The Redesign of an Existing Intranet System and its Effect on Usability and User Experience

Sid Jotsingani

28th October 2014

Abstract

This paper covers methods used to improve an existing intranet system by adding new functionalities to try and provide solutions to existing problems. The goal is to improve overall usability and to create a better user experience in the hope of improving internal business practices by staff members. The proposed changes to the system begin with a redesigned theme with a responsive layout that supports multiple browsers and devices, along with reasoning behind the design decisions. In addition to the design changes, new functionalities were also implemented such as a new profile management concept, a mapping application to provide a visual overview of user and project locations and a system to submit applications and generate graphs from statistics. Some statistically insignificant preliminary user testing was executed with a small sample size, however due to the nature of the testing this did not provide a conclusive insight as to how usability or user experience was affected.

Acknowledgements

I would like to thank everyone at Opus International Consultants Ltd. and at The University of Auckland who have supported me throughout this project. In particular I would also like to thank the following people for their continuous guidance, support and feedback:

Dr. Sulo Shanmuganathan

Industry Supervisor.
Opus International Consultants Ltd.

Dr. Xinfeng Ye

Academic Supervisor.
The University of Auckland.

Dr. Sathiamoorthy Manoharan

Information Technology Coordinator.
The University of Auckland.

Table of Contents

Introduction

Opus International Consultants is a multi-disciplinary infrastructure consultancy firm with over 3,000 employees in more than five countries. They are focused in 8 major sectors which include buildings, transport, water, energy and more. Opus deals directly with clients and most of the projects are completed on site at various locations depending on the client's requirements.

The current intranet system is primarily used to manage and search profiles of employees, and to view internal documents from inside the company. However the usability of the system is quite low, and performing tasks which are meant to be trivial, such as editing one's profile, can turn out to be quite complex. Along with the complexity of the system, it is missing some key pieces of functionality that the company wants implemented in order to automate and simplify some tasks that are normally performed manually by employees.

One such issue this project addresses is that user's do not keep their profiles maintained. This includes keeping their current location to date. This is quite an important aspect which is needed in a firm where users are constantly travelling between different sites and locations. In order to improve this practice, the design of the intranet needs to be improved without being too complex a change where users have to re-learn the system. Research needs to be done on how usability can be improved in the current intranet system without increasing overall complexity, and to understand what functional and design decisions need to be made in order to enhance the user experience in the current intranet system.

A Quick Note About The Project

My supervisor and peers at Opus have been very supportive of throughout this project. One of the issues that arose in the earlier stages of this project was that a transition was meant to be occurring from Opus's current intranet system to a SharePoint solution whilst I was to work on the intranet system. What this meant for me is that my academic supervisor and I had to discuss whether I develop for the older system which only had a limited lifespan left or if I wait and develop for the newer SharePoint system. The IT team then kept any changes that were to be done to the intranet on hold, so the timeframe as to when the migration was to happen, became uncertain. With these constraints in mind, we still had to keep development moving and so any project work done so far has had to be done locally on my own server and not on Opus's actual systems.

This will require changes later in the future (possibly significant changes) when migrating – but at least the fundamental aspects of the logic and design will remain the same. I also acknowledge that the work I have completed was not migrated to Opus's servers by the time this project ended. However, it can be used as something to present to the IT team about what we are able to redesign which they could then port themselves. It could also be used as a proof of concept for future intranet iterations. Apart from this I was never able to meet the IT team in Wellington and was unable to look at how the backend of the intranet was actually working or what its infrastructure was like.

Despite the fact that a lot of the items I have developed may have to be modified before it can be migrated to work on the newer SharePoint system, the project was still pushed forward and there was always something to work on and test (locally). Opus was very co-operative and helpful throughout the process.

Related Work

The goal of this project is to improve the usability in the system and to enhance the user experience of the intranet system. Usability and user experience are often used interchangeably, however there is a distinction between these two terms.

N. Bevan wrote that a survey at Nokia stated that User Experience was perceived as being usability plus anticipation and the pleasure from the usage (Bevan, 2009). N. McNamara & J. Kirakowski stated the three areas of concern in Human Computer Interaction are Functionality, Usability and User Experience. They define usability as a "characteristic of the interaction between the user and the product" and state usability answers the question "can I make the product do what I want it to do"? This differs from their definition of user experience which is "the wider relationship between the product and the user, in order to investigate the individual's personal experience of using it". User experience should answer "how the person felt about the experience, what it meant to them, whether it was important to them, and whether it sat comfortably with their other values and goals" (McNamara & Kirakowski, 2006).

Both Bevan and McNamara & Kwiatkowski agree that user experience links with usability plus the emotional response attached after using the product. This is also quite similar to what Domain7, a group of designers from around the world say about usability and user experience. They state usability plus user experience, are both essential to the success of a website/application. Usability is about task-based interactions and is allowing something to be done easily and intuitively whereas user experience is how a person feels when they interact with your product and their emotional connection to the task (Domain7, 2014). What this means in the context of the project, is just because the website is easy to use – it does not mean that users are enjoying using the site or are having a good experience using it. The intranet must be easy and simple to use (usable), but must be designed in such a way that it is delightful and a pleasure to interact with, and that there are no aspects that give the user frustration or inconvenience.

One of the most known authors in the world of Human Computer Interaction is Jakob Nielsen. In his research, (Nielsen, Usability 101: Introduction to Usability, 2003) Nielsen defines usability as "a quality attribute that assesses how easy user interfaces are to use". In the same paper he also states that for intranet systems usability relates to employee productivity. So if an employee has to spend time deciphering their next actions, then that time wasted is money lost by the company without any work or productivity accomplished. J. Nielsen & R. Molich also list out 10 usability heuristics for interface design. These heuristics are meant to solve common problems that people often face in interaction design (Nielsen & Molich, Heuristic evaluation of user interfaces, 1990). They 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 recognize, diagnose, and recover from errors
- Help and documentation

The proposed work will be using these points, commonly known as “Nielsen’s heuristics” as guidelines for some design decisions.

A. Hinchliffe & W. Mummery performed a similar task of attempting to improve a health promotion website – where the main goal was improving usability. They broke down their usability improvements into 6 different categories. The Design which was about changing actual visual elements; the feedback which was the response the system gave; the format such as entering dates in the same format consistently throughout the website; The instructions given to the user; The navigation and how easy it was for a user to get back to where they came from; And the terminology. Upon improving these 6 categories they found an overall improvement in Usability (Hinchliffe & Mummery, 2008). There are again similarities between the changes performed by Hinchliffe & Mummery and Nielsen’s Heuristics. The instructions, representing the help and documentation, the terminology relating to the match between the system and real world, the navigation relating to the visibility of system status and so forth. This again indicates that Nielsen’s heuristics appear to be a reliable method to asses and increase usability of a system and that usability can possibly be improved on a website by following these set of guidelines.

A recent book by R. Lal states that the best interfaces are those which have a minimum design, simplicity, accessibility, consistency, feedback, forgiveness and are user driven (Lal, 2013). This also matches with what Nielsen also states in his heuristics, 10 years later. The core concepts appear to still be the same. Instead what appears to be changing are how designs are expressed and our perception of what consists of good design.

Figure 1 shows the top most English websites sites on Alexa.com (Alexa, 2014). After looking at each of these websites in detail, I aggregated a list of design patterns that were similar between most of the sites. This list shows showed some common design trends that are slowly emerging from my observations:

- The use of white and grey to distinguish foreground and background. Most of the sites are using grey as either the body background or navigation bar colour, and using a white for the background of the main content of the site.
- “Card styled” layouts using squares and rectangles filled with the main content. These cards generally have sharp corners, thin margins between other content and contain a combination of text and images.
- Shadows to give depth to the main content. Despite moving away from gradients and using single “flat” colours, most of the areas that contained content (cards) had shadows to make them appear as if they are sitting “on-top” of the background.
- Highlights/outlines on input boxes when focused onto them. This is a form of feedback when clicking inside a textbox to show which box you are typing in.
- Larger than the default sized input boxes with large fonts. Most of the input boxes had modified CSS to make them larger than the HTML default size, along with larger font than the size of the text used in the body content and paragraphs.

- Some form of separation with the logo and search, and the rest of the content. The logos mostly sat in the top left corner with the navigation horizontally across or vertically down on the left hand side in some cases. There would either be some break in colour, border lines, or by padding to break where the navigation ends to where the content starts.
- Consistent colour schemes and styling throughout webpages. Each website had their own set of colours they would use throughout on multiple elements. The layout would also be consistent throughout pages. An exception was normally the homepage which may have had a different layout – but which still used the same colours and styles.

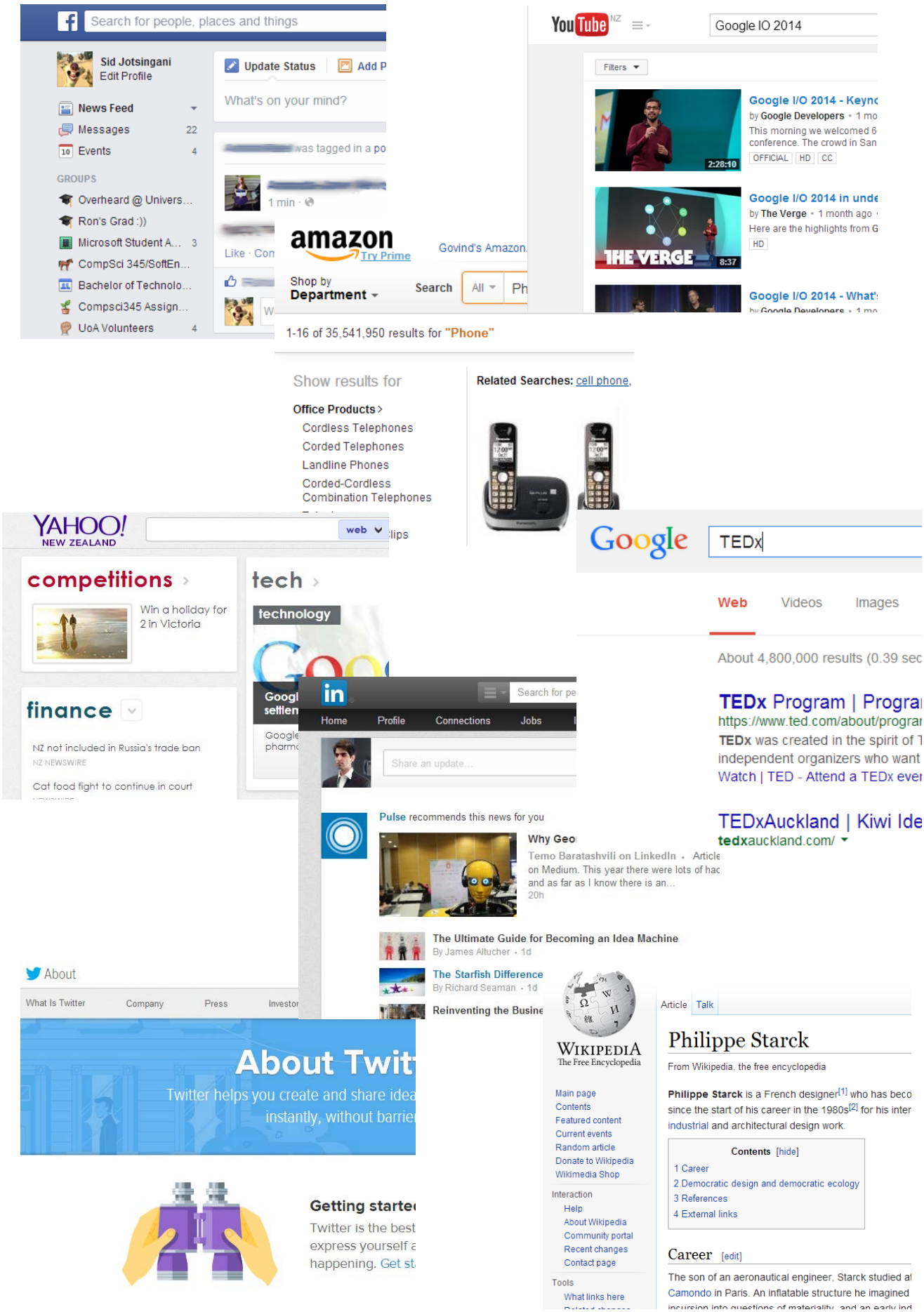


Figure 1: Top Alexa.com websites from left to right: Facebook.com, Youtube.com, Amazon.com, Yahoo.com, Google.com, LinkedIn.com, Twitter.com and Wikipedia.org

On most of the researched websites, an emerging trend is the use of the "card-style" layout where the overall design is flat – yet depth is perceived. This combination of a card layout along with flatness is not only emerging in web technologies, but is also advancing in mobile design too. In 2013, Apple overhauled the iOS layout with the release of iOS7 to a flattened layout – yet added depth using parallax (Figure 2). Windows 8 and Windows Phone 8 use a tiled layout with and have adapted a flatter style of icons too. Whilst Android has been using flat card style layouts for some time now, and recently with the announcement of Android 5.0, Lollipop has changed their design language to "material design" which follows the idea of cards with depth and layers (again using shadows). This increase in popularity in flat design contrasts skeuomorphic design which came into play when the term Web 2.0 *(*and understanding what it was about) gained popularity in 2004 (O'Reilly, 2005).

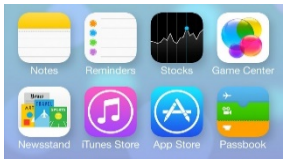


Figure 2: iOS7's flat design still uses shadows behind flat icons to perceive depth

Until about 2010 iconography was focused on skeuomorphism, meaning making icons look as close as they can to their real life counterparts. This meant the use of gradients, 3D perspectives and excessive use of shadows to give it real life characteristics. As shown in Figure 3, common design trends have changed since then to flatter designs. More and more users and firms have begun adapting to this new flat design style and this trend is still only gaining in popularity. However as A. Turner points out, one of the biggest drawback of flat design has to do with icons losing meaning, and that flat icons often do not offer feedback when clicked or hovered upon (Turner, 2014). From a usability perspective, this lack of feedback is a step backwards. Nielsen states there should be visibility of system status – which includes providing feedback on a performed action in a reasonable amount of time. It also violates one of Shneiderman's 8 golden rules of principle design which states an interface should offer informative feedback (Shneiderman & Plaisant, 2010). So despite having a flat design – some type of feedback should be given to a user, even when using flat icons in order to maintain a certain level of usability. This is a balance that needs to be achieved throughout the entire design process.

Figure 3: A change in design trends to flat logos

Ilya Pozin, a writer on Forbes.com mentions that responsive web design is the new emerging trend of web design. He says we need to embrace higher resolutions, so from a web design perspective we need to make our websites more responsive and fluid. Long gone are the days of fixed size. But along with responsive web design he mentions about how the other biggest change is that design is becoming flatter and is something we need to adapt to. (Pozin, 2014)

The Design & Theme

When deciding what design elements should be changed for the current intranet system, I decided to have a look at what Opus's current public facing website looks like (Figure 4). The colour scheme appeared to pop out immediately on viewing the page. The foreground and background were separated using two shades of grey. The background was a lighter shade of grey, with text a darker grey colour sitting on top. The text for any important information (main headings and the current navigation item) was all uppercase, and in a bold red font. Other headings (sub-headings, and other navigation items) used the same font but in a darker grey colour. The use of red was what was standing amongst the shades of grey, and that was being used for any important information.

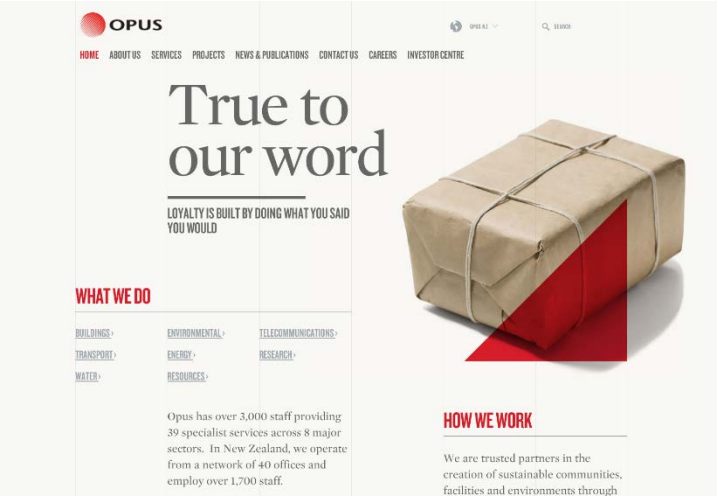


Figure 4: A screenshot of the homepage from Opus's public website

Apart from the colour scheme, the logo was placed in the top left corner with a horizontal navigation bar below it (which fits in with other common design trends). The text used throughout was a large serif font, and the design was overall very flat. The background colour was not a standard white as most websites use, but instead a lighter shade of grey. I personally believed this still worked quite well since the text still had a large contrast with the background colour. This public facing website was overall very pleasing to look at. When this is compared to a page from the current intranet system, the differences immediately become visible.

Figure 5 and Figure 6 show screenshots from Opus's current intranet system. The first is a screenshot from the profile page of another user, the second being the results of a search for an employee in the company. The first noticeable difference is the lack of consistency between the two pages itself – despite being a part of the same intranet. The logo and header has remained the same, however the navigation bar has changed completely upon searching for a new user. Secondly the layout of the main content has changed quite drastically as well in terms of width. Where in the first screenshot the content is centred inside a beige box which has a fixed width, in the second screenshot the content takes up almost the full size of my screen (despite still being a fixed size). It has also been placed inside the background itself – rather than inside a content box like on the profile page. Both these pages had layouts of completely different widths – one looked portrait, whereas the order looked landscape. One of Nielsen's principles was about consistency and standards. This is something that will need to be addressed in my implementation ensuring that there is consistency between the pages. Secondly Nielsen also mentions about aesthetics and a minimalist design. Compared to the website counterpart, the intranet interface looks much less aesthetically pleasing and there a lot of content has been crammed together. This is especially noticeable on the profile page where there is a lot of information packed by the name.

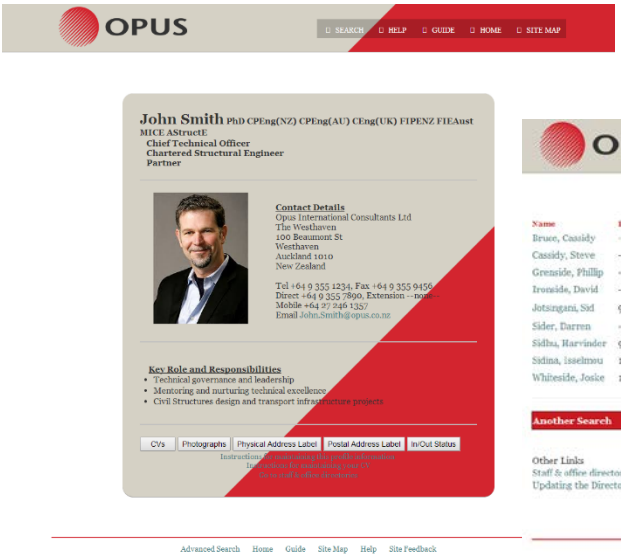


Figure 5: A screenshot from the profile page from Opus's current Intranet system

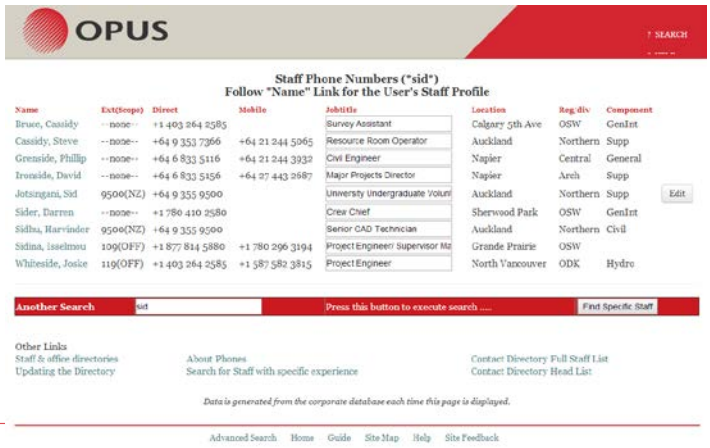


Figure 6: A screenshot of search results from Opus's Intranet

From the people I met at Opus, it was clear that not all of them were very "tech-savvy". So the design I choose to implement definitely had to be very intuitive and easy to use. One of the decisions I made was to attempt to improve the functionality and design of the intranet system, without making significant visual changes. Any changes I wanted to do needed to have a reason, and they had to be simple, yet effective modifications. The reason for this was that the end-user may feel intimidated seeing the system appearing to have changed so much. Changing a system entirely may also require (or make the end-user feel) as if they must relearn how to use the basic functions of the system. This would not only be difficult for the end-user, but they may also feel frustrated having to re-learn something they may have already previously known how to do. Thus the changes made, had to keep the overall layout of the website similar and familiar.

Another design choices made during the design process of the intranet, was to also bring consistency between the different pages. Not only that, I also wanted to bring consistency between the intranet system and Opus's website so that employees can recognize the system easier and possibly have a better understanding of it. Nielsen in his book, Designing Web Usability: The Practice of Simplicity, mentions the structure and organisation of graphical elements of the web-site should have uniformity. He says "uniformity of design re-enforces ease of use, as users do not have to 're-learn each new page in the site" (Nielsen, Designing Web Usability: The Practice of Simplicity, 2000). This again relates to the fact that consistency and uniformity should try to be achieved between the intranet and internet system, without the cost of complexity.

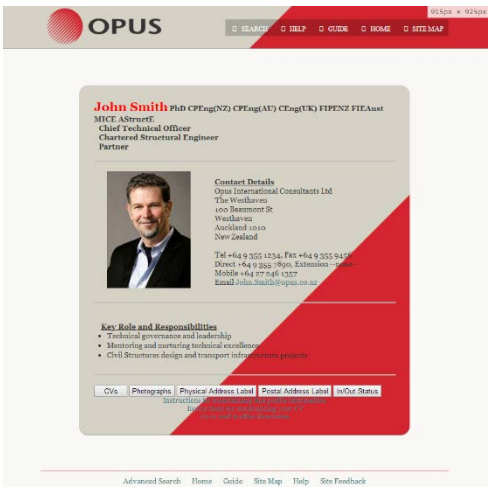


Figure 7: The first modification to the existing intranet including a grey background and bold red headings

To achieve this, this meant bringing certain elements and design features from the public

website to the intranet system. Some of these design features were the bold red headings, the lighter grey background and the minimalist navigation system over to the intranet system whilst trying giving it an overall cleaner layout. To do this, the colours used on the website were sampled, copied and then used on the intranet page. The colour of the headings was now a darker red rather than black, and the background was also a lighter shade of grey as used on the website. As you see Figure 7 shows how the design has changed, it is barely noticeable, but it was a start at trying to bring consistency between these two individual systems.

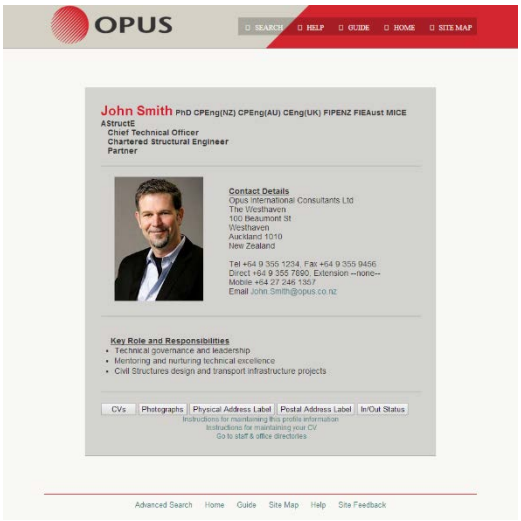


Figure 8: The next modification including sharper corners, new colours and a sans-serif font

The next biggest change came when redesigning the content box. First of all the background colour was changed from beige to a darker shade of grey to better match the website background colour. The edges of the boxes and input buttons were squared off rather than being rounded. Sharp edges were more common than rounded corners in recent design patterns (such as Google card layouts, Yahoo news categories, Facebook news feed items, and LinkedIn updates all using square rather than rounded corners). The square corners also seemed to be a better match fit due to the fact the Opus triangle used in their branding (which appears on most pages) has sharp corners and not rounded ones. The font was also changed to a much more legible and common, sans-serif font Arial rather the previous serif font, Georgia. The use of Arial does not follow what's on the Opus website, but was chosen because of its easy readability at small sizes, simple design and it being acknowledged as a CSS safe web font to use for web design (W3Schools). The background image of the red triangle in the content box was removed (temporarily). The screenshot, Figure 8 shows the changes after this.

Figure 9: The third modification cycle consisting of a changed navigation bar

As you can see in Figure 9, the navigation bar was also changed to better match that of the Opus website. A similar horizontal layout was used. The reason for this was to enable users to recognise the navigation layout that they may be familiar with from the Opus website, and to again bring consistency between the intranet and website. This newer navigation bar, also better suits the common design pattern of keeping the navigation aligned to the left rather to the right. The header font and navigation font is not the same as the font on Opus's website. Opus's public facing website uses a premium font called "Knockout" whilst instead I used a free font called "League Gothic Regular" was used. In the future, this font should be replaced with "Knockout" if permission is given by Opus to use it. Just these changes alone, appeared to visually better match the look and feel of the Opus

website. However even at this point it felt like some things were lacking and could still be improved upon.

The next changes that were made were made based upon looking at the top few Alexa.com websites (Figure 1) and the original intranet page rather than Opus's website. This was to bring more design elements that were seen as modern and contemporary whilst at the same time trying to retain the existing look and feel of the original intranet page. This was so it did not seem like a completely overhauled system design that may frighten some users.

The current design was already pretty flat (and was thus suited to modern design patterns), however with most of the top sites, some form of shadow was still being used to distinguish foreground from background. A box shadow was added around the content box, to give it depth as if it were sitting on top of the background. The width of the content box and some elements on the page were also modified to be more fluid and to change according to how a website scales for different sized screens. The logo and navigation was aligned with the left side of the content. The background of the content-box was also changed to be transparent, the red Opus triangle was also added back which was made using pure CSS rather than an image with its z-index set behind the content box. This also helped give the effect of depth using a flat design. Figure 10 shows the template minus any content (that will vary per page) for my proposed solution. To bring consistency this will be used on every page filled with content inside the grey area. The style also follows the card-style layout used on most of the top sites of the web. Figure 11 shows the layout of the profile page using the new theme. Note that the layout also has some new content such as the addition of a map, which wasn't part of the original layout. This map is further discussed on Page 39.

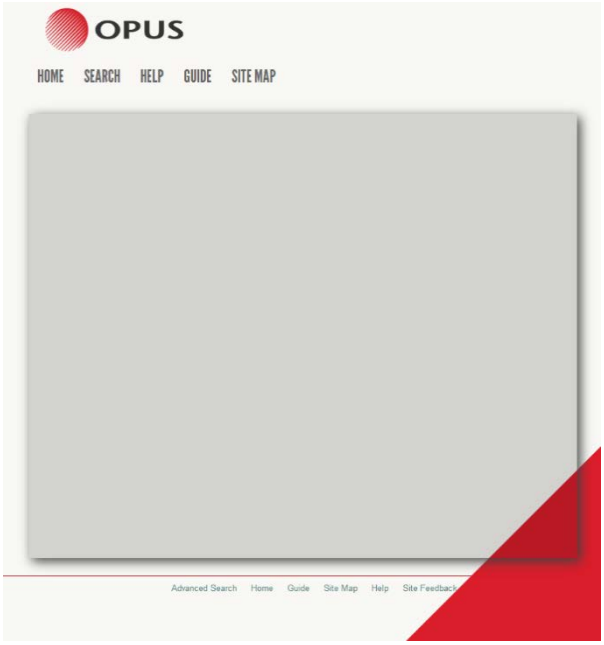


Figure 10, The proposed theme with no content

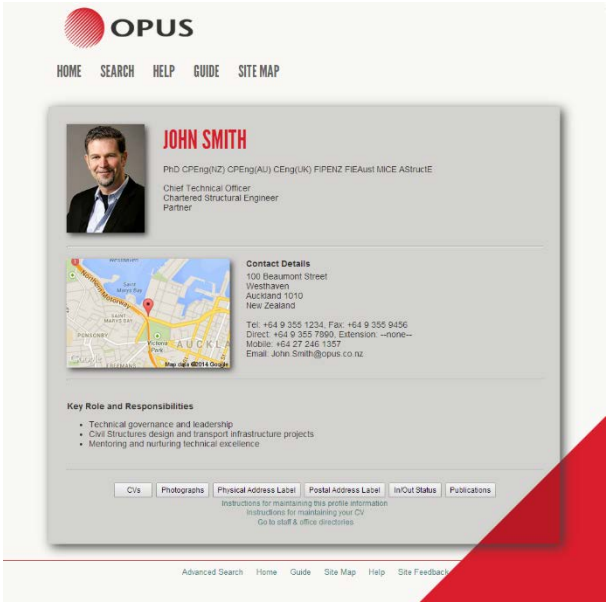


Figure 11, The profile page with the new proposed theme

The design shown also consists of some embedded features which are not directly visible in a static screenshot. These include a lightbox (pop up overlay window inside the current page) that opens when needed, and the use of some jQuery animations. The animations are being used as an attempt to enhance user experience by making the overall feeling of using the site "fun" and

making the experience a joyful one. They also help provide feedback to the user on their actions performed.

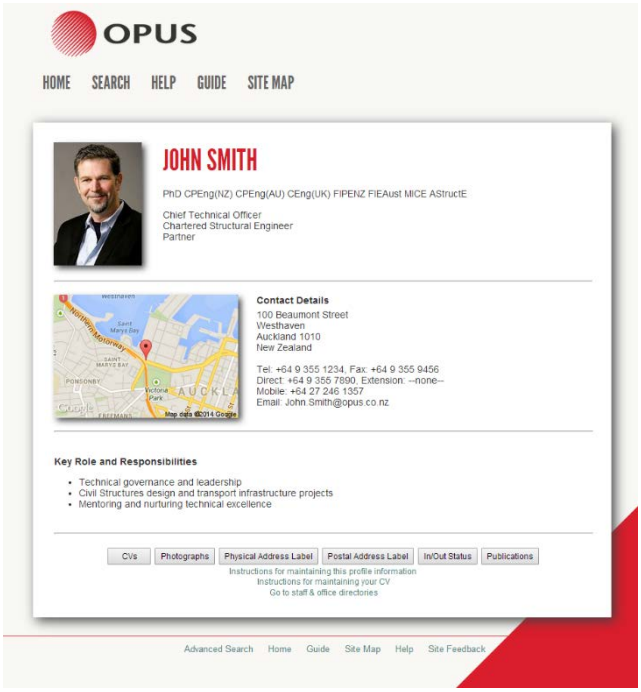


Figure 12, An alternative to the current theme where the background colour of the content box has changed to white

One of the major differences between my proposed design above and common design patterns, is that the background I am using for the main content is not white. Most of the top websites use a white background for the content followed by a grey colour for the rest of the background. The left figure (Figure 12) shows an alternate version of the theme using a white background for the content as opposed to a dark grey one. The reason for the dark grey was used, was to better fit the Opus website colour-scheme that did not contain any white elements at all. Having the darker background for content and a lighter background for the body was a closer match to that of the original opus intranet page, so it did not look like significantly difference.

I do not know which of the designs will be perceived as more usable or which may provide a better user experience, so some further research or testing on users to deduce whether the white or grey background provides a better user experience. Upon showing both designs to my industrial supervisor, the preference was the darker background option, so for the rest of the project this is the design that I have continued to use (Figure 11).

I have attempted to cover as many of Nielsen's Heuristics as I could in my proposed design:

- Showing the system status at any given point by giving feedback on actions such as hover on links, adding loading bars for certain events and the use of modal messages upon errors or successful actions (See implementation section on Technical Funding Applications on Page 45).
- Matching between the system and the real world was something I could only do to a certain extent. However this seemed to be quite fine to begin with. I am using the same text that was on the original intranet pages and so I had no control over the jargon. However when it came to me using natural language rather than jargon, any messages I was showing/display to the user are using simple non-technical terms. I could have also tried to use icons that relate as best as they can to real world objects (See universal search section of the mapping application on Page 26).
- User control and freedom in the with the ability to reverse an action such as close a lightbox when not needed, and the ability to modify content on the page itself. (See double-click to edit feature in profile management on Page 33).

- Consistency and standards is being addressed by creating a single theme that is to be used amongst different pages, along with keeping some design aspects of the intranet similar to the website. Modern standards have also been used throughout such as the use of shadows.
- Error prevention by red modal boxes containing the error in a simplified language upon an incorrect action such as uploading an image as opposed to a PDF (See implementation section on technical funding applications Page 45).
- Recognition rather than recall – most of the common actions have been kept in the same location between pages (navigation and footer). However in future iterations of the design, a more prominent and improved help system could be implemented to help new users understand the system better.
- Flexibility and efficiency of use - there are no shortcuts implemented for experienced users. Both novice and experienced users go through the same interface to accomplish tasks. Future improvements could add this functionality to assist more advanced users.
- Aesthetic and minimalist design – the proposed design appears to have been simplified down from the original, however there is always room for improvement. There are very little frills (such as the Opus triangle), apart from the core elements on the page.
- Help users recognize, diagnose, and recover from errors – all error messages give a clear indication of what the user did wrong and how to fix the error. Also any errors in inputting data are immediately highlighted in a red colour around the border of the input box.
- Help and documentation – currently only the help section which is accessed in the footer. Instructions have been made visible where possible. However in future a more prominent and improved help system could be implemented on a page-by-page basis to help new users understand the system better.

When I started looking into usability and design, what I kept reading about responsive web design and the enormous advantages of creating websites for responsive design. All the articles and journals contained information about the responsive web. However after reading some papers – I soon learnt that fluid web design and adaptive web design also have some similarities to responsive web design. Although there is a fine difference between them – I wasn't really sure which one I was designing for.

G. Carlos states in his book "Responsive Web Design with jQuery" (Carlos, 2013), what the difference between fluid and responsive design are. Fluid design involves adjusting the dimensions of elements but without varying the layout structure. So you generally use percentages for all the elements. Adaptive design changes the layout for different resolutions, but it's more like having a bunch of static layouts and is not fluid (Davison, 2013). Finally responsive web is having fluid layouts while also changing the layout structure for devices with different resolutions. An example for responsive is where all the content moves to a more vertical layout on mobile phones (since phones are larger vertically), and horizontally on traditional monitors (since displays are larger horizontally). J. Campbell & B. Shelley raise a good point about static layouts – that static layouts are always consistent, (and therefore look the same and are recognisable) no matter what the resolution is. However the

drawback here is that for screens with extremely small or large resolutions it may appear too small or too big (Campbell & Shelly, 2014).

Now the question was what was I designing for? The current Opus layout was a static layout that did not respond to any changes in resolution. Until halfway through the project, the layout style I had begun working on was a fluid layout. My initial reasoning behind this choice was that adaptive layouts appear to be less responsive to changes compared to fluid and responsive layouts since they do not respond to changes as well as those, whilst static layouts do not respond at all. Although a responsive layout consists of a form of fluidity layout plus changes in structure (and is therefore generally seen as being the better choice) – in the scenario of this project – the Intranet (at least currently) will only ever be accessed on desktop/laptop devices with larger screen resolutions.

However in the later stages of the project, I made the decision to try and convert what I have made so far into a responsive layout. Despite connectivity to the intranet was only possible through a wired Ethernet connection (that is only available on desktop/laptops or larger tablets), I felt that in the future it is possible that the scope of this project may extend beyond the intranet. It is possible that some of the information that is only available inside the office now, may be available to be accessed remotely in future versions of the intranet system. Therefore I decided to "design and plan for the future", I would attempt at converting my designed site to a responsive one. I felt it was best to do this now when it's not too complex, rather than someone else having to modify it, when it may be harder to do so later. Responsive layouts are ideal for scaling a website down for mobile browsing. Again this currently wasn't needed, but it may be in the future.

Responsive layout use media queries. A media query simply is a conditional CSS statement that will apply a set of styles depending on a parameter, such as the width or resolution of the screen. They can be used to detect the end users screen width, and accordingly apply different style-sheets that allow the page to better fit. After some research on how to structure my media queries I came across a book by Benjamin LaGrone. In his book on responsive web design (LaGrone, 2013), LaGrone shows us an example where he classifies screen sizes into three main groups:

- Small screen sizes – width is less than or equal to 800px
- Medium screen size - Between 800px to 1024px
- Larger screen sizes - 1024px and bigger

These three sizes appear to be sufficient for the time being to cover phones, tablets and desktops. However we can always add support for more screen sizes, such as devices less than 480px and testing for 1920px which are also common media queries.

In my design so far I was using a larger size content box than the original did. So the main content box width had increased by default – however for devices with not as much screen width, the content box needs to decrease down to a smaller size to better fit the screen. I used media queries to fill the width of the content box to 100% if the size is smaller than 1024px. This completely hides the margins on the side and maximises the space we have for displaying the main information on the page.

At the same time we want the content box to increase for larger screen sizes, such as screen sizes larger than 1024px. Rather than use a separate media query, we can emulate this feature by re-sizing using the CSS2 property of max-width, rather than using a fixed width command. Using the max-width command we can still allow the width of the content box to increase without the structure changing any more. So if the window is getting bigger, the size will keep increasing until the maximum width specified. This brought some fluidity on the page that works even when a user is changing the size of a window.

Figure 13, A screenshot of the updated responsive page showing how the content width fills to 100% for lower resolution devices.

Using the max-width command with width percentages, we can then apply this to many elements. For example the profile picture on the page fluidly changes its width to ensure it is not too large on the page. The upper limit helps it from scaling too large as could otherwise occur on resolutions with very high resolutions.

Another structural change performed, was maximising the size of the added map to fit the screen width when the screen size was small or medium. This was so that it was easier to see on lower resolutions screen since it was an image which contained text on it (such as street names). I also made changes so that some elements would now stack vertically rather than horizontally, (such as text that normally sat next to the map). The layout was now adapting better to mobile devices, and had become more vertical, meaning it took up more vertical screen space than it did horizontal.

One of the most noticeable things about the updated design in Figure 13 was that the navigation bar was no longer fitting in one line. It was wrapping on to the next line. I went back to look at the top websites from Alexa.com, and realised there was a common trend between most major websites as to how they handled the navigation menu for smaller screen sizes.

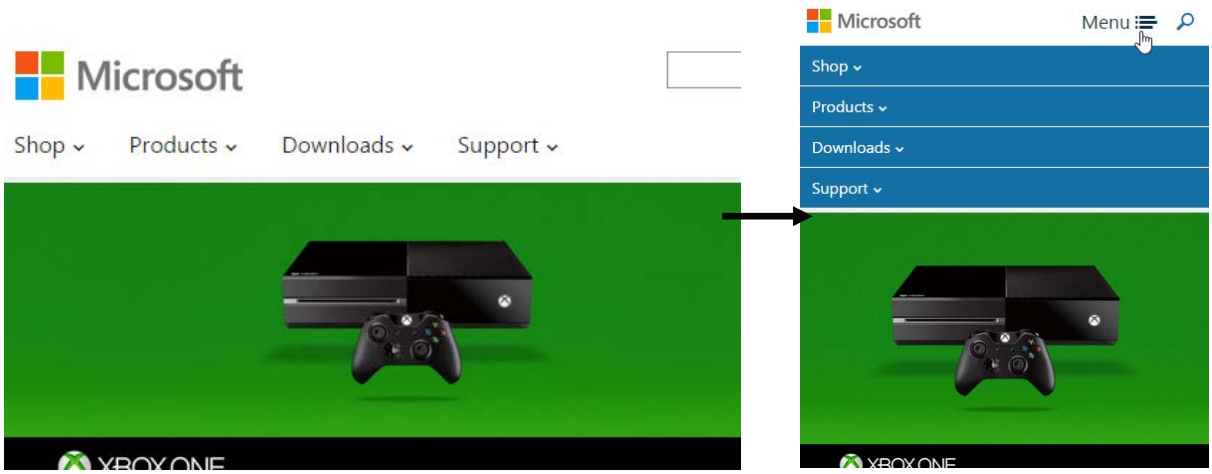


Figure 14, A screenshot of Microsoft.com showing how their horizontal navigation bar changes to a vertical navigation bar for devices with smaller resolutions

Figure 14 shows how Microsoft's navigation menu changes to become vertically stacked when the screen width is reduced. Figure 15 shows Yahoo's mobile navigation bar. It has also changed from a horizontal to a vertical layout, however appears from the side. This trend appeared to be very common in many of the top websites, when viewing their mobile versions. These menus were normally not shown by default however. To access them usually required one click to open these menus, and another click (or moving focus from the menu) in order to shut them.

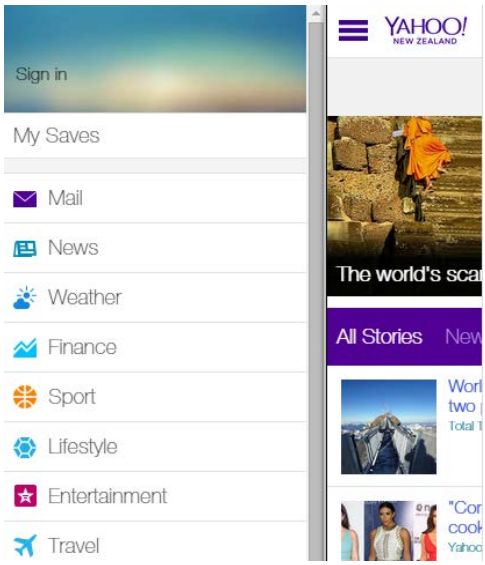


Figure 15, The vertical navigation bar visible at smaller resolutions on Yahoo.com

Thomas Michaund in his book says that horizontal navigation bars when viewed with a phone with a lower resolution did not look as appealing. He also states that a way to resolve this is by using media queries to display a vertical navigation bar instead. He says the use of lines and additional spacing helps make each item in the navigation menu clickable (Michaud, 2013). Kris Hadlock also mentions that when viewing a normal website on a mobile device, it may appear zoomed out. Because of this reason he acknowledges that adding more spacing (padding) can help make an object more "touchable" (Hadlock, 2012). Both Hadlock and Michaund's statements relate to one of the most known laws in human-computer interaction, Fitts's Law. Fitts's Law is a model which predicts that the time required to rapidly move to a target area is depends on the distance to the object and size of it. (Fitts, 1954). Fitts's law also holds well for gestural interfaces, including those that support touch. As mobile displays are smaller than their desktop counterparts, and since websites tend to appear smaller on mobile devices, elements appear smaller. As per Fitts's law, they will be harder to click since they are much smaller. The accuracy of successful presses decreases. By making elements bigger by using padding, we can help make it easier for a user to press buttons on a smaller mobile device.

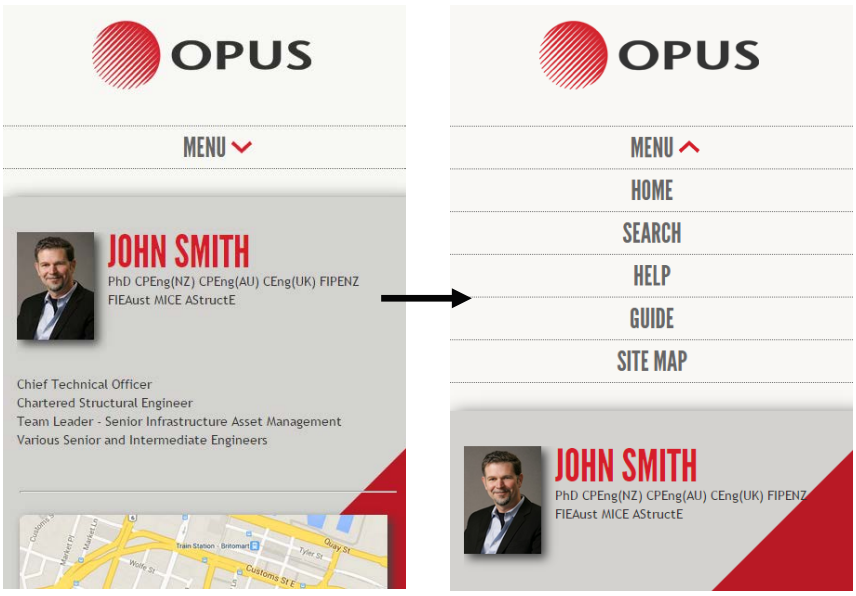


Figure 16, The design of vertical navigation bar used for devices with smaller resolutions. The bar animates and expands upon clicking the menu button, and shuts on pressing it again

I followed the design of the vertical menu, and created a menu that was expanded and contracted upon clicking the menu button. Along with this, additional padding around the text was added to make it easier to press the buttons and to help reduce the chances of mistakes. Dashed borders were also added in order to separate each menu item. Animations were also used as transitions for

when the menu was opening and shutting. These transitions appear help smoothen the process of the expansion and contraction of the navigation menu. The arrow also rotates on press, which acts as another form of feedback to the user as to where they pressed.

Finally upon completing the navigation bar and responsive design and functionality, the design had to be extended to work across multiple browsers. This included adding support for not only desktop browsers, but also mobile browsers render the site correctly. As we cannot anticipate what browser an end user chooses to use, for the design to work, it should be functional and maintain its look and feel across all browsers. This required adding extra CSS rules to ensure that the website appears to be the same when viewed on all browsers. The screenshots in Figure 17 show the same webpage viewed in multiple browsers.

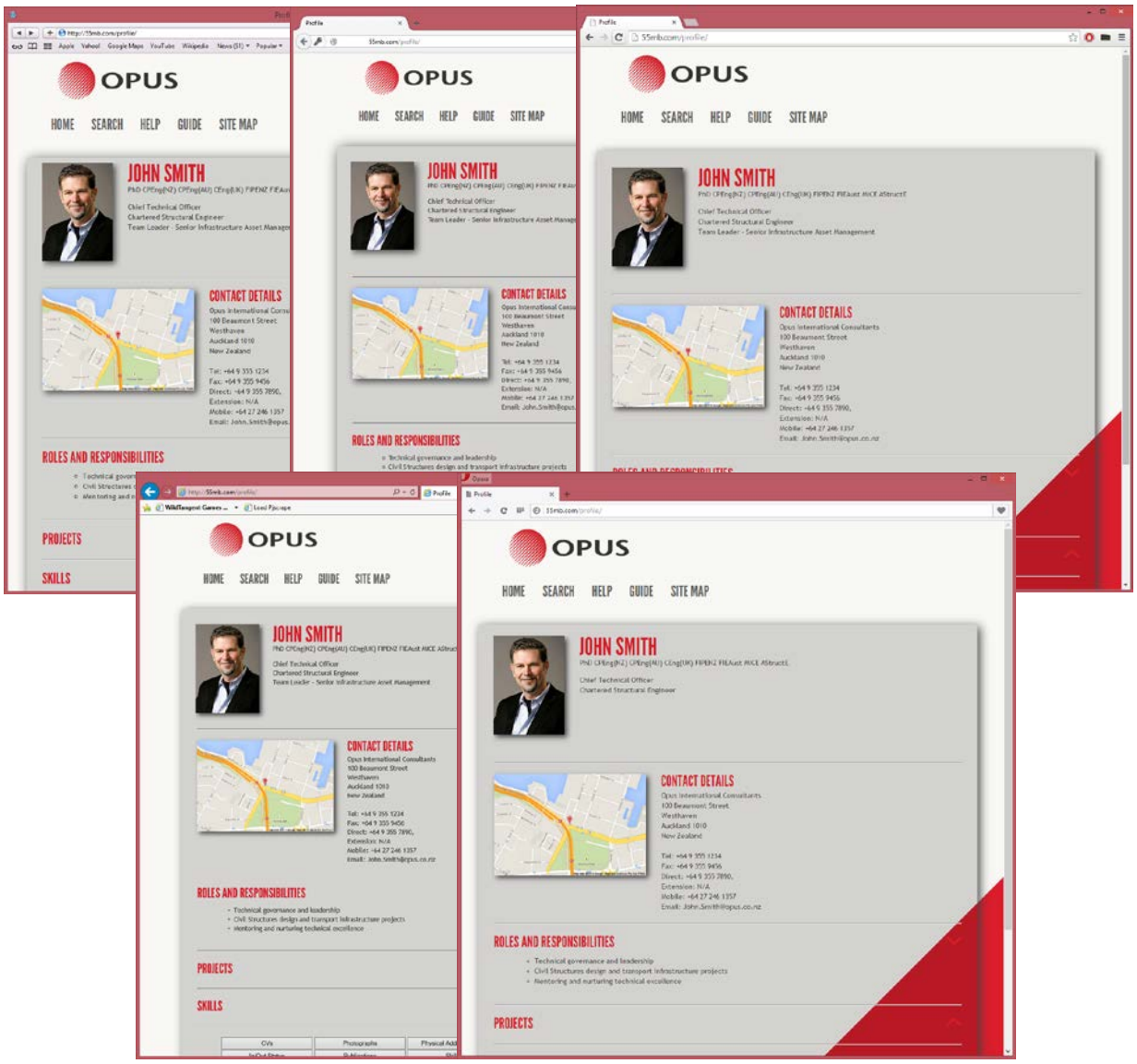


Figure 17, The proposed design of the profile page: Safari 5.1 (top-left), Firefox 30 (top-middle), Google Chrome 36 (top-right), Internet Explorer 9 (bottom-left), Opera 25 (bottom-right).

Implementation and Interactive Design

Throughout the process, an agile methodology was followed when going from ideas to prototypes. This meant creating rapid prototypes and gathering continuous feedback on them. I would go to meet my industry supervisor almost every week to find out what was needed. Upon gathering that information I would search for ways to accomplish what my supervisor wants, and try and find solutions to those problems myself. I did not create many mock-ups or sketches. Instead I would make something that I felt best fitted what my supervisor wanted and would create a quick semi/fully-functional prototype of it. During my next meeting I would then show this prototype to my supervisor and gain feedback on how to improve it. This process repeated over several iterations. I would also not be working on a single piece of functionality at once; generally I was working on two-three different parts of the project at a single time focusing on one the most, while doing minor revisions on the others. As mentioned earlier, there were some communication issues I faced with the IT team. However over a call, my supervisor and I did find out, that the backend system possibly uses PHP. So this was the justification to use it throughout the project. JavaScript was used due to its portability and functionality.

Map Overview for People and Projects:

Introduction

One of the new functionalities to add to the intranet system was a mapping application that could show the locations of all the staff at Opus, and locations of all the projects taken by Opus. The requirements were that initially the map would show a list of all users and projects that exist in Opus's database, but to then have the ability to filter by certain criteria. The main filtering function was to be added by integrating a skills matrix. This skills matrix contained skills that certain users specialised in, such as "Asset Management" or "Enterprise Systems". By selecting a set of skills, the map should be able to show the subset of all users who are skilled in that area. The point of this mapping application was to view not just the searched user's location, but to also get an overview of all staff and projects that exist in the company. Along with this, a greater level of interactivity and granularity was required, such as filtering down to certain groups of users and projects, and also viewing more detailed information about a user or project.

Because of this requirement of filters and viewing a large set of data on a map, the map had to be highly interactive. So instead of trying to integrate this functionality with the existing profile page, it was decided that a standalone application would be a better choice due to the high level of functionality and user-control required. This standalone mapping application was to operate on its own separate webpage, but it would use the same databases used for maintaining user's profiles, and project information, so updating information (such as the users location) on the users profile page would reflect the changes on this map.

Whilst the embedded map only required the ability to view a single marker on a static map, plus your own location, this more extensive standalone map application required the following features.

- The ability to view multiple users and their location on the map.
- The ability to view multiple projects and their location on the map.
- The ability to get more information about a user by clicking on the appropriate user marker on the map. This information should contain their name, contact information, and a photo.
- The ability to get more information about a project by clicking on the appropriate project marker on the map. This information should contain the project title, the client, and the date of creation.
- The ability to view groups of users at one time. Users need to be grouped by skills and the ability to view only a group of users with certain skills needs to be implemented.
- The ability to search for a user or project and show the location on the map.
- The ability to search by for a group of users by their skill, and show only those users on the map.
- Clustering map markers together that are in a near radius. Being able to click on a cluster and view more information about that group of people or project.
- Geocoding addresses to latitude and longitudes.

Comparison of mapping software:

There are many mapping services available to use from the internet such as OpenStreetMaps, Bing maps and Google maps. When it came to a comparison between such services, Cipeluch et al., found no clear winner between either of the services and stated it was down to one's personal requirements (Cipeluch, Jacob, Winstanley, & Mooney, 2011). After researching online I created a table showing the features required for the project along with what each Vendor provides.

	Google Maps	Bing Maps	OpenStreetMaps
Clustering of Map Markers	Yes with plugin	Yes with plugin	Yes with plugin
Gecoding of Information	Yes	Yes	Yes
Group markers by category	Yes	Yes	Yes
Free API	Yes	Yes	Yes
Routing between two points	Yes	Yes	Yes with plugin
Supports static map images	Yes	Yes	Yes

Table 1, A comparison of desktop mapping applications

Table 1 above summarizes the functionality available between different mapping platforms. All three services offered all the features and functionality that were required for this project. After using all the three different mapping applications, I also found them to be quite similar

in terms of aesthetics and functionality. The final decision was made to go with Google Maps due to the fact that it was the most popular choice for mapping software used online. Not only is Google Maps the most popular standalone mapping application, but it is also the most popular embedded map used online, with a market share of 68% of all mapping application users (BuiltWith, 2014). Therefore with such a majority, it made sense to go with the mapping software which users would be most familiar with. This probability shows a user would most likely have used Google Maps either on the Google Maps website or Google Maps embedded in another website. This past-experience (if any) could help users to recognize the tools and controls used to operate the map, which in turn could hopefully make the map more familiar, recognisable and easier to use (as compared to any other competitor mapping application). Secondly the API has a free limit of 25,000 map loads per day which should be sufficient for a company the size of Opus which has around 3000 staff, (Google, 2014).

Implementation

A new webpage was created with Google Maps as the choice of mapping application. Compared to the maps used for the profile page (which is discussed later), since this map was meant to work standalone, it was created to take up 100% of the width and height of the browser's window. A PHP connection was made to the database to retrieve a list of all users and projects. The results were then split and placed into two separate arrays, one for people and one for projects. Since a copy of the existing projects or users database could not be retrieved from Opus, dummy data was used for the project details and the same dummy data used in the profile sections was used again. The dummy data I had created contained 25 users and 10 projects.

Looping through both of these arrays, the location information used to create map makers for each user and project. A red marker indicating a person, whilst a grey marker indicating a project. Using the longitudes and latitudes, the marker was then added to map at the right position. Although all the markers were visible on the map, as anticipated there was a lot of overlapping taking place between markers. This follows a real life scenario where they may be multiple users located at the same office. Since they were all stacked on top of each other it was impossible to tell how many users were at that location.

Marker Clustering

The proposed solution to above problem was to implement a form of marker clustering. What this meant was for example, as opposed to having 20 markers in the same position, rather show one big marker with the number 20 written on it. This clustering would need to be separate for both people and projects.

Google Maps does not come with a standard method to cluster markers together. However there is an open source utility library for Google Maps called "MarkerClustererPlus". Using this library we can get our desired result of showing one big cluster as opposed to many small markers in one place. After importing the plugin files the usage was as follows:

```
cluster = new MarkerClusterer(map, markersArray, options);
```

It was immediately clear that there would need to be two instances of the clustering happening at once. One for clustering people together, and one for clustering projects together.

```
peopleCluster = new MarkerClusterer(map, usersArray, peopleOptions);
projectCluster = new MarkerClusterer(map, projectArray,
projectOptions);
```

Although this solution works at first sight, both clusters are working independently of each other. This was an issue.

Take the following example where there are 3 clusters (A,B,C) created by the people instance of MarkerClusterer, and 5 clusters (D,E,F,G,H) created by the project instance of MarkerClusterer. Because of how clustering works, all the clusters created by one instance of MarkerClusterer will not overlap between themselves. So none of the user clusters will overlap with any other user clusters (A,B,C will never overlap between themselves). And none of the project clusters will overlap with any other project clusters (D,E,F,G,H will never overlap between themselves). However because we are running 2 instances of MarkerClusterer, it is possible that clusters from each instance may overlap with clusters from another instance. This is shown in the Figure 18 with clusters A and F. If both instances are run together, one of the two clusters, either A or F will completely cover the other one so it will not be visible and clickable.

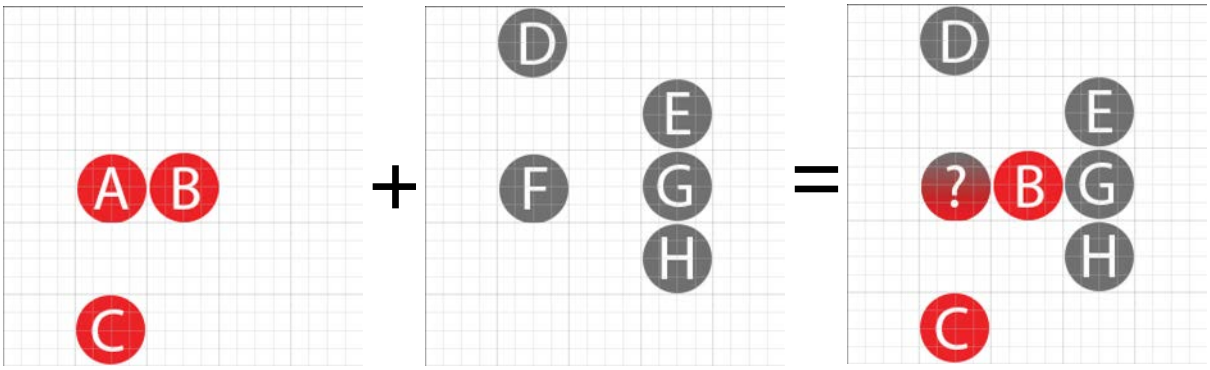


Figure 18, The red cluster of people, plus the grey cluster of projects added together will cause overlapping between A and F

This means in certain scenarios, if a group of users was nearby a group of projects then the projects and people clusters themselves could overlap each other. Reading the documentation and other online resources shows that the MarkerClustererPlus library is only meant to be running at one instance at a time (Google, 2014). In order to stop overlapping between clusters would require modifying the library itself, but despite rigorous searching there was no one who had shared any method as to how to do so online. One of the ways to do this could have been to let both cluster instances “talk” to each other and check if there were any two overlapping clusters by comparing X and Y values plus the radius of the cluster. However this was not only a difficult solution, but because the library was meant for only one instance of the cluster, after drawing one instance of the cluster on screen, the new cluster would replace the data of the other cluster instance as they were using the same resources and variables. Without trying to heavily modify the library, a partly working solution I came up with was to offset both the project clusters and people clusters in opposite directions.

One of the features of the library is that as there are more markers grouped in a cluster, the size of a cluster grows. The size of the cluster will either be 35px (small), 45px (medium) or 55px (large). This is customizable as per ones requirements. For the worst case scenario we will assume all clusters are the largest size, 55px. Clusters always sit in the centre of their grid as shown in the figures above (Figure 18).

If our clusters overlap by less than 50% then the number on the cluster will still be visible. So to do this we will offset clusters by a certain amount, 25% of the diameter of the cluster. User clusters will now always be drawn slightly northwest↖ by about 25%, whilst the projects cluster would be drawn southeast↘ by about 25%.

$$55px * 0.25 \approx 14px$$

Therefore we need to offset our clusters by 14px in diagonally opposite directions. A diagonal offset is simply a combination of a vertical and horizontal offset. To calculate what the horizontal and vertical offset would be we can use Pythagoras's theorem. To maximize the diagonal offset, the angle we shift the clusters, is 45 degrees. Therefore A and B in Pythagoras's theorem are equal.

$$\begin{aligned} C &\approx 14px \\ A^2 + B^2 &= C^2 \\ A^2 + B^2 &= 14^2 \\ A^2 + B^2 &= 196 \\ A &= B \\ A^2 + A^2 &= 196 \\ 2A^2 &= 196 \\ A^2 &= 98 \\ A &\approx 10px \end{aligned}$$

This means we have to shift our cluster 10px horizontally and 10px vertically in opposite directions for each cluster. So the user clusters were all shifted 10px horizontally and 10px vertically, and the project clusters were shifted -10px horizontally and -10px vertically. This would mean although the location accuracy of the cluster is slightly off, if two clusters were in the same grid they would no longer overlap due to their offsets. The figure below shows how the clusters are affected by offsetting them.

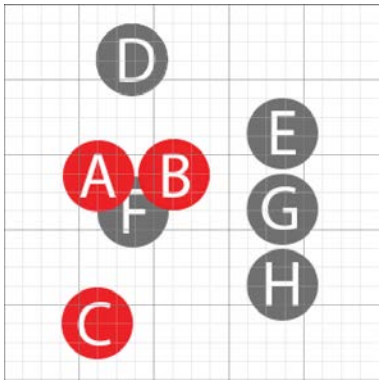


Figure 19, The combination of two cluster instances after offsetting each cluster. Notice how despite partial overlapping the text is still readable

Each cluster has moved 14px away from their original positions in perfectly opposite directions. So even if two clusters were originally in the exact same position, their centres are now 28px apart. This would mean they are less than 50% overlapping. If we also increase the grid size, then this can help achieve even lower overlapping percentages at the cost of larger sized clusters. The method to work out what the offset values should be set to can be generalized to the formula below. The formula guarantees a maximum overlap of 50% for two clusters that are at the same position. *m* should be the size of your biggest cluster in pixels, whilst the result, *x*

is the value of the offset you should set both horizontally and vertically. g is the grid size you set for the map clusters.

$$x = \frac{m}{4\sqrt{2}} + 1 \qquad\qquad g = m + x$$

I modified the plugin so offset values could be set as an option when creating the instance of the MarkerClusterer. They then get added or subtracted from the original clusters centre position. I called these the offset values.

```
pos.x -= this.offsetX_; //Offset cluster horizontally. Default is 0.
pos.y -= this.offsetY_; //Offset cluster vertically. Default is 0.
```

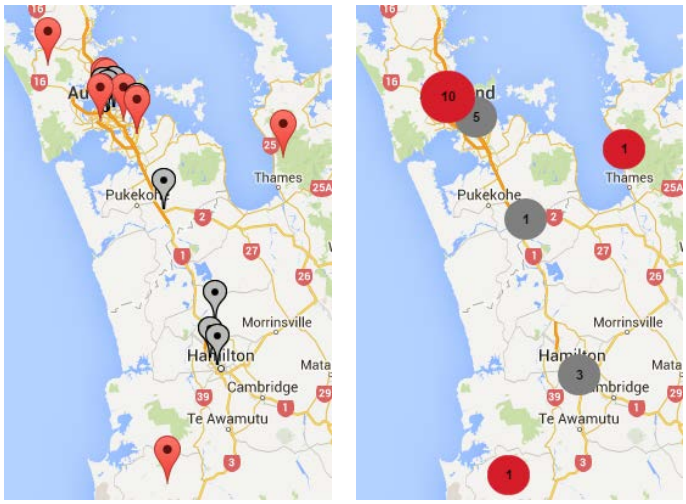


Figure 20, No marker clustering (left) vs. marker clustering (right)

A better solution for the future would be to implement true non-overlapping clusters possibly by modifying the plugin to detect for cluster overlaps.

The next change to the library was that the by default, the MarkerClusterer library zoomed in on the cluster when clicking on a cluster. It did not support events for showing information when clicking a cluster or for handling your own cluster click events. The idea that I wanted to implement was that if there were many clusters, then to be able to click one and then view the individual people or projects made up of that cluster in some form of popup or infowindow. I did manage to implement this. The pseudocode has been shown below for simplicity reasons.

```
If a cluster is clicked, trigger a Cluster click event

Cluster click event:
      String s;
      if the map is zoomed out past the zoom level set by the user
            zoom in
      else if the map is already zoomed in past set zoom level then
            for each marker in the clicked cluster
                  get the "simple" value for this marker and
                  concatenate it to string s
             end for
            create an infowindow popup for the cluster
            output string s as HTML in a new infowindow
            close all other previously opened infowindows
            add this infowindow to the map
      end if
```

The code above shows the changes made to the plugin to allow it to display info windows upon clicking a cluster. The "simple" attribute had to also be added to each map marker upon initialization. This simple attribute contains the formatted output HTML to display in the

infowindow. If the marker is a person marker, then the simple attribute contains the image of the person, their full name, their location and the onClick action if someone clicks it. For a project it contains the project title, the project location and the onClick action if someone clicks it

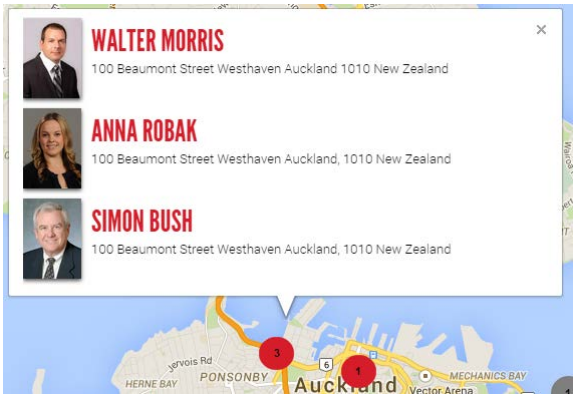


Figure 21, The infowindow when clicking a cluster

Now that a cluster can be clicked, and information is shown about the group of people or project above that cluster in an infowindow (Figure 21), the next step is to handle what happens if a person clicks one of the displayed people or projects in that infowindow. As mentioned above an onClick handler event was added to each marker. Clicking a person should show more information about that person, and also show their location on the map. Clicking a project should show more information about that project and the project's location on the map. In order to differentiate between both a person and project, a "profile card" was overlaid above the map that will show information about a selected person, whilst a sidebar was added to show information about a project (Figure 22). This means a person can view the full details of 1 user and 1 project at any given time. This separation between users and project seemed to be a viable option, as a common scenario could be where someone views information about a user and then views information about a project by that user, without wanting to lose that users information. Another scenario is where someone may be look up a project and want to see users of that project without losing information about the project.

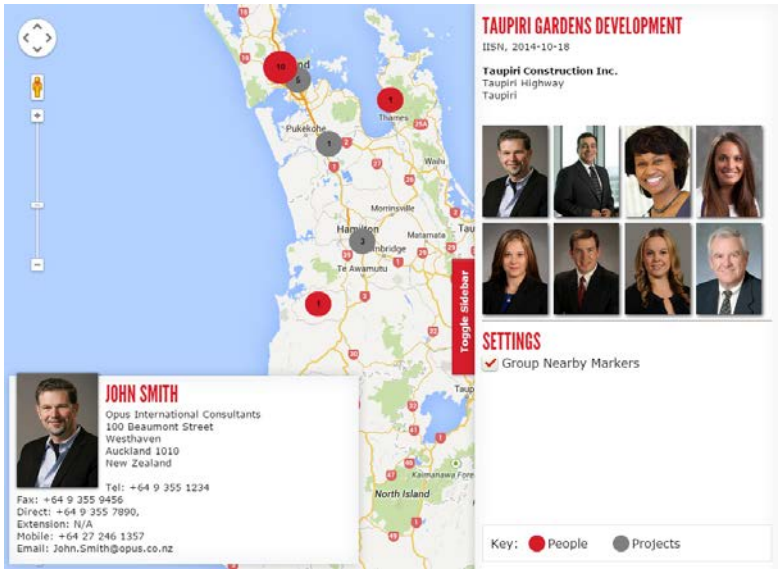


Figure 22, The mapping application showing the profile card to view user information (bottom left) and the sidebar to view project information (right)

Clicking a user updates the profile card, which then shows the users name, image, contact information, location information and PIN group. This profile card can be dragged around the map. By moving the mouse over the user's image, the cursor changes to the draggable cursor giving feedback to the user they can move it around. jQuery was used in order to achieve this functionality using jQuery-UI's draggable method as follows:

```
$("#over_map").draggable({handle: ".avatar"});
```

The "over_map" was the DIV identity of the profile card, while the handle "avatar", was the users image, thus only allowing the profile card to be moved around by dragging the users

image. The reason for this was that by allowing a user to drag the whole card itself, this became annoying, as by default it overrides the text-selection feature. This meant it became almost impossible to select text with the mouse (if supposing you wish to copy text) without dragging the profile card.

Clicking a project updates the sidebar, which then shows the project name, image, contact information, location information and PIN group. The sidebar is opened by default, and the opening animation (done using jQuery) is played initially upon loading the map. This initial animation is done to indirectly inform the user that the sidebar can be expanded and contracted, and that it is not hidden by default. Apart from showing the project information, a call is also made to the database using AJAX about which users are a part of this project. Because of the many to many relationship between projects and users (many users can be assigned to many projects), the database is structured so that the projects and users table has a junction table that assigns project ID's to users ID's. This combination of projectID's and userID's is unique and forms a composite primary key.

The AJAX response returns the list of users assigned to that project, which are then displayed with CSS hover effects to display the name of the user on hover. This hover gives an instant preview about the user. Clicking any one of these users sets the user profile card to be information about that user.

Until now, marker clustering has been completed. However it is possible that the end user may not want to enable marker clustering and would rather see the overview of markers. For this scenario the sidebar was utilized again with settings to enable or disable marker clustering. A settings section was added with a checkbox to disable or enable clustering (Figure 22). Using jQuery we can check the status of the checkbox, whether it is checked (true), or unchecked (false).

```
if($('#cluster-checkbox').is(":checked"));
```

jQuery also provides an event listener that triggers if the state of the checkbox is changed. Upon changing the state, the clustering can be shown or hidden

```
$('#cluster-checkbox').change(function(){});
```

However there is no standard way to enable or disable marker clustering. Individual markers can be set to be visible or invisible; however the MarkerClusterer library does not have the same support for clusters. The MarkerClusterer library has a setting that allows you to set the maximum zoom level for when clusters are visible. The work around here is by setting this value to null or -1, since the zoom level will always be greater than -1, the clusters will not cluster, which is the equivalent of the clusters not being shown. To show them simply set the zoom level back to the default value. A legend was also added to explain to the user that the red markers are for people and grey is for projects. Toggling the clusters visibility will also change the images on the legend to be either a marker or a cluster. This is again done whenever the checkbox state changes, and is set using jQuery

In the scenario where marker clustering has been turned off, clicking on a normal marker (not a cluster) will resort to the default actions. If the clicked marker is a user marker the profile card will be set or if the marker is a project marker then the project information and project

users in the sidebar will be set. This default marker click handling is not handled by the cluster click event, but is instead handled by the native Google maps marker click event.

Universal Search

The last part of functionality that needs to be added to this map is the search function. The search functions should allow us to filter down what we can currently see on the map. By default all the project and user markers are shown.

Some examples of searches that we wish to see implemented are:

- Search for a user and view their information and location on the map.
- Search for a project and view the project information and location on the map.
- Search by a user's skill area, and find other users with the same skill.
- Search for a project and find other users involved in that project.
- The standard map functionality of searching for a normal address.

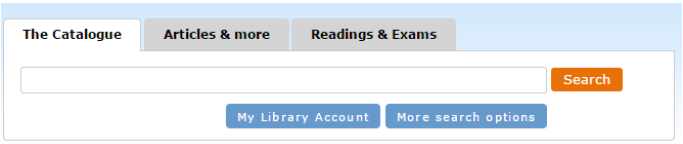


Figure 23, Three separate search boxes separated by tabs on the University of Auckland's Library search (http://library.auckland.ac.nz)

The simplest option was to have separate search boxes for projects, users, skills and address searches. However having four search boxes may be confusing to the users as to which box is for which. Secondly it would take up an excessive amount of room on the page. Figure 23 shows The University of Auckland's library search page. Although it has three separate search boxes for three different types of searches, it has each box on a separate tab so a user only sees one at a time. This is a better solution than having three individual search boxes visible at the same time as you are not losing out on screen real estate, however there is the additional cost of a click to switch between tabs.

Figure 24 shows IMDb and Amazon have similar search concepts to each other. The search automatically searches multiple types of data such as Actors and Movies for IMDb, and for Amazon, it searches multiple categories of products. The search also has a drop-down box if a user wishes to specify what they want to search. Amazon's search function has the added benefit over IMDb's search, as next to each result it shows what category the result is from such (as seen in Figure 24) where you can see "All Departments" and "Toys & Games". The IMDb search however lacks this tagging. On first sight it is very hard to differentiate between what results are Actors and

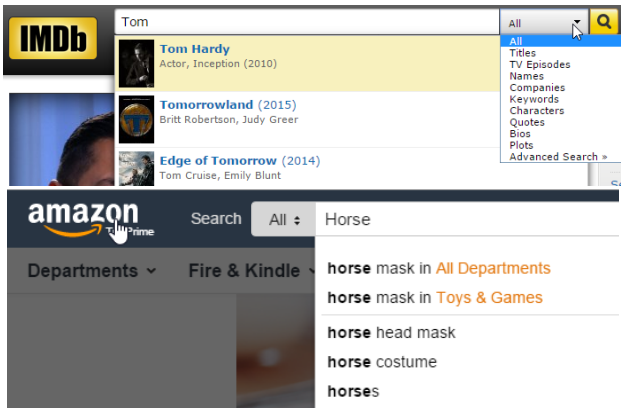


Figure 24, IMDb.com search bar (top) and Amazon.com search bar (bottom). Both also contain a dropdown for more filtering

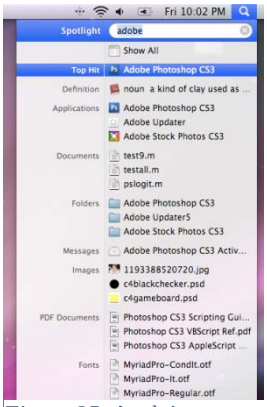


Figure 25, Apple's spotlight search for Mac

what results are movies. Having a dropdown box however, has the cost of two additional clicks, along with the time spent by the user reading and selecting the options, however it could save the user time when selecting between results.

The third search option is what is used by Apple's spotlight search and Facebook's graph search. There is only one search box; however this search box searches across multiple tables for different types of data. As with Apple's spotlight search in Figure 25, searching "Adobe" shows multiple result types, and an icon and text next to each result is used to distinguish what the result is e.g. document, image or application. Other than selecting the result no additional clicks are needed. The user also has flexibility as to how they understand the meaning of the search. They can read the result type text, or they can look at the icons.

Facebook (Figure 26) also uses icons with its search results to distinguish result types such as groups, pages and people. Facebook also appends text to the search results to help understand the meaning of the search. As shown in the figure, searching "St" showed a search option "Find all groups named St" and "Find all people name St". This additional text helps the user understand what it is they are searching for. Along with this, their top results were different results type. A group, a person and a page were the top 3 items. Just like Apple's spotlight search the icons can be used to help the user understand the meaning of the search, with the appended text also helping.

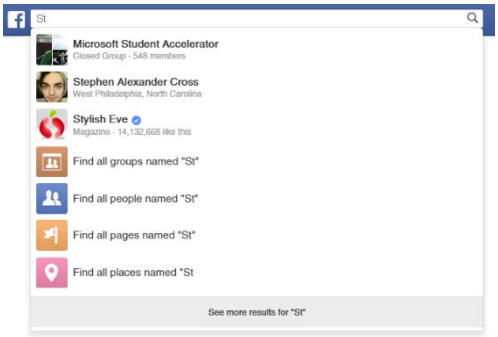


Figure 26, Facebook.com's universal graph search

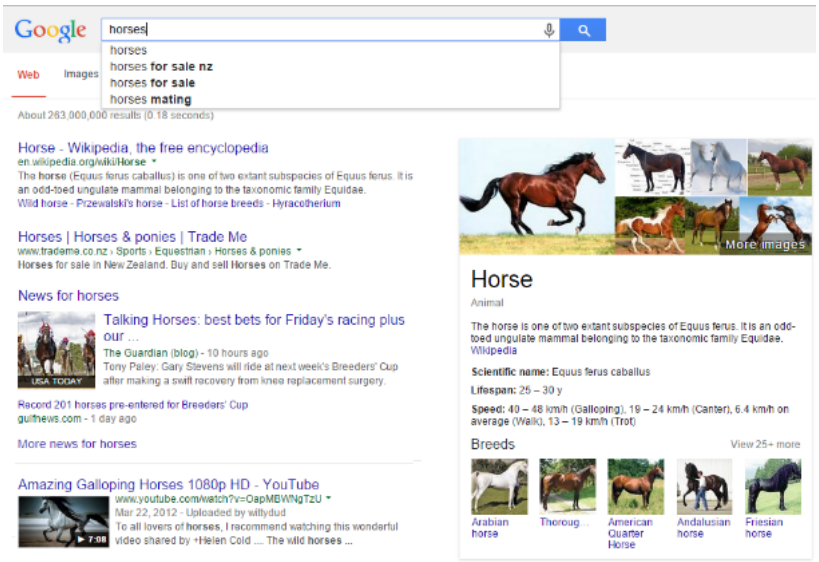


Figure 27, Google's search results

Other than autocomplete, Google's dropdown box does not adapt to the users search query. Instead Google uses the results page, that shows multiple result types on the same page which and uses the layout to distinguish between them. Links to webpages are standard, news results are also included in their own section, videos have thumbnails and duration and there is also a preview for image results on the right hand side (Figure 27). This option gives a lot of information to the user, at the cost of requiring a lot of screen real estate. This works for Google since the application's core functionality is search. For the mapping application however, most of the screen real estate will be dedicated to the map as it is the core feature, not search.

All search boxes, had their pros and cons. The implementation that is being used for this Mapping application is a style similar to Facebook's graph search and Apple's spotlight search. This involves having an auto-complete box which searches multiple tables, and divides results into types using different icons and additional text.

One of the common themes between all the above search figures, except the Apple one, was the use of larger than normal textbox sizes with large legible fonts. A second trend was the use of placeholder text (default grey text), that sits inside a textbox that helps indicate what the function of the search box is for. Since there is no other information about the search, inside the search box, I am showing all the options a user can search for. We would like to auto-complete results for the user as they type. In order to do this we need to attach an event listener to whenever the users presses a key using jQuery.

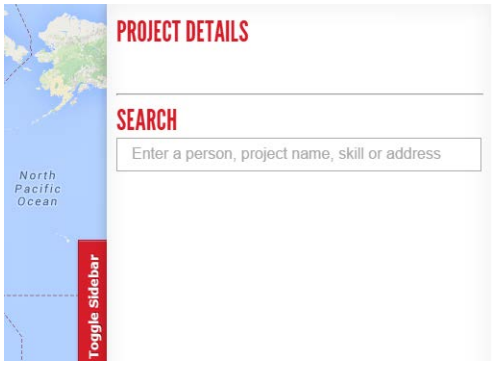


Figure 28, The design of the maps search-box

```
$('#filterfield').keyup(function(){
      $.post('get.php',function(value){
            $('#tab').html(value);
      });
});
```

What this does, is upon a user entering a key, the current entered text is posted to the get.php page using AJAX. This get.php page is what creates a persistent connection with the MySQL database, executes the query and formats and sends back results via an AJAX call-back.

This get.php page needs to send back different types of results, such as people, projects and skills. In order to display multiple results from different table we need to search multiple tables. My SQL offers a UNION method, which allows you to combine the result of two or more SQL queries. Using this we can now combine three SELECT queries for the people table, the skills table and the projects table using 1 SQL query.

```
(SELECT CONCAT(firstname, ' ', lastname) AS result, 'person' AS type
      FROM users
      WHERE firstname LIKE '%$data%' OR
      lastname LIKE '%$data%')
UNION (SELECT...'project' AS type FROM projects...WHERE...LIKE...)
UNION (SELECT...'skill' AS type FROM skills ... WHERE...LIKE...)
```

The above code consists of 3 SQL statements combined into one. The first statement is concatenating the first name and last name together which will be the output result. We are also adding a "type" row with every result. This type can be a person, project or skill. This will help us identify the row type later. SQL's LIKE statement is used to search for a pattern in a column. We are performing a check on the first row to see if the search query text exists in the anyone's first name. For example searching "sh" will be true for "Shane" and also true for "Ashley". The second LIKE repeats the process but for last names. This pattern matching is repeated in two more select statements, but for projects and skills

Multiple rows of data have now been received, however the order is random. We want to give the user the most relevant results at the top, and the least relevant results at the bottom. We need some way to weight and rank our results by some priority level. To do this I created a simple algorithm that detects where the user query string is in the result.

Example searching for "St":

1. If the query string is an exact match. e.g. "**St**"
2. If the query string is at the front of the result. e.g. **St**eve Smith", "**St**ain Removal"
3. If the query string is at the front of any word of the result. "John **St**anley", "Cleaning **St**airs"
4. If the query string is anywhere in the string. E.g. "Kyle A**st**ley", "We**st** Park Development"

There is also the scenario where there is a draw. If two result types, both have priority level 1, then users take preference over projects, and projects takes preference over skills. This simply acted as a tiebreaker; however there are better solutions as to how to break such ties.

The way this is implemented in SQL is by using WHEN which is a conditional clause, CASE and ORDER BY. $data is the query string.

```
ORDER BY CASE
      WHEN type = "person" THEN CASE
            WHEN result LIKE '$data' THEN 0
            WHEN result LIKE '$data%' THEN 4
            WHEN result LIKE '_% $data%' THEN 7
            ELSE 10
      END
      WHEN type = "project" THEN CASE
            WHEN result LIKE '$data' THEN 1
            WHEN result LIKE '$data%' THEN 5
            WHEN result LIKE '_% $data%' THEN 8
            ELSE 11
      END
      WHEN type = "skill" THEN CASE
            WHEN result LIKE '$data' THEN 2
            WHEN result LIKE '$data%' THEN 5
            WHEN result LIKE '_% $data%' THEN 9
      END
END ASC
```

One such possibility is by outputting words where the percentage of the matched word is higher. E.g. "**St**ar" gets more priority over "**St**ainless", as more of the word has been matched. Another tiebreaker can be accomplished by keeping a counter when a search has occurred and serving the most popular results first, or if possible, then by looking at a user's past searches basing it on the frequency of the type of search e.g. projects, people or skills. This second option however may have to overcome some privacy issues.

Figure 29, The four search result icons representing: users, users, projects, skills and locations

So after the results have been ordered, the get.php page formats these results correctly into HTML, by adding the HTML containers, displaying the correct icon for each result type (Figure 29), appending on the correct text, and handling what clicking each row now does. So whenever a user enters a new letter, the keyup function causes this 3-in-1 query to be called. The get.php page formats the data and sends it back via the AJAX call-back to be

outputted inside a DIV using the jQuery html() command. This is as a simple method to create an autocomplete box.

Apart from all of this, the main functionality of any mapping application which is searching for a location and displaying it had not been added yet. This location search functionality must also be added. The way this works is a human-readable address needs to be converted to a latitude and longitude. A map marker is then created at the location of the longitude and latitude. This conversion can be done using Google's geocoding service as shown below.

```
geocoder = new google.maps.Geocoder();
geocoder.geocode({
      'address': address
}, function(results, status) {
      if (status == google.maps.GeocoderStatus.OK) {
            //create new map marker and display on map
      }});
```

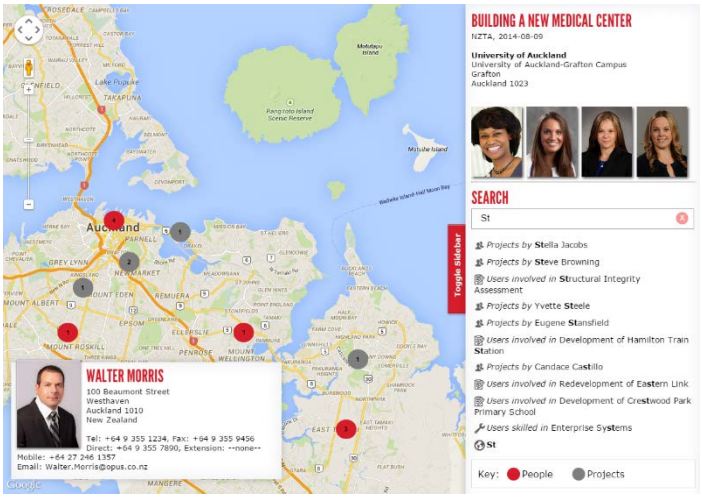


Figure 30, The final mapping application showing search functionality with a variety of result types with their respective icons

At the bottom of each set of results, another row is appended on with the location icon and the text the user searched. This data is not compared to any row in the database; therefore this row will always be visible in the search query even if there are no other matches from projects, skills or people. Clicking this, will make a call to Google Maps geocoder and convert the query data string to an address. The geocoder will try converting the query string to a latitude and longitude, and will pan to that location on the map.

Profile & Profile Management:

Introduction

Staff members were not maintaining their profile pages. Some of the information visible on the profile page is the user's name, qualifications, job titles, address, phone numbers, email and roles and responsibilities. Some staff members did not have profile pictures, some members had very out-of-date information on their profile page, and some barely had any information at all. Staff members were also not keeping their physical locations up to date.

The profile page had to be modified and redesigned in order to make it easy for a user to maintain their profile. Although a simpler design will not guarantee users will keep their profile updated, it may act as an incentive to do so due to a simplified process. Secondly after speaking with some staff, one of the reasons I discovered as to why they were not keeping their profile updated was because they simply found it too hard of a task to do. They therefore could not be bothered to do it, or want to learn how to do so.

As covered in the design section above, an overall change to the theme was performed. The visual changes made were not drastically different in order to make the new proposed system still appear recognisable to previous users. A system which is recognisable will be easier for a user to use and adapt to rather than a system that is unrecognisable. These visual changes were more to bring a clean look, to reduce clutter and to make the system appear more aesthetically pleasing rather than to be a solution as to why users were not maintaining their profiles

Research

Upon further investigation, I tried to find out why profile management was not being done and why it was such a challenge. I realised the issue may be due to the page that is used to edit a profile was overly complicated. Figure 31 shows a screenshot from the edit my profile page. There are an excessive number of fields and the overall layout is very crammed and displeasing. I felt this would be one of the best places to start, since a user cannot maintain their profile, if they don't know how to edit it.

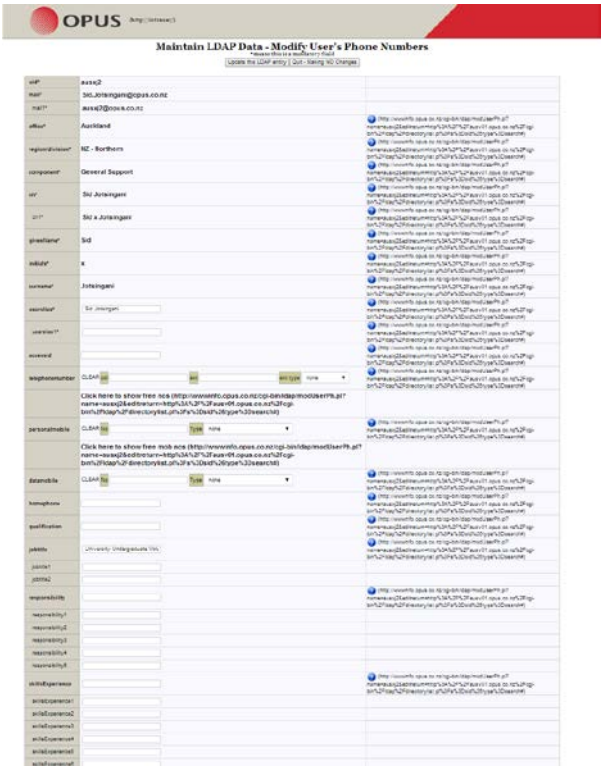


Figure 31, A screenshot from the profile edit page

There were a significant amount of textboxes on the page and all the input fields for text were only textboxes. Dropdown boxes were not being used anywhere. For data validation and data aggregation, this is quite bad. One user may call themself a *Structural Engineer* whilst another may say *Engineer – Structural.* Although they both have the same meaning, they will be seen as two different things by the database. So some form validation needs to be implemented. Secondly it is also harder for the user to fill out. A dropdown box gives the user pre-populated options they can choose from, where as a textbox means the user can enter anything they want. A textbox requires the user to think more, and is also more work for the user. Clicking two buttons for a drop down menu is normally much easier than typing in a textbox, however a dropdown box may restrict answers and limits flexibility for the end user if they cannot find what they want in the dropdown box. Also if a dropdown box has too many options, it may take the user a significant amount of time to decide between the options. An alternate to a dropdown box, is an autocomplete box. This is a balanced option that could be used in order to allow users to select an option from a predetermined set of data for better validation, whilst at the same time giving the user flexibility to type what they want if they cannot find what they want in the dropdown list.

I started to look up different options of input boxes that could help with data validation. To achieve autocomplete and responsive input boxes, the choice to go with jQuery since it would be a good choice to use for autocomplete that needs quick, dynamic and responsive results. Plus I was anticipating using a lot of JavaScript and jQuery for some other planned functions

for the profile page. Some research showed that there were a lot of autocomplete/dropdown plugins based in jQuery. I decided to try out some to see which best suited my needs.

There were four types of autocomplete/dropdown menus that I put together on one page (Figure 32). They all had the basic functionality required, but were slightly different such as having different styles or the ability to select multiple items vs. single items. I showed all these four to my supervisor, and there seemed to be an obvious choice between the four. The one selected, named token-input. It had the benefit of being able to hold multiple values and could easily delete any one of the multiple values.

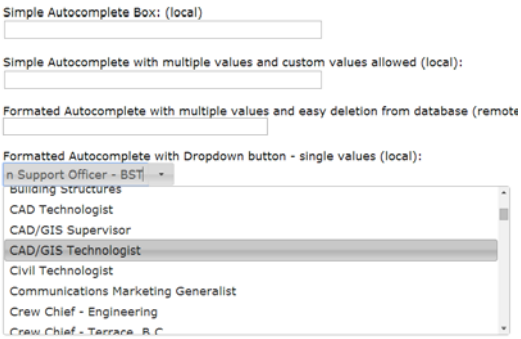


Figure 32, Four of the different auto-complete box plugins that were contemplated between

I had to setup a MySQL database and create a table containing the different types of roles at Opus. The connection was done via PHP. Since I did not have access to actual data, I gathered this list through the list of open jobs on the careers page of the Opus website to use as sample data. Once the table was set up the values were then being fetched from the database and being fed into the autocomplete box. To do this I had to set up a persistent SQL connection with the database. A persistent connection was used over a normal connection as it has the advantage of attempting to find an already open link with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection which saves time and processing power. This is handy for an auto-complete scenario where a query will need to be made every time the user enters a new letter in the input box.

```
mysql_pconnect($host, $user, $password);
mysql_select_db($database);
$query = sprintf("SELECT id, name from roles WHERE name LIKE '%%%s%%'
ORDER BY name DESC", mysql_real_escape_string($_GET["q"]));
$results = mysql_query($query);
```

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	id	int(11)			No	None	AUTO_INCREMENT
2	firstname	text	utf8_unicode_ci		No	None	
3	lastname	text	utf8_unicode_ci		No	None	
4	roles	varchar(255)	utf8_unicode_ci		No	None	
5	contact	text	utf8_unicode_ci		No	None	
6	latitude	float			No	None	
7	longitude	float			No	None	
8	type	int(11)			No	None	
9	image	varchar(255)	utf8_unicode_ci		No	None	
10	responsibilities	varchar(255)	utf8_unicode_ci		No	None	
11	jobtitle	varchar(255)	utf8_unicode_ci		No	None	
12	qualifications	varchar(255)	utf8_unicode_ci		No	None	
13	address	text	utf8_unicode_ci		No	None	

Figure 33, The structure of the "Users" table in the MySQL database

The next part was adding a table for users in the database. I wanted the data shown on the profile page to be data that was dynamically retrieved from the database when the page loads. Again since I have no access to Opus's actual database or servers, this was all done locally including the connection using dummy data. The structure of the table created, was based on the information I could view from the old profile page, and from the old edit my profile page. This helped act as a guide to figure out some of the fields that would exist in the database. Figure 33 shows the structure used for the Users table of the database.

Now all the data displayed on the profile page was being grabbed from the database. I now needed to implement the profile editing page. As Figure 31 showed, the current editing page

had a very complex look and feel to it. So the goal was to create a way to simplify down the editing process to the lowest level possible, to make it significantly easier to edit ones profile. The options were:

- To improve on the existing GUI and use the standard form based layout to edit data. This would mean a user has to navigate to a new page, and they have to find the fields they wish to edit, change the values in that field and hit save. In the user's mind there would be some form of mapping going on such as which field relates to what part of the profile page
- The second option was to implement some form of WYSIWYG editor. This however would not bring any more simplicity than the standard method of editing forms, and a WYSIWYG editor leaves the design decisions up to the end user. For example in a rich text box a user may decide to use a bold green font throughout. This may look unappealing and would lack consistency between profile pages. Plus the layout of information may be in different places since the end user now has all this additional flexibility.
- The third option was the one I implemented. This was a feature I created from scratch which I like to call "double click to edit".

Double Click to Edit Feature

The double click to edit feature, means that a user could double click any text on their profile page, and the text would then be replaced by a textbox/text-area according to what the context of the content was. The user could then enter an updated value and click a save button that would appear next to the textbox. This would require no mental mapping between text-fields and actual text on the page (since it is not requiring the user to navigate to another page). It would also act similar to a limited WYSIWYG editor, except only for changing content and not overall styles.

The feature was initially implemented as being hardcoded, but later this was changed to make it work more like a library/plugin. I created new HTML tags called `<editable></editable>` that wrap around a normal HTML element. The editable tags understand what the context is, by reading the inner HTML of its child nodes (the wrapped content). Depending on what the wrapped element is, the double clicked content will be replaced with an editable version of the element. For example, a list will be converted to a token input box (the jQuery library list box), paragraph text will be replaced with a plain text-area, and an image will open a file picker where a user can select a new image to replace the old one. Examples on how to use the editable tags are shown:

- A single line text area when the content only contains one line
```
<EDITABLE id="role"><p>graduate</p></EDITABLE>
```

- A multiple line text area when the content contains more than one line
```
<EDITABLE id="role"><p>line 1<br/>line 2</p></EDITABLE>
```

- An input field containing list items with when the content contains list items and the attribute autocomplete is set to off.
```
<EDITABLE id="fruit" autocomplete="off">
```

```
  <ul width="50"><li>apple</li><li>pear</li></ul>
</EDITABLE>
```

- A file picker allowing the user to select an image to replace when the content is an image.

```
<EDITABLE id="drinks">
  <img src="http://example.com/image.png" />
</EDITABLE>
```

- Example with default value being retrieved from a database

```
<EDITABLE class="list-auto" id="fruit">
  <UL><?php echo $fruitList;?></UL>
</EDITABLE>
```

- Example with autocomplete enabled. Note the auto-complete has to be setup separately.

```
<EDITABLE class="list-auto" id="drinks" autocomplete="on">
  <UL><?php echo $drinksList;?></UL>
</EDITABLE>
load_autocomplete("#drinks");
```

The editable tags work by getting the inner contents of the double clicked item and figuring out what the content type is e.g. text, list or image. A save button is appended that makes an AJAX call to a PHP page which updates the database with the new values. A persistent database connection is also made if the textbox is supporting autocomplete to get results in real time. If the content is text, then it is replaced by a textbox which grows and shrinks by determining the size of the text and the number of lines the user has entered upon every user input. If the content is a list, then it is replaced by the token-input box and autocomplete functionality is enabled or disabled depending on the autocomplete parameter. If the content is an image then the image is animated to wiggle (to indicate that it is the selected image that will be replaced), and a file picker is opened allowing the user to select a new image to replace it with. The save button updates the value of the content in the database with the new content, and the textbox/token-input box is replaced with the content formatted into HTML.

Figure 34, The double click to edit features showing how textboxes and autocomplete-list boxes are replaced with the text the user wishes to edit

The very simplified jQuery code and pseudocode below shows how the editable tags work in the background.

```
//On double clicking an editable tag
$("editable").dblclick(function(event) {

//If the item is in editable mode already then break here
if($(this).attr('clicked') == 'true'){ return; }

//Else get the inner HTML content of the editable tag
var content = ($(this).html());

//Deduce what the type of content is e.g. Image, list, text
if ($(this).children()[0].tagName == "UL")

//If a list then split the list items from the HTML and
//convert the textbox to a token input
content = content.split("<li>").join("");

//If it is image, then open the file picker and animate the image
$('input[type=file]').trigger('click');

//If it is paragraph text, then replace the content with a textarea
$(this).html("<textarea>" + content + "</textarea>”);

//Append a save button that saves the new content in the database
$(this).append("<input type='save' class=’save’>");
$('.save').on('click', '', function(){//connect to db and update}

//Enable the autocomplete Ajax callback requests to the database
$(this).tokenInput(<?php echo $select_all_rows_from_db;?>, {});

//Detect the lines of text and grow or shrink the textbox if needed
$(this).animate({ height:new+"px" },200);

//On save, submit the updated content via Ajax and replace the steps
$.ajax({ type: "POST", url: "update.php", data: data

//Replace the input boxes back by doing the above steps backwards to
//convert the listbox/texbox objects to HTML code to be displayed.
```

There are currently some limitations in terms of functionality, since the double click to edit feature, still has a lot of room for improvement:

- Every editable tag needs a unique ID which currently has to be the name of the row in the database that the information will be saved in. This is a current limitation, but however in future revisions, can be improved on by using a dictionary or by saving the information about which editable field ID’s relate to which column in the database itself.
- As shown in Figure 34, the contact details are all saved in one row. The telephone, mobile, and fax number would normally be separated as different columns when saved in the database. As I did not have a copy of the database, and thus could not view the structure, this was done for simplicity purposes

and also to act as a concept. These fields could be split up to represent different columns in the future.

- Thirdly there is currently no way to define where the source of the autocomplete is coming from. All input boxes currently share the same source for auto-complete data by default. This in future iterations will need to be solved so that different input boxes can get their data from different sources. A possible solution may be to make every editable input box that requires autocomplete have additional parameters in either jQuery or as HTML attributes where you can define the source of the data.

There are most likely other small things which need to be improved on, along with extending functionality to other HTML elements such as tables, and other media, however this prototype shows the possibility of creating a simplified profile editing system that does not require one to navigate to a new page.

In creating this double-click to edit feature, three other minor features were implemented that may not seem as obvious at first. The first was the animating auto-growing text area.

The first thing we had to detect was when double clicking the content, how big the text area that replaces the content will be. The textarea that replaces the content should be big enough to fit all the content inside of it without scrolling. To do this we the inner HTML and count the line breaks.

```
content = child.innerHTML.split("<br>").join("&#10;");
content = content.replace(new RegExp( "\\n", "g" ),"");
var lines = (content.split("&#10;").length)
```

This appeared to be a very straightforward fix to a simple problem, however another inconvenience appeared. When a user double clicks a paragraph of text the text is replaced by a text-area, as it should be. Although now the text-area created had a height that fit perfectly to the paragraph it replaced, the textarea would not grow if the user wanted to enter more content. When a user entered a new line, the text area size would stay the same, so a vertical scrollbar would appear. Although being a small issue, upon using the double-click to edit feature many times myself, I realised how annoying this could be. In order to solve this problem I looked at a solution of making the textbox grow or shrink depending on whether you add a new line or delete a new line. I thought if the user is entering more information that what was previously in the textbox, then the textarea grows in size, else if the user is deleting lines of content then the textarea shrinks.

At first I thought about adding a key listener whenever the enter button is pressed and to expanding the textbox size, but this did not take into account the text running on a new line by itself, or copying and pasting text inside or going smaller when deleting text. I looked online and saw there were already existing plugins available for similar problems, however for some reason they were all very large in size. *autogrow.js* takes up over 120 lines, *Autogrow-TextArea* was almost 100 lines without any animations and therefore worked very roughly. The most popular solution, Jack Moore's AutoGrow was over 270 lines and again worked very roughly. I felt I could implement this myself in a shorter way, and include animations so it smoothens the transitions of the growing text area so the end user understand what is happening. I implemented my own solution with 12 lines with one statement per line

including the jQuery event handling. It accomplishes the same job in a much simpler way by using some math and matching the special new line characters, to calculate what the height should be based on the font size and the line-height after every key stroke. Upon adding a new line, the textbox animates and "grows", while it "shrinks" on deleting one.

```
var prevTextareaHeight;
$("textarea").keydown(function(e) {
    var counter = 0;
    n= this.value.match(/\n/g) ? this.value.match(/\n/g).length:0;
    lines = content.split('\n');
    for (i = 0; i < lines.length; i++) //
        counter += (Math.floor(lines[i].length / 50));
    textareaHeight = (counter + n) * 20;
    if(textareaHeight!= prevTextareaHeight)
        $(this).animate({ height: textareaHeight +"px" }, 100);
    prevTextareaHeight = textareaHeight;
});
```

The algorithm above counts the number of new lines it sees in the textbox. It then splits each line and sees if the length is more than 50. 50 is the width of the textbox in number of columns. If the length is greater than 50 it means a line is wrapping on to the next line. We do this for every line and add it on to the counter. Finally to detect the height of what the textbox should be, we sum up the number of newline characters, plus the number of lines that are wrapping and multiply this number by 20. 20px is the line height for size 14 font which sits in the textbox. We then animate this change in size over a 100ms delay assuming the height has changed. This is a simple solution for the textbox growing and shrinking problem.

The second feature implemented relates to when the double-click to edit feature when used on an image. When the file chooser window is open, the selected image also wiggles to indicate it is the image being replaced on the page. This helps the user identify the image on the page that is going to be replaced which may come in handy if there are multiple images on the page. The wiggling animation can help prevent errors where a user could replace the wrong image (which also relates to Nielsen's heuristic of helping prevent errors).

This wiggle animation is similar to the animation used on iOS when holding on to an icon to move around. On iOS it means the apps are an editable mode where they can be moved around. For our project it has a similar meaning that the image is and can now me modified. The animation is made using the CSS3 animation feature and is simple to implement, however gives important feedback to the user that can help the user detect a wrong click.

```
@keyframes wiggle{
    0%{-transform:rotate( 2deg);}
   50%{-transform:rotate(-2deg);}
  100%{-transform:rotate( 2deg);}
}
```

We then attach this animation to a class.

```
.shake{
  animation:wiggle .2s infinite;
}
```

To then add this animation to the image, when the image has been double clicked we add this shake class to the image using jQuery:

```
$('#profile-pic').addClass('shake');
```

The image now animates until a new image is selected. After a file is selected the change function will be invoked by jQuery

```
$('input[type=file]').change(function() {......}
```

At which point we can remove the shake class from the profile picture

```
$('#profile-pic').removeClass('shake');
```

This works fine, except in the scenario when a user presses cancel and does not select an image. Neither HTML nor JavaScript provides an event handler or any method to detect when the user presses cancel. After a lot of research I found that there is no straightforward way to detect if the cancel button is pressed. What this means for us is if a user does not select an image, the removeClass function will never be called and the image will continue to keep shaking.

An inelegant work around to this problem however can be done by checking to see when the document gets focus again, after double clicking an image.

```
var img = document.getElementById('profile-pic');
img.ondblclick = charge;

function charge(){
     document.body.onfocus = function () {
          document.body.onfocus = null;
            $(img).removeClass('shake');
      };
}
```

So if the file window is open the focus will be on the file picker and not on the document. Upon the body getting focus again, the shake class is removed and the focus is nulled. This is a very simple work around to emulate the event of the cancel button being pressed.

The third and final minor feature implemented was ensuring that the double-click to edit feature works on mobile phones. Upon first trying to load it on a mobile phone, I realised the feature was not working and instead my web page was being zoomed into. Although pinch-to-zoom is generally associated with zooming, Hinckley & Song from Microsoft research note that pinch-to-zoom is difficult to perform, one-handed. It is for that reason touch-screen devices often use double-tap for one-handed zooming (Hinckley & Song, 2011). I realised that my double-click to edit feature's listener was being overwritten by this standard browser control, and therefore the content was not going into editable mode. In order to resolve this problem, I looked into alternatives that could be used in order to change to editable mode on mobile phones.

One point made by Oh et al., was even when two systems share a common gesture, such as tap-and-hold, the details may be different (e.g., duration of a short vs. long tap) (Oh, Kane, & Findlater, 2013). Tap-and-hold on mobile phones is generally associated with selecting an element or opening up more options/settings about the tapped element. This feature holds true for iOS, Android and Windows Phone. This is a gesture that is recognised by most users. So I added the ability to tap-and-hold on mobile phones to switch to an editable mode. However the hold time was reduced to a smaller amount (250ms) so it does not interfere with the standard controls that popup after a longer period of time on all mobile platforms. This not only gives instant recognisability, but it also was a simple yet universal gesture that appeared to work fine on different mobile platforms.

Embedded Map on the Profile Page

Because Opus has many locations and a lot of projects are on-site, large amounts of travelling are required in some jobs. It was very bad practice if user's physical locations were not kept up to date. It can become very hard to figure out where user is located. The only way to figure this out is to ask around or call the user, but even then time-zones can impact on communication if the other person is overseas.

One of the solutions that was looked into was the creation of a mobile application where one's location is automatically updated and saved on the intranet, or a that allow users to check in to the system. This was ruled out very early due to the constraints of the intranet only being accessible internally and secondly since a mobile application was not part of the scope or requirements for Opus.

Instead my proposed solution was to implement a method to simplify the process of saving one's location, in order to encourage location maintenance. Along with this, the addition of a visual map on the profile page of the intranet, where the location can be viewed rather than just plain text.

The mapping application was to be a small map embedded in the profile page of a user next to their contact address. The picture of the map needed to be a visual representation of the address on the right, so the map needs to grab that contact address. This would act as a visual aid to go along with the text address. For compactness and simplicity I decided to simply display a static image of the user's location on the map. Upon clicking that map, would open a larger HTML5 version of the map with basic interactivity such as panning and zooming along with showing your current location. This map will open as an overlay on the same page in a lightbox/popup box.

This appeared to be a better choice rather than embedding a very tiny interactive map on the page. The reason for this was also to ensure speed and responsiveness when loading the profile page, and secondly to hide all the extra controls bits of functionality (such as the geocoding search) that exist on the map page already. Secondly due to the space constraints, embedding a small interactive map would not serve much purpose due to it being too small and the controls of the map taking up most of the room. So rather than trying to add all

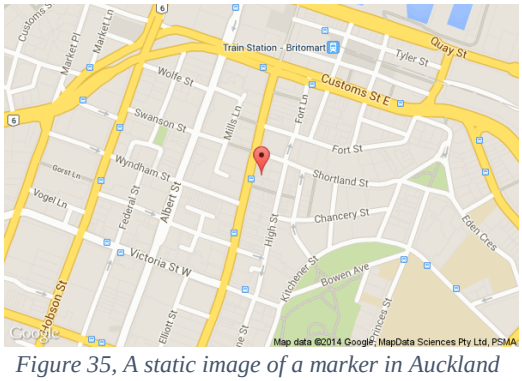


Figure 35, A static image of a marker in Auckland

the functions of the maps page and place it on to the profile page (where it would just take up space and may not even be needed) instead I am just showing a simple image which can be clicked to access the full functionality of the mapping application

If the user wants more information, they can click the image at which point a larger interactive map can overlay above the current window. The cost of this method is an additional click; however we are not overwhelming the user by showing them the additional map functions if they do not require it. We also save on a lot of screen real estate, which we can use for other elements.

The following features were required for this embedded profile map:

- The ability to view a single user's location on the map.
- Converting the map with the user's location into a static screenshot.
- Geocoding addresses to latitude and longitudes.
- Displaying the users own location on a map using HTML5 geo-location tools.

Google Maps was to be used for the same reasons as used in the other standalone maps application (See Page 19 for comparison of mapping software). Google offered all the functionality required, and plus it makes sense to bring consistency by using the same mapping service throughout the intranet system, since we are using it in two different places.

The first step was to show a small map of the user's location next to where their address is shown as text on the profile page. The size of this should be around 1/3rd the total column width, whilst 2/3rd of the space should be kept for the text address. The reason for this is due to wanting the map to serve as a visual aid to the text, and keeping the text the focus rather than the map being the focus. This is a scenario where the text plays a more important part than having an associative image. A specific address etc. "100 Beaumont Street", will give more precise information than a map marker dropped at approximately 100 Beaumont Street at a quick glance. Secondly there could be an unlikely scenario, where it is possible the map may not be able to geo-code the address correctly, and despite the address being the same as the user entered, the location shown on the map may be incorrect. It is for this reason priority and focus should be given directly to what the user inputs.

The way the static image is created is by using the Google Maps Static API. The API works by taking in an address and geocoding it to a latitude and longitude which it displays on the map. The result is an image that can be embedded by using normal HTML `<img>` tags. The code below shows us the call made to Google to retrieve a static image for the address: "100 Beaumont Street, Westhaven, Auckland 1010". Note: to get the address in the right format some whitespace, newlines and special characters need to be removed. Also the URL will need to be encoded before making the request.

```
https://maps.googleapis.com/maps/api/staticmap?
zoom=14&size=300x200&maptype=roadmap&markers=color:red
|address:100%20Beaumont%20Street%20Westhaven%20Auckland%201010%20New
%20Zealand
```

So when loading a profile page, we retrieve the location information from the MySQL Database to display this location as text. Along with this, we now also make a call to this

static maps API with the location details of the user. The address is sent with other parameters such as output image size and the level of zoom. An image is returned of a map with a marker at the address location. This image is done added next to the user's location information on the profile page using HTML. Although Google's API handles the conversion from location to latitude and longitude, it does not know (and therefore cannot inform us) whether the position it believes the human readable location is at, has been converted to the correct position. Therefore it is possible that the image returned may not be of the correct address that the user entered. Such a case can be seen if a user simply enters a road name without a city such as "100 Queen Street", without also including Auckland. In this case a map image will be returned, but it will be of 100 Queen Street, USA when in fact the end user may have meant Auckland.

Adding a Lightbox

So now when we click the static maps image, the interactive maps page is opened. As opposed to opening the maps in a new window, I decided to make it open in a lightbox. A lightbox is simply a popup box that overlays and appears on the same page as the current window (as seen in Figure 36). This makes it easy for a user to go back to the profile page and to exit the map quickly without having to reload the page or close a new tab or window. Different solutions were looked into for a lightbox library, due to the large amount of options.

The most popular options were all implemented in jQuery. jQuery LightBox, ColorBox, FancyBox, ThickBox, OrangeBox and SlimBox. The functionality though differed slightly between all the options. There were three pieces of functionality that I felt were the main required.

- The ability to embed an iFrame inside the lightbox. This iFrame would be required in order to open the enhanced maps page inside the lightbox.
- Also for the technical funding applications (See Page 44), one of the planned features that would be needed, is the ability to embed a PDF inside a lightbox.
- The lightbox should be able to handle images (which is what a lightbox is designed for)

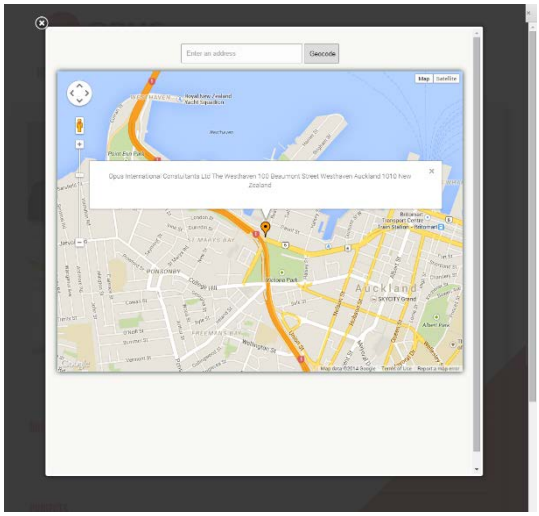


Figure 36, The lightbox library - OrangeBox as implemented on the profile page to display a popup modal window of the interactive map.

S. Deering had a quick comparison between different libraries in which he states the differences between the lightbox libraries are minimal, and your selection should be based upon the optional extras and customizability you require (Deering, 2011). Between all the available libraries, *Thickbox* can view PDF's in iFrames, but you will still need to implement some PDF viewer yourself for those browsers that do not support it. *OrangeBox* and *FancyBox* both could support PDF's natively. The rest were ruled out since they could not display inline PDF's. Both could also support iFrames and images. The deciding factor between the two options was that FancyBox was not free for commercial use, whereas OrangeBox

was. Therefore the lightbox library chosen was OrangeBox.

So now clicking the static map would open the lightbox, and the contents of the lightbox was now set to be an iFrame. An iFrame is an inline frame used to embed another document within the current HTML. So the lightbox contained an iFrame, and the iFrame was set to contain a new webpage with this HTML5 interactive Google map.

Creating the enhanced map was a simple procedure of embedding an HTML5 canvas which uses JavaScript to manipulate the canvas and display Google Maps. After successfully embedding the application the first thing implemented was the ability to show your own location on the map. Most mobile devices have a GPS built in to do this; however for an intranet system that is being used entirely on laptops, this is not a possibility since laptops do not contain a GPS. Instead I am using one of HTML5's new features for geo-location. This native feature can approximate your location by using information data from a variety of sources such as Wi-Fi triangulation and your IP address to calculate a rough approximation.

```
navigator.geolocation.getCurrentPosition(function(position) {
   var pos = new google.maps.LatLng(position.coords.latitude,
   position.coords.longitude);
   var marker = new google.maps.Marker({
         position: pos,
         map: map
   });
});
```

The above code uses the HTML5 geo-location service to approximate the current location of a user. However the location of the user cannot be determined, unless the user explicitly allows the web browser to send its location to the web server. There is no way by default for a webserver to enable this from the user, due to privacy reasons. Not all users will be comfortable sharing their location, and so the application has to also function if a user does not want to share their location (the default option). In this case the map will still function, but only the searched user's location will be shown on the map, and not the users own location.

If the user does allow the web browser to share their current location, then their location is then their position (latitude and longitude) is created into a map marker by Google's Map services. This marker is then shown on the map. The standard convention for showing your own location in a mapping application appears to be using a blue circle as seen on Google Map's Website and application and on Apple Maps (for iOS). To copy this norm, I created an animated GIF consisting of blue circles that increase in size to use for the marker for your own location. Due to this blue circular style icon being commonly used on most platforms to echo your own location, it should hopefully be easily recognisable by the user that it is representing the users own location.

What is important to remember is that the interactive map is on a separate page to the current page. This interactive map is displayed inside an iFrame in a lightbox, therefore when the lightbox opens it appears to be as if it is on the same page. To get the lightbox to show the correct location upon opening it, data had to be sent from the profile page to the inline iFrame. This was done with standard HTTP GET parameters implemented in JavaScript. The

address is copied from the “Contact Details” paragraph using jQuery, and is sent over the URL to the maps page

```
var address_encoded =
encodeURIComponent(document.getElementById('address').innerText ||
document.getElementById('address').textContent);
document.getElementById('maplink').href = "/map/?address=" +
address_encoded;
```

We also have to update the OrangeBox script, since it only does one static check when the window loads for links that have to be converted to lightbox links rather than links that open in new windows.

```
var head= document.getElementsByTagName('head')[0];
var script= document.createElement('script');
script.type= 'text/javascript';
script.src= orangebox.js';
head.appendChild(script);
```

Re-appending the script is an easy way to refresh it, without modifying the OrangeBox library itself. Along with this the OrangeBox plugin itself also needed to be tweaked. Although I wanted to keep the plugin native, I could not find a way at the time to allow the lightbox iFrame to update without having to modify the plugin files.

To retrieve the position data in the iFrame, we grab the GET parameter, remove any extra whitespaces and decode it.

```
function getUrlVars() {
      var vars = {};
      var parts =
window.location.href.replace(/[?&]+([^=&]+)=([^&]*)/gi, function (m,
key, value) {
            vars[key] = value;
      });
      return vars;
}

codeAddress(decodeURI((getUrlVars()["address"]).replace(/(\r\n|\n|\r
)/gm,"")));  //codeAddress is the geo-coding function
```

After getting the position data, it is passed on to the Google Maps geo-coding service to be converted. This map will then show a new map marker created at that latitude and longitude with an information box above showing the address, with the map centred on it along with the end users current location (should they allow it). Along with those core features mentioned above, other little frills were added such as animations on pin drops, popup information boxes on clicking a map marker and basic error handling. The final map which is displayed inside the lightbox is shown back in Figure 36.

The last part was combining the double click to edit feature with the static profile map. In the scenario a user double clicks to change their location, we would also like the static map to be updated to the entered location. An additional condition was added that when the address

field is being modified, upon save we make another call to Google's static map service, with the new address that the user just saved. We then instantly replace the previous static map image with the new one.

```
//If address has changed then update the static Google map image
$map.href = $url + encodeURIComponent($new_address);
```

This gave instant visual feedback to the user about the contact address they entered. As mentioned earlier, Google's static map service will always return an image even if it cannot geocode the address correctly. Now since the image is updated instantly, the user can see if the returned image is showing the correct location they entered. If it isn't the correct location, they can double-click to edit again and add more detail to the address (such as adding the suburb/city/country) in order to help Google correctly geo-locate the address. Upon saving again, the image will update once more and hopefully the returned image now will be of the location the user entered. This instant feedback again relates to giving the user feedback and user control as per what Nielsen mentioned in his heuristics.

So the static map visual aid, plus the ability to update your location by double clicking, could potentially simplify the process of updating ones address. If it does simplify the process, the goal would be to encourage users to use this feature as it may be seen as easier to use. This could then possibly help solve the problem of users keeping their locations out of date and make it easier for another user to locate another user's physical location

Technical Funding Applications and Management:

Introduction

Every year staff apply for funding for the projects they are working on. These can be small applications with low budget requirements, to projects that depend heavily on funding. Every application has to be filled out using an online form. This form is not easy to use and one user even pointed out (without even me asking), they do not fill up the form themselves since the application process never works for them.

All applications have to include a project director, manager, a budget, the country the project is being worked on, the completion date of the project and a PDF with further details about the project itself. In the back end of the system the data is aggregated by hand. Someone manually creates an Excel spreadsheet after accumulating all the information entered from all the applications at the end of every year. There is also a time delay between when all the applications are submitted to when the excel spreadsheet is created. My academic supervisor who then sees these applications has to wait and rely on someone else to create the spreadsheet and send it to them. She wanted this process to be automated.

The Application Form

The first part of my solution involved redesigning the application form online**.** To do this I created basic fields in HTML all designed with usability in mind. I used the new HTML5 form input types and functions to provide better control and validation (W3Schools). The reason for this is to try and improve on the usability, as users will receive appropriate

responses and feedback upon performing actions. E.g. by entering letters in a "number" input field, the textbox will highlight red telling the user something is incorrect. Another example is the when the user has to enter dates; they will be shown a date picker modal box. The user will also receive messages about required input boxes which they have left blank. Standard conventions have also been used along with these such as red highlights for incorrectly/blank fields, green highlights for correctly entered fields and light neutral highlights on focus.

For the design aspect I am using large textboxes with large padding, width and a bigger font-size. The goal is to make the application process as easy as possible and to retain focus on the content entered in the input fields. The bigger font and input boxes are an attempt to stand out, and helps to ensure focus is retained on those elements. This follows this pattern used by other top Alexa.com websites which follow similar conventions.

Figure 37, Custom animated loading screen created using purely CSS and images

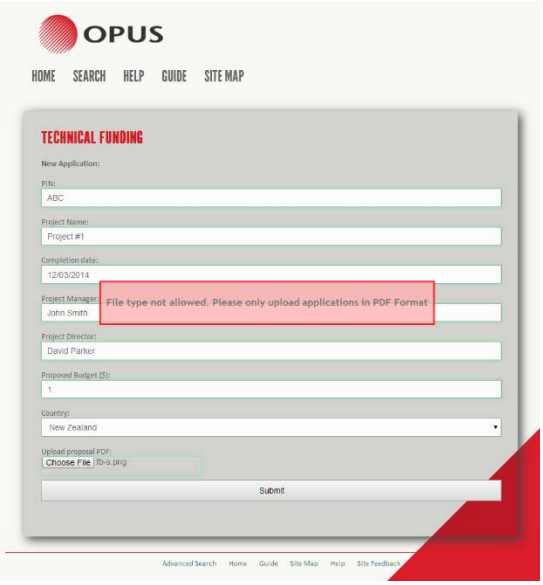


Figure 38, Example of a CSS modal box appearing on invalid file type input

I have used a combination of Ajax and PHP to process the forms contents. Although HTML already handles some errors such as blank fields - some additional error handling had to also be implemented. Upon submitting the form, the contents are grabbed and are posted to a PHP page using Ajax. The PHP page processes the data by first looking at the attachment. It checks the file size to see if it's less than 20MB, and then the extension to ensure it is only a PDF file and nothing else. Whilst this is happening I created a custom animated loading screen using purely CSS and an animated GIF (red, for consistency to suit the Opus colours), that fades in to look like a lightbox (overlay on the webpage where the webpage is dimmed out) as seen on the previous page (Figure 37).

After the PHP page has processed the form content it will send a call back to the Ajax function telling it whether it could successfully save the form in the SQL database or whether an error occurred. An error could occur due to the file size being too big, the extension not being a PDF or other reasons such as it could not simply connect to the database (Figure 38). Whatever the reason, a custom modal box (which I created using purely CSS) is transitioned on screen in either a red colour with the appropriate error message, or in a green colour saying the application was successfully submitted. This whole process happens on the one

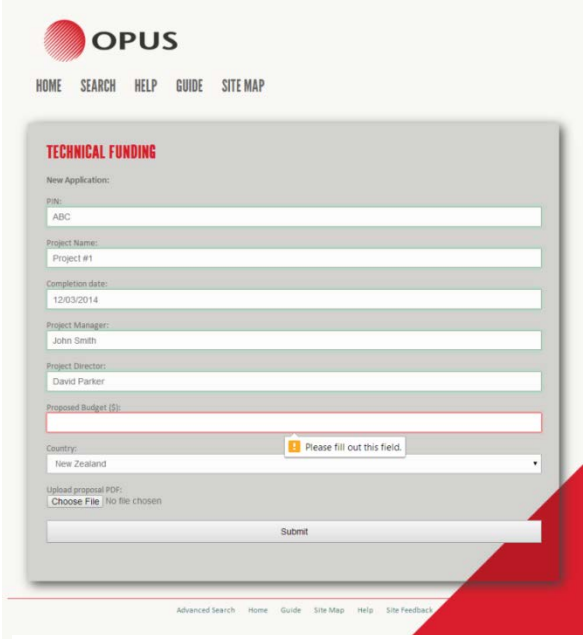


Figure 39, The final application form for technical funding showing validation and an example message and highlighted textbox when leaving a field empty

page for the user – and if an error message is sent back then none of the fields of the form are cleared. This allows the user to simply edit the incorrect data rather than having to await a page refresh and possibly having to enter all the information again. All of the above functionality was my implementation for the front-end process on submitting a technical funding application.

Graphing and Aggregating Application Data

The second problem relates to the back-end of the system. My supervisor wanted this data to be aggregated somehow. Until now someone had to manually do this data aggregation. They had to create a spreadsheet from scratch, fill it with this information and then send it along with all the individual project PDFs to my supervisor. This not only was a time consuming process, but it required my supervisor to await someone else to do this aggregation. This was a bottleneck as she would have to wait until this information was aggregated by someone else. Because the aggregation was done manually by someone, it was also subject to human/calculation errors.

My solution to this issue was to create a basic management panel to view all the applications (which should only be accessible by my supervisor or anyone with permissions). Since the front-end system saves all the applications in a MySQL database, we can now access this information, aggregate it together and display the output. Because each field has its own column, aggregating this data is now significantly easier. This management panel acts as an interface between the end user and the applications table in the database.

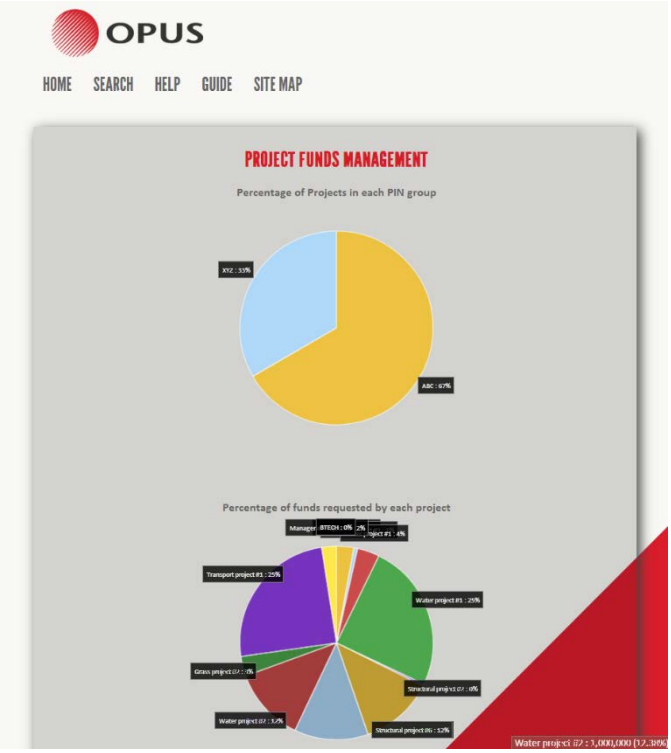


Figure 40, The management panel for technical funding applications, showing pie graphs created using the Flot library

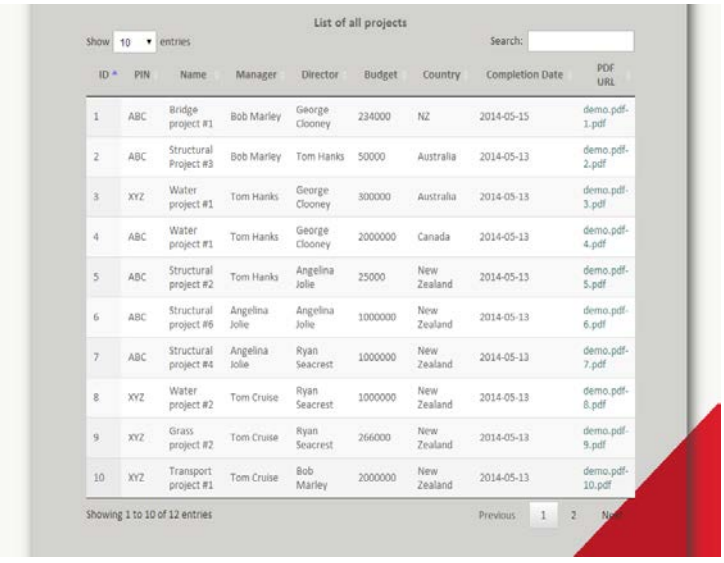
The management (administration) panel primarily uses jQuery and HTML5 to display the application information. I am also using to libraries for jQuery which I have customised to work with a SQL database. The first is called Flot, which is a plotting library for creating different types of graphs. And the second is DataTables, which is used for enhancing standard HTML5 tables.

Upon visiting the management panel, one can view graphs about all the applications that have been submitted till date. To do this I had to first make a connection to retrieve the applications table from the SQL database, then to convert each of the rows into an array of objects into a format that could be understood by the Flot Library. With some additional parameters and information Flot then outputs a HTML5 canvas which contains a graphed version of the inputted data. I am using bar graphs and pie graphs because for this application they are both useful at displaying a large amount of data in a relatively simple manner.

There were many other libraries available for plotting graphs in many different languages. JavaScript/jQuery again seemed like a good choice as it is a web language that will fit well with all the other pages I have created so far. Also JavaScript is highly responsive and interactive, which is ideal for viewing graphs online as users can interact with it. A comparison of JavaScript graphing engines was done by P. Whelan (Whelan, 2010) and by SocialCompare.com (Social Compare, 2014). Despite all libraries having their differences most of the graphing software will cover the basic functions needed to generate graphs. The best libraries seemed to be HighCharts, D3 and Flot because of their customisability and resultant output – which were much more aesthetically pleasing than the others. HighCharts appeared to be the best choice, except it costs from $90 – $3600 to use for commercial organisations. D3 and Flot were the free options. D3's output charts seemed to be more for complex data analysis and seemed too complex with excessive information, compared to what was required for this application. In this case simplicity and usability triumphs over the amount of functions the program contains. I therefore chose Flot as the library to use.

The resultant design of the graphs is mostly standard to what came with the default program. Labels for each slice of the pie chart were added along with the mouse over function. This was to view the percentages in the bottom right corner as some information may be obscured due to being very small slices in the pie chart. Secondly multiple graphs were added so the same information can be viewed and grouped by different criteria. Along with the pie graphs a simple bar graph was also created to view the budget required by each project.

The Data Table


List of all projects

Show 10 entries Search:

ID	PIN	Name	Manager	Director	Budget	Country	Completion Date	PDF URL
1	ABC	Bridge project #1	Bob Marley	George Clooney	234000	NZ	2014-05-15	demo.pdf-1.pdf
2	ABC	Structural Project #3	Bob Marley	Tom Hanks	50000	Australia	2014-05-13	demo.pdf-2.pdf
3	XYZ	Water project #1	Tom Hanks	George Clooney	300000	Australia	2014-05-13	demo.pdf-3.pdf
4	ABC	Water project #1	Tom Hanks	George Clooney	2000000	Canada	2014-05-13	demo.pdf-4.pdf
5	ABC	Structural project #2	Tom Hanks	Angelina Jolie	25000	New Zealand	2014-05-13	demo.pdf-5.pdf
6	ABC	Structural project #6	Angelina Jolie	Angelina Jolie	1000000	New Zealand	2014-05-13	demo.pdf-6.pdf
7	ABC	Structural project #4	Angelina Jolie	Ryan Seacrest	1000000	New Zealand	2014-05-13	demo.pdf-7.pdf
8	XYZ	Water project #2	Tom Cruise	Ryan Seacrest	1000000	New Zealand	2014-05-13	demo.pdf-8.pdf
9	XYZ	Grass project #2	Tom Cruise	Ryan Seacrest	266000	New Zealand	2014-05-13	demo.pdf-9.pdf
10	XYZ	Transport project #1	Tom Cruise	Bob Marley	2000000	New Zealand	2014-05-13	demo.pdf-10.pdf

Showing 1 to 10 of 12 entries Previous 1 2 Next

Figure 41, The overview of technical funding applications as shown using the DataTables library

Graphs are great to get an overview of information in a small amount of time. However in this scenario it would also be handy to view information about each project individually. A table with all the information could help to view all the information about each application. Unlike with the graphing applications where the usage of each library was strongly spread, with tables there were two main libraries that the majority of users seemed to use; DataTables and JQGrid. (Kumar, 2011)

Despite doing the same job – the libraries differed significantly on how they work. DataTables works by on top of a normal Table in HTML. The library then uses JavaScript and CSS to modify the table to give it the additional functionality and styling automatically such as the odd-even row colouring and the ability to limit the number of rows per page. JQGrid on the other hand required passing in data through a source such as JSON/XML without you having to do the HTML structure yourself. Both options had extensive support and documentation, but for commercial usage, JQGrid required a licence which costs $299 – thus DataTables was the option I used. Figure 41 shows the implementation of DataTables in the management section. The OrangeBox plugin was also used to be able to view any PDFs that were attached with a funding application in a lightbox on the same page itself. This meant a user did not have to navigate away to a new tab in order to view a PDF attachment, and to go back they could simply close the lightbox allowing easy reversal or actions.

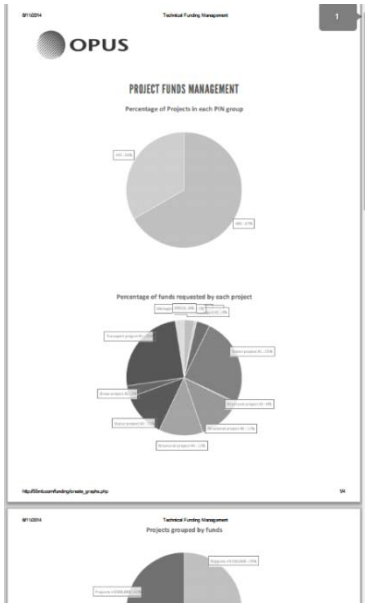


Figure 42, A print preview of the technical funding management panel

Since graphs would also be printed out, an adjustment that was done was to ensure that all the content will fit and look good when the page is printed. CSS allows you to specify the layout of your pages when printing to ensure a printer-friendly format using the @media rule. Figure 42 shows the how the graphic page appears when printing. Note that the navigation, background images and shadows are all removed in the printing process and only the graphs and tables are being printed out.

Research Publications Search:

Introduction

One of the functionalities that my academic supervisor wanted to add to the existing profile pages was the ability to view a list of publications and research papers that have been written by the user you search for. My supervisor wanted to implement this by adding a button that links to these publications at the bottom of the profile page. Hosting the publications may not be an option due to copyright and licensing issues, so to solve this problem the page would simply contain links to the articles which are hosted on the scholarly publication website.

However the larger problem remained that users are currently not in the habit of updating their profile pages. So users may still not have any incentive to fill up this information and keep it maintained even if we were to add these additional fields of information. So to overcome this, the plan was to gather this information dynamically off internet and automatically attach this information to the profile of a user (without requiring them to enter any details themselves). This would be done as a scheduled task in the background to ensure information that is retained is up to date. The sources we gather this information from are scholarly publication websites.

Comparison of Scholarly Websites

Falagas et al. conducted a comparison of scholarly websites. Their results stated that Scopus missed out on older articles whereas Web of Science didn't. However Scopus had a larger coverage of articles than Web of Science did and more than double the number when compared to PubMed. PubMed was excellent for producing biomedical articles. Google Scholar was good at finding many results (but that also included publications which may be of a lower standard) – but covered all range of topics (Falagas, Eleni, Malietzis, & Pappas, 2008). As Opus is an engineering firm in many sectors, a wide range of articles from many different sources would need to be aggregated together, in order to maximise on the results. Li et al. also compared a study using citations as an index for comparisons between Scopus, Web of Science and Google Scholar. Their results showed that Google Scholar was finding the most citations, however this included some duplicates. Scopus had more references than Web of Science, although the articles retrieved in search for both Scopus and Web of Science were quite different. This meant there was only a small subset of overlapping publications between these scholarly websites (Li, Burnham, Lemley, & Britton, 2010). So in order to maximise the number of publications retrieved, we may possibly have to aggregate results from multiple sources.

Initially the plan was to start off retrieving results from one publication source by iterating through every user in the Opus Database and figuring out if they had written any publications. Additional parameters such as the country they are staying in would be used to identify the correct user on the publication site. The future plans were to then collect data from multiple sources, aggregate them together and remove duplicate results. This would have ensured we collected most of the publications available without requiring to do anything themselves. However many limitations soon kicked in upon further research into the solution.

PubMed and Google Scholar both did not offer any API's to access resources. Web of Science requires a large fee to access their resources. Only Scopus (which includes Science Direct) had a free API, but that too was limited. Their paid API version again was a significant subscription fee. Upon further investigation into other academic databases I discovered smaller ones such as CiteSeer which supports OAI (Open Archives Initiative) but is limited in size and has no public API that can be used for search. JournalSeek and BASE have no API at all. After taking all this into account I decided I would still try to see if I could gather resources using Scopus's free API access – since it still was a significantly sized library. I had to register and had to submit the IP address that I would be using the API from. In the future this would have been something that would need to be addressed, since it would be a tedious process adding every single IP address in the company to the list of allowed API's on Scopus's website. A better solution would be to make all API requests route through a single proxy that handles all communication with Scopus's servers. This would have been needed to be implemented in the future.

Implementation

The initial prototype (Figure 43) used a search box to search for an author and using *REST* retrieve a list of publications written by the entered name in a JSON format. Due to some search limitations with the free API, a lot of instances when I was searching an author I was getting an empty result set. So to overcome this issue, if the first and last name combination

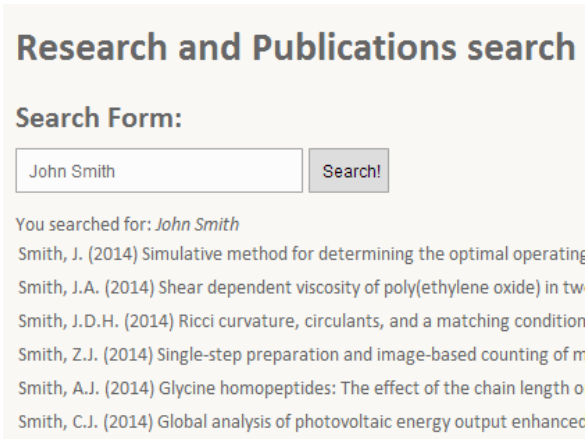


Figure 43, The initial publications search page prototype

could not be found then a second search is automatically repeated using the initials of the authors first name and their full last name. Although the set received back would be substantially bigger, it at least was a quick and simple fix to what potentially was looking at being quite a large problem.

The received information is parsed using PHP as shown below.

```
$results = $url."&query=firstauth(".$firstName.",".$lastName.")";
$file = file_get_contents($results);
$json = json_decode($file, true);
```

Depending on whether it's a Journal or Article the appropriate formatting is applied. The problem I realized upon creating this is that people have the same name – so we need a way to identify whether the name searched maps to the correct person in real life – e.g. Is John Smith the correct John Smith who we searched for in real life. A proposed solution to this that would require minimum input from the user, is to use filters (e.g. entering a country to narrow down results) plus producing all the results from the search to the user. From these results let the user select anyone article that they have written themselves. With the article they selected, we would get the Authors ID that we could then use to search for more publications from that author.

I added radio buttons next to each search that were to be used to select which entry was written by the user themselves. I then animated the process of display more publications using jQuery. However what I realized is despite the documentation for Scopus stating that you retrieve an Author ID with each result – in the free version of the API you do not. So even if a user selects a publication they have written we cannot get their ID to find more publications by them. Secondly searching by Author ID is also not allowed in the free version of the API.

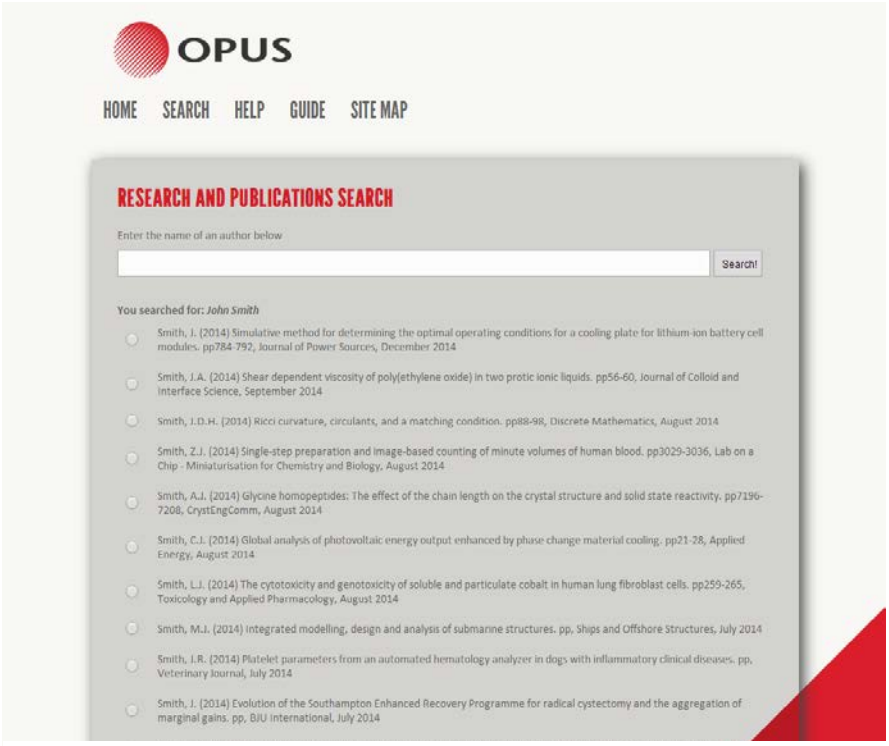


Figure 44, The final research and publications Search page

The above limitation became a real issue and discussions were made about workarounds to the solution, but nothing seemed to be a viable option. This was the last bit of development that was done for the publications search till date. Another option was looked into which was scraping webpages to gain additional information or results. Scholar.py is a python based scraper for Google Scholar that when running locally, will scrape any results from Google Scholar, and acts like an API to your client side application. This however was not a viable solution again as web scraping was a part of Google's Terms of Violation, and for a commercial organization is not an ethical option to do. Secondly if the website or structure of Google Scholar results change, then this may break functionality in the web scraper that will need to be patched before it can be used again. Therefore this piece of functionality was put on hold and it still remains unfinished even now.

Extracting Location Information From Project Data

Introduction

This was a piece of functionality that was begun in the later stages of the project, however was not finished due to limitations and other reason that will be further discussed.

Throughout the span of the project, my supervisor and I were attempting to get access to the database or even a copy of the database so that I have some sample data to work with. One of the databases that we were looking to get access to was the projects database which contained information about different projects undertaken by Opus. We wanted this for the mapping application to be able to display the locations of projects on the map using the geo-coding conversion process mentioned earlier. Opus's Intranet had a projects search page where you could search for different information about projects. What we learnt later was that the location field for each project contained very vague information. As this projects database was relatively new, when data was migrated paper to the digital database, only generic information about locations was entered. An example for this was that most of the locations in New Zealand would say either "North Island", or "South Island". Only a few projects would also contain a city and rarely any would have a proper address. We realised even if we did get a copy of the database, the location field would not contain sufficient information to plot on a map. It would have hundreds of markers that were generalised as either North or South Island.

The proposed solution to this was that we try and extract some sample data from the projects search functionality and retrieve as much information as we can about each project. I would then create an algorithm that would go through that information and try and extract any information about locations. After extracting this location information I would try to convert it to an address, and this address could then be geo-coded into a latitude and longitude to be mapped.

After looking at some sample projects, around 30 rows I realised that although the location field was generally vague, the project titles and client name generally contained enough information to deduce a location. Some examples of project names were as follows (the names have been slightly altered but the structure remains the same).

- **SomeName Road** Pond Upgrade
- **SomeName Street** Walking and Cycling Improvements Detailed Design (123-4567)
- **SomeName Railway** Station
- Detailed Design of the SomeName and **SomeName Park** Stream
- **SomeName Street** and **SomeName Street** Realm Upgrade, **SomeName Quarter**: Engineering Design Services
- 123-456/789 Proposal for **SomeName Tunnel** Drainage Design
- 789/123 Redevelopment of The **SomeName** area

And some example of client names

- **SomePlace Transport**
- **SomePlace Railways**
- **SomeCity Council**

After going through some of the sample project and client names, it was clear that these two fields had a fair amount of information that could be used to better estimate the projects location. The initial idea was to make this algorithm in JavaScript as it could be injected in the page with ease. For testing purposes I run the algorithm in the console window of Google Chrome on the actual search page.

Implementation

The idea and first step of the algorithm was to iterate through all the results, and find all the titles of the projects, and the clients. To do this the recursive function iterateDOM was created.

```
var iterateDOM = function (node, func) {
    func(node);
    node = node.firstChild;

    while(node) {
        iterateDOM (node, func);
        node = node.nextSibling;
    }
};
```

This function takes in a node retrieved in JavaScript and some function to apply to that node. To use it we pass in the first node with the matching criteria, in this case each search result row had the class project-results, so we find the first element with that class and pass that in the function. Along with the first node we also pass someFunction() in which will be the function used on the found nodes.

```
iterateDOM(document.getElementsByClassName ('project-results')[0],
someFunction());
```

iterateDOM will then apply the parameter function on that first element and continue to traverse through the DOM and repeat the process on other nodes that match the criteria until it reaches the end of the document.

The function we were applying to each node was to get its project title and client name, and see if the name or client contained any one of the words in the arrays below.

```
    var suffix = ["Road", "Drive", "Street", "Park", "Lane", "Avenue",
"Quarter", "Reserve", "Tunnel", "Station", "Waterfront", "Railway",
"Railways", "Council", "Transport"];

    var prefix = ["Creation of", "Development of", "Redevelopment of",
"Regeneration                                                    of",];
```

If the project title or the client name did contain one of the phrases in the array above, then it is likely the word(s) before or after that word would be location information. Just like the examples above, in the suffix array we have common location terms that normally act as suffixes to location information, e.g. “Dominion Road” or “Britomart Station”. The prefix array contained phrases commonly used as prefixes to location information e.g. “Development of Crestwood Primary School” or “Redevelopment of the Hamilton Gardens”. By searching for the keywords in those arrays, we could make an assumption that the words before or after will help us with the location information. We next have to rank these keywords, by some order in case we get multiple matches in the title or client name.

The way the arrays are structured is the words with higher importance are at the front of the array with lower indexes, while words at the end of the array have lower importance. “Dominion Road”, “Queen Street”, “Avondale Station” are very precise locations which will have a much higher accuracy when mapped. Whereas the assumption that words ending with Railways or Transport etc. Auckland Railways or Auckland Transport may only give us a generic location, such as Auckland.

In the next example where the project name is “Dominion Road Widening” and the client is “Auckland Council”, the words “Road” and “Council” will both be found in the suffix array. However because the word “Road” is in front of “Council” in the array, the word “Road” takes priority and the assumption is made that “Dominion Road” is more a more precise location than “Auckland Council”. This may not always work, but it is an assumption that has been built into the algorithm, and upon trial and error this method produced some fairly accurate results.

The next set of ranking is between the two arrays. In terms of priority, suffix array has priority over prefix array. That is why the algorithm searches suffix array after prefix array. If a word was found in prefix array and suffix array, then the suffix array word will replace the prefix array word. This is because of the way road names work. The name of the road always comes first then the type of the road e.g. street, avenue, and drive. If something is found in the suffix array it will possibly very precise. Even something ranked lower in the suffix array like Britomart Railways or Auckland Waterfront is still precise enough to give us a location to display on the map. This is the assumption that words in the prefix array tend to be more general, such as “creation of” or “development of” could be very general locations, or may be locations which do not exist yet (creating of Sylvia park, before it existed), therefore it ranks lower. Again this may not always be the case, and there will be better ways to approach this ranking and algorithm; however this is the first implementation of the algorithm I chose to do.

To test this algorithm, I had to first extract some sample data. I had to perform a blank search to get a set of results. The results were shown on multiple pages. The algorithm works on a single page result, which contains about 30 rows of data. The pseudocode of the algorithm was as follows, and my implementation was completed except for the last step of attempting to geocode the location.

```
Get the inner text of the node and split it column name.

For each column
  If the column name is the title of the project
    Split the project name into words
    For each word in prefix array
      If the word is in the project name OR if the word is in the
clients name
        Save the words after the matched word
        break
      end if
    end for

    For each word in suffix array
      If the word is in the project name OR if the word is in the
clients name
        Take this word and prepend to the current word
        Save these concatenated words
        Break
      End if
    End for

    Substring to saved words from the current location to the next
“and”
    Take these updated words and highlight them
    Append the updated words to the location field
    //Attempt to geocode this location
  End if
End for
```

As mentioned algorithm was only my attempt at trying to extract this location information and was left unfinished. There are improvements and better ways would also exist as to how to extract this location information. After getting to this point, this part of the project was put on hold as my supervisor advised she would instead just try to get us a copy of the database, and there is no point trying to work with half-baked data.

Evaluation

Due to time constraints a full usability evaluation was not conducted, however a System Usability Scale (SUS) was filled out by 10 willing participants. This evaluation statistically was insignificant and it is not conclusive and does not prove validity. The sample size was extremely small, and the participants were not selected randomly, nor were they target users

of the system. The users were also all from the same area and were known to me which may have caused some bias, and these users were only giving their evaluation based on their experience with the current system. None of these users had used the previous intranet system at Opus.

The participants tested were all based in Auckland, New Zealand. Half the participants were male, and half female. Four participants were aged between 20-24, two 24-28, two 30-34, and two 50-54. Only 2 participants were in computer related fields, with the other participants in a wide range of fields such as commerce, architecture and law.

The questions in the SUS were:

Q1. I think that I would like to use this system frequently.
Q2. I found the system unnecessarily complex.
Q3. I thought the system was easy to use.
Q4. I think that I would need the support of a technical person to be able to use this system.
Q5. I found the various functions in this system were well integrated.
Q6. I thought there was too much inconsistency in this system.
Q7. I would imagine that most people would learn to use this system very quickly.
Q8. I found the system very cumbersome to use.
Q9. I felt very confident using the system.
Q10. I needed to learn a lot of things before I could get going with this system.

	A	B	C	D	E	F	G	H	I	J
Q1	3	4	4	4	4	3	4	3	2	4
Q2	0	0	0	0	0	0	0	0	2	0
Q3	4	4	4	4	4	4	4	4	4	4
Q4	0	0	0	0	0	0	0	0	0	0
Q5	3	3	4	4	4	4	4	4	3	4
Q6	0	0	0	0	0	0	0	0	2	0
Q7	4	4	4	4	4	4	4	4	3	4
Q8	0	1	0	1	0	0	0	0	0	0
Q9	4	4	4	4	4	4	4	4	2	4
Q10	1	0	0	0	0	0	0	0	3	0
Normalised Score	92.5	95	100	97.5	100	97.5	100	97.5	67.5	100

Table 2, Results from a System Usability Scale (SUS)

The final SUS score was calculated by normalising the results, and then taking an average of all 10 participants as per the guidelines from Usability.gov (U.S. Department of Health & Human Services). The final normalised SUS score was 95.5. Although on its own this score is above average, as mentioned earlier, this score holds no weighting, relevance or correlation to the relationship with usability or end user experience.

Along with these usability tests, some basic performance testing was done just to be used as an insight as to what performance optimizations can be done on the site. Rather than to use the tool as a comparison index, it was used to gather feedback about what changes could be implemented to enable the site to load faster, and what performance optimisations were needed to improve on the site. Since a live intranet system would be hosted internally, there

anyway wouldn't be a real comparison index as it would be unfair testing between an intranet site and an externally hosted website.

The testing tool that I used was Pingdom Tools. It is used by a large number of firms including Google, Microsoft, Apple, Twitter, Github and Instagram (Pingdom AB, 2014). V. Sripathi & V. Sandru also recommend Pingdom Tools as a tool for testing website speed and performance (Sripathi & Sandru, 2013). Pingdom's website speed test offers feedback on improvements that can be made to help optimise a website. The intranet site was put online on a basic webhost, with shared hosting. The three pages tested with their results were:

- The profile page - received a score of 77/100 for performance.
- The standalone maps application - received a 81/100 for performance.
- The technical funding application page - received a score of 89/100 for performance.

This was an informal test, used to just gain feedback on optimisations that could be made rather than for the results. Some of the feedback received from Pingdom was to specify a cache validator, leverage browser caching, specify a Vary: Accept-Encoding header, and to combine external JavaScript. The mapping application and profile page seemed to receive lower scores due to the redirects and use of query strings on static resources that was being used by the Google Maps library, so this is something that I could not control.

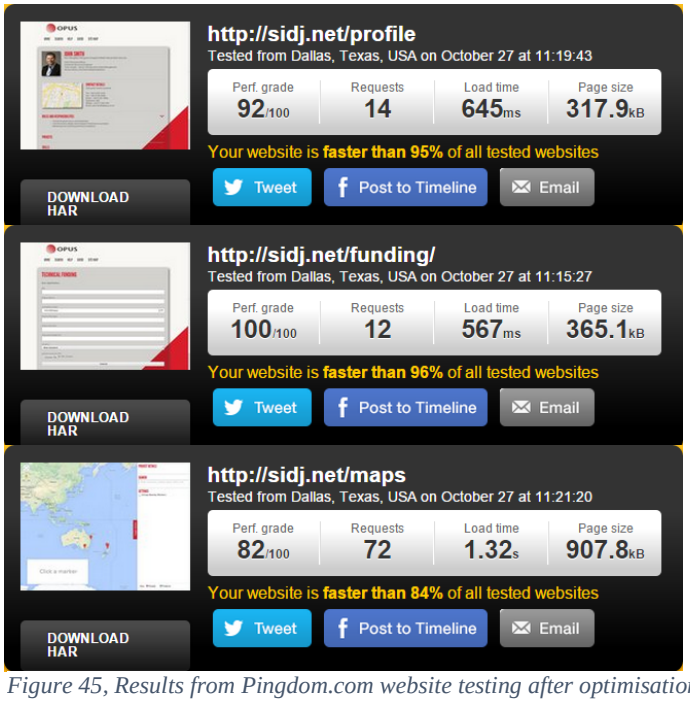


Figure 45, Results from Pingdom.com website testing after optimisations

After receiving this feedback, changes were made so the website better supports caching and compression. This was done by specifying these parameters in the sites .htaccess file. One of the other changes was to move some externally hosted JavaScript files (such as the jQuery source file) locally. This meant that the number extra requests being made to external sites had reduced, and most of the resources needed were all available locally. Upon testing again the scores had improved as shown in Figure 45.

Although according to Pingdom, since the last test the performance of the website had been improved, this was just one tool, and overall this does not mean by any standards that the website is a fast and well performing one. There were still improvements that could be made such as font-caching and merging and minifying JavaScript files together to reduce the number of requests, especially on the maps page. What is also important to note is that Pingdom and other website speed test tools only test for the website's speed and responsiveness on load. They do not measure the responsiveness and speeds of dynamic content that is generated from user interaction with the system.

Future Work

It is still currently unknown whether my proposed solution actually does help improve usability and user experience. Some further testing and usability evaluations should be performed in order to find out how the proposed changes affect end user experience. Along with usability evaluations, it may also be wise to do some performance testing to see how the newer system compares with the older system or other similar websites and systems out there. As well as both of these, because the system supports the facilitation and storage of confidential user and project data, some web application security tests, such as penetrative testing should also be performed. This could help ensure the system is secure and can only be accessed by those who are authorized to do so.

For the double click to edit feature, it would be nice to try and continue working on the feature, in order to bypass some limitations which currently exist. If one could establish a better method to map connections between text areas and the rows they correspond to in the database that would be a significant improvement as it would be easier to configure. Secondly one could also possibly look at making it possible to configure where autocomplete gathers its data from, for each individual editable block. Both these functions could help allow the feature to be converted to a more universal plugin that could simply be imported and used on any webpage.

Changes to the map could include a better method at preventing overlapping between marker clusters. The concept of the offset algorithm only works with a maximum of 2 groups of clusters. A better solution maybe to modify the plugin and to allow a cluster to be assigned to a group and check for overlaps when clusters are created. This could guarantee 0% overlapping of clusters and would allow multiple cluster groups to better fit on the map together.

The standalone mapping application's universal search algorithm is ranking results by looking at where the position of query, is matched in the string. Along with this, the way ties are dealt with is unfair as users will always win any ties. Improvements to this algorithm could be done by taking into account other factors such as frequency of each search result, a user's past search history and also just an overall improved weighting system. This could possibly help better estimate what the user is trying to search for.

Finally if proven successful, then this proposed solution will need to be migrated to Opus's intranet system. This may require some porting to be done in order to make it work with the new system. It could also be used as a proof-of-concept and be recreated in order to make it compatible with the SharePoint solution that Opus is implementing.

Conclusion

The above project my attempt at the goal of enhancing Opus's intranet system's usability and user experience. The changes I have made so far have been based upon the research I have conducted so far, and what I personally felt will be a design that enhances user experience. I have attempted to identify and follow emerging design trends and patterns, along with well-known heuristics. However since design is very subjective, I cannot be sure whether my proposed solution will improve, worsen or affect the usability and user experience at all. I was unable to conduct sufficient user tests to deduce how my proposed solution affects the existing system, so this is something that will need to be done in the future. If this proposed solution is seen as an improvement, then this project would need to be migrated on to Opus's servers, should their IT team choose to do so. I hope that in the future, these new functionalities and design changes can be improved on and further tested by someone, with the desire to provide a better user experience.

Bibliography

Alexa. (2014). *The top 500 sites on the web*. Retrieved from Alexa: http://www.alexa.com/topsites

Bevan, N. (2009). *Wha is the difference between the purpose of usablity and user experience evaluation methods?* Uppalsa: INTERACT.

BuiltWith. (2014, August 4). *Mapping Usage Statistics*. Retrieved from BuiltWith: http://trends.builtwith.com/mapping#

Campbell, J., & Shelly, G. B. (2014). *Web Design: Introductory.* Course Technology.

Carlos, G. (2013). *Responsive Web Design with jQuery.* Packt Publishing.

Cipeluch, B., Jacob, R., Winstanley, A., & Mooney, P. (2011). *Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps.* Dublin: Environmental Research Center Environmental Protection Agency.

Davison, N. (2013). *Liquidapsive (Liqui-dap-sive)*. Retrieved from Liquidapsive (Liqui-dap-sive): http://liquidapsive.com/

Deering, S. (2011, November 17). *JQuery LightBox vs. ColorBox vs. FancyBox vs. ThickBox - What are the key differences?* Retrieved from SitePoint: http://www.sitepoint.com/jquery-lightbox-colorbox-fancybox-thickbox/

Domain7. (2014, April 8). *Difference Between Usability and User Experience*. Retrieved from Domain7: http://domain7.com/blog/slideshare-difference-between-usability-and-user-experience

Falagas, M. E., Eleni, P. I., Malietzis, G. A., & Pappas, G. (2008). Comparison of PubMed, Scopus, Web of Science, and Google Scholar: strengths and weaknesses. *The FASEB Journal*, 338-342.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology 47*, 381-391.

Google. (2014, Auguest 1). *FAQ*. Retrieved from Google Maps API: https://developers.google.com/maps/faq#usagelimits

Google. (2014). *MarkerClustererPlus for Google Maps V3*. Retrieved from Google Code: http://google-maps-utility-library-v3.googlecode.com/svn/trunk/markerclustererplus/docs/reference.html

Hadlock, K. (2012). *jQuery Mobile: Develop and Design.* Berkeley: Peachpit Press.

Hinchliffe, A., & Mummery, W. K. (2008). Applying usability testing techniques to improve a health promotion website. *Journal of Austria 19*, 29-35.

Hinckley, K., & Song, H. (2011). *Sensor Synaesthesia: Touch in Motion, and Motion in Touch.* Redmond: Microsoft Research.

Kumar, A. (2011, October 5). *Two Best Display Components :JqGrid And Datatables.Net*. Retrieved from MSGuy: http://www.msguy.com/2011/10/two-best-display-components-jqgrid-and.html

LaGrone, B. (2013). *HTML5 and CSS3 Responsive Web Design Cookbook.* Packt Publishing Ltd.

Lal, R. (2013). *Digital Design Essentials: 100 Ways to Design Better Desktop, Web, and Mobile Interfaces.* Beverly: Rockport Publishers.

Li, J., Burnham, J. F., Lemley, T., & Britton, R. M. (2010). *Citation Analysis: Comparison of Web.* Alabama: Routledge.

McNamara, N., & Kirakowski, J. (2006). Functionality, usability, and user experience: three areas of concern. *Magazine Interactions - Waits & Measures, Volume 13 Issue 6*, 26-28.

Michaud, T. (2013). *Foundations of Web Design: Introduction to HTML & CSS.* New Riders.

Nielsen, J. (2000). *Designing Web Usability: The Practice of Simplicity.* Indianapolis: New Riders Publishing.

Nielsen, J. (2003). Usability 101: Introduction to Usability. *Jakob Nielsen on Usability and Web Design*.

Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. *ACM CHI'90 Conference.* , (pp. 249-256). Seattle.

Oh, U., Kane, S. K., & Findlater, L. (2013). Follow That Sound: Using Sonification and Corrective Verbal Feedback to Teach Touchscreen Gestures. *Proc. ASSETS'13*.

O'Reilly, T. (2005, September 30). *What is Web 2.0*. Retrieved from O'Reily: http://oreilly.com/web2/archive/what-is-web-20.html

Pingdom AB. (2014). *Pingdom - Website Monitoring*. Retrieved from Pingdom: http://www.pingdom.com

Pozin, I. (2014, May 15). *Let It Go: Say Farewell To These 5 Web Design Trends.* Retrieved from Forbes: http://www.forbes.com/sites/ilyapozin/2014/05/15/let-it-go-say-farewell-to-these-5-web-design-trends/

Shneiderman, B., & Plaisant, C. (2010). *Designing the User Interface: Strategies for Effective Human-Computer Interaction:*. Reading: Addison-Wesley Publ. Co.

Social Compare. (2014, July 22). *Javascript Graphs and Chart Libraries*. Retrieved from Social Compare: http://socialcompare.com/en/comparison/javascript-graphs-and-charts-libraries

Sripathi, V., & Sandru, V. (2013). Effective Usability Testing – Knowledge of User Centered Design is a Key Requirement. *International Journal of Emerging Technology and Advanced Engineering*, 627-635.

Turner, A. L. (2014, March 19). *The history of flat design: How efficiency and minimalism turned the digital world flat*. Retrieved from The Next Web: http://thenextweb.com/dd/2014/03/19/history-flat-design-efficiency-minimalism-made-digital-world-flat/4/

U.S. Department of Health & Human Services. (n.d.). *System Usability Scale (SUS).* Retrieved from Usability.gov: http://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html

W3Schools. (n.d.). *CSS Safe Web Fonts*. Retrieved from W3Schools: http://www.w3schools.com/cssref/css_websafe_fonts.asp

W3Schools. (n.d.). *HTML5 Input Types*. Retrieved from W3 Schools: http://www.w3schools.com/html/html5_form_input_types.asp

Whelan, P. (2010, December 3). *Highchart Vs Flot.js - Comparing Javascript Graphing Engines*. Retrieved from Big Fast Blog: http://www.bigfastblog.com/highchart-vs-flot-js-comparing-javascript-graphing-engines