and create a particular personality for that site. All these sites had great usability and incorporated an ease of learning. All navigated displayed intuitive descriptions of navigation.

4.2 Available job logging sites

After identifying the features of user experience and what makes a website user-friendly, the next stage would be to analyze websites that have been designed for managing project tasks. The goal of this is to identify the key features and functionality of these sites and evaluate them in relation to the OpusCOMMS project. Doing this will give an insight into the designs and functions that enhance the usability and adaptability of the OpusCOMMS site that I will be designing.

4.2.1 WorkFlowMax

The WorkFlowMax is job logging system that is available for businesses to use for a monthly fee. For the purpose of this project, I obtained a month free trial with the intention of research. This particular solution to a job logging system implemented the essential features for a job logging site. The purpose of this site is to maintain tasks that need to be assigned and completed and to keep track of the costs and hours involved. To enable these purposes, it implemented the features of the ability to add a job, enter a value or enter the amount of time spent working towards the completion of that job.

The initial page was a 'dashboard' page which displays all the user's jobs that they must work towards completing that week, as shown in figure 4. The page displayed them on a Monday – Sunday calendar for that week. The page collected data about the hours the user had worked that week, calculated from their progress on the jobs. It also displayed a list of today's jobs. Although the calendar feature may be useful, it was not a very interactive dashboard page. The page did not allow for any direct user interaction. The colour scheme and layout are very simplistic, and the system just contains the core functionality of a job logging site.

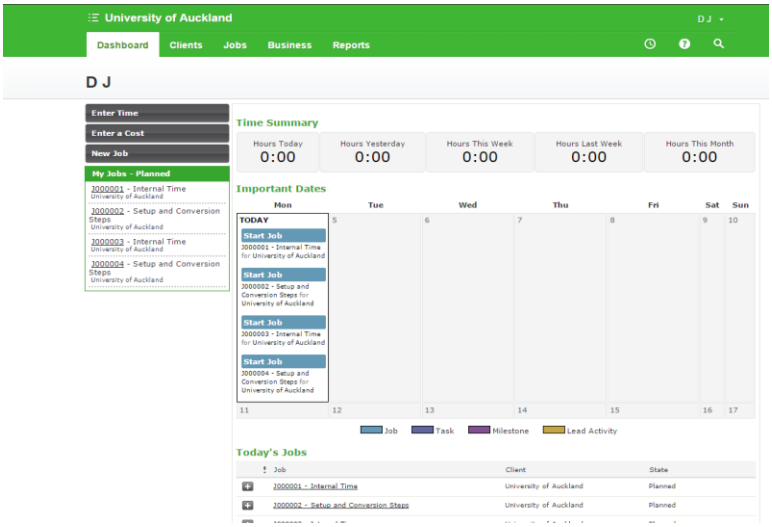

Figure 4. 'Dashboard' page of WorkFlowMax, a comparative job logging system. (WorkFlowMax, 2015).

It was simple to add a job, and not all the fields were required, allowing jobs to be quickly created or created by only disclosing the relevant or required fields. However, it was not very customizable and the functionality in terms of job logging and tasks associated with them were very simple. As this site should be focused on gaining and retaining businesses as their paying customers, the simplicity and the inability to customize was surprising. However, the form contains many drop down lists creating a more user-friendly input process and a reduction in errors.

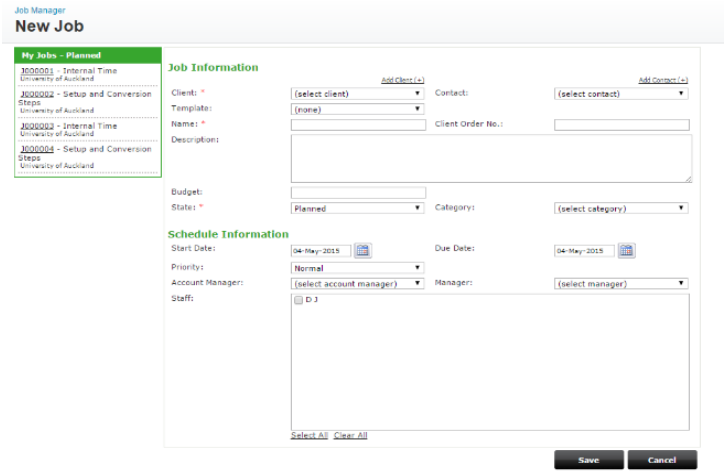

Figure 5. "New Job" page of WorkFlowMax, shows how a new job is created (WorkFlowMax, 2015)

Overall, it was very simplistic and easy to use, but the site lacked personality and styling. I also believe that this site would restrict certain businesses as it does not cater for customizability and the difference in the information that companies may need to record. Furthermore, this does not allow the system to be flexible and efficient in terms of business needs. Therefore, I do not evaluate this system to comply with the usability standards.

In terms of this project, the information displayed on the 'dashboard' page is different to what would be displayed on the new OpusCOMMS site. This difference is due to the OpusCOMMS site requiring more interactivity on the dashboard page than a calendar and lists. Displaying 'Today's Jobs' is a useful tool, and something that I may consider when creating the dashboard page for the project. The ability to add a job using a drop down lists was very efficient in error prevention. I also believe this would enable users to have a better experience when creating communications in relation to OpusCOMMS. However, OpusCOMMS will require greater flexibility due to their plans to extend its usage.

4.2.2 ProWorkFlow

ProWorkflow is a site that caters for project, task and time management. It is available to businesses to use on various pricing plans that vary based on the number of projects and file storage size. These pricing plans are charged per user. I obtained a free trial of this site for the purpose of this project.

The 'Dashboard', shown in figure 6, page of this job-logging site contained more visuals and interactivity when compared to the WorkFlowMax site. The colour scheme incorporates bright colours, and this made the system more enjoyable to use and increased the usability due to the aesthetics. The visuals on this page were relevant to current tasks, displaying the number of active tasks. It also had visuals related to the cost income of tasks and time spent that week. On this page, tasks were displayed in the form of a list, grouped by the date. This grouping was logical and also very relevant to what work the user needs to complete.

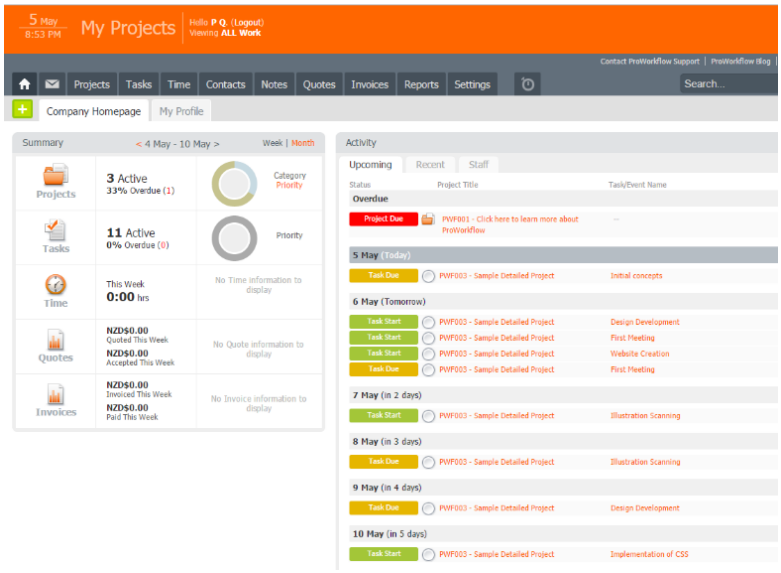

Figure 6. 'Dashboard' page of ProWorkflow (ProWorkflow, 2015)

This system allowed the creation of Tasks, which can be ticked when completed. The ability to "tick-off" tasks created a match between the system and the real world. This feature also required minimal input and work from the user. Both these points increase the usability of the system. Due to this functionality, and the simplicity of creating tasks, this site could be described as a To-Do List management site. It does not allow tasks or actions to be added to the tasks after they have been created. The system that Opus requires is more complicated than the functionality of ProWorkflow.

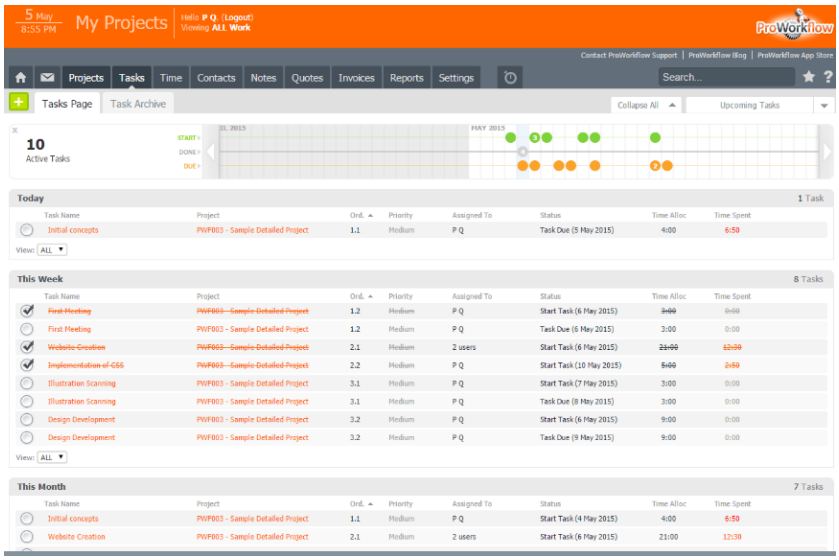

Figure 7. 'Tasks' page of ProWorkflow, shoring all the tasks and details of those tasks. (ProWorkflow, 2015)

To conclude, ProWorkflow has been designed well and incorporates usability. However, in terms of the OpusCOMMS project, this site did not fit the needs as OpusCOMMS. OpusCOMMS requires extra functionalities, such as adding Actions and changing the Status of the jobs. However, the ease of job completion was very simplistic, and this simplicity should be incorporated into the project. The dashboard page displayed the tasks that need to be completed is something that could be integrated into OpusCOMMS. Moving away from the table design that the current OpusCOMMS system implements would be desirable. The layout of the tasks in ProWorkflow is an idea that could be implemented into the design of the new OpusCOMMS site.

4.3 User Feedback

As well as looking at the functionality and layouts of successful job logging sites, I also was able to gather feedback from users of the current OpusCOMMS system. I obtained this information by creating a survey that asked current users about their usage of the site. The intention of these questions was to gain insights about the current system and identify any areas that the new system can enhance the user's experience and the usability of the site. The feedback will ensure that the new OpusCOMMS site will be an improvement to users of the current OpusCOMMS site. The conducted survey consisted of the following questions:

- What is your main purpose of using OpusCOMMS?

This purpose of this question was to gain an understanding of the goal of using OpusCOMMS in terms of the user's role within the company. Users may use the system for tasks frequently and understanding how it fits into their work activity can be beneficial.

- When creating communications, activities or people, did OpusCOMMS ask for the appropriate information?

One of the criteria for the inputting of the forms is to create flexible forms that cater for lots of different industries and different types of requests. This flexibility is essential as Opus provides services to many industry types that have different information that is necessary for collection when communications are created. The goal here was to determine if the data collection forms are currently catering to what is needed.

- If you could change one thing about OpusCOMMS, what would it be?

This question allowed some insights to be obtained about the usability of the current system. It also provides the user an opportunity to identify areas where the system is lacking in terms of user experience.

- Are there any additional features or tools that you would like to see on OpusCOMMS that would enable your use of it be more efficient or easier?

This open-ended question provides the opportunity for users to provide direction in terms of what would enhance their use of the system and essentially make their job easier.

- How would you describe your overall experience with OpusCOMMS?

This final question has the purpose of giving me the opportunity to gauge the feelings and user experience towards the current system. This level of user experience can be used as the basis for improvement.

4.3.1 Feedback

After obtaining the feedback from current users of the OpusCOMMS system, I correlated the collected answers and identified some interesting areas. These areas consisted of users either offering an interesting answer or where multiple users provided a very similar answer. From these points of interest, I am then able to use these findings to influence my designs.

Question	Feedback Points of Interest

What is the main purpose of using OpusCOMMS?	• To input COMMS forms
	  ○ Accident that happen on state highways
	  ○ All enquiries from the public
	• Covering the Professional Services contract with the NZ Transport Agency, "maintain a communications database with all the incoming correspondence."
	  ○ Provided to client as a report monthly
	• Track communications and actions required by external contractors
	• To generate a report at the end of the month to product to clients
	• Inputting stakeholder complaints

When creating communications, activities or people, did OpusCOMMS ask for the appropriate information?	• Only IT department can create a Person • Lack of flexibility "it does what it does." • Arrangements for adding contacts was very poor, "learnt to put the absolute minimum detail" to avoid system freezing

If you could change one thing about OpusCOMMS, what would it be?	• To be more user-friendly
	  ○ Have to "Save and Close" when you want to save
	  ○ Have to write person responsible many times

	• Summary Reports do not print the full title of the communication if the title is too long.
	• Process of changing communication status to Close is not intuitive
	  ○ Have to "Change Status", click "Close", then "Confirm", then "Confirm" then enter Date and Notes of the action taken.
	  ○ Complicated processes where a "Save" button would be nicer
	• Search feature is hard to use as it does not always find things with the keywords known to the user
	• Easier access - provide easy start-up and entry of initial details
	• "The whole system as it does not meet our needs."

Are there any additional features or tools that you would like to see on OpusCOMMS that would enable your use of it to be more efficient or easier?	• An App for the employees who are based on the road so they can type COMMS as currently they send it through to the office for someone else to type up • When an Action is added, an email is sent to the assigned employee, would be nice if the reference and description were included in this email. • More comprehensive reporting filters • Ability to record all files against tasks – currently just PDFs can be added • Ability to edit a communication if something was saved incorrectly.
How would you describe your overall experience with OpusCOMMS?	• "It needs some work." • Good as it increases systemization • "It's not user-friendly" • "Frustrating" • Would not "recommend its use." • "Not much better than a glorified spreadsheet."

After gathering the feedback from real users of the current OpusCOMMS system, I felt that it is important to take into account this feedback when designing and implementing the new OpusCOMMS system. Some important things that I will take into consideration when designing my system are:

- The enabling of employees to benefit from a flexible form when inputting data to create communication or add activities. This will enable the system to meet better the needs of the users.
- The creation of a better process for adding a Person, enabling an 'Add Person' during the phase of adding a communication. Thus, allowing a user-friendly process that completes the two tasks in an easier way.
- Incorporating user-friendliness into the site.
- Enabling a simplistic, intuitive process, eliminating unnecessary processes and allow staff to update the Status of jobs easily.
- Enabling staff to 'Add A Communication' from their homepage and ideally implement auto-complete functionality and helpers.
- Displaying greater details in emails sent to staff.
- Allowing a Search feature with a keyword search on words that the user would relate to the comm.
- Implementation of greater sorting functionality for reports.

Due to the abundance of features desired by the employees, the suggestions will be taken into account when the related task implementations are being considered. Time constraints of the project may affect the ability to implement all of these requirements, and some may be left for future work on the system.

5 Design and Theme

As an agile development approach has been adopted for the approach of the project, the planning phase of the site will be conducted in parallel with the development phase. The reason for this approach is to be able to obtain regular feedback from my academic supervisor and reconsider the plans accordingly based on that feedback. It also allows for the simultaneous work on different tasks that require implementation in terms of the project.

When creating my designs, I felt it was important to incorporate Opus's brand image to create consistency and familiarity for users. It is also important for brand awareness and creating a strong reputation, as this site will be exposed to external contractors. Creating a positive brand image promotes the Opus brand to these external users. The Opus website has a strong sense of a brand image, as shown in figure 8. This brand is built through the bold capitalization and the strong colour themes. Red is an intense, aggressive tone and produces a more potent image for Opus. The website also incorporates grey into their colour scheme. The background is not white. Instead, it uses a striped background that includes grey into the background.

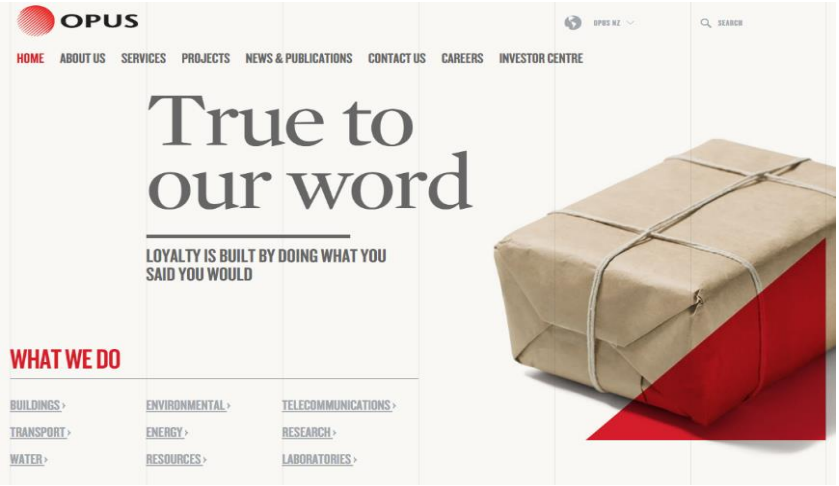

Figure 8. Opus online homepage. (Opus, 2015)

Opus's external OpusCOMMS system is used by outside contractors to manage projects with Opus, review communications sent to them by Opus employees and log actions against those communications. Looking at this site, it does not have a strong connection to the image created by the Opus website, nor does it implement any of the designs that represent Opus.

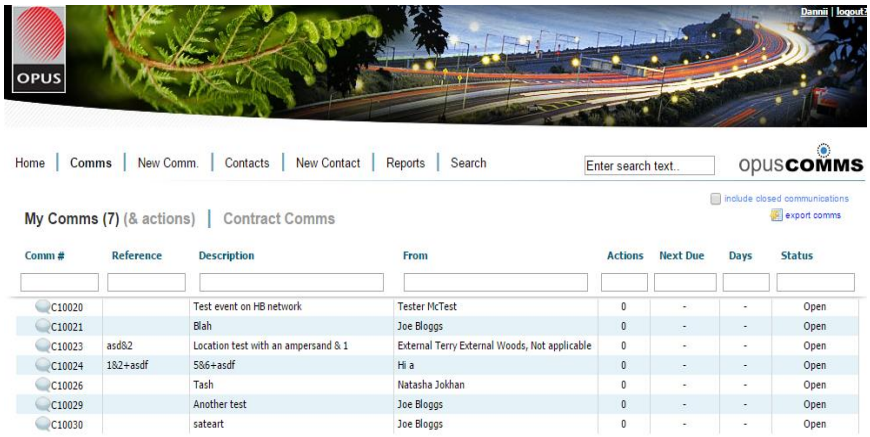

Figure 9. Existing OpusCOMMS site, displaying the homepage. This page displays all the communications that have been assigned to the user (OpusCOMMS, 2015)

The external OpusCOMMS site does not incorporate this theme, and I feel it lacks strong brand recognition for Opus, especially when outside companies use this. The external site has received negative feedback from outside contractors as it is not intuitive to use and is not user-friendly. There is also some programming bugs that exist on this external site.

As Opus has a very strong connection with the colour red, I choose this as my base colour and then picked other colours which complemented the base colour. I wanted to capture the robust design that the Opus website captures. I must also capture the user's attention and incorporate bright colours, creating a fun user experience. I used Opus's current systems as a basis for my colour schemes.

My final colour scheme implements the grey and red that is consistent throughout Opus's site and also incorporates another colour which will aid the positivity of the new OpusCOMMS site.

Figure 10, proposed colour scheme for the new OpusCOMMS site.

5.1 Mock-up and implementation of design

Shown below is my mockup of the layout I wanted to implement into the new OpusCOMMS site.

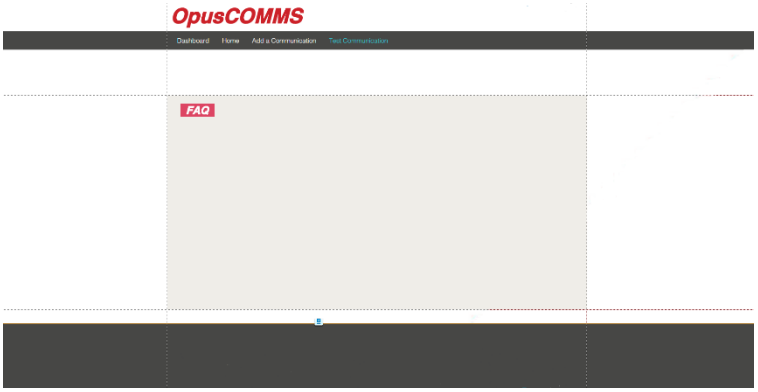

Figure 11, proposed mockup of layout, including the desired colour scheme

To begin with, the first implementation of the Home page mimicked that of the current OpusCOMMS system. The reason for this was to ensure that the layout was appropriate in terms of the styling, it is also important that the new system is not at a lower value of user experience. The first implementation will be reconsidered during the implementation phase.

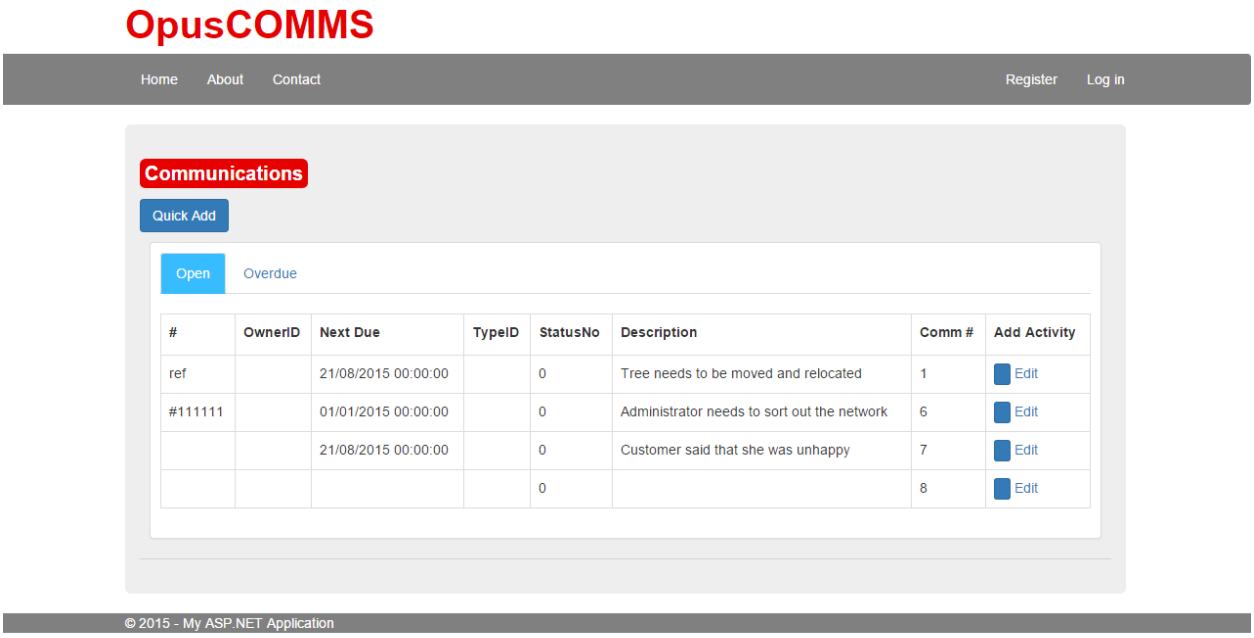

Figure 12, first implementation of the Home page.

6 Implementation Decisions

For the implementation of this project, I continued to use an agile approach. For this stage of the process, this meant that I discussed the decisions supervisor within Opus before implementation. This process enabled me to get feedback and adapt things based on feedback.

6.1 Technological Evaluations

Before advancement in the implementation of the project, it is important to compare the technologies that are available to use. Identifying the advantages, disadvantages and uses will enable a decision to be made upon which technology is most useful for this particular project. One of the choices that I was presented with was whether I wanted to use the existing system and update the existing code, or to create a new OpusCOMMS system. The existing system is written in PHP, so this would mean learning PHP and understanding the code that is currently in place. Being able to manipulate the current site may prove difficult as I would have to have a deeper understanding of how the system has been implemented and written. According to the Opus IT team, the site has been poorly implemented, and not maintained over time. The current system contains a lot of bugs that have not been fixed, and many features do not perform as expected. Creating a system from scratch presents the opportunity of choosing from what technologies are available, and being able to complete a full stack project.

To make this decision, I evaluated the use of PHP and ASP.NET MVC, as these were both acceptable solutions available for my use. As the existing internal and external site use different databases, a new database will be designed and created for the utilization of the new OpusCOMMS system. Therefore, I also evaluated the different relational database management systems to be able to compare their uses and suggest an appropriate system.

6.1.1 PHP VS ASP.NET MVC

Hypertext Preprocessor, better known as PHP, is a scripting language designed for the purpose of web development. It is a server-side scripting language, meaning that after the code is executed, the web server sends the result to the client. Code can be written from any text editor. Whereas, ASP.NET MVC is a powerful tool that allows the developer to build web applications based on the .NET Framework (Paz, 2013). It allows for a lot of control

over the creation of the application. However, due to the ASP.NET framework, this leads to PHP being more flexible than ASP.NET MVC (Ranjan et al. 2012), so the purpose of the use must be considered. In terms of this particular project, the configuration flexibility is not something that must be considered as the application could be created under the MVC model. Both the security of PHP and ASP.NET are considered at least good (ebit.), although this is not very informative, it does lead to the assurance that no security issues need to influence the decision.

According to Erack Network (2008), ASP.NET is preferred within businesses that operate using Microsoft as the projects have been optimized to run on Windows operating systems. Opus uses Microsoft within their business operations. With this in mind, it may be more compatible to use Microsoft based software such as Visual Studio. Using Visual Studio will also enable easy pushes to Microsoft Azure, Microsoft's cloud computing software. Microsoft Azure can be used to host the web services as a Web App, and this hosting is a desirable goal of Opus when my project is complete, to aid the easy transition from my local development to their work environment for future work.

MVC stands for Model-View-Controller, it is a framework that separates the model, view and controller. The model manages the business layer of the application, and it accounts for the logic of the data. The view handles the display or the output of the data. Finally, the controller achieves the user interaction and can read the data from the view and send the data to the model. This setup can enable ease of creation for complex applications as the separation allows the task to be broken into segments. The advantage of the MVC framework is a separation of concerns due to the separation of the data model, html views and logic controllers; this leads to greater maintainability (Paz, 2013).

As PHP is a scripting language, and ASP.NET MVC is a tool, for this evaluation to be fair, I must also consider the languages involved, rather than comparing a language with a tool. As PHP is a scripting language and ASP.NET MVC is a tool, for this evaluation to be fair, I must also consider the languages involved, rather than comparing a language with a tool.

6.1.1.1 PHP VS C#

Although Visual Studio has the option to create a web application in different languages, including Visual Basic, Python, J# and C#, due to my experience levels, I would be most inclined to program in C#. Therefore, I will be comparing the use of C# and PHP.

C# is an object-oriented programming language that was developed to combine the benefits of C++ and Visual Basic. Hypertext Preprocessor, better known as PHP, is used for web based programming, and unlike C#, it is not object-oriented. C# can be used for a much wider range of projects and not limited to web development.

In terms of performance of the languages, the intention of C# was to sacrifice some of the performance to create a language that was easier for developers to create quick projects. Although, PHP does not allow any threading, whereas C# does support threading that could be very advantageous depending on the project. C# also has many optimizations built into the language and can be written to outperform PHP (ebit.). Therefore, in terms of performance, it is project dependent and also the use of latest hardware components means that the noticeability in the differences in performance time leans towards zero.

Overall, there are benefits to using both PHP and C# for this project. In terms of the evaluation, I like how ASP.NET MVC framework separates the components into logical layers. I also believe that using a Microsoft tool will suit Opus and be compatible with their other applications. Therefore, I have decided to use ASP.NET MVC for this project.

6.1.2 Relational Database Management Systems

A database is an organized collection of data which has been logically collected in tables. Database management systems (DBMS) are applications that enable the management of databases. A Relational Database Management System (RDBMS) can manage databases that are made up of tables, columns or attributes that hold data types. Each table contains a primary key, which is a unique field within each record, and each row represents a record when all related together this is known as the relational model. AS SQLite, MySQL and PostgreSQL are all popular RDBMS, I will analyze their features and assess their usage in relation to my project.

6.1.2.1 *SQLite*

SQLite uses SQL and offers tools to handle data with ease when compared to server-based relational databases. The reason for this comparison is because SQLite has the advantage of being a file-based database. Therefore, the entire database is a single file on a disk giving it the advantage of its ability to transport. It directly calls the files that contain the data rather than using an interface such as a socket to communicate. Thus, enabling SQLite to execute its libraries quickly and efficiently. SQLite offers the least amount of datatypes (Tezer, 2015),

and this could be seen as an advantage or disadvantage depending on the use of the database.

The main situations that would make use of SQLite are:

- Applications that do not require to be scalable such as mobile applications
- Applications that need to read/write files directly to disk, however, SQLite only allows one write operation at any one time. (SQLite, 2015)

6.1.2.2 *MySQL*

MySQL has a lot of features and is an open-source product. It is used in a lot of website and online applications. Due to its popularity, there are a lot of resources and information about MySQL online, which means that help is readily available when required. This also means that a lot of third-party applications and tools are compatible with MySQL. The advantages that MySQL has is that it allows for a lot of SQL functionality, which means that compared to SQLite, there is more functionality available. However, some features have been given up to improve the speed of the RDBMS, such as rendering (Tezer, 2015). MySQL has security features that have been built in which implements protection for data access and use, this is an enormous advantage as users do not need to implement further security features when creating their projects. As it has been designed for larger databases, it can handle a lot of data and can handle scaling if needed.

The main situations that would make use of MySQL are:

- When more advanced features are required than what SQLite can offer
- Application that requires security
- Applications that may require flexibility and scalability such as websites and web applications

6.1.2.3 *PostgreSQL*

PostgreSQL has the intention of being compliant with standards and extensible. It supports object-oriented and relational database functionality, which means it completes support for reliable transactions, these are further enhanced by the use of Multiversion Concurrency Control which allows concurrency of tasks (ebit., 2015). However, read-heavy operations make PostgreSQL appear less efficient than MySQL. It also allows the creation of stored procedures, and these allow common tasks to be stored and executed when required.

PostgreSQL also offers the greatest range of data types, which may benefit particular projects and helps to ensure data integrity when inputting data into the database.

The main situations that would make use of PostgreSQL are:

- Applications that require reliability data
- Applications that require stored procedures
- Databases that are of a complex design

6.1.2.4 Evaluation

After evaluating the uses and advantages of SQLite, MySQL and PostgreSQL, it is important to assess the three RDBMS in terms of what would be best suited for this project.

The criteria of this project involve:

- a complex database that is stored on a server
- flexibility and scalability
- moderate level of security
- the reliability of data
- Reading and writing operations from the website to the database

After evaluating these criteria, the most viable options would be either MySQL or PostgreSQL. As they both have advantages over each other, for this project, I will be using PostgreSQL as Opus currently has use of pgAdmin3.

6.1.2.5 DBMS Usage

After learning to use PostgreSQL within PgAdmin3, I was given the option of whether I would prefer to use Microsoft SQL Server for the database management tool, which using an implementation of SQL known as Transact-SQL. After installing the MS SQL Server and using some of the tools involved, I found that ASP.NET MVC had very nice integration tools for MS SQL Server. These tools allowed more flexibility in how the databases could be added and used within projects. Although I have not used T-SQL before, I found that there is an online community that is willing to help and offer advice to fellow users who are unfamiliar with the language differences. Therefore, I have decided to use MS SQL Server for this project rather than PgAdmin3, and I feel it adds more value to this particular project and better integrates with my project technologies.

7 Implementation and Interactive Design

7.1 Database

As the current OpusCOMMS system has a separate database for internal and external communications, the database is being redesigned for the new OpusCOMMS system, as it requires just one database. Currently, the two databases store different information and have different attributes. With this in mind, it was important to look at the data being stored and collected by the internal use and external use of the system. The ability to design a new database also allows the freedom of selecting the data types and the data fields that require collection.

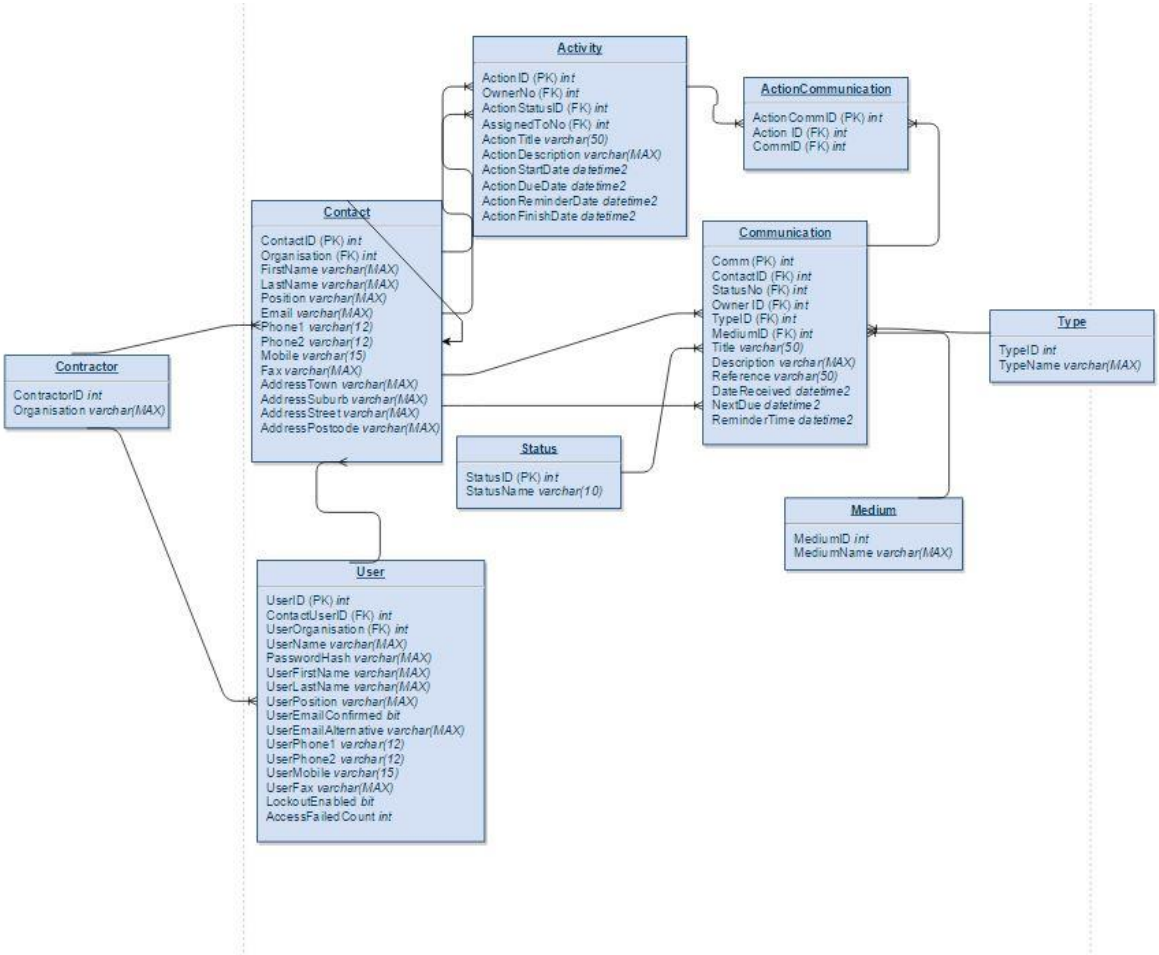

Figure 13, ERD that represents the database that was created using MS SQL Server to connect to the OpusCOMMS project.

7.1.1 Database Logic

The starting point for the creation of this database was the Communication table. It is important that sufficient data is collected to meet the user requirements. From here, some pre-defined options were identified. These options include the type of communication, the medium associated with that communication and the communication status. Creating these as their table allows for the easy management of these data lists. Easy management includes the ability to make the options assessable at the system side as well as the more natural addition of items to that list when compared with code injections.

The Action to Communication was identified as a many-to-many relationship. Although uncommon, there may be an event where one action has an effect on multiple communications. Also, a communication can have multiple actions or steps involved in the completion of that task. This implementation also secures the expansion of the system as it may be under greater demands in the future.

The contractor-contact inheritance relationship is to solve the cohesion of the Opus employees and the external contractors. An employee of Opus does not need to have an organization or industry associated with them. Whereas, an external contractor would be identifiable with the company name that they are employed within. The contractor will also be associated with the industry in which they provide services.

Another issue that arose for the creation of an ERD was the difference between a Contact and a User of the system. A Contact can create and be assigned communications, whereas a User can log into the system. There can be instances where an employee is both a User and a Contact, and instances where an employee is a member of just one of these groups. To account for these situations in the database, both a User and Contact table were created. The User table also contains a ContactUserID that associates to the ContactID, if a User also has a Contact profile. There is logic associated with this situation within the project.

Currently, I have implemented this database onto my local database server so that I can use this database for testing purposes. Opus are pleased with this ERD and intended to send a spreadsheet of the real data to be imported into the created database, to be able to test the application with real data. However, this was never received due to time constraints and other commitments of my supervisor within Opus.

7.2 Dashboard page

One of the pages that will be implemented in the new OpusCOMMS site is a dashboard page. A dashboard page is one which should be interactive and entertaining for the user to view when they first login. The goal is to enhance user experience and interactivity through the use of this page. The suggestions for this page are the number of weekly tasks closed and overdue by particular person, team or project. For this dashboard page, I looked at open-source solutions that would cater to the needs of the project.

I believe the dashboard page, shown in figure 14, will fit the criteria of this project, as it has a interactive and statistic based layout. I implemented the design into my ASP.NET MVC project and made some changes in terms of the layout to be consistent throughout the project.

Figure 14. Screenshot of the default implementation of the dashboard page.

However, as the dashboard page is not one of my functional priorities, I will not be further enhancing the data for the page, this is due to it not being in the scope of tasks for this project. The final solution will include the imported CSS styles and default layout to be manipulated as future work for this project within Opus.

7.3 My Communications

When I was thinking about designs for the user's Homepage, I wanted to create something different to the designs Opus is currently using. For the existing OpusCOMMS site, a spreadsheet layout is used, with details displayed under the appropriate columns.

Comm #	Reference	Description	From	Actions	Next Due	Days	Status
C10020		Test event on HB network	Tester McTest	0	-	-	Open
C10021		Blah	Joe Bloggs	0	-	-	Open
C10023	asd&2	Location test with an ampersand & 1	External Terry External Woods, Not applicable	0	-	-	Open
C10024	1&2+asdf	5&6+asdf	Hi a	0	-	-	Open
C10026		Tash	Natasha Jokhan	0	-	-	Open
C10029		Another test	Joe Bloggs	0	-	-	Open
C10030		sateart	Joe Bloggs	0	-	-	Open
C10033	email #3	Too noisy	a a	2	2015-05-14	-12	Overdue

Figure 15, current OpusCOMMS display of communications (OpusCOMMS, 2015)

The feedback obtained from the user survey revealed that some respondents were unhappy with this form of design. It is challenging to read and becomes hectic when staff have been assigned a vast quantity of communications. To accommodate these users, and to create a better user experience, I created a layout that displays all of this information, incorporating progressive disclosure for those particular communications of interest to the user.

Progressive disclosure is a technique of grouping less used features on a secondary screen (Nielson, 2006). Incorporating this feature adds to usability experience (Spillers, 2004). Jakob Nielson (2006) believed that using progressive disclosure exposes your system to one of the best interactive design methods. Using progressive disclosure when displaying the information of the communication assigned to staff has the advantage of limiting what is shown on the screen and also allows staff to focus on only communications on communications of interest.

I grouped the data by the 'Next Due' date, which corresponds to when the next activity is due, or the competition tasks are required, in ascending order. Tasks with no date are displayed on the "Open Comms" tab, which displayed communications which have not been assigned a date.

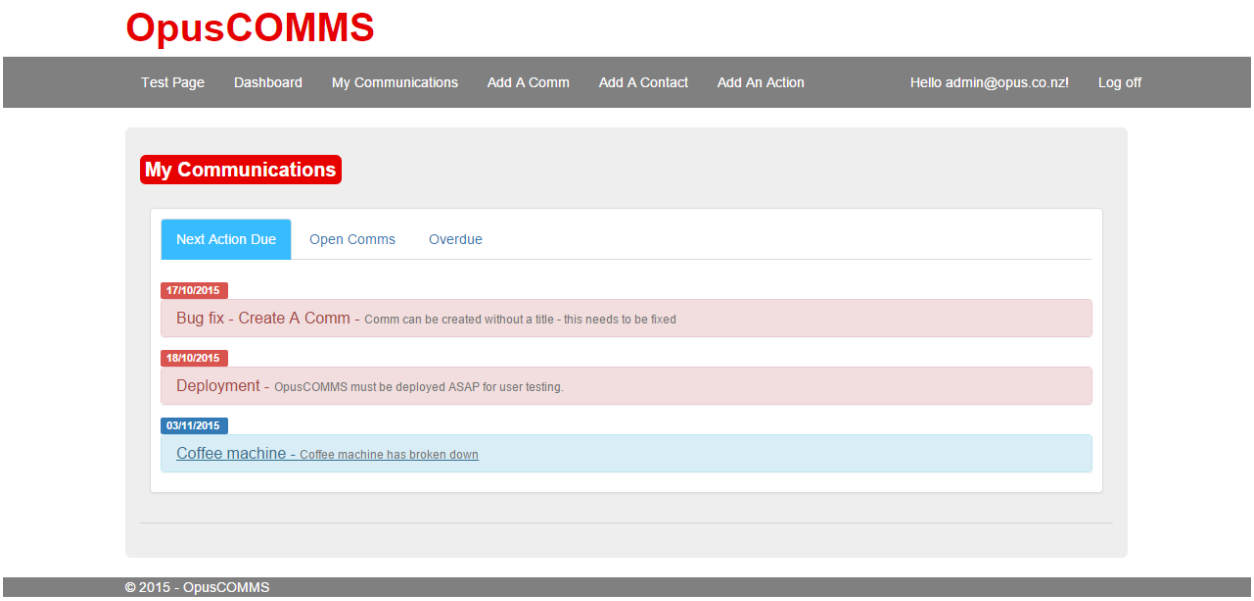

Figure 16, Homepage of the new OpusCOMMS

The colored banners refer to whether the communication is overdue or current. The red banners represent overdue items that need to be updated, and the blue show the current communications. The communications are grouped by date and this is shown at the top of the banners within that section.

To obtain this grouped data, I used the following query:

```
var query = (from item in Model.Comms
                    orderby item.NextDue ascending
                    select item).Where(item => item.StatusNo !=
                        1).GroupBy(group => group.NextDue);
```

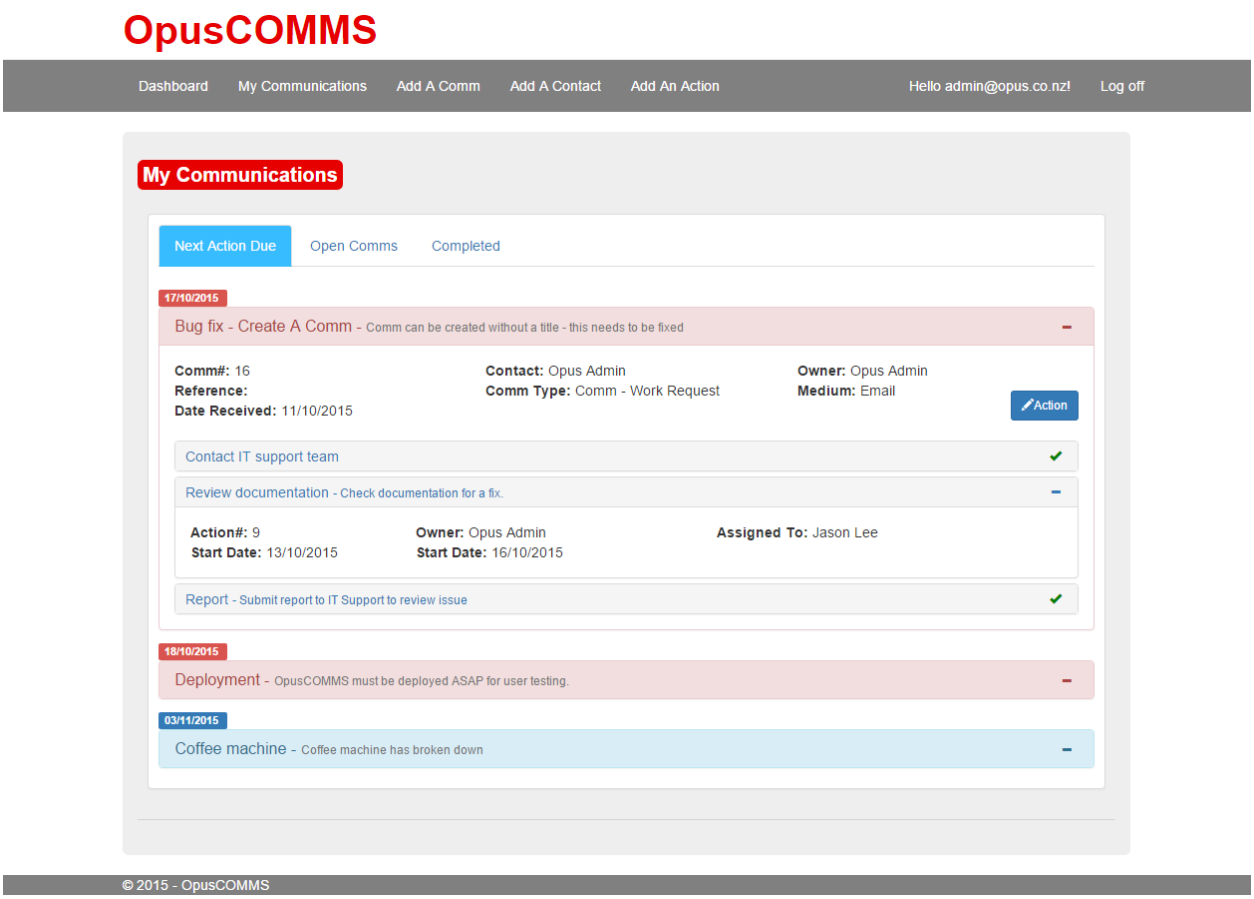

Figure 17, details have been expanded on the OpusCOMMS homepage.

Figure 17 shows how the homepage implements progressive disclosure. When a banner is clicked, the details of that communications are shown underneath. This allows for only the interested communications to be viewed. Under the communications, if actions are associated to that communication, they are shown underneath. These action also implement progressive disclosure and can be expanded on click.

7.3.1 Communication evaluation

For this page to be accepted by the users of the system, I evaluated my progress in relation to Haberbusch's six user experience approaches that I outlined in section 4.1. The communication will not be the homepage of OpusCOMMS. The homepage will be the dashboard page. For this iteration of the project, I have been focusing on pages other than this homepage and focusing on user experience broadly across all pages, as required by Haberbusch. The navigation that I have implemented so far attempts to be descriptive and

concise, as shown on the top navigation bar in figure 17. Therefore, I have met Haberbusch's first two criteria's of user experience.

The "My Communications" page does not display a substantial body of text. Instead, the text displayed is easily readable and presented in a logical order. Compared to the original table layout, the new layout has a much greater user experience. Although I did meet his criterion of Haberbusch, I did not have the opportunity to implement the F-pattern into this page as it does not display a large body of text.

In terms of error handling, I have taken precautions to avoid null pointer exceptions. Error handling is an important aspect of encourage user experience, and there is sufficient error handling implemented on this page. The page queries the User that is logged in, and displays their assigned communications, checking the UserContactID. If the User does not have any associated communications, then the page will not display any communications. If the User does not have an associated UserContactID as this field is null, the Controller checks the User details against those in the Contact table. This check is done by comparing the email addresses as this data would be consistent in both as Opus employees use their company email. If an email address is found in the Contact table that corresponds to that in the User table, then this page will display the associated communications, and save the UserContactID in the User table, for future references.

7.4 'Add A Communication'

The 'Add A Communication' page is a high priority task as it is one of the core functionalities of the site. The purpose of this page is to enable the user to create a communication, which is assigned to an employee, who will complete that task. The primary data input this page must include is:

- Title
- Contact
- (Owner)
- Type
- Medium
- Reference
- Description
- Remind me on

- Date Received

These data input values have been taken from the current internal and external sites. Previously, these sites asked for different input data as they used different databases. Due to the complexity of the database, these data values are stored in multiple tables. Therefore the saving of the data input is more complicated than would be with a single table.

To minimize input data errors and to increase user experience, it must be considered how users can input communications most efficiently, with the least amount of manual typing or input processing. With this in mind, it is important to consider the use of auto-complete or drop down boxes to allow the users to select possible input values rather than typing them.

To begin with, I created a simple form that only entered the data into one table, which was the Communication table. This stored the value of Title, Reference and Description without any further processing or table mapping needed. These values will also not require any auto-complete or drop down boxes as the user is free to enter text in there that they believe appropriate.

Extending this functionality, the implementation of the database requires the following data to be mapped and saved in these databases in the correct format. Each communication will have the following data stored within the Communication table:

Input Field	Format of user input	Lookup in table	Field saved in communication
Comm# (uniquely generated)			Comm#
Title	Text		Title
Contact	Contact option	Contact	Contact ID (FK)
Type	Type option	Type	Type ID (FK)
Medium	Medium option	Medium	Medium ID (FK)
Reference	Text		Reference
Description	Text		Description
Remind me on	Date/Time		ReminderTime
Date received	Date/Time		DateRecieved
Owner		Contact	OwnerID (FK)

The primary key, Comm# was created by an increment of the last inserted record. In the future, this model could be improved by recycling previously used primary keys where the associated records have been deleted. However, it was undecided whether a user should have the ability to delete a communication that has been created or if their only solution is to update its status to close. This is something that will may be considered in the future, as the exsisting OpusCOMMS system does not allow for the deletion of communications, so primary keys would not need to be recycled.

The Title, Reference and Description fields are able to be saved into the Communication table as the user has entered them into the 'Add a Communication' form. These input values are stored as NVARCHAR(MAX) to not restrict the user on a particular input size. I have also ensured they do not have an upper limit as my judgements of what the user may enter in the textbox are my assumptions. These assumptions may differ from the actual use, and this should not be a limitation of the system.

Using NVARCHAR as the data type was more desirable when compared to NTEXT. The reasons for this are that NVARCHAR is, arguably, easier to work with as there are more functions that can be used when manipulating the data of this type. An example of a function that could be relevant is the LEN() function. This function can be used on NVARCHAR data type, but it cannot be used directly on a data type of NTEXT. Another reason is NTEXT is being depreciated within MS SQL Server, so using this data type would result in the database being incompatible with an upgrade when it is fully depreciated.

For the Contact field, the user is allowed to select another staff member's name and the site needs to be able to map this string FullName to the unique ID to be stored in the ContactID field of the Communication table. This string was created in the ViewModel, and enabled the conncatinated FirstName and LastName to display for the user in the dropdownlist.

```
public string NameFull { get { return Contact.FirstName + " " +
Contact..LastName; } }
```

Acquiring the Owner of the creating communication, by default, this is the user who is logged into the system. The logged in user is queried in the Controller, and this data is not part of the user input form. The UserID is added to the communication as the Owner of the communication.

For both the Type and Medium fields, the user is able to select from a pre-defined set of options. For these fields, I used the pre-defined options from the current OpusCOMMS site. These options are stored in the Type and Medium tables within the database. Storing these options within their own tables enables the easy access and displaying of the pre-defined options to the user. When the user selects their option from the displayed drop-down box, this selection is mapped back to the associated primary key. This unique value is stored in the TypeID and MediumID fields within the Communication table.

The 'Remind me on' and 'Date Received' are both displayed are both saved to the Communication table in the datetime2 format. By default, the date received will be the date of the communication made. However, there may be a case where the user was out of the office at the time and is creating the communication after the period the task was created outside of the system. The 'Remind me on' is a date of the user's choice that they wish to be sent an email to update them about the status of the job on that date. Both these options will involve a date chooser option to enable ease of use for the user. There are many different possible inputs that could be used for inserting a date and users may be unfamiliar with which input will be accepted by the system. This issue of confusion is eliminated with a date picker.

7.4.1 Future Work

Extension of this data input includes the ability for users to create their own data fields and add values to them, this is to make the input more flexible. This could be a potential goal for future work on the new opusCOMMS site. The added flexibility could be used for a wider range of communications and projects than the current system is used for. Due to the use across different industries and project types, the communications may involve different data inputs such as location or sub-category.

7.5 Twitter Bootstrap

Twitter Bootstrap is a development framework, released in 2011 (Lerner, 2012) used to aid developers in their front-end designs. This framework was made available by the popular online social networking provider known as Twitter (Twitter Inc, 2015). The framework is available for free and anyone can benefit from the provided stylesheets and scripts files. These files are modifiable, and within this project have been modified to create customized features desired for the design of this project. The provided files allow for responsive layouts, allowing web environments to adapt to the various devices and screen sizes that

users may have available (Cochran, 2012). Responsive layouts are a huge advantage to developers as previously, this may have been an issue when designing web-based projects, perhaps sacrificing an optimized environment for particular devices. The Twitter Bootstrap framework has a large community of users whom are willing to offer advice and information regarding the framework (ebit.).

Lerner (2012), argues that using frameworks requires the developers to give up the freedom that developers have when they create customized CSS. However, he believes that this sacrifice of freedom leads to the added benefits of the CSS being shorter and more understandable. It also has the advantage of developers being able to focus more of the development tasks as opposed to ensuring that the CSS is appropriate.

Download of Twitter Bootstrap is available from their site, http://getbootstrap.com/, this site also contains documentation on the features this framework provides. The documentation is informative, and provides code snippets of how to set up the available features. Once the zip file has been downloaded, the files can be transported into the ASP.NET MVC project. The following html tag allows the stylesheet to be added to the project and the related CSS will be used that match the tags within the html document.

```
<link href="~/Content/bootstrap.css" rel="stylesheet">
```

Although I used the Twitter Bootstrap css and JS libraries throughout the project, the imported sheets were edited frequently and heavily during the creation of the project. For example, the progressive disclosure on the My Communications page was not a built-in feature.

7.6 AJAX and JavaScript

AJAX is a solution of incorporating a few different technologies to create an approach to eliminate the traditional flow of interaction (Garrett, 2005). AJAX combines the technologies of CSS, DOM, XML, XMLHttpRequest and JavaScript (ebit.). The traditional flow includes the stop/start process of interaction. A HTTP request is sent to the server, and once completed, the page refreshes as it is rendered again. AJAX is the solution to eliminate this flow, it allows for greater user experience as the user does not need to stop their interaction with the website each time a response is needed from the server. AJAX

allows for the request to be sent to the server and is able to just render the part of the website that is required, as opposed to the entire page.

7.6.1 Update an Action

One of the issues of the existing system is that the process of marking an action as completed was a difficult process that involved many stages. These stages included locating the action, clicking the action to open a detailed page regarding the action, changing the status and writing a comment regarding the action that had been completed. This process resulted in many actions not being marked as completed, and as a result users had long lists of the actions marked open. These long lists would eventually result in an unorganized work space and a call from a project manager offering little incentive for the employees to engage in the strenuous process of ticking off their many Open tasks.

My goal was to create a simplistic process where actions can easily be marked as completed without the need for this unnecessary and tedious process. I thought back to my research of other job logging systems, and although very simply, I was influenced by the job logging system that I described as a to-do list. The to-do list functionality allowed tasks of this job logging system to be easily ticked off. I wanted to implement something as simple and intuitive as this within my project. If actions were able to be quickly ticked off, it is not a tedious process for employees, instead it is very simple and of no extra work for employees.

My solution involved using the minus and tick glyphicons provided by Bootstrap, which has already been integrated with my solution. When the action is loaded from the database and displayed, the status of the action is checked. If the status corresponds to "Open" or "Overdue" then a Minus sign is displayed. If the status matches "Completed", then a green tick is displayed next to the corresponding action. Using JavaScript I was able to change the icon from a minus glyphicon to a tick glyphicon on a click, and vice-versa. This was the functionality that I wanted in terms of the interaction with the user.

To update the corresponding data within the database, I used AJAX. AJAX allows the updated data to be sent to the controller to update the model, without the user being able to notice this change as the page is not refreshed and behavior remains constant. I chose this method as it is the least amount of effort from a user interaction point of view and this enables usability to be high. When testing this process, the correct data was being updated in the database. I ensured this by changing multiple action statuses, in some cases, multiple times. I confirmed that the database had the correct values and when refreshing the

browser to reload the page, the correct glyphicons were loaded, relating to the updated actions.

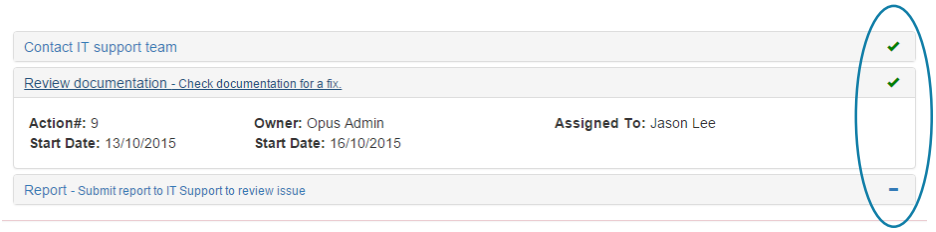

Figure 18, example of how the actions displays graphically the status of the action. When clicked, the icon changes and the database is updated.

7.6.2 Related usability work

This process related to the usability heuristic known as match between system and the real world. Ticking off a task if often done when humans create lists of tasks that they need to complete. The process mimics the creation of a list. The changing of the icon enables to the user to see the visibility of system status, as the change in the icon is a result of the change that has happened in the database.

7.7 Form Validation

Validation of the forms that are present in the OpusCOMMS site is important for integrity of the site in terms of security and user experience. Data validation is essential for the security of a web based application as well as sanitizing the input of trustworthy users of the system. Security is an important aspect of form validation as attackers could form inputs that have the potential to gain unauthorized access to the system, or alter some of the application functionality (Li & Xue, 2014). Validation can be used to escape particular characters to reduce some of these exploits being executed. A compromised system could lead to an access to unauthorized information, resulting in a release of sensitive data, or company secrets. Other risks include financial, ethical and legal issues arising (Li & Xue, 2014). However, this is in relation to the security of the application. The data input obtained from the user must also be meaningful and understandable to other users who may see the input that was created by another user. A situation of this is when one user creates a communication to be assigned to another employee to complete. Various mechanisms were placed in order to encourage the creation of meaningful communications, and the completion of other relevant forms.

Web applications created using ASP.NET MVC have solutions to the exposure to the security issues that arise when your application is hosted on the Internet. Some of these solutions have been incorporated into the solution to enable the protection against threats this site could be exposed to. A threat that the solution combats is the Cross Site Request Forgery (CSRF). CSRF is the process of a malicious site sending a request to a targeted site from a user's browser. The aim of this process is to disrupt the authentication that is granted by tricking the browser into offering access that the user did not endorse, but uses their authority (Kafer, 2008). This disruption affects the integrity of the victim's session with the targeted site. ASP.NET MVC provides a solution to this threat by offering the AntiForgeryToken HTML helper. The AntiForgeryToken() enables the site to be able to create a security token on the rendering of a View. In the controller of the project, a ValidateAntiForgeryToken is used to ensure that the request contains this security token. If the request does not contain the security token, an error is thrown.

In terms of the user input, it is vital that the data is entered in the correct format. This is important to ensure the integrity of the object that is being created in terms of a valid ModelState. This means that the data is able to be successfully added and saved to the correlating tables. It is also important to ensure that users are creating understandable communications that other users of the system deem readable and comprehensible.

7.7.1 Client-side validation

Although ASP.NET MVC has a simple solution to client-side validation that allows the developer to easily implemented client-side validation to their project, I implemented client-side validation using both the Model solution and a JavaScript solution. My reason for completing the same task twice was purely from a learning aspect. This task enabled me to have the opportunity that would allow me to learn some JavaScript and learn the ASP.NET MVC implementation of client-side validation. Both approaches use different parts of the MVC infrastructure, the ASP.NET MVC solution is implemented in the Model, whereas the JavaScript solution is implemented in the View.

The ASP.NET MVC solution provides a robust approach to client-side validation, with minimal code. Validation rules can be declared in the Model. The input is not to be restricted too much, as the rules are created upon assumptions. One of the issues that this presented was that the data within my database is dummy data. It was promised that the database would eventually contain the real data held within the existing system, and would need to be

manipulated to suit the newly created database. However, due to time constraints and issues, this data was never received. Therefore, the validation checks are basic ones that are based upon my assumptions of what is expected in the input fields.

7.7.2 Add a Communication

When adding a communication, it is vital that all the necessary data is created by the user corresponding to that communication. In the case of a communication, it is essential that the employee provides a title for the communication. The Title is required as it is vital for the identification of the communication when other employees are concerned, as it may be assigned to another employee. The minimum length of the Title is set to two characters, this was done as the actual minimal size is not known, and this could be amended for the live deployment of the new system. The maximum length was set to 50 characters. This check was implemented on the client size using JavaScript, and on the server side as a method within the Controller.

The Date Received, Comm Type and Medium are auto-set to default values, these being today's date, Comm-Work Request and Email, respectively. These default values are the most popular used to create communications, but can be changed by the user. The auto-set values create convenience for the user completing the form.

The Comm Type, Medium and Contact are restricted to drop-down boxes so that the user can only choose one of the pre-determined options from these lists. This reduces user error and the creation of many different types of mediums and communication types, created from users creating similar types of the same type. For example, there could be many variations of the form email, or users could type very specific or unrecognizable types.

There are also restriction related to the input type of the dates, it is preferred that the users choose to use the datepicker, however, users can type in dates if preferred, but these dates must follow a DD/MM/YYYY format. This is checked on both the client side and the server side. Rules exist around the collection of dates in the form. These include the Next Action Due date cannot be less than the Date Received field. Also the Send a Reminder date field cannot be less than today's date. Users can create communications that have been received prior to today's date as there is the scenario that users collect communications outside the office and log them at a later date. These rules have been implemented.

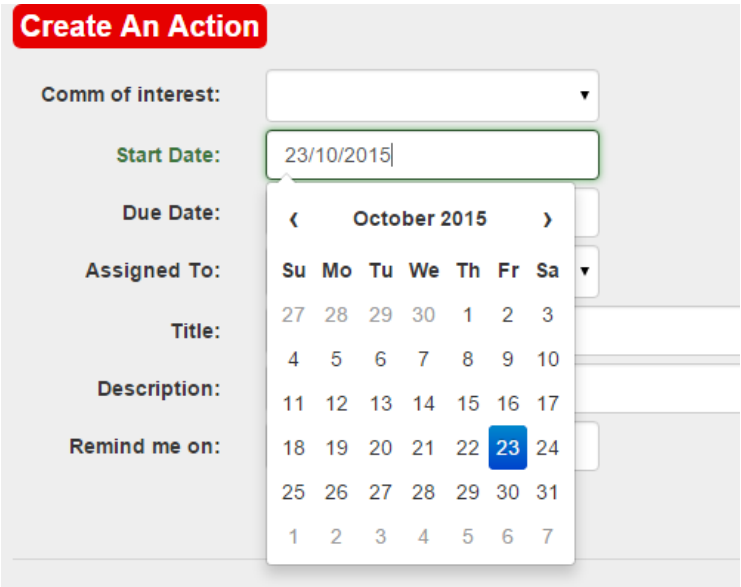

Figure 19, datepicker implementation. Start Date is auto-filled to today's date.

Omission of a Title would make the communication unidentifiable to other employees. Therefore the Title field is a required field. Omission of a Title results in the form not being submitted due to the client side validation. This check is also enforced through server side validation.

Figure 20, client side validation that displays while the user is completing the form, ideally before the form has been submitted. All error messages need to be fixed before the form is able to be submitted.

7.7.3 Add a Contact

The least necessary data required for the completion of this form is First Name and Last Name. These are the vital data components necessary for the recognition of a contact. Validation has been added to the name fields to ensure that they do not allow any incorrect characters such as numbers and special characters. There is also necessary checks on the

Phone number fields (Phone1, Phone2 and Fax), and Email fields to ensure the correct format has been entered. These checks are implemented on both the client side and server side.

The intention for the Organization field would be a restriction to a drop down list, and this would limit the number of similar organizations created by a user. If the organization applicable to this contact is not in that list, the user would able to create an organization to assign to that user, and is available for use later. However, this will be left for future work due to the complexity of the AJAX script needed to insert the created Organization into the dropdownlist and display this created option as the selected option and the timeframe of this project.

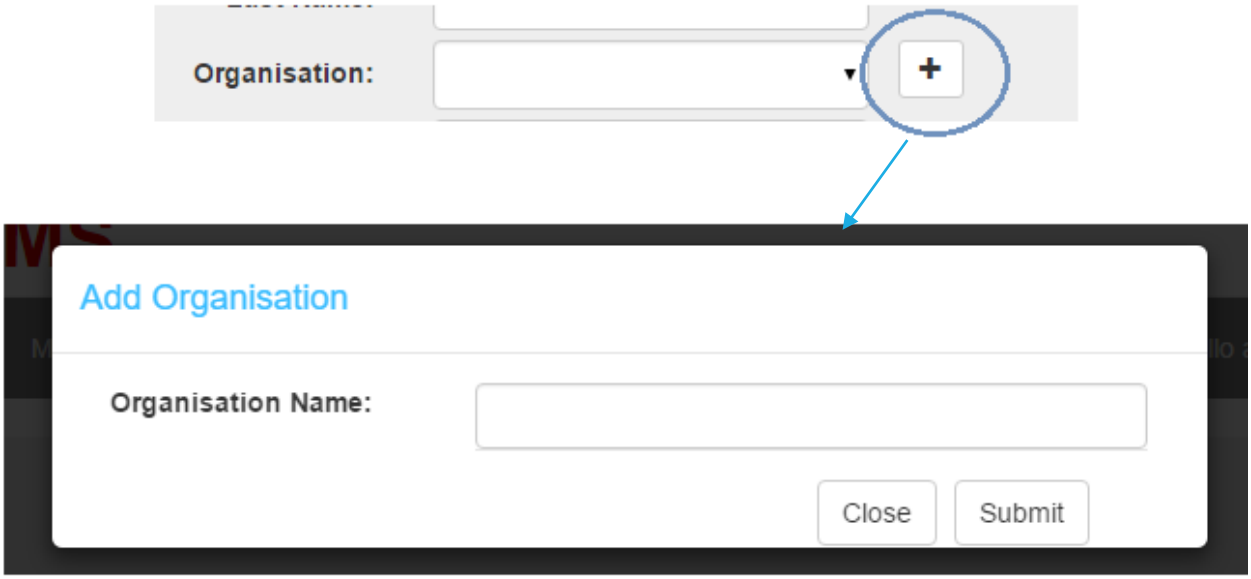

Figure 21, the Organization field was intended to be displayed as a dropdownlist with a "+" button to display a modal box, allowing the user to create a new Organization to be added to the dropdownlist.

7.7.4 Add an Action

When creating an action, it must be associated with a communication. The association allows the action to be kept track of and is organized for users to understand its origin. The required data also includes an Action Title, as well the Comm of Interest or Associated Comm. There is sufficient checks on the Start Date and Due Date fields, which checks the format of these fields is in the format DD/MM/YYYY and that the Due Date is after the Start

Date of this action. There is also a check to ensure that the chosen Remind Me On date is after today's date.

When an Action is created from the My Communications page, the Comm of Interest is automatically filled. Whereas, creating an Action from the Add an Action page could refer to any communication, and so the user is able to choose the communication that they are creating the action for.

7.8 Log in page

Microsoft enables an easy authentication process. The authentication process has been set up so that developers do not need to focus on security mechanisms of their projects. Default infrastructure has been created by default when a new ASP.NET MVC project is created. This infrastructure involves the creation of a default login page requiring a username and password, a table within a database has been created to store user data. The data includes the username and a password hash. When registering an account, by default, the solution asks for a username and a password to be written twice. The username must be unique and the password must be equivalent in both the fields requiring password.

I created a table within the database known as User. This table stored information applicable to the user so that the created users can be kept track of. This table does not store the password or the password hash.

7.8.1 Contact/User conflicts

An issue that arose when thinking about the registration process was the differences between a User and a Contact. A User is an employee or contractor that has been allowed access to the OpusCOMMS system, however, a Contact is an employee or contractor who can be assigned communications. Although these roles are similar, a User does not necessarily have to be a Contact, a User could want to assign communications, or want to view communications, but does not necessarily have to be a Contact. To solve this issue, I added some extra fields to the Registration page to be able to collect this information. I added a First Name, Last Name field to enable the collection of the user's first name and last name. Along with these fields, I added a checkbox that asks the user who is registering whether they want to allow create a Contact at the same time. Checking this checkbox confirms that the User registering will be also created as a Contact. When the form is submitted, a check is done to see if the User exists within the Contact table already. This match is done using the Email address field. The user is redirected to the Add A Contact

page. If a matching email address is found, the Add A Contact form is populated with the corresponding Contact information. The user can then choose to add any missing fields or update the contact information. If no contact is found, the field will be populated with the input User data.

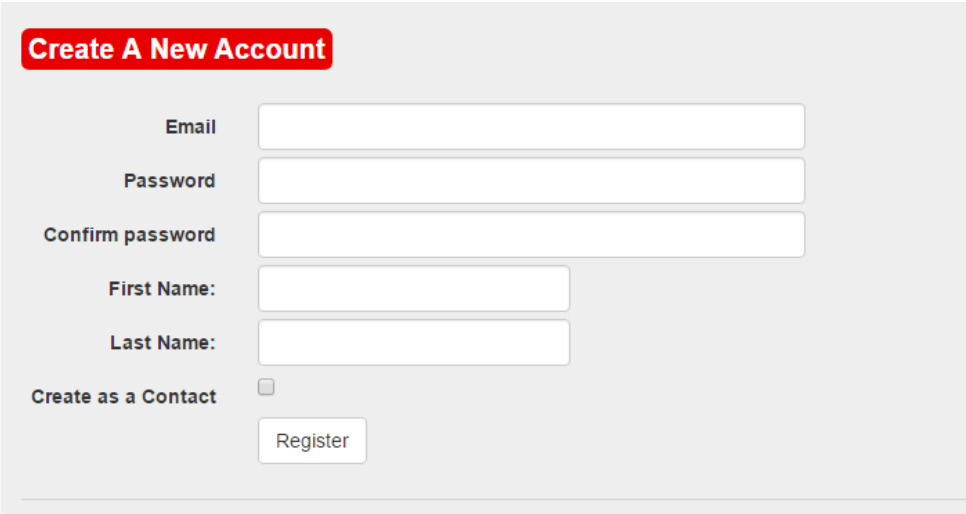

Figure 22, Register form that shows the extra details collected to solve the Contact/User conflicts.

The registration form implements all the necessary client side and server-side checks that are present throughout the other pages that implement forms.

A New user can be registered, and log in details can be used in future instances to log in, as shown in figure 23.

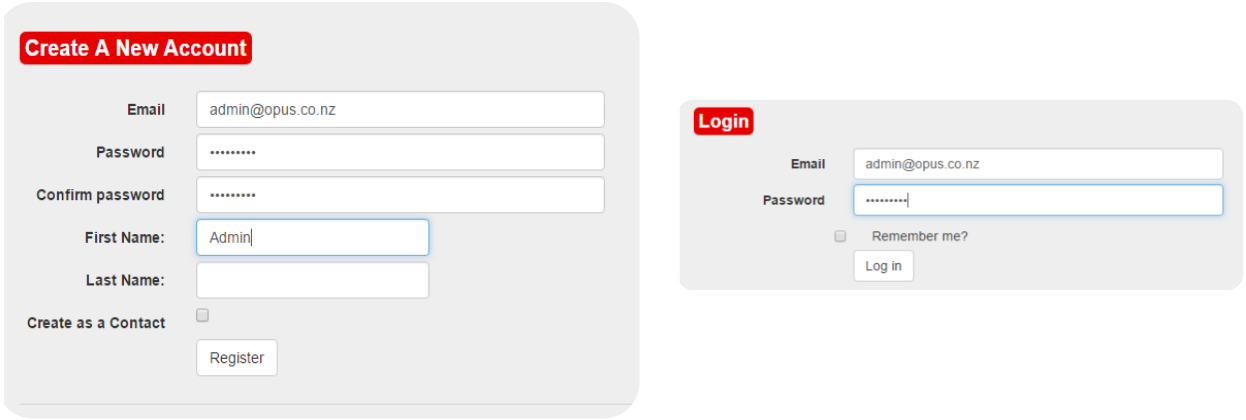

Figure 23, A user can be registered via the Register form, and then is able to Login during future accesses to the system.

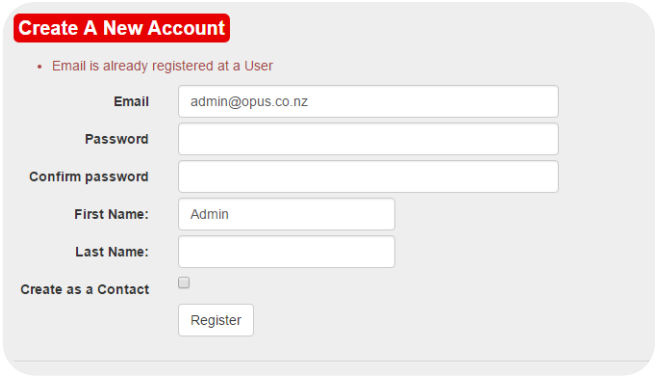

Figure 24, Validation of this form is implemented via the modal of the MVC infrastructure.

This page uses client-side validation implemented via the model of the MVC infrastructure, as shown in figure 23. Due to time constraints on the project, I was not able to implement client-side validation on this page using JavaScript. This time constraint creates an inconsistency within the pages of the web project. However, the registration of a new user would not be done by the average user, this task would be delegated to those with the required administration rights. Therefore, the difference on this page is not a hindrance to the usability of the web site, and these validation rules would have been added if the project timeline allowed.

A New user can be registered who is already a contact, without 'Add An Contact' tickbox, as shown in figure 24. This scenario creates a new user, but does not add them as a contact. However, the email address used to register the User is looked for within the Contact table, and if it exists, the ContactID is added as a foreign key in the UserContactID field within the User table.

A New user can be registered who is already a contact, with 'Add An Contact' tickbox. When the 'Add as a Contact" checkbox is ticked, the user is redirected to the "Create A Contact" page once the User's profile has been created. In the User already exists as a Contact, then this information is found, by matching the email address, and populated into the form. The user can edit this information if required and save it. However, if no Contact relating to the User is found, then the user is redirected to the Create A Contact page, yet the form is blank, allowing for the Contact profile to be created.

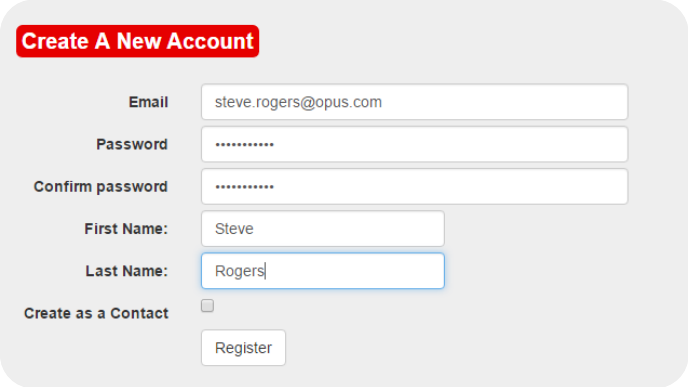

Figure 25, a new User is registered, but a Contact profile is not wishing to be created.

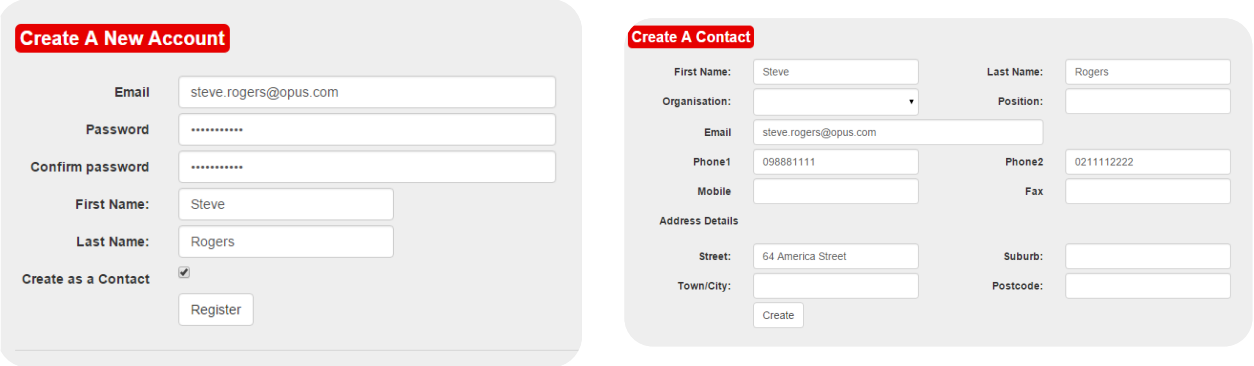

Figure 26, a new User is registered, and a Contact profile is also wishing to be created for this User. This User has previously been created as a Contact, and is found within the database. The redirect is populated with the Contact details to be edited and saved.

8 Deployment

8.1 Microsoft Azure

Microsoft Azure is cloud computing solution for businesses to host various areas of their business activity online. The desirable deployment of this web project was for the site to be hosted as a Web App on a Microsoft Azure account. This would have allowed for the staff of Opus to be able to access the Web App to see the status of the website and for more regular advice and feedback from my design decisions.

However, there were some issues that with these process. I was given access to the Opus International Consultants Ltd directory with the hope to be able to deploy my project through this directory. However, due to the permissions access that I was granted, I was unable to deploy my project to Microsoft Azure and these issues were not resolved. The issues were not resolved due to the Azure management team being located in Wellington and my Opus contact located in Christchurch. In attempt to resolve this issue, I created my own student account with Microsoft Azure and deploy the web application on this account. This process took a lot of time and the student subscription only allowed a connection to a MySQL database. As the project did not have a MySQL database, I did not want to spend even more time on this issue as it was becoming a major hindrance to the progression of the development of the project.

8.2 Visual Studio Online

Visual Studio Online was a solution to enable some source control within Visual Studio. A free subscription of Visual Studio Online allows a project to be shared with up to five people. The code is then displayed on the account under the directory of the OpusCOMMS site. The directory contains all the files that have been created within the ASP.NET MVC project, and this code is deployed to the directory from Visual Studio. After each session of working on the project, I was able to check in the code that I had been working on. This was a simple process done from Visual Studio that automatically selects the files updated since the last check in.

Sharing my code via Visual Studio Online acted as the method of deploying my code from my personal machine to the Opus development team.

9 Evaluation

Ideally, I would have liked to conduct a usability study on a selection of users at Opus in order to gather an understanding of how my system compared to the existing one. However, due to the complications with deploying the project as a web application on Microsoft Azure, there was not a system that could be used for a usability study. A usability study would have been appropriate as a large focus on this project was to increase the usability of the system. Instead, a usability evaluation was conducted based on heuristics to evaluate the usability of the project. Also, a performance evaluation was conducted using the Y-slow metrics.

9.1 Usability Evaluation

A usability evaluation can be used to evaluate the project in terms of its user friendliness and ease of use. This evaluation is important as usability was one of the focuses of this project as it was an element that the existing OpusCOMMS system lacked. The most appropriate usability evaluation would be one that observes users of the system carrying out various tasks related to the system. However, due to the time constraints of the project, sufficient time was not available for the planning and conduction of a full usability study. As a result, I will be conducting a heuristic evaluation based on the ten Nielson usability heuristics.

9.1.1 Visibility of system status

This heuristic involves the use of feedback for the user so that the user is aware of the current status of the system. The forms throughout the project implement JavaScript client-side validation. This validation allows for immediate feedback of any errors that the user may have entered. For example, an incorrect date format. This type of validation allows for the immediate feedback and visibility of the system status. The forms cannot be submitted if they do not comply with the validation rules. As a result, the user should be able to see that their form has not been submitted, as there are errors related to the form. When the form is submitted, a logical redirection is made and the user should be able to see their data submitted through this redirection. For example, when a user creates a communication using the form, after submission, they are redirected to the My Communications page, where the user is able to see their created communication.

To improve the visibility of system status, the actions created from the My Communications page would have been dynamically inserted under their associated communication through

the use of AJAX. This method was something that was planned for this project, however, will have to be left for the suggested future work of this project.

9.1.2 Match between system and the real world

The relation between the system and the real world is well balanced. The forms used throughout the project collect the data that would be necessary for the comprehensibility of the item. Also, the ease in which an Action can be "ticked off" mimics a To-Do list and is a very intuitive process, especially when compared to the existing system.

9.1.3 User control and freedom

The user has control over what they create when they are logged into the system. The user is able to create communications, associated actions and contacts. However, if they create something by mistake, there is no option to edit or delete the items. Deletion was something that is not allowed in the existing system, and the same rule may be kept with this new system. However, the only time that a user can edit is when their User profile is created and they want to amend their associated Contact profile. The ability to edit will need to be left for future work on the system. The ability to edit may be a restricted feature only available to privileged users, therefore, it was appropriate to leave these rules for future work.

9.1.4 Consistency and standards

Throughout the project, there has been careful use of the various vocabulary involved, and it has been ensured that the use of this vocabulary has remained consistent. For example, the words communication and action have been used appropriately and steadily throughout the system. This consistency would also ensure a smooth changeover from the existing system to the new system, when deployed. The existing system was used as a basis to understand the logic and expectations of the users of the system. Many of the forms implemented have taken data fields from both existing systems, internal and external, to ensure that the expected and correct data is collected.

9.1.5 Error prevention

A lot of work went into both the client-side and the server-side forms of validation. These validation checks ensure that the user is aware of the errors that they have made, and that they are able to amend those errors before the form can be submitted to be saved in the database. The error messages are also short and informative, for example "Please enter at least 2 characters." These messages ensure that the user knowns what is required from the

data entry field. The field also has a green outline when the user has entered an acceptable input.

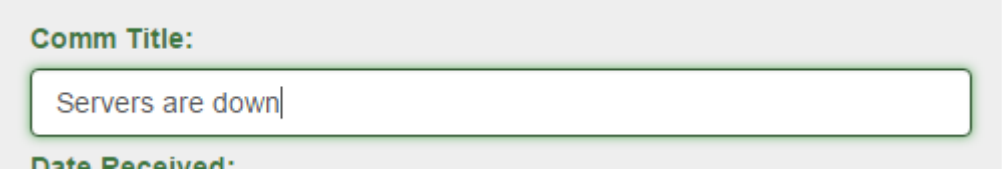

Figure 27, the data field border is green when the input means the client-side validation rules.

The use of datepickers also aid to reduce the errors made as the user is able to choose a date using the datepickers. Also, some data fields can be completed using a dropdownlist, this also prevents errors.

9.1.6 Recognition rather than recall

All the data required from the user as part of form input is clearly labelled. This labelling is fairly consistent with the existing system. This consistency allows for the forms to be easily recognizable and easy to understand. All the navigation throughout the site is descriptive of the page, for example "Add A Comm" and "Add An Action" are clearly navigations to the forms that complete the completions actions. Throughout all the pages, there is very little recall as the pages have been created with a focus of user friendliness and all have intuitive processes.

9.1.7 Flexibility and efficiency of use

The created OpusCOMMS system does not implement any efficiency of use features for experienced users, such as shortcuts to frequent tasks they perform. However, this could be an implementation of the Dashboard page. One of the ways that efficiency of use is created in through the "My communications" page. This page only displays communications that are relevant to the user that is logged in. Therefore, this creates a lot of ease of use as when the user logs in, this page can be referred to. The communications are grouped by when they are next due to create a list that the user can refer to and see their tasks that need updating for that day.

9.1.8 Aesthetic and minimalist design

Aesthetics and minimalist design are important features for the usability of the system. Referring to the project, minimalist design has been implemented as there is no unnecessary information shown on any of the pages. The pages are labelled descriptively. Also the user of progressive disclosure on the "My communications" page ensures that unnecessary information is not shown about communications the user is not currently interested in.

Hiding this information aids the aesthetics of the page. The aesthetics are consistent with those of the Opus brand image and incorporates their brand reputation. When compared to the existing system, I believe that the new system holds a lot more aesthetic appeal, and by not implementing the tabulated design, the new site results in greater readability and minimal design.

9.1.9 Help users to recognize, diagnose and recover from errors

The use of the validation rules, as mentioned in section 9.1.5 display clearing the error that is present for the user. The error also contains a description of what the issue is and helps user to recover from this error. If a form is not submitted, then the page is not redirected and the data still remains in the data fields. In these instances, a validation rule should be present. There are no paths in the code that lead to pages that are not available. If the user enters an incorrect link, then a 404 cannot be found error appears.

9.1.10 Help and documentation

Documentation for this system was not created. The reason for this was the intuitive process did not require additional prompts and help messages to describe the types of text that were required by the user. The majority of the users that would interact with this system would be familiar with the terminology and processes as they would be users of the previous system. Also, due to the size of the created system, there is currently not a need for documentation. However, in the future, if the project is maintained and developed, there may become a need for documentation, especially if the functionalities are vastly changing from those of the previous system.

9.1.11 Usability heuristic evaluation

Overall, the majority of Nielson's usability heuristics have been met through the development of this project. As usability was a fundamental focus of this project, meeting these heuristics is a valuable, summative evaluation for this particular creation. Through the evaluation, there was also improvements that were identified. These improvements included more focus the user control and freedom and the visibility of system status heuristics. Suggested work to reach these standards includes a frequently performed action suggestions and the use of AJAX to dynamically add actions created by the user.

9.2 Performance Evaluation

Y-slow is an open-source project that provides website developers with guidelines of enhancing the performance of their web pages. It analyses the web pages are provides suggestions of how to improve the performance of the website.

I used Y-slow to analyze the new OpusCOMMS system, and the overall performance scored 92, corresponding to an A.

A	Make fewer HTTP requests
A	Use a Content Delivery Network (CDN)
A	Avoid empty src or href
F	Add Expires headers
C	Compress components with gzip
A	Put CSS at top
A	Put JavaScript at bottom
A	Avoid CSS expressions
n/a	Make JavaScript and CSS external
A	Reduce DNS lookups
A	Minify JavaScript and CSS
A	Avoid URL redirects
A	Remove duplicate JavaScript and CSS
E	Configure entity tags (ETags)
A	Make AJAX cacheable
A	Use GET for AJAX requests
A	Reduce the number of DOM elements
A	Avoid HTTP 404 (Not Found) error
A	Reduce cookie size
B	Use cookie-free domains
A	Avoid AlphaImageLoader filter
A	Do not scale images in HTML
A	Make favicon small and cacheable

Figure 28, results from the Y-slow performance evaluation.

These results show that the performance of the OpusCOMMS system is good, scoring a grade A, however, as it is not a perfect score, Y-slow has revealed some room for improvements. Conduction of these suggested improvements may lead to a more robust, scalable project that is more user-friendly. The improvements of the Y-slow metrics would lead to greater usability as it would reduce the wait time for the user, and result in a more responsive web site design.

The results indicate that the main issues are with the imported CSS and JS files that were used to perform specific functionalities within these projects. For example, the bootstrap.css file that was imported for this project was flagged as not having an expire header or configure entity tags and was not cookie-free. These elements led to a lower performance score overall. From this, it shows that performance enhancements can be made by careful creation and execution of these stylesheets.

Overall, there is room for improvement related to the performance of this system. This performance rating could be enhanced if time permitted. The Y-slow analyze tool is an easy-to-use tool that provides valuable guidelines.

10 Future Work

The redesign of OpusCOMMS has a lot of potential and flexibility in terms of the future work. Throughout the project, there were ideas that were not fulfilled, and these were mentioned throughout the report to create an image of what was intended for the growth of this system. Ideally, the system will continue to develop in terms of usability to create an intuitive process for the user.

The following list shows the tasks that I would have loved to implement into this project, but due to time constraints, I was not able to complete them. In the future, it would be desirable to see these implemented to increase the user experience.

- Usability evaluation that allows users of the existing system to use the new system, to obtain feedback.
- Dynamic insertion of the created actions and communications into the "My Communications" page.
- Ability to Add An Organization on the "Add A Contact" page, to be displayed in a dropdownlist.
- TypeAhead feature implemented in the Contact allocation field, rather than a dropdownlist.
- The setup of the Dashboard page, this requires absorption of the data from the tables to display statistics for the logged in User.
- Enhanced security throughout the system. This area was not an area of focus due to the scope of the project and the environment in which it was contained. However, future deployment of the system would require a greater focus on security regarding the encryption of data and defense from a wider range of attacks.

11 Conclusion

Overall, this project has faced many challenges, and has enabled an effective learning experience throughout the year. Many different technologies have been incorporated into the final creation of this project. The goal to create a new OpusCOMMS system that was available for users both internally and externally of the company was completed. This was completed by the development of a single application, which took into account the combined functionalities of both of the systems. The project also incorporated user experience and user-friendliness, and when compared to the existing system, performs much greater in these areas, in my opinion.

The usability evaluation on related work due to the issues that arose during the final stages of the project, restricting a usability study on users of the existing system. The conducted evaluation of the system may be subjective, due to my opinions expressed during the evaluation. Therefore, the success of the enhanced usability, and the new OpusCOMMS may not be known until the deployment or acceptance testing phases. However, these phases would be left to the future work of the project, due to time constraints. A new database was also created for the system, and this took into account the future of the project. This new solution to a project management system may be used within Opus in the future to replace the existing OpusCOMMS system.

12 References

Chisnell, D. (2015). *What You Really Get From a Heuristic Evaluation | UX Magazine*. *Uxmag.com*. Retrieved 30 April 2015, from http://uxmag.com/articles/what-you-really-get-from-a-heuristic-evaluation

Cochran, D. (2012). *Twitter Bootstrap Web Development How-To*. Packt Publishing Ltd.

Erack Network,. (2015). *ASP Tutorial - vs PHP*. *Tizag.com*. Retrieved 6 June 2015, from http://www.tizag.com/aspTutorial/aspVersusPHP.php

Garrett, J. J. (2005). Ajax: A new approach to web applications.

Haberbusch, K. (2014). *6 UX Strategies to Make Your Site More User-Friendly*. *GA Blog*. Retrieved 29 April 2015, from https://blog.generalassemb.ly/six-ux-strategies-make-site-user-friendly/

Käfer, K. (2008). Cross site request forgery.

Latimes.com,. (2015). *Los Angeles Times - California, national and world news - Los Angeles Times*. Retrieved 29 April 2015, from http://www.latimes.com/

Latimer, J. (2015). *OmniUpdate | "User Friendly," Usability, and User Experience Decoded.Blog.omniupdate.com*. Retrieved 30 April 2015, from http://blog.omniupdate.com/posts/2014/user-friendly-usability-and-user-experience-decoded.html

Lerner, R. M. (2012). At the forge: twitter bootstrap. *Linux Journal*, *2012*(218), 6.

Macefield, R. (2014). *An Overview of Expert Heuristic Evaluations:: UXmatters*. *Uxmatters.com*. Retrieved 30 April 2015, from http://www.uxmatters.com/mt/archives/2014/06/an-overview-of-expert-heuristic-evaluations.php

Nielsen, J. (2006). *F-Shaped Pattern For Reading Web Content*. Retrieved from http://www.nngroup.com/articles/f-shaped-pattern-reading-web-content/

Nielson, J. (2015). *Progressive Disclosure*. *Nngroup.com*. Retrieved 22 May 2015, from http://www.nngroup.com/articles/progressive-disclosure/

Nielsen, J. (2015). *Heuristic Evaluation: How-To: Article by Jakob Nielsen*. *Nngroup.com*. Retrieved 30 April 2015, from http://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/

Paz, J. R. G. (2013). *Beginning ASP. NET MVC 4*. Apress.

Ranjan, A., Kumar, R., & Dhar, J. (2012). A comparative study between dynamic web scripting languages. In *Data Engineering and Management* (pp. 288-295). Springer Berlin Heidelberg.

Sortie en mer,. (2015). *Sortie en mer*. Retrieved 29 April 2015, from http://sortieenmer.com/

Spillers, F. (2015). *Progressive Disclosure*. *The Interaction Design Foundation*. Retrieved 22 May 2015, from https://www.interaction-design.org/encyclopedia/progressive_disclosure.html

Tezer,. (2015). *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems | DigitalOcean*. *Digitalocean.com*. Retrieved 6 June 2015, from https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems

The Webby Awards,. (2015). *2015 19th Annual Webby Awards*. Retrieved 29 April 2015, from http://www.webbyawards.com/winners/2015/

Twitter Inc, (2015). *Welcome to Twitter - Login or Sign up*. Retrieved 20 September 2015, Available at: https://twitter.com/

Li, X., & Xue, Y. (2014). A survey on server-side approaches to securing web applications. CSUR,46(4), 1-29. http://dx.doi.org/10.1145/2541315