BTech Project - HazApp

The Geospatial Hazard Management System

Rowan Carmichael
University of Auckland
June 2015

Abstract

This paper will be following my progress as a fourth year BTech student working with the software development team of Opus International Consultants Limited to solve a company-wide problem of on-site hazard reporting and management. The idea is a web-based application for desktop, mobile and tablets named HazApp, which will function as a geospatial hazard management system. The most fundamental challenge of this project is how will I map out, plan, develop, and evaluate an entire hazard management application based purely on the idea of a single problem.

Everything from the highest level of planning for the problem, to the lowest level programming will be documented, as well as research and recommendations for the technologies I see best fit for any design problems that arise, and any testing that has been to help us complete this project to the highest of quality. This report will also cover the results of a usability study as well as several heuristic evaluations on the application's overall usability.

Contents

Figures

Tables

Acknowledgements

A special thanks to Mano for giving me this wonderful opportunity. Also a massive thank you to the Opus team: Kodie Wixon, Taylor Carnell, Sulo Shanmuganathan, Roquito Lim and Mitchel Bennett, who have all made this process as equally enjoyable as it is fulfilling.

Dr. Sathiamoorthy Manoharan

Academic Supervisor
Senior Lecturer Computer Science - University of Auckland

Kodie Wixon

Industry Manager
Senior Software Developer & Software Team Manager - Opus International Consultants

Taylor Carnell

Industry Supervisor
Geospacial Software Analyst - Opus International Consultants

Chapter 1 - Planning

1. Project Introduction

The HazApp system is a proposed geospatial hazard management system to be used by Opus employees and contractors. It will consist of both a mobile app for on-site hazard reporting and real-time hazard alerts, and a desktop app for statistical analysis and management of hazards. The original proposal was an outcome of Opus' Big Ideas Competition [1] as an improvement to Opus' current paper-based hazard reporting system.

1.1 The Company

Opus International Consultants Limited is a multi-disciplinary international consultancy company consisting of over 3,000 engineers, designers, planners, researchers and advisors, situated across 5 countries (New Zealand, Australia, Canada, America, and the United Kingdom). Their work services include transport asset development, building design, water, and other infrastructure. Because of the nature of their work fields, many Opus employees and contractors working for Opus are very often found on work sites (rather than in the office).

Some of Opus' more recent projects include, but are not limited to, the Newmarket Rail Station redevelopment, Ngatamariki Geothermal Power Plant construction, the Waikato Expressway, and the Carterton Events Centre. All of these projects are of massive scale and as such are relatively prone to on-site hazards. HazApp hope to minimise the time spent towards reporting and managing on-site hazards so that work can continue on the more important aspects of planning and construction.

1.2 The Problem

Offering professional consultancy services in asset development and management often requires Opus employees to be working on-site where if any hazards occur they must be reported and stored for management, however the current hazard reporting system is a paper-based form (See Appendix Item 1 for a sample hazard reporting sheet) which is tedious to complete, takes time to be transferred into a database, and does not offer any advice or alerts to the reporter or anyone else working on the same site.

As well as having an inefficient paper-based hazard reporting system, each different Opus office (both nationally and internationally) has a different means of reporting and storing the hazard data. This lack of connectivity has misaligned Opus' safety and business practices and is continuing to promote an absence of interconnectivity within the company.

1.3 Project Goals

HazApp was created to rectify these problems by both moving the hazard reporting away from paper towards utilisation of smartphones and tablets, and realigning databases to make the best use of the reported hazard data. Using a mobile map-based hazard reporting not only allows for real-time hazard reporting and management, it also has room to offer immediate advice and mitigation

devices to best handle reported hazards. Using a desktop management system with a single global database allows for powerful statistical analysis.

HazApp needs to perform well in areas with and without stable internet connections, it needs to have adequate security measures to protect Opus' data, and it needs to be as user friendly as possible (as it will eventually be offered to users of all different technological abilities).

1.4 Related Work

Due to the practical nature of this project and the fact that it is a relatively original idea it was difficult to find many sources of related work (especially those of academic nature, such as journal entries or conferences). However from my research I have come across one application which seems to be very similar to what we intend on developing ourselves. I believe it is very important while we are still in the early planning stages of the project to seek out similar projects for analysis on what they have done, and what we can do better.

ThunderMaps

ThunderMaps [2] is a mobile application used to report and manage workplace risks in real time much like how HazApp is planning to do. While simply using ThunderMaps to solve our problem may seem like a decent and quick solution, there are definitely some drawbacks to simply buying vendor software for such a specific problem. The most important benefit of creating an in-house application from scratch is that we will have the maximum amount of freedom in terms of design and development. Instead of having to try to find the "perfect" software for the problem, or trying to find a "close enough" software and asking the vendor to modify it (which can be very expensive and time consuming), we can create the application purely based on our needs. This includes getting input from knowledgeable sources within the company (such as health and safety managers), as well as getting real users to test the application and give us feedback while we develop. This freedom also extends to the data itself, by designing and creating an application ourselves we can mould our database design around current Opus databases. And finally, as HazApp is planned to eventually extend to all Opus employees worldwide (assuming the deployment within New Zealand is a success), the scalability and agility to respond to problems is significantly more important. As such it will be a necessity to have in-house developers that understand and can react to anything that may damage the effectiveness of HazApp.

The features that ThunderMaps shares with the HazApp idea is location based risk reporting, utilising "alert areas" which allows users (for example health and safety managers) to monitor a specifically selected area, automatic alerts to staff when they approach a hazard, and some form of report generation. However there are some critical differences which separates ThunderMaps and HazApp, in particular while ThunderMaps does allow for public and private commenting on a particular hazard, it doesn't have nearly as much depth in its report generation when compared to HazApp. [2] The HazApp idea also has a strong focus on generating statistical analysis from all the information input into the database, this is a crucial aspect which will aid in the improvement of safe business practices.

2. Technologies

Due to the general complexities and cross-platform nature of the project, HazApp will be heavily reliant on technologies if it is to be a success. The discussion following will outline key decisions defining how the project is to be made and how it will function. The discussion and ultimate decisions will be based on a review of current technologies that are able to fit the required function specifications. These will range from development languages, technical application program interfaces (APIs), user interface (UI) frameworks, and database systems.

I will also note that in many of these cases, decisions will not be made until development starts, and any decisions that are made here may be subject to change when actual development does commence.

2.1 Programming Language

Seeing as this project requires both a mobile (smartphone and tablet) and desktop component it is crucial to decide on a language (or languages) that accommodate the platforms and functionality of the application. The primary programming languages that I will be considering for this project are; native app for mobile (consisting of some/all of Android, iOS, and Windows Phone), native application for desktop computer (likely in either C#.net or Java), a web based application (HTML5) for both mobile and desktop, or an HTML-built application utilising PhoneGap.

Native Mobile App

Perhaps the most obvious solution for the mobile side of this project, native mobile apps allow for very powerful functionalities and generally faster speeds for functionalities such as use of GPS tracking and photo capturing (when compared to web based applications). Another attractive benefit for developing a native mobile app is the ease of use for either offline work or in areas that have sporadic network connectivity. This is going to be something that may not be a primary decision factor now, but will definitely need to be considered for the completion of this project.

One big drawback for both forms of native applications, in comparison to a web application, is that if the project is ever updated (which will most definitely occur), the native applications will have to be manually updated. This is not the case for the web application as all of the updates will occur server-side so whenever a user goes to the web page it will be showing the most updated version. Along with having to update the application where necessary, both native apps require disk space on the device being used. While this may not be an issue for a desktop computer, it is something we have to keep in mind for mobile devices with limited storage space.

Unfortunately another major drawback of creating a native mobile app is that it would likely like more than one language codebase (as some combination of Android, iOS, and Windows Phone) for the mobile side alone. Taking into considerations the limitations of time, expertise, and money we have decided that a native mobile app approach would not suit what we hope to achieve.

Native Desktop App

Similar to native mobile apps, a native desktop application can offer increased speed and functionality in comparison to a web application. However it also shares the same drawback of requiring separate code bases for both the mobile and desktop sides. With this in mind it is clear

that the two options we could take in terms of coding languages are either a native mobile app and a native desktop app, or a web app for both mobile and desktop.

Web App

The limitations of web applications compared to native mobile and desktop applications somewhat numerous, however due to the relatively simple functionality of the project, we feel as though a HTML5 web application using JavaScript would more than suffice the primary needed functionality of mapping, geolocation tracking, photo/video capturing, and connectivity to a server side database.

As mentioned earlier, having some functionality for offline use or intermittent internet connectivity needs to be considered. Fortunately there are options utilising HTML5 and JavaScript which allow for such functionalities. The possible solutions to this problem will be assessed later.

Finally, using a HTML5 web based application will allow for a lot of code sharing between the mobile and desktop application. This is an incredibly attractive trait of web applications and is a main contributor to why we have ultimately chosen to develop a web application in HTML5, utilising JavaScript APIs, and a PHP database.

PhoneGap

PhoneGap is a special case that we will be also looking into specifically for the mobile and tablet side of our project. In terms of development it should be exactly the same as a web application using HTML5, CSS, and JavaScript. However it differs in how it is deployed, instead of being a web based application PhoneGap would allow for out HTML/CSS/JavaScript codebase to be converted into native mobile applications for Android, iOS, and Windows Phone. This would eliminate the problem of having multiple code bases for each different device type and could also allow for better local storage on the device for offline or poor-connection usage. While in theory this sounds great it may not be as easy to include such functionalities. [3]

Although this seems like a good middle ground decision it does not come without its own faults. The most glaring problem that would most likely arise is decreased performance compared to a regular native application or a web based application. From what has been suggested from some of the other Opus developers, PhoneGap may not be the best choice if we are interested in having decent performance speeds from out application. From what I have gathered PhoneGap's conversion comes with a cost, and seeing as it is important for our app to function fast enough as to not have a negative impact on the user's view of the app, PhoneGap may not be the most obvious choice.

Decision

Due to the very small size of our development team and the large size of the project itself we have decided that we will be developing using HTML5 for both the desktop and mobile/tablet side of the application. As mentioned earlier HTML5 comes with more than enough functionalities to accommodate the project requirements and by ch0osing HTML5 we will be able to share a decent amount of code between the two sides of the application.

2.2 Database Management Language

One of the key inspirations for this project was to connect the entire Opus community through a global database system, as such the decision for the database management language is very important to the longevity of the project. As discussed earlier we are going to be creating a HTML5

based web application and as such we will be using PHP to connect client and server databases. In terms of database implementations we will be looking into two of the most popular options being a regular SQL database and a NoSQL database.

SQL

SQL (structured query language) databases have been around for many years, with their first appearance in 1974 and initial release in 1986 and as such have been the dominating framework for databases up to present day. The biggest difference between SQL and NoSQL is that SQL databases are primarily relational databases utilising tables containing data fitted into predefined categories. While SQL has been around significantly longer than NoSQL it does come with its limitations. The major limitations to note are scalability and complexity. As SQL uses relational databases scaling the size of a database is an expensive and difficult task which requires powerful servers. The other major drawback as compared to NoSQL databases is the complexity of relationships within the database. SQL requires a network of tables all connected through some means of relationship strings. An implication of this is that altering the design of the database structure can be very complex and can downright break your database (especially in the case of deleting data/tables).

One benefit that regular SQL databases have over NoSQL databases is that, for complex queries, SQL offers standard interfaces aiding in working with such queries. In general SQL databases are best fit for heavy duty transactional type applications, the reason being is that they offer more stability and promise atomicity as well as integrity of the data. This is emphasised through SQL's ACID properties (Atomicity, Consistency, Isolation and Durability). The HazApp project will be including some form of transactions (most likely on the management side of the application) so the benefit of stability and atomicity will be kept in mind.

Finally the last advantage to note that SQL has over NoSQL is that, as it has been around for so much longer, SQL offers excellent support for their databases from vendors. Whereas NoSQL largely has to rely on community support. [4]

NoSQL

NoSQL databases have surged in popularity since their release in 1998. The aim of NoSQL was to move away from the idea of concrete relational tables for a more flexible framework. NoSQL databases focus more on key-value pairs, no longer requiring fixed table schemas and relational join operations. They have traded off the ACID properties for Brewer's CAP (Consistency, Availability, Partition tolerance) theorem.

One of the bigger positives NoSQL has over has over regular SQL is that no schema are required. That is to say data can be inserted into a database without having to define a rigid database schema. This also allows the format of the data being inserted to be changed at any time without application disruption, leading to massive application flexibility. In general NoSQL databases process data faster than relational databases as their data models are more flexible and often simpler. [5]

While both SQL and NoSQL databases have their own differences and benefits, we are yet to make a decision on what database type we will be implementing. This decision will be made closer to development start.

2.3 JavaScript Frameworks

Selecting appropriate JavaScript frameworks can greatly reduce the need for tedious manual calculations. Essentially what we are looking for in a JavaScript framework is functionalities which will aid help with manipulation of the webpage's data through things such as functions and bindings. In this case we will not necessarily be settling for a single JavaScript framework, but instead we may use several in different areas which we see fit. For general JavaScript we will be considering Knockout.js, Angular.js and Backbone.js. This decision will likely be influenced by how the JavaScript framework combined with the web app framework. For offline storage functionalities we will be looking at Lawnchair and Local Forage.

Knockout

While Knockout, AngularJS, and Backbone all offer some of the same very useful functionalities that regular JavaScript does not naturally support (such as data binding and DOM templating of code into smaller maintainable pieces), Knockout is different as it is primarily a lightweight data-binding library. Unlike AngularJS it has explicitly put work towards focusing on unobtrusive code, which could be important for our project. While at times each of these three frameworks may outperform the others in terms of performance speed, Knockout has a stronger focus on speed and should offer better performance than the other two for common tasks that we will be implementing. [6]

Angular

Angular is different to both Knockout and Backbone as it is a full-fledged framework (rather than a lightweight one). It has be built from testability and as such of this can clear project organisation more effectively that the other two alternatives. It is the "heaviest" of the three frameworks, and because of this it can offer more luxury functionalities such as custom elements. It is difficult to say whether or not these extra functionalities will be of any benefit without having started development yet. [7]

Backbone

More similar to Knockout, Backbone is a lightweight JavaScript framework and as such in general it will also perform better that AngularJS in terms of speed. Unfortunately this comes at a cost; while Backbone excels in simple applications, it may fall behind when dealing with heavy built-in data interactivity or extensive scaling. As mentioned earlier, it is difficult to tell which of these frameworks will suit out problem, although in terms of the mobile side of the application we are going to try and make it as simple as possible so both Knockout and Backbone may have the slight edge at the moment. It is going to be our job when we start developing to identify and handle the balancing act between these frameworks. [8]

With all of this in mind I will conclude that all of these frameworks essentially are solving the same problems. There are small differences between the functionalities and syntax between the three but ultimately any of these frameworks seem as though they would adequately work for our project. While this decision is still undecided, it will probably be influenced by more specific problems we encounter while actually developing. If any one framework can manage a specific problem better than the others, it will most likely be the one we use, however that will be judged on a case-by-case basis.

Lawnchair

Again Lawnchair and Local Forage share the same roll of maintaining data offline. Although we are still in a very early stage of planning and offline data management is rather low on our priority list I still thought it would be beneficial to review two of the most popular options for local HTML5 storage using json.

Lawnchair has been designed with mobile in mind, which is great to hear as the mobile hazard reporting side of our project is where we will be wanting offline support. It has a few very simple but powerful functionalities which cover the basis of offline data storage. These include mapping key-value pairs, saving them to a local "store", and accessing them later. Unlike Local Forage, Lawnchair has stopped releasing new builds and has been "finished" as a project. This means that as time goes on it most likely will fall further and further behind the regularly updated Local Forage. [9]

Local Forage

Mozilla's Local Forage seems to be a more complete solution to the problem of local storage for HTML5. It shares a similar methodology of saving and retrieving data as Lawnchair, but it also offers built in error handling. This essentially means it is slightly more complex but covers lightly more functionality than Lawnchair. As mentioned earlier it is continually updated and from what I have gathered has far more extensive documentation and support. [10]

At this time in our research, offering offline storage functionalities has one of the lowest priorities and will probably not be mentioned again until we are nearing our final release.

2.4 Mobile Web App Frameworks

The web app frameworks we will be deciding to use will primarily be based on user interface and ease of use. With this in mind we will only be looking at a select few (although there are a numerous amount of potential contenders) that we deem most likely to fit our needs. We will consider JQuery Mobile, Ionic Review, and Bootstrap.

JQuery Mobile

JQuery Mobile is a very well know and very popular choice for HTML5/CSS/JavaScript development for smart phones and tablets. It is a very easy to use framework that does a lot of useful work for you (especially in terms of automatically generated user interfaces). It is such a popular choice for smart phones and tablets as it includes a very clever built in scaling system so that you program can easily be transferable between many different screen sizes. In essence JQuery Mobile is a minimalistic upgrade to JQuery designed for responsive web pages and platform independent applications.

Another great benefit of JQuery Mobile is that seeing as it is so simple, it is incredibly easy to extend further JavaScript libraries. As such it should be able to fit well with any of the JavaScript frameworks discussed above. As well as working great for mobile devices JQuery Mobile also offers smart designs and implementations on desktop applications. This may come into our decision making process as it is important to have coherency between out mobile and desktop application which can be boosted by having a similar user interface style for both.

The final benefit that JQuery Mobile which is very attractive for our project is the fact that it offers a lot of mobile-specific function handling such as swipe-events, page transitions and touch-friendly

components. However while it does offer a lot of useful functions it can be have very slow performance, especially if the application is not designed properly. [11]

Ionic

The Ionic framework is the most recently created web app framework we will be considering, with the alpha release in November 2013. Similar to JQuery Mobile, Ionic Review primarily focusses on the user interface, however it differs in the fact that it is built on top of Google's AngularJS framework. This pairing is a necessity for Ionic to function to its fullest potential so if we were to choose it we would also have to be working with AngularJS.

Another similarity Ionic shares with JQuery Mobile is having a strong focus on responsive web design, which is a big plus. We will be wanting to have our application provide optimal viewing and interaction experience, as well as quick and easy navigation, and the ability to function on a wide variety of devices. By utilising a responsive web design to its maximum potential, we should be able to share a lot of code between out desktop and mobile applications. [12]

Bootstrap

Bootstrap is a front-end framework which also offers a number of great user interface components such as dropdowns, breadcrumb navigations, and button groups. Unlike JQuery Mobile, it has not been designed to primarily focus on mobile applications and as such seems to have the appearance of a desktop application (even when on a smartphone screen). To fix this, custom code would be necessary. As it is less dependent of JQuery, it generally will exhibit better performance. [13]

While we will be developing for both mobile and desktop we are yet to decide if we will use any of these frameworks for both sides of the application or if we will divide our application by using different frameworks for the two sizes (for example JQuery Mobile for the mobile/tablet side, and Bootstrap for the desktop side).

2.5 Stylesheet Languages

While not a major priority for the project, utilising an effective stylesheet language that can be compiled into CSS, can make the CSS code easier to understand and simpler to create. The two big names in this area that we will be considering are Less and Sass. It should be noted that while we will be considering both, we may end up not using either and just stick to regular CSS.

Both Less and Sass share a lot of syntax and functionalities, and are essentially attempting to solve the same problem of decreasing the amount of code needed for stylesheets through added functionalities. These include but are not limited to:

- Mixins: which allow embedding of properties of a class into other classes, creating a soft of variable which can be repeatedly used.
- Parametric Mixins: act as functions by allowing passing of parameters
- Nesting: similar to a nesting in a language like Java, cuts down on repetitive code
- Functions and Operators: allows for mathematical equations within your CSS code (for example taking a colour variable and making it slightly darker by adding to the RGB value)
- Namespaces: which are groups of styles that can be called by references (rather than requiring several CSS files)

Less

While both Less and Sass are pre-processors for CSS, Less has been greatly influenced by Sass. This is very apparent in the shared functionalities of the two. One difference between the two is that Less is a JavaScript library and is processed client-side. Being a JavaScript library it is incredibly easy to incorporate into a web based application. All that is needed is two extra lines of code in the HTML file, one referring to the .less file and one referring the less.js file. [14]

Sass

As stated earlier the one significant difference between Less and Sass is that Sass is not a JavaScript library, it instead uses Ruby. However it seems as though this is not a big deal at all as if I were to develop using either of these frameworks I wold have to be learning new syntax anyway. [15]

It seems as though if we do decide to use either of these frameworks it will most likely come down to personal preference as the differences between the two seem to be minimal. However we will probably be ignoring these for the start of our development as we will be wanting to only focus on the most necessary functionalities.

2.6 Mapping API

It is of immense importance to get the mapping technology that will best suit our project. The primary properties we will need from our mapping API is an attractive interface, capabilities to effectively send user input to, and retrieve and map data from a server's database. We would also really like to have easy and flexible movement options for the user (in particular moving the map location and zooming), and a fast overall performance. Although the mapping API ArcGIS has already been chosen by Opus for this project (as for all mapping-based projects within Opus ArcGIS has been used) I will still review this API in hope that I will gain a better understanding on how I will end up developing with it.

ArcGIS

The ArcGIS engine allows for adding dynamic mapping and geographic information system (GIS) capabilities to both existing applications and custom built mapping applications. Some very useful features include creating custom and prebuilt drawing/graphics features, such as points, lines, and polygons. These graphical features are not just for show, ArcGIS offers powerful manipulation and geographic operations on these shapes, such as calculating differences, finding intersections, and even assigning points on a map to database objects. This is exactly the kind of functionality we are looking for in our HazApp project. As well as the interfacing side of the mapping technology, ArcGIS also offers network analysis which is another crucial part to the success of our project. All this is very well documented and there are many demos on their site which will most definitely speed up my learning process as so far I have had no exposure to ArcGIS. [16]

In terms of the visuals of the mapping, ArcGIS seems to offer an abundance of choices. Again this is a massive positive as we can customise how the map looks to best represent the data, and maximise the ease of use for users. I will be reviewing some of the more specific visualisation options later in the planning phase.

3. Planning and Design

Now that the majority of decisions have been made on the technologies we will be using to construct HazApp we now have the opportunity to move onto planning specifics of the project. Again, seeing as we are at an early stage in development, it is very important to create a sound foundation before development starts. Our planning and design phase will cover the underlying database design, a basic user interface design, and an initial plan for our development phase.

It should be noted that while we are still only in the planning and design phase any decisions made here are subject to change.

3.1 Database Design

The decision has been made for this project to be implemented using a PostgreSQL database [17]. The reason we have chosen to go with a more typical SQL database rather than a NoSQL database is due to the relational nature of the data we will be storing. The figure below shows our first ERD (Entity Relationship Diagram) including the fundamental tables, attributes, and relations.

To begin the planning we started with mapping the most important entity in our database: the Hazards. From this we expanded the database outwards, keeping in mind the relationships that would follow the new tables. The most important tables that we identified were: Hazards, Projects, Users, Lookups, Attachments, User_history, and Sync_history. While the Hazards, Projects, and Users tables are fairly self-explanatory, I will be briefly covering a few of the details of the less obvious tables. To start, the Lookups table is to help define and categorise specific hazards. Also closely related to the Hazards table is the Attachments table, this is to be used for linking photos/videos with reported hazards. And finally, User_history and Sync_history are there to offer some form of documentation on users and to help synchronising the database (for example in times of a lack of internet connectivity).

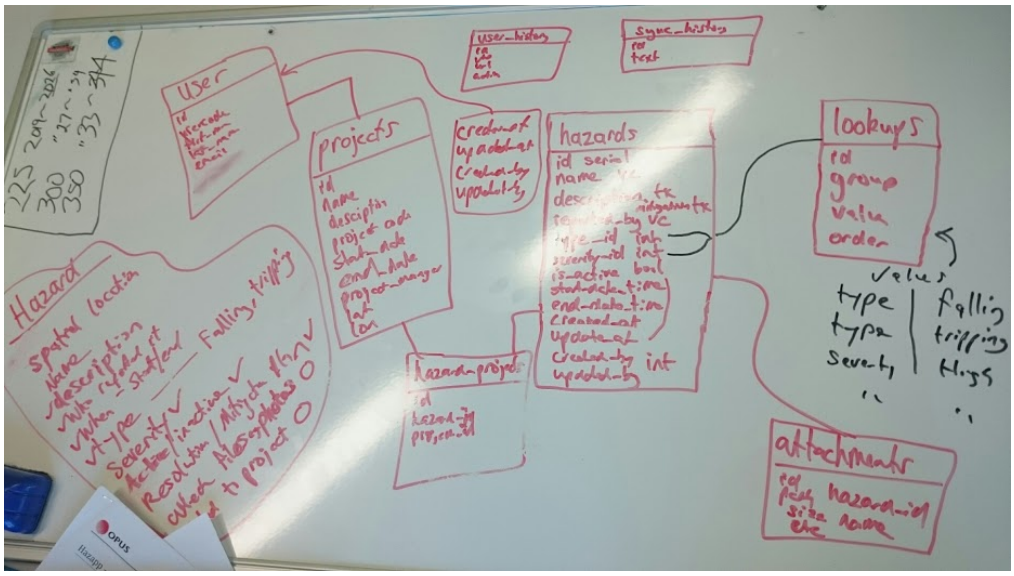

Figure 1 - First ERD draft

As this was our very first draft of the database design there was a lot of room for change. Through several iterations of reviewing the ERD we finalised the design (as shown below) with several changes. These include but are not limited to the addition of attribute variables, further defining the

relationships (as well as the relationship types), and the addition of new relational tables (in this case the table User_project which is used to resolve the many-to-many relationship between the User and Project tables).

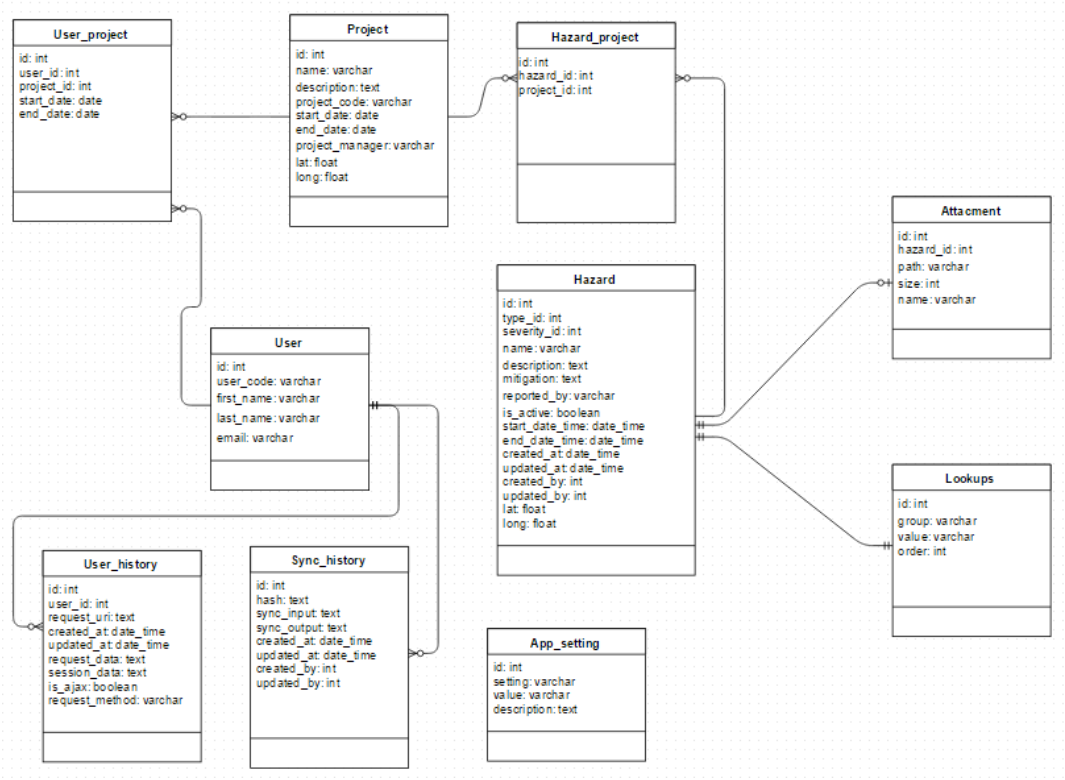

Figure 2 - First ERD

This ERD represents the basis of our database for the first implementation of our project. After creating it and going through several iterations (using draw.io [18]), we have now converted this design into a PostgreSQL database using pgAdmin 3 [19]. This allows us to quickly and easily edit the database through a simple interface, and it also has allowed me to set up a local version of the database on my machine (for testing purposes).

3.2 User Interface Design

The user interface is not the highest priority at the moment, however it is important to create an easy to use interface that the first testers (Opus employees) will be able to understand and use effectively. For our first prototype not much time will be allocated into making the user interface look “nice.” This time will instead be put towards the primary functionalities of the application. With that being said, the first prototype will ultimately be used by a group of Opus employees, and as such we don’t want the user interface to have any negative connotation due to the way it looks. It should be as simple and clean as possible.

The following figures are mock ups created at the very first idea proposal stage and were used as a tool to aid HazApp’s application process. While they are slightly dated and a lot has changed since my involvement in the project, I will be using these as a very rough template for the user interface design I will be working on.

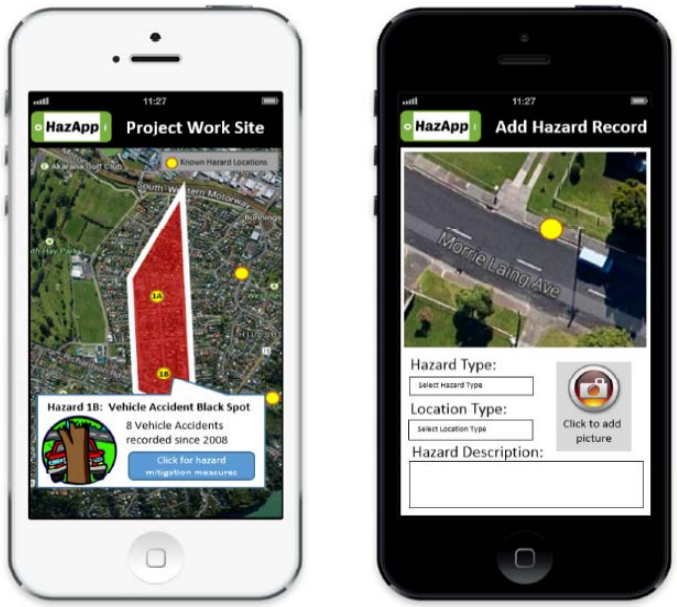

Figure 3 - Mobile user interface prototype [1]

I really like the idea of having a full screen map view that showing the local area's project sites and hazards. Due to the limited screen sizes of mobiles, maximising the efficiency of space by minimising the amount of unneeded clutter is a must. Similar to the above figure I will also include a small hazard form popup that will take up about half a screen when someone wishes to view or log a hazard. This should still leave enough room for the user to accurately select the correct area on the map.

3.3 Progress Plan

With a more concrete game plan forming we are now working on a scheduled plan for the first release of the mobile and desktop application. The table below is our finalised development plan for this stage. It should be noted that this plan covers all aspects project and there will be some areas that I will not be directly involved in (mainly the desktop side of the application). This is primarily to allow me to focus purely on the mobile side of the application. In terms of general development the areas that are of critical importance to me are the database design (which will be implemented as a PostgreSQL database), accessing the database and performing simple CRUD (Create, Read, Update and Delete) operations using PHP, the ability to add hazards to the database using a mobile interface I will be creating (this should also incorporate the mapping technology ArcGIS), and the ability to retrieve all local hazards from the database and display them via the ArcGIS map on the mobile application.

Type	Feature	Description	Iteration (Week)	Due Date
Design Stage	Database	Design database (ERD)	0	28th April
Project Initiation	Planning	Low level task assignment and plan	0	29th April
First Release Dev	Database	Setup dev, uat databases with Postgis	0	6th May
First Release Dev	Database	Implement database	1	13th May

First Release Dev	Base Application	Base CI app with required libraries etc	1	13th May
First Release Dev	Base Application	Mapping requirements	2	20th May
First Release Dev	Base Application	Theme, styling, look / feel. mock pages	2	20th May
First Release Dev	Adding a Hazard on Desktop	CRUD for hazards	3	27th May
First Release Dev	Hazard List	Prepare a list of the hazards in the system, with an ability to search and query	3	27th May
First Release Dev	Base Application	Configure to authenticate with AD	4	3rd June
First Release Dev	Base Mobile Application	Base app setup for the HTML5 WebApps	4	3rd June
First Release Dev	Mobile App Add Hazards	Ability to add hazards on the mobile interface	5	10th June
First Release Dev	Mobile App View Hazards	Ability to view all hazards on the mobile device (using GPS as well)	5	10th June
First Release Dev	Hazard Map	Map view of all hazards with the additional of other spatial queries	6	17th June
First Release Admin	UAT Feedback	To prepare users for UAT and any adjustments before commencing full pilot	7	24th June

Table 1 – Progress plan for first release

As seen in the table above, the final due date we are aiming for (at least for our initial prototype) is the 24th of June. At this point we hope to have a working desktop and mobile application that can be tested in a real world environment by a small group of Opus employees. As it is the first prototype we will only be focussing on the most important aspects of the application that we deem necessary. This means we will not be implementing functions such as offline use or statistical visualisations until a later release. Assuming all goes well the first prototype will be the most important part of the project so far. It will enable us to truly see if the idea is viable, and it will definitely highlight aspects that are weak in the design.

Chapter 2 - Development

4. First Prototype

After the successful completion of our planning phases we will now be beginning the development on our first prototype. As mentioned in the project plan, I will primarily be focussing my development on the mobile side of the project, and as such this chapter will be following development from my point of view. Throughout this chapter I will be making reference to the desktop side of the project however my involvement in that part of the project may be limited for the time being.

The first working prototype will be due at the end of June. By then I hope to have the mobile application accessing the database servers and preforming simple CRUD operations for hazards, a basic form for adding hazards, and utilisation of the ArcGIS mapping technology which will show localised hazards and have some functionality for reporting of hazards through a geospatial navigation interface.

As this is only the first prototype and we have a limited time frame before the first iteration of user testing commences, several “luxuries” will be left out depending on time. As mentioned earlier we will not be bothering applying any functionality for offline testing at this stage and I will most likely not be spending a lot of time on the user interface and other things like fancy transitions between screens. As such I will probably be ignoring the mobile web app frameworks, stylesheet languages, and will only be using JavaScript frameworks where absolutely necessary.

It should be noted that due to a differential between the submission time for this report and the time for our first application testing phase, several of the above features may not be completed by the time this report is completed. With this in mind the final area in this report will summarise what will be completed in the coming days (leading towards the first deployment).

4.1 Accessing Database using PHP

While I have had some experience with HTML, JavaScript and CSS, this is my first exposure to PHP programming. Fortunately I have also had some experience with SQL databases and the functions of such databases. Because of this PHP for database management felt very familiar, and no real difficulties were encountered when trying out the basic CRUD operations. One minor setback I did encounter was when trying to access the database tables. At first I did not realise that when using PostgreSQL queries in PHP, all table names in single quotations were automatically converted to lower case. This essentially meant my PHP code was not recognising any of the tables in my queries (as they were defined in the database with a capital first letter, for example ‘Hazards’). After some minor confusion I contacted one of the PHP experts at Opus who informed me of the problem, and a

few solutions. One being changing the table names in the database to be all lower case, and the other to slightly alter my queries to include an extra set of quotations around the table name so that it keeps its case.

I have included below the two most important aspects of using PHP to access the database; creating a connection to the database, and defining a query (in this case a simply lookup). [20]

```
$db = pg_connect("host=localhost port=5432 dbname=postgres
user=postgres password=password")
```

Figure 4 - Connecting to the PostgreSQL database using PHP

If you notice in the above code I have set the host to localhost, I have done this as while we are yet to deploy our first version, I have been testing my code using a localised copied version of the database. To access the database through the localhost on my computer I have used the free database connection program XAMPP [21] which opens specific connection modules such as Apache or MySQL. When we release the first version of our applications to some Opus testing staff I will instead be connecting to a server database created by the company.

```
$query = 'SELECT * FROM hazards;
$result = pg_query($query)
```

Figure 5 - A simple SELECT PHP query

This PHP query is the most simple of queries, all it does is get all of the entries in the 'hazards' table and stores them in a result. For testing purposes I continued this followed this code with a few simple echo statements to view the data.

For purposes of error handling and safety I have also included catches to handle and display any errors that may occur while either connecting to the database or querying the database. Again this is very simple to do but it holds massive benefits in terms of resolving errors in the early development stages. This is done within a single line that follows either an attempted connection or query (as seen below):

```
or die("Error: ".pg_last_error());
```

Figure 6 - Basic PHP error handling

4.2 Creating the Mobile Side Using HTML5

The second piece of the mobile prototype is the actual HTML5 application, it will essentially be the interface between the user, the map, and the database. Due to its complex nature I have separated this task into 4 smaller subtasks. These are; the ArcGIS map, the HTML5 form, adding hazards to the database via the HTML5 form and a PHP connection, and retrieving and displaying local hazards from the database onto the map.

ArcGIS Mapping

While I have had some experience using enterprise mapping API's (in particular Google Maps), this is my first experience using Esri's ArcGIS mapping technologies. As such I have had to put in quite a lot of time towards learning the API as there are a lot of different functionalities that I will have to be utilising to effectively solve the problems. To start, I simply tried to get a map displayed within my

testing browser. This was done purely using JavaScript and some of Esri's libraries [16]. The code snippet I have included below is the most basic of maps which is located above Auckland city.

```
require(["esri/map", "dojo/domReady!"], function(Map) {
  var map = new Map("map", {
    center: [-174.7400, 36.8406],
    zoom: 8,
    basemap: "topo"
  });
});
```

Figure 7 - Basic ArcGIS JavaScript code

The most important aspect to using ArcGIS through JavaScript is the first line of the above code. The '`require`' keyword defines what libraries are to be downloaded from Esri as well as the functions that will operate on the given map. This can be expanded immensely to cater for all different functionalities used by the ArcGIS map.

The first functionality that I have decided to add is the ability to zoom to the current device's location (using Geolocation to get a latitude and longitude value). The reason I have prioritised this functionality so highly is because even for this first prototype, the users will be wanting to log hazards using their mobile devices while on work-sites. By allowing them to automatically zoom the map to their current location by the click of a button, Opus employees will be able to quickly locate the area where they wish to report a hazard and can easily view all hazards in the local area (to be completed later).

In terms of the map style I want to have something that is clean and not overly complicated, but also shows enough detail of streets, building groups, and environmental areas to not only be visually appealing, but also practically useful.

Below are a few of the alternative map styles I have considered. While there are many more map styles available (both created by Esri and created open source), these were the four that I found to best fit the design requirements.

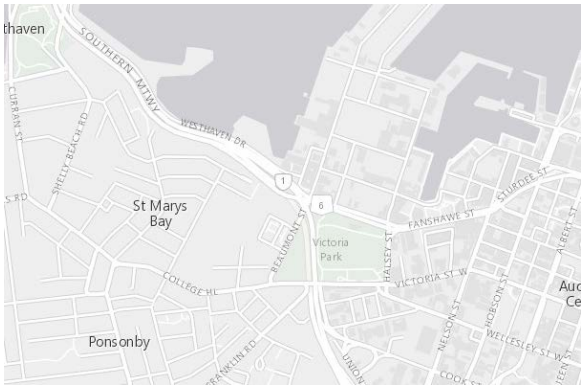

Figure 8 - Gray view

Figure 9 - Hybrid view

Figure 10 - Topo view

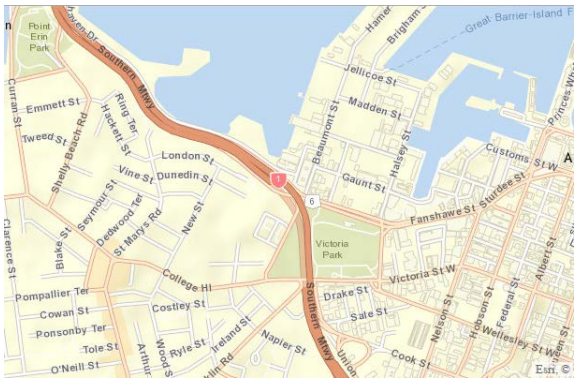

Figure 11 - Streets view

My decision was narrowed down to either the 'streets' view or the 'topo' view. Both ultimately function very similar but have slight differences that I have used in my decision making process. The 'streets' view offers more detail of roads; it has several different shade to categorise the different types of roads (for example the darkest road in the above image is State Highway 1). Whereas the 'topo' view does not focus so heavily on the roads (although it does show them in adequate detail), instead it offers better visualisations of building areas. In the above image it is clear what area is more residential (the left half) and what area has a higher density of larger commercial buildings (the right half). While both the 'streets' view and the 'topo' view have only minor differences, I have decided to go with the 'topo' view as I believe it gives enough detail of roads, it is clean and easy to look at, and the building density shading system will be massively beneficial to users when locating particular work sites.

HTML5 Form Creation

The form view for HazApp is vitally important to the project as with a broken or unusable form, no data may be logged, therefore the very core idea of HazApp won't be able to function. While there are several aspects to the form view already, below I will be covering the most important areas and those which I had the most difficulty with.

Hazard Type Input

Crucial to reporting hazards is the form detailing the specifics of a given hazard. I will be reducing the possible inputs to only those I consider a necessity. First and foremost is a hazard categorisation input, which will give the option to choose a pre-set hazard type. The hazard categorisation input is based off the databases Hazard Type Lookup table. The reason we have decided not to allow users to input a custom hazard type is that the hazard type list needs to be up to Opus' health and safety standards. As such users should only be able to select hazard types of an approved list. Another benefit from having a fully defined hazard type list to choose from is that statistical analysis will be far easier to process and far clearer to understand, and it will also open avenues for things such as symbol filtering on the map view (which may be implemented later). Fortunately the selectable inputs will be easily modifiable in the future as all that is necessary is the addition of a new column into the PostgreSQL database.

Latitude and Longitude Fields

This will be followed by the location (latitude and longitude) of the current hazard, which will either take the values from the area selected on the map or another value directly input by the user. Seeing as the latitude and longitude fields are currently editable (this may be changed later down

the track), It is very important at this stage to make sure that they have the correct input validation as any malformed input will make that particular hazard unusable in the database. To do this I have used a Knockout binding. This particular Knockout binding limits the potential input keys available to the user when they have the latitude or longitude field. As seen below, it only allows the simple editable inputs (such as backspace, delete and enter), and the numerical inputs (including negative numbers as well as decimal points).

```
// Custom binding to limit a text input to a floating point number
ko.bindingHandlers.numeric = {
    init: function (element, valueAccessor) {
        $(element).on("keydown", function (event) {
            // Allow: backspace, delete, tab, escape, enter, and -
            if (event.keyCode == 46 || event.keyCode == 8 || event.keyCode == 9 || event.keyCode == 27 || event.keyCode == 13 || event.keyCode == 189 ||
                // Allow: Ctrl+
                (event.ctrlKey === true) ||
                // Allow: . (decimal point)
                (event.keyCode == 190 || event.keyCode == 110) ||
                // Allow: home, end, left, right
                (event.keyCode >= 35 && event.keyCode <= 39)) {
                // allow input
                return;
            }
            else {
                // Ensure that it is a number and stop the keypress
                if (event.shiftKey || (event.keyCode < 48 || event.keyCode > 57) && (event.keyCode < 96 || event.keyCode > 105)) {
                    event.preventDefault();
                }
            }
        });
    }
};
```

Figure 12 - Knockout numeric binding

This binding is then simply applied to both the latitude and longitude fields by including:

```
data-bind= "numeric”
```

within the input tag.

Calendar Inputs

It is very important from a usability point of view to offer some form of date picker for the calendar inputs (start date and end date). While HTML5 does offer a very simple and easily implementable date picker via:

```
input type= "date"
```

it unfortunately is not supported in Opus’ company standard browser (Internet Explorer 10). As such I have had to find another option for a date picker.

I have decided to go with JQuery UI’s date picker which can be implemented and modified using Knockout. As seen in the code snippet below, I have decided to use two primary functions within the date picker; ‘init’ (to initialise the date picker itself as well as set the current date for the start date field) and ‘update’ (in the event that the user wishes to input/change a new date).

```
// Custom binding for a datepicker input field
ko.bindingHandlers.datepicker = {
    init: function(element, valueAccessor, allBindingsAccessor) {
        //initialize datepicker with some optional options
        var options = allBindingsAccessor().datepickerOptions || {},
            el = $(element);
        el.datepicker(options);
        //handle the field changing by registering datepicker's changeDate event
        ko.utils.registerEventHandler(element, "changeDate", function () {
            var observable = valueAccessor();
            observable(el.datepicker("getDate"));
        });
        //handle disposal (if KO removes by the template binding)
        ko.utils.domNodeDisposal.addDisposeCallback(element, function() {
            el.datepicker("destroy");
        });
    },
    update: function(element, valueAccessor) {
        var value = ko.utils.unwrapObservable(valueAccessor()),
            el = $(element);
        //handle date data coming via json from Microsoft
        if (String(value).indexOf('/Date(') == 0) {
            value = new Date(parseInt(value.replace(/\/Date\((.*?)\)\//gi, "$1")));
        }
        var current = el.datepicker("getDate");
        if (value - current !== 0) {
            el.datepicker("setDate", value);
        }
    }
};
```

Figure 13 - Custom Knockout binding for date picker

As demonstrated in the picture below, a user simply selects the date field (in this case the active date) and the date picker appears below. Then instead of having to input the date numerically they can simple select the desired date on the popup calendar. I believe this makes the date selection task significantly easier for the user, and it also negates any need for custom input validation as the user cannot directly manipulate the input via keyboard.

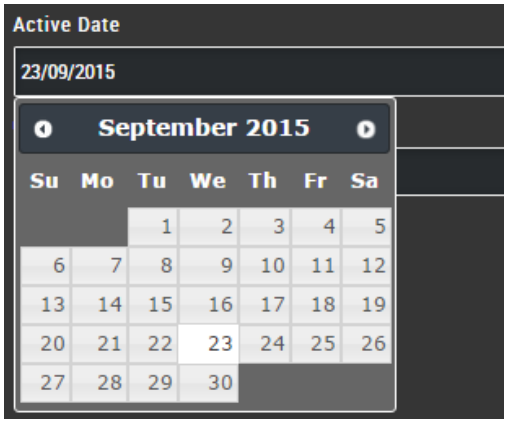

Figure 14 - Date picker user interface

5. Second Development Phase

After successfully completing the first prototype many ideas have been refactored, most importantly the desktop and mobile side for the hazard reporting has been decided to merge into a single application. This combined application will primarily be targeting the screen resolution of mobile tablets as this is what Opus' hazard site managers will most often be using. With that in mind, we will also have to cater for both larger screens (i.e. desktop computers) and significantly smaller screens (i.e. mobile phones). As such we will be primarily be utilising Bootstrap's functionality of auto adjusting widgets to create a responsive web design.

To stay within Opus' software development standards we will now be managing our code base using CodeIgniter, which is an open source web application development framework used in building dynamic sites with PHP. This means that rather than having a few colossal code bases such as a single html file, one CSS file, and one JavaScript file, we instead split the development areas into better defined and more manageable parts. For example we will be having a separate file for the ArcGIS mapping JavaScript, the Bootstrap JavaScript, and the main JavaScript (which includes Knockout). While this has been a challenge adjusting to, it ensures that after I have completed this project the application will be up to Opus' software standards and as such will still be able to be maintained and updated by any of Opus' software developers.

This next area will be covering the finalised development of the hazard reporting side of the application as I will be expanding on the ideas and techniques used for the first prototype. It will be highlighting all significant points we have faced all the way from the base application to the final product.

5.1 Base Application

Getting the base application up and working holds the greatest of importance at the current stage. While we will not be using the first prototype as a skeleton for further development, many of the techniques and concepts gained by creating the prototype will likely prove invaluable for this next stage. It should have the core functionality working before we develop any of the more intricate functionalities. The core functions needed include:

- Having a map view which shows all of the current hazards in the given area (the hazards and their information will be pulled from the PostgreSQL database and superimposed on the ArcGIS map).
- Having functionality to select a particular hazard on the map view to identify its specific information (for example the hazard type, severity, description, when it was logged and who by).
- Having functionality to add a new hazard to a specific area on the map view by clicking on the chosen location.
- When adding a new hazard, a hazard form needs to appear which will allow the user to log the additional information about the hazard.
- Some simple form validation so that incorrect data (which could potentially break the database) cannot make its way into the database.

- Some simple photo/video attachment option.
- Having functionality to submit the hazard and its form to the PostgreSQL database so that it will show up as a new hazard on the map view.

While many of these seem trivial and/or very obvious to include in such an application, it is of much importance that we map out exactly what is needed so that later down the road we don't realise we have missed a critical functionality (which may be significantly harder to implement later rather than earlier in development). By having all of the listed functionalities working as expected, we will have a strong foundation of which we will hopefully have ease implementing some of the more difficult functionalities which will be coming later.

I will now be covering over the core functionalities in more detail, focussing specifically on any points of interest and any surprises or difficulties I have encountered. I will also be including some of the more influential screens so that comparisons between old and new user interfaces may be easily made.

Map View

It is of vital importance that the map view is not only easy to understand and visually appealing, but also offers all the expected functionality of being to add and view different hazards. The base map view tab has maximised the size of the map itself so that users (especially using devices with smaller screens) can get the most information possible. The ArcGIS map offers all the normal functionality of a digital map (such as relocation and zooming) but it also allows for different hazards pulled from the PostgreSQL database to be treated as entities on the map. This not only allows them to move and zoom with the map, but it also offers other functionalities such as selecting a particular hazard and viewing its information or even editing it. As seen in the screenshot below, all of the currently active hazards in the designated area are shown, however at the present time the hazards offer very little information on their own other that *where* the hazard was reported. If a user wanted to see what a particular hazard's 'type' was, they would have to open the hazards information dialogue. This will most definitely be improved in the future as I believe the hazards on the map view have the potential to offer a lot more information on their own. For example by changing the colour of the hazard symbol to represent different severity levels or by changing the hazard symbol itself to represent different hazard types.

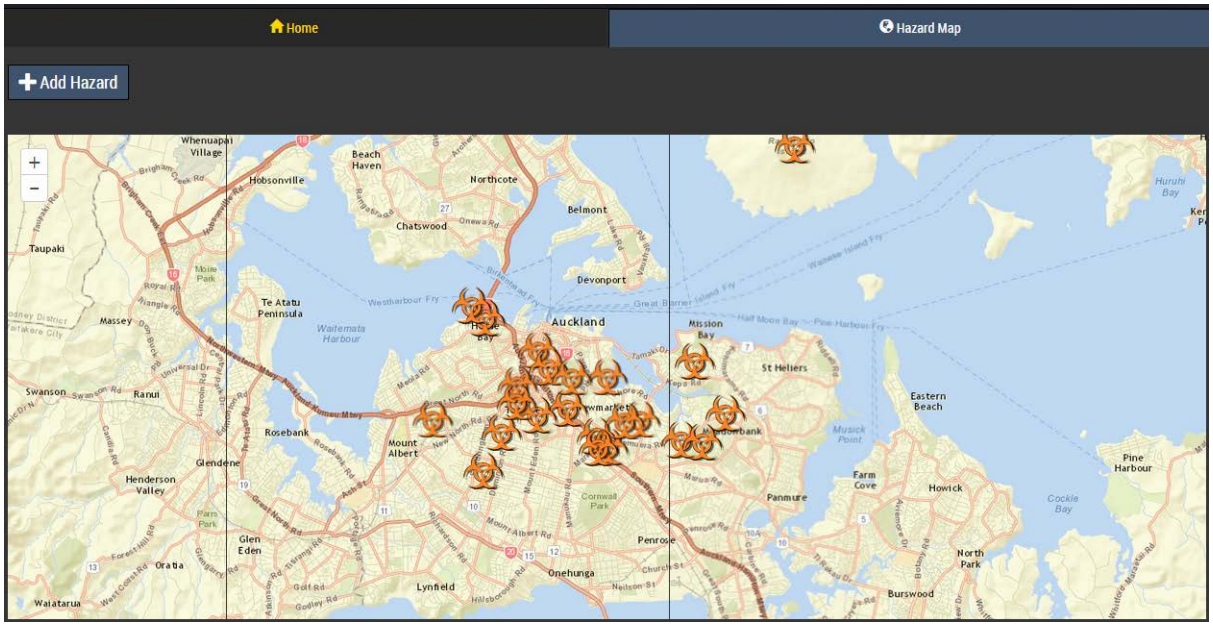

Figure 15 - Map view version 1

Information Dialogue

While the map view does have some potential to give additional information such as severity or type, it is still necessary to be able to view additional information about a particular hazard. This is where the information dialogues come into play. Currently the information dialogues pull all attributes and attribute data of a particular hazard from the PostgreSQL database and display it in a very simple alert popup. As seen below this is unattractive and difficult to read. While it is just a filler for the meantime, it will definitely be changed to something more visually appealing and easier to understand. We will most likely be using a Bootstrap modal which follows our user interface theme and some filtering will be necessary (i.e. removing any unhelpful information).

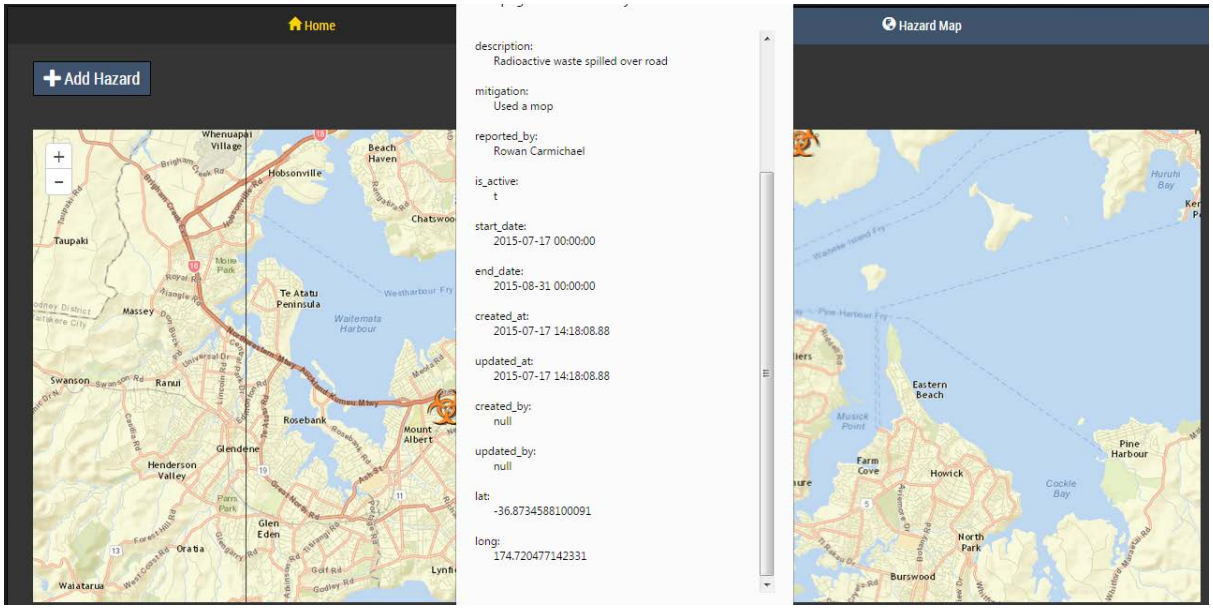

Figure 16 - Hazard dialogue version 1

Unfortunately seeing as we have utilised lookup tables for hazard type and severity, the information shown about these two attributes is only a reference ID to another table. As such instead of showing

the hazard type as ‘mechanical’ it would instead simply display ‘2’. Again this is something that will need to be fixed before we are ready to push it out for the first round of user acceptance testing. Fortunately this shouldn’t be too difficult of a task, all that is needed is some SQL code which will create a new view which contains the appropriate information.

Hazard Form

As seen in the two screenshots above, we have implemented a simple ‘Add Hazard’ button to the map view. Once clicked a small prompt appears telling the user to “Click on the map to add a new hazard” and once the user has done so it changes the tab to the hazard form. The hazard form (shown below has some very simple inputs for all the necessary attributes. At the moment all of the attributes in the hazard table have inputs in the hazard form, however in the not too distant future we will be evaluating whether or not all of these fields are necessary. So far we plan on replacing the latitude and longitude fields with something that is more easily understandable to humans, such as a smaller version of the map view or perhaps a dropdown list of active project sites. By the time we roll out for the first stage of user acceptance testing we will also have a log in system so the “Reported By” field may become redundant. And finally, we intend on removing the basic photo/video attachment button with a function that allows a user to capture a photo/video within the application if using a device with a camera (rather than having to capture it outside the application, then save it, and then select it as an attachment). As it stands I don’t believe anyone will bother going through all the additional steps, and as such this usability problem needs to be solved.

Figure 17 - Hazard form version 1

At the moment the only necessary fields are the location (latitude and longitude), the hazard type, and the reported date. Once these and any of the optional fields are completed, the user may submit the form to the PostgreSQL database and it should be instantly available for all other Opus employees to view on the map view.

5.2 User Interface Improvements

After the successfully completion of the base application version 1, it is clear there is a lot of room for some user interface improvements. In the previous section I mentioned some of the more obvious changes that were required and in this section I will be highlighting how we went about solving those (and any other) user interface problems.

The changes outlined below are in preparation for the first stage of user acceptance testing, which I will be covering in more detail in a later section.

Map View

Seeing as the map view is so influential to the entire application, it is of much importance to make it as visually appealing and usable as possible. The first change we made to the map view was differentiating different hazard types by using different representative symbols. While the symbols have been designed to be easily understandable, we have also included a legend to clarify any misunderstandings.

The symbols themselves have been based on the hazard type lookup table in our database. While many have been designed, and many more will likely be designed in the future, the screenshot below shows the new map view with the new symbols and the legend side tab (which can be expanded and collapsed at any time). When compared with the map view version 1 screenshot, it is now clearly more readable and visually appealing.

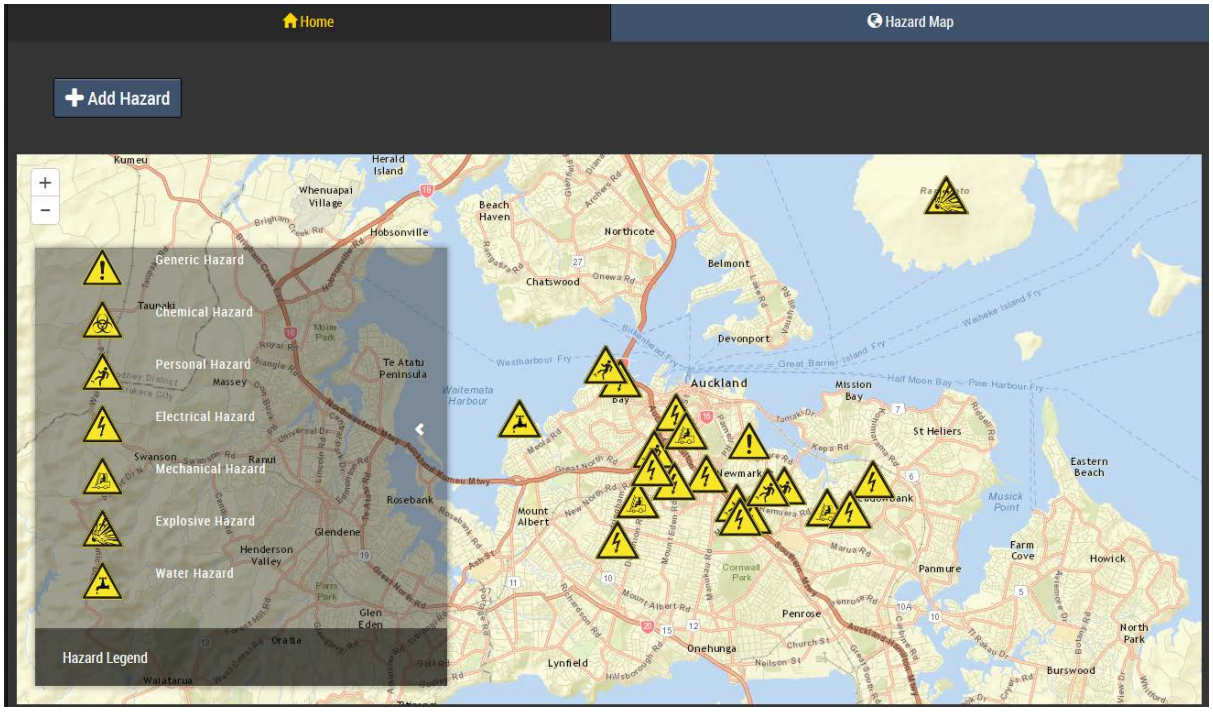

Figure 18 - Map view version 2

Information Dialogue

The old information dialogue was definitely in need of a clean-up of both what information was shown and how the information was shown. We have improved the old hazard information dialogue into a new hazard information modal. A decent amount of the unnecessary information has been

removed and the design has been upgraded to a Bootstrap modal. This way the user interface can more easily align with our overall design. As seen in the screenshot below the new modal also shows a visualisation of any attached pictures which were submitted with the hazard. Again this is further increasing the richness of information available, while reducing any distracting or unnecessary factors.

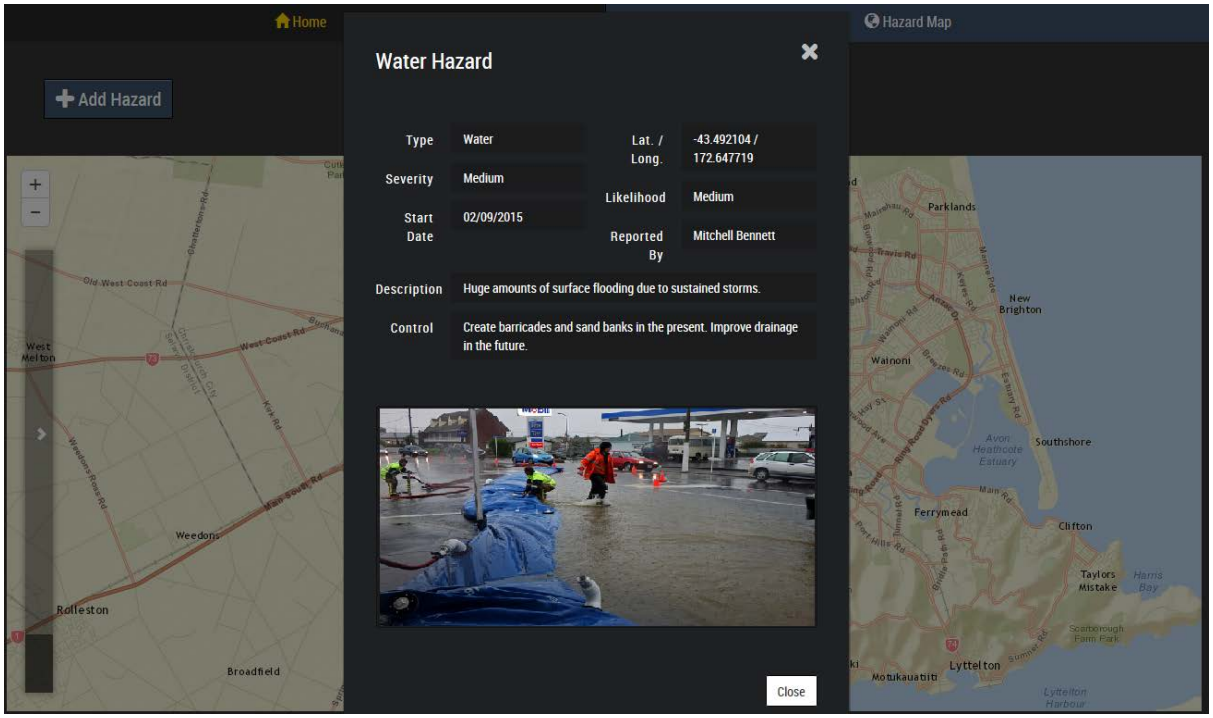

Figure 19 - Hazard Information version 2

Hazard Form

One of the more important improvements made was a suggestion from the business/health management team. They have stated that the old severity field does not align with their business safety standards. In particular, their safety scales are based on a combination of severity and likelihood of an event happening (the scale is shown in *item 2* of the appendix). As seen in the screenshot below, we have included this suggestion by creating a two dimensional grid which combines a severity scale and a likelihood scale.

The newly refactored hazard form also include the mini-map of the location where the hazard has been reported. This is significantly easier to understand than what we had before (which was simply numerical values for the latitude and longitude fields). As an added bonus the new mini-map only shows the symbol of the hazard which is currently being reported. This is especially useful if a user is reporting several hazards at the same site or reporting a hazard in an area which is already highly populated with hazards. By only showing the hazard that is being reported, the user can more easily identify whether or not the location selected on the map view is correct.

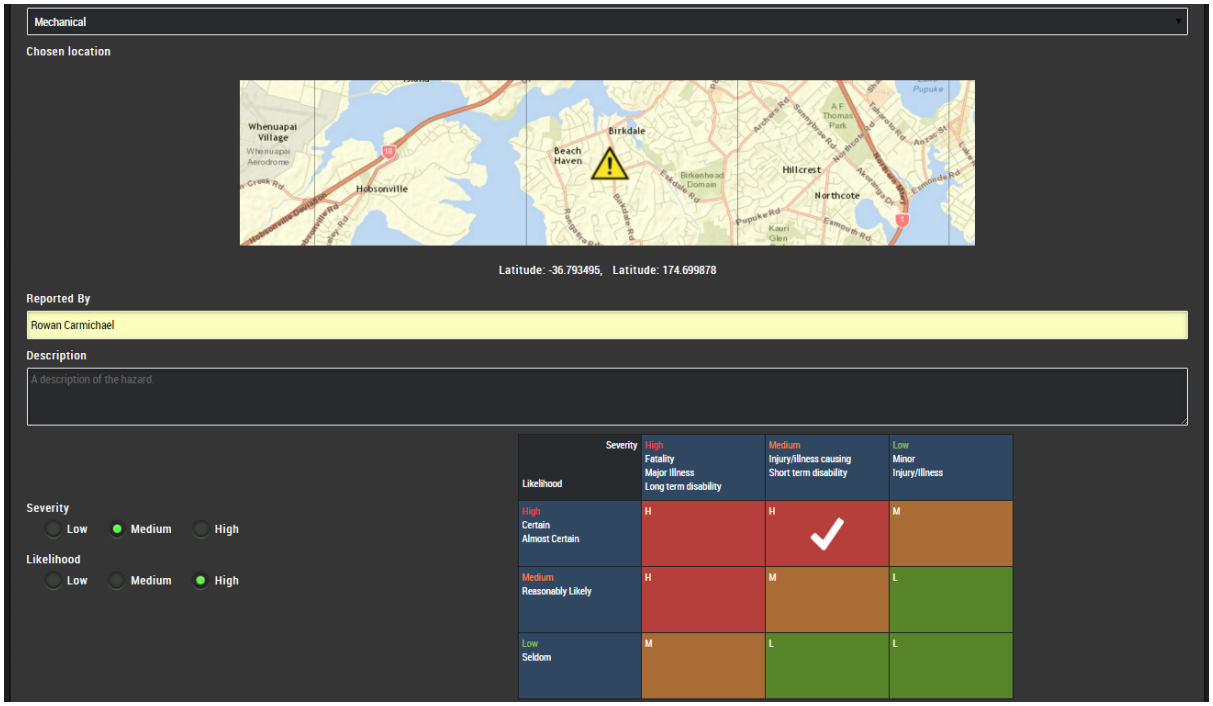

Figure 20 - Hazard Form version 2

6. UAT Development Phase

At this point in development we are now ready to put our current version of HazApp online so that the first stage of user acceptance testing may begin. In the following section I will make reference to some of the feedback we have received from the user acceptance test and how we have implemented the necessary changes. This section will primarily focus on the actual development done due to the feedback from the user acceptance testing, rather than focussing on the user acceptance testing itself (which will be covered more thoroughly in a later section).

Some of the most immediate feedback received was about very small bugs such as incorrect geolocation tracking, and small features such as capturing photos within the application. In this development phase we will not only be making changes based off the user acceptance test feedback, but also several intended improvements as well. Most notably, the beginning of the data analysis.

6.1 Correcting the Geolocation Zoom

An interesting bug that became apparent after giving our application to the user acceptance test participants was that in some instances when the ‘auto-zoom to current location’ function activates, the GPS recognises that the user is somewhere where they aren’t. More specifically, this only happens when users use HazApp within the Opus network (i.e. desktop computers in Opus offices). After noticing this problem, recreating it consistently, and discussing it with some of the other software developers at Opus, we have come to the conclusion that it likely has something to do with the way in which the desktop computers at Opus are set up on the Opus backbone network. The network itself begins and ends in Christchurch which is the location it automatically zooms to in

these instances. While this seems like a relatively minute problem as users can still scroll to their desired location on the map (like any normal mapping interface), it is important for us to have the best usability for our application as possible, and as such we felt the need to correct it.

Our first thought was to simply try another geolocator framework, however after testing the problem on a different framework (in this case GeoPlugin) we found we had the exact same problem. Being unable to solve the problem this way we decided to consider other workarounds. In the end we decided to simply include a local bookmarks feature in which users have stored locations which they can easily jump between. Following the consistent design principles, we have made the location bookmark feature in a way that it doesn't unnecessarily distract the user or take up too much screen real estate, while still functioning as intended.

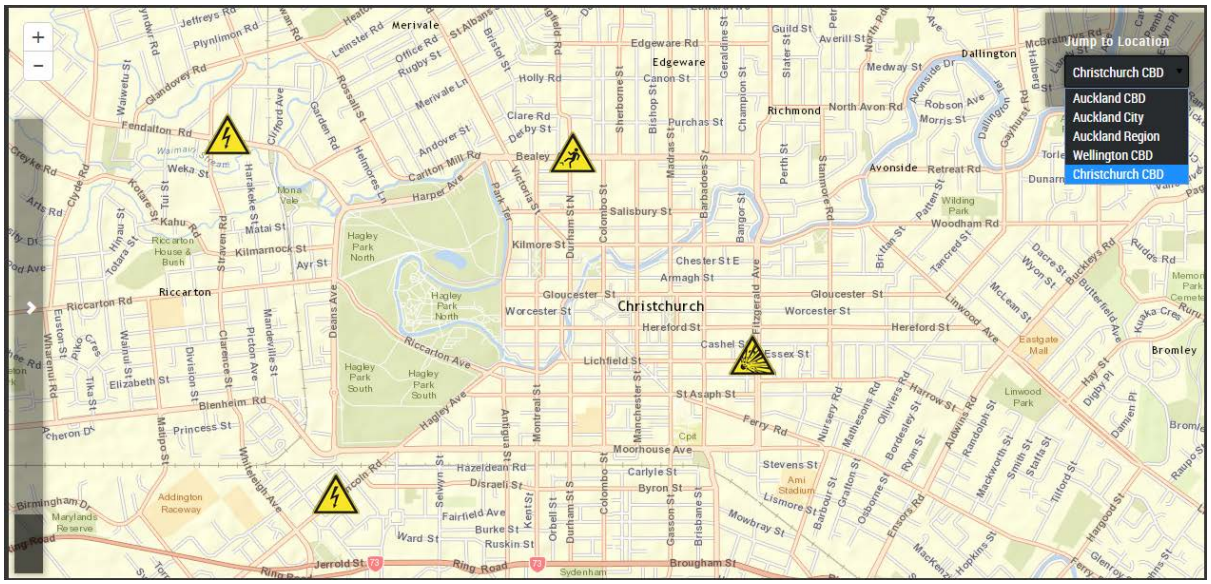

Figure 21 - Map view with location bookmark feature

The location bookmark list can easily be expanded/edited in the future to fit the needs of the users. However at this stage by default it only consists of the three major cities in New Zealand which Opus operates at.

6.2 Mobile Specific Functionalities

When uploading a photo(s) for a particular reported hazard while using a mobile phone or tablet, the user now has the option to either choose an image from their gallery/documents, or to capture a new photo within the application (as seen in the screenshot below). This is important as the vast majority of the hazard reporting will be done on project sites where the users may not have immediate access to a desktop computer or laptop. As such the hazard reporting via smartphone or tablet needs to be as easy as possible, which is why the seamless transition between the hazard reporting form and the photo capturing is so important. This greatly improves the user experience as now the user doesn't have to exit the application to take a picture then return to the application to upload it.

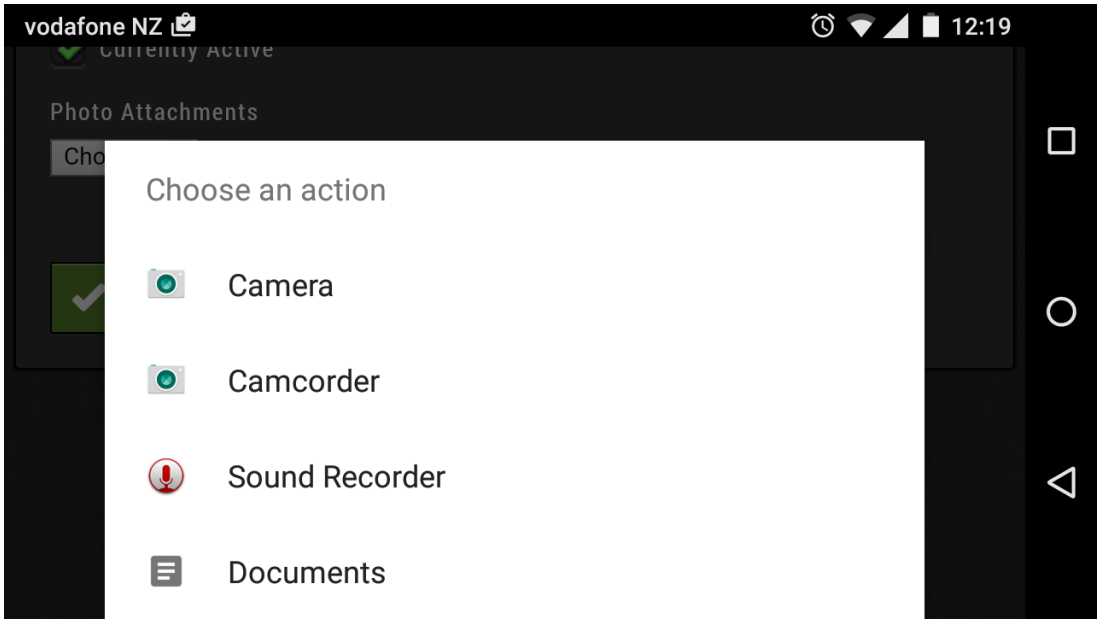

Figure 22 - Adding photo attachment via smartphone

As seen above, when the user selects to add a photo attachment, several options appears. This should be consistent across all mobile browsers and all modern smartphones and tablets, so there shouldn't be any limitations in that aspect.

6.3 Data Analysis Using HighCharts

While the majority of this project has been focussing on the hazard reporting side of HazApp, we have also included the beginnings of the data analysis side of the project. At the time being we have only been analysing the data in ways that seem most useful (for example the hazard type distribution chart shown below). This has been done by feeding the PostgreSQL data to the data analytics JavaScript framework Highcharts. This generates interactive chart visualisations which are significantly easier to comprehend than the data that they have been composed from alone.

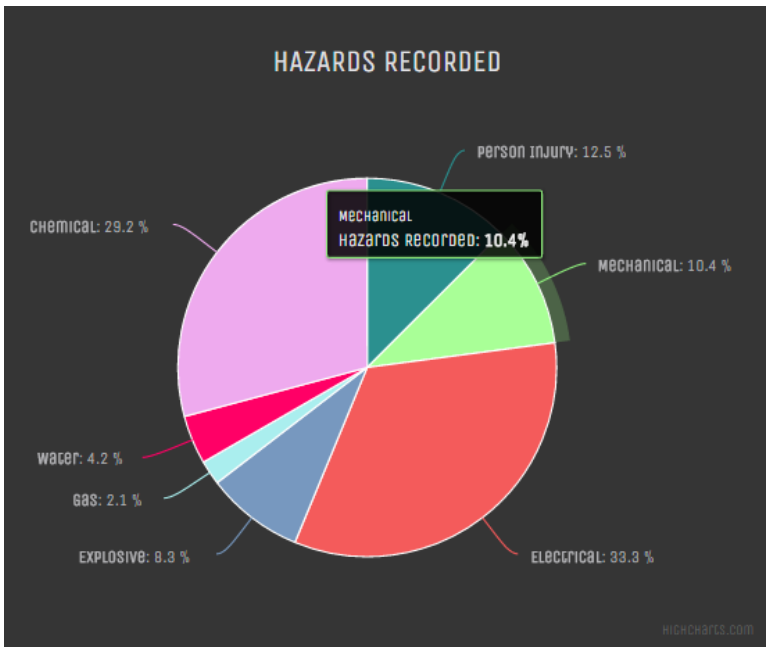

Figure 23 - Highcharts analysis of hazard type distribution

This infographic may be used to highlight areas which Opus needs to deploy further training regimes or better safety practices. For example if 50% of the reported hazards were of the “Personal Injury” hazard type, Opus may need to consider ways on how to combat this.

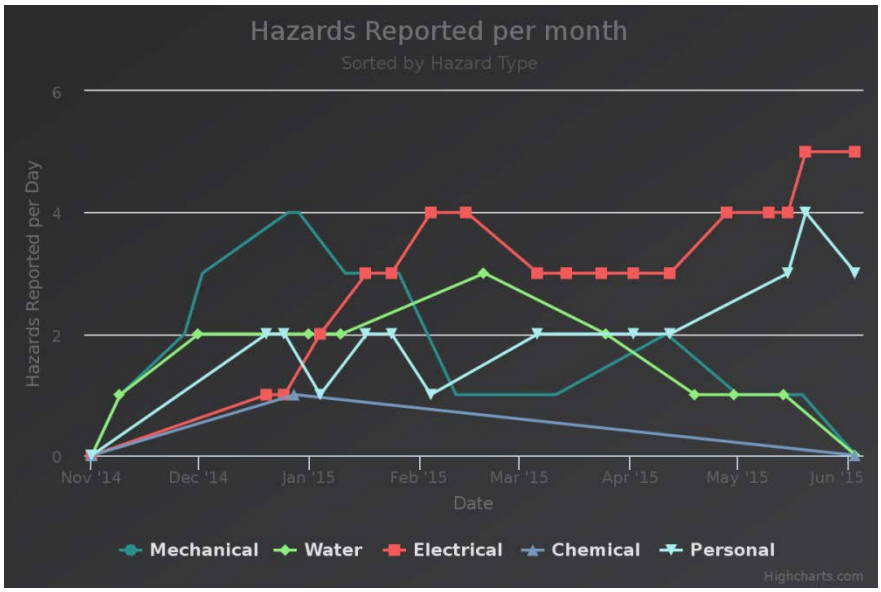

Figure 24 - Number of hazards reported per day via line chart

The analysis side of HazApp also allows a user to view the distribution of reported hazards on a daily basis. By highlighting any single data point the graph shows the exact date in question and how many hazards were reported for that particular type of hazard on that particular day. There is also a filtering functionality in which a user may select a type of hazard(s) to filter out of the graph which will reduce the cluttering. This tool is an incredibly useful way of visualising general trends within a given time period, for example there may be an increase in personal injuries over the Summer period. If this were the case Opus’ managers could be prompted into questioning why and potentially improving work practices to reduce the number of personal injuries occurring over this time. It may also help give insight into the effectiveness of Opus’ health and safety training sessions. For example it would be expected that if new health and safety regulations were to be introduced around how mechanical vehicles/devices are managed there would be a decrease in the number of mechanical hazards reported.

While we have only captured what I consider the most important data analysis forms as of yet, we have also began planning on what other visualisations may be useful to both the average user as well as the managers. This may include:

- As well as a line graph that shows how many hazards have been reported daily within a given time frame, showing how many of the reported hazards have been “solved” daily within a given time frame.
- Dividing major areas up (for example Auckland CBD, Wellington CBD, and Christchurch CBD), so that analysis may be performed and compared on a regional basis. For example are there certain types of hazards that occur more regularly in Auckland CBD compared to Christchurch CDB.

- For each type of hazard, are there differences in the severity and/or likelihood as compared to other hazard types. This may again give insight on which types of hazards are more important to be taught on.
- Are particular project sites reporting an unsafe number of hazards, and if so what can be done to reduce the number (for example is more training on safe business practices needed and/or is the site's safety manager doing an adequate job).

While there are many more features planned for the upcoming development phases, time has been a massive factor which has narrowed my scope to only what I have considered more important to getting my original plans of having a working application up and running.

Chapter 3 – Evaluation and Results

7. Usability

Seeing as HazApp is intended to be offered to all New Zealand Opus employees/contractors and eventually (assuming success) all Opus employees worldwide, it is of critical importance to have the application as user friendly as possible. As such I have decided to compare and criticise our current application against what I believe to be the most relevant aspects of Nielsen's 10 Heuristics and Shneiderman's 8 Golden Rules. From this I hope to find any missteps that may damage the overall usability of the application so that I may attempt to rectify them.

After this I will also be evaluating some of the feedback received from the first stage of user acceptance testing. Similarly to the heuristic evaluations, I hope to find any areas that may be difficult to understand by the average user or any other glaring issues with the application.

7.1 Nielsen's Heuristics

Nielsen's Heuristics is an incredibly influential evaluation tool commonly used in areas of Human-Computer Interaction. It specifically involves evaluators examining the interface and judging its compliance with recognized usability principles (the heuristics). The main goal of heuristic evaluations is to identify any problems associated with the design of user interfaces, and while it is a somewhat informal methods of usability inspection, it can still highlight any potential holes in our application's usability. [22] [23] [24]

In this section I will be focussing on the heuristics that I have found to be most relevant to our project, and evaluating how HazApp stands up to them. They are as follows:

Match between system and real world

To do this we have attempted to make Map view as similar to a real map as possible, this was relatively easy as ArcGIS (much like any modern mapping API) has been created in a way to be as recognisable and usable as possible. We have also included several features that would be found on a normal map as well, for example a legend to better understand the symbols. The form view has also been designed in a way to be as similar to a real hazard form as possible. It contains all the expected fields in order, and utilises radio buttons, dropdown lists, and check boxes to increase the overall usability.

User control and freedom

Having the user locked into a pre-set path can greatly reduce the user's feel of control. As such we have developed HazApp in a way as to never lock the user in to making a decision. At any time can

they may move between tabs without disrupting the system, and they may jump in and out of the form as they please without any repercussions.

Consistency and standards

We have utilised bootstrap and CSS to keep a consistent look over the entire application and we have used consistent wording in the form (for example “Controls” for how the hazard has been mitigated) so that at no point different words, situations, or actions mean the same thing. Also the form view is based upon Opus’ health and safety standards, so any users familiar with such standards will have no amount of confusion.

Error prevention

As HazApp is not an incredibly complex system, error prevention was not that difficult to develop for. Essentially the only area which errors may arrive is the form view. To combat any potential spawns of errors we have primarily used input validation (including image validation/filtering) thus eliminating error-prone conditions.

Recognition rather than recall

We have attempted to minimise the user’s memory load by making object, actions, and options as visible as possible. We have used very clear functions on the map view (for example popup legend and create new hazard function are both very clear and prompts for hazard creation), and the form tab has clear and concise field input names with hints where necessary.

Aesthetics and minimalistic design

We have kept the design as consistent, simple and easy to understand as possible, this has been done in a way to reduce the cognitive load on the user without compromising on the fundamental functionality. Along with this the hazard information modals only contain information which we consider most relevant and useful.

Help users recognize, diagnose, and recover from errors

When submitting a hazard form the form validation will check to see if all of the required fields have been filled correctly. If any haven’t the user will be told so and the particular input field(s) that have either been missed or entered incorrectly are highlighted to let the user know what area they need to fix.

7.2 Shneiderman’s Golden Rules

Similar to Nielsen’s Heuristics, Shneiderman’s Golden Rules are intended to help you create a well-designed user interface and thereby improve the usability of the system. [25]

Following on from the Nielsen’s Heuristics, this section will highlight the most important aspects of Shneiderman’s 8 Golden Rules (which have not already been covered by in the section above). They are as follows:

Offer informative feedback

All actions have some form of system feedback. While most are subtle some cases require more detailed feedback. For example when submitting a hazard correctly, or when submitting a hazard with an incorrect image (due to incorrect file typing or invalid path), the data will still go through if

submitted, however the user will receive an alert that the form has been completed and submitted but the image has not been saved (as seen in the screenshot snippets below).

Figure 25 - Information popup for correctly submitted form

Figure 26 - Information popup for submitting an incorrect file to the database

Permit easy reversal of actions

At any time before submitting the user can edit and of the form fields or cancel the entire form submission (for example if the location selected is incorrect), and if necessary the user may edit and/or delete their hazard form submission through the management portal.

Support internal locus of control

HazApp makes the users initiators rather than responders by having all functions requiring a user action before they do anything. For example if a user wishes to view the information of a particular hazard they will have to select that hazard on the map, or if they want to create a new hazard report they have to select to do so. At no point will the application commence any of its functions without the user's input. The only changes that will be made within HazApp without a particular user making an action, is when a different user has submitted a hazard which will make the hazard appear on all the other users' maps.

Reduce short-term memory load

The only short-term memory load required at any time should be of a single hazard that a user will be submitting at the time. As the reporting is done on a single case basis there is no need to remember past reports, and if any information is needed about past reports it is available through the information dialogue by selecting the hazard of interest on the map view. We have also avoided any conditions with multiple page displays, and we have minimised window-motion frequency where possible.

7.3 User Acceptance Test

The first user acceptance test is in the process of being completed with a small team (approximately 5 people) of Opus' Geotech Site Managers in the Auckland region. These users have been selected as they perfectly fit into the demographic of users we are trying to target (i.e. site managers that spend time reviewing business practices on-site). While we have developed the hazard reporting side of HazApp to function on all screen sizes from smartphones to desktop computers, the majority of the time the participants in our first user acceptance test will be reporting on-site hazards using tablets.

It is of great importance that we work closely with some real users and that feedback is received and discussed early and often. At this stage it is still relatively easy to improve, remove or add new user interfaces or features. Positive user experience is of critical importance for overall acceptance and ultimately the success of the application. If the application is difficult to understand and/or a hassle to use, it simply will not be used. Fortunately we do have users at our disposal who are willing to

help us test our application as at this stage success is not nearly as important as learning from the users' feedback and addressing the issues accordingly.

From the first round of feedback we have received the following feedback on additional functionalities which would greatly benefit the Geotech team:

- Integration with ArcGIS - If the hazards added via the app could reference points and/or projects in existing ArcGIS maps, Geotech has suggested this would save time and be a great advantage for them.
  This essentially means they would like a function which would allow the information from the HazApp map to be added with their ArcGIS mapping applications (which they use on a daily basis). This means that all of information the Geotech team needs would be in one single application (rather than one application for hazards and one application for everything else. This could be as simple as adding a link/reference to the hazard however for the meantime it is undecided on how it will be implemented.
- Photo exports - The ability to bulk export images for a project area, for use in reports and team or project summaries.
- Hazard statistics/summaries for a project area - This is similar to an already planned functionality and will be easy to adjust this to defined areas. Essentially it will require some sort of pdf generator which could also include photo exports (as mentioned above).

While the majority of these feedback points were already planned for in the future it is great to see that the direction in which we are heading is aligning well with not just the business and health & safety managers (based off the feedback received at the Executive Leadership Team demonstration), but also the people who will eventually be making use of our application.

From the first round of feedback it has also been encouraging that the users haven't raised any complaints with the user interface or flow of the application. I would like to believe that this is due to the fact they are happy with how the application currently looks and feels. However for the next round of feedback we will be prompting the users to focus on and discuss the overall user interface and flow of HazApp, just to see if there are any easy to fix aspects which may be damaging the general user experience.

8. Security

While HazApp is still in the current development stage and is only available to a few specially selected individuals within Opus, security seems somewhat unnecessary. However as HazApp is eventually deployed to a wider audience it will become more and more important to ensure that any confidential data is safe so that the Opus and any individuals employed by Opus are never in a position to be harmed. As such we have focussed on both keeping unwanted users out via a secure login system, and keeping authorised users from damaging the system/data via input validation and SQL injection counter measures.

8.1 Database Security

Making sure no malformed or malicious code is inserted into the database is very important to keep the confidentiality of data at a high, as well as allowing the application to run smoothly and as intended without any crashes/errors. For the most part this has been done within the form fields'

validation parameters. As the form fields are the only area which adds data directly back into the database we have put strict limitations on what may be input for the form fields using Knockout input validation (as highlighted earlier in 'First Prototype' section), however to combat any potential form of SQL injection we have also used prepared statements which essentially takes the characters input from the form field(s) and places them directly in the database as text. This means that even if a user bypasses the input validation, their input will never affect the INSERT statement in any way.

8.2 User Login

Seeing as the hazard reporting side of HazApp will be available via internet browser it is of much importance to have some form of user login/verification system to limit users to only intended users. This will not only protect the potentially confidential information about Opus' business operations, but also it will negate the possibility of unwanted users "griefing" the system by logging incorrect and misleading information that may affect real users (for example someone reporting there has been a chemical explosion in a certain area when in fact everything is fine could halt work in said area until the confusion was cleared up).

To protect the system from unwanted users we have implemented an htaccess login system which prevents users from accessing any part of the application without a verified username and password. If a correct username and password is entered, the user will have full access to HazApp's reporting features. However if an incorrect username and/or password is entered the authentication prompt will remain and the application will not be loaded.

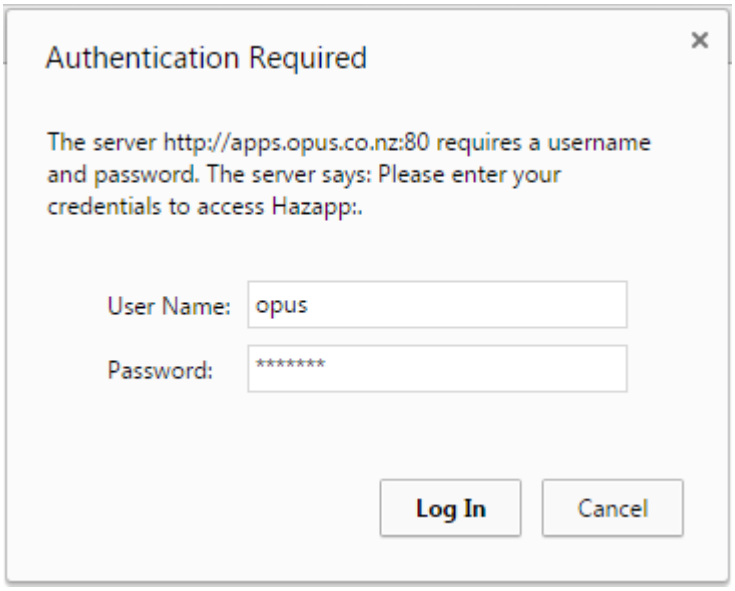

Figure 27 - htaccess login system prompt

When attempting to access HazApp via browser the prompt above appears before loading any of the application, as such no information is obtainable without a correct username and password. This most importantly applies to the page's html code. As seen in the two screenshots below, no information is retrievable about the page before the login has been successfully completed, but after the login a lot of information about the structure and design is obtainable. While this information is likely safe to be seen by non-approved users, it is still important to keep any information that could potentially damage Opus' business confidential where possible (for example if a competitor wanted

to copy and create a similar application for themselves they wouldn't get any useful information without a login).

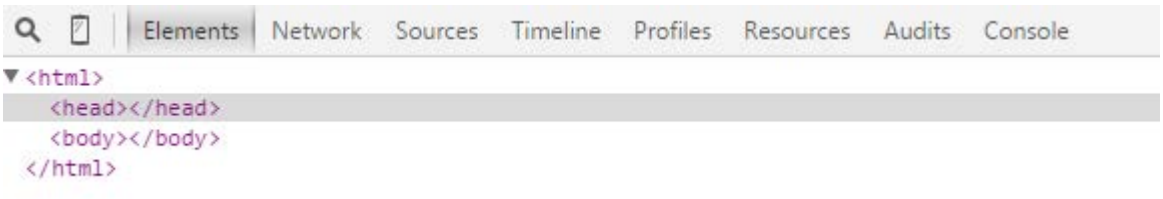

Figure 28 -Empty html code before login verification

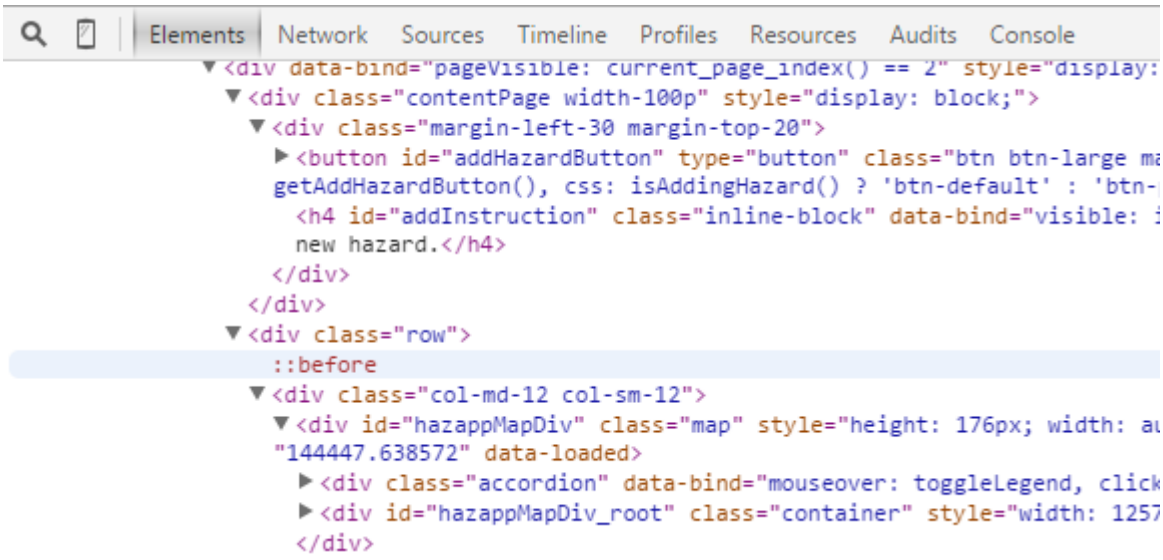

Figure 29 - html code after login verification

9. Performance

As maximising positive user experience is a priority for the success of HazApp, having a good performance for the web application is very important. Research has shown that having a web page take too long to load can juristically hurt the overall usability of an application [26]. As such I have reviewed our current application using YSlow [27], a tool which analyses and grades a web page based on a set of rules highlighting important areas in regards to performance.

9.1 YSlow Grading

As seen in the screenshot below HazApp (hosted at http://apps.opus.co.nz/HazApp) has a rating of 90 out of 100 (A-Grade). The only area which is not at the A-Grade is the section of "Making fewer HTTP requests". The reason this section has not been given an A-Grade is because our application has several external JavaScript scripts, stylesheets and background images.

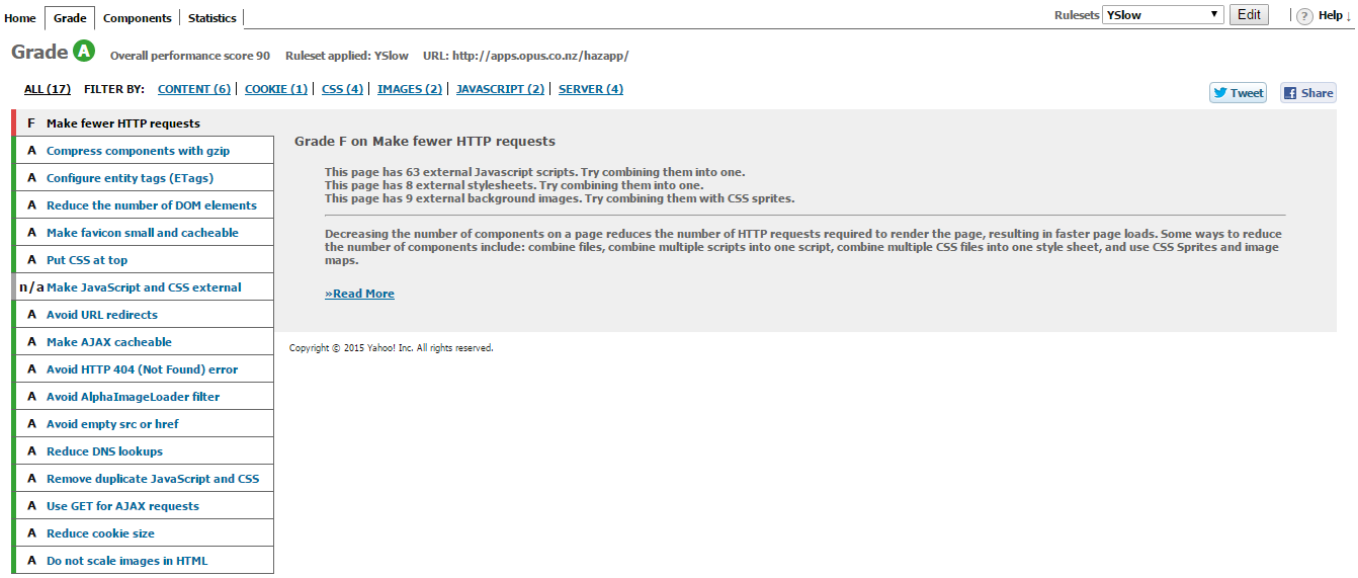

Figure 30 - YSlow Performance Review

YSlow's suggestion is to combine the external JavaScript scripts together and the external stylesheets together, however this conflicts with one of my original goals of having the application up to Opus' software development standards. We have used Code Igniter to structure the entire application in a way that related code is grouped together, but separated from other areas. For example the JavaScript code for the ArcGIS mapping is separated from the JavaScript code for Knockout JS. If all the JavaScript code were to be combined into a single script it would be so large and would contain so much information that it would be significantly more difficult to understand and it would take a lot of time to find any single area you were interested in. As such I have decided to stick with the structure that has been implemented. Although it may give a minor hit to the performance of the web application, it is a necessary sacrifice that should increase the overall longevity of the application.

Other than the section of "Maker fewer HTTP requests" HazApp has been given A's all round. Overall I am very pleased with the grading and I feel as though in its current state its performance is at a high enough standard to be used without damaging the user experience.

9.2 YSlow Cache Statistics

The screenshot below shows the statistics page of HazApp, unsurprisingly for both the empty cache and the primed cache JavaScript code takes up the majority. I believe that this is the case as we have used many different JavaScript APIs/libraries to get the functionality needed to having HazApp working as intended.

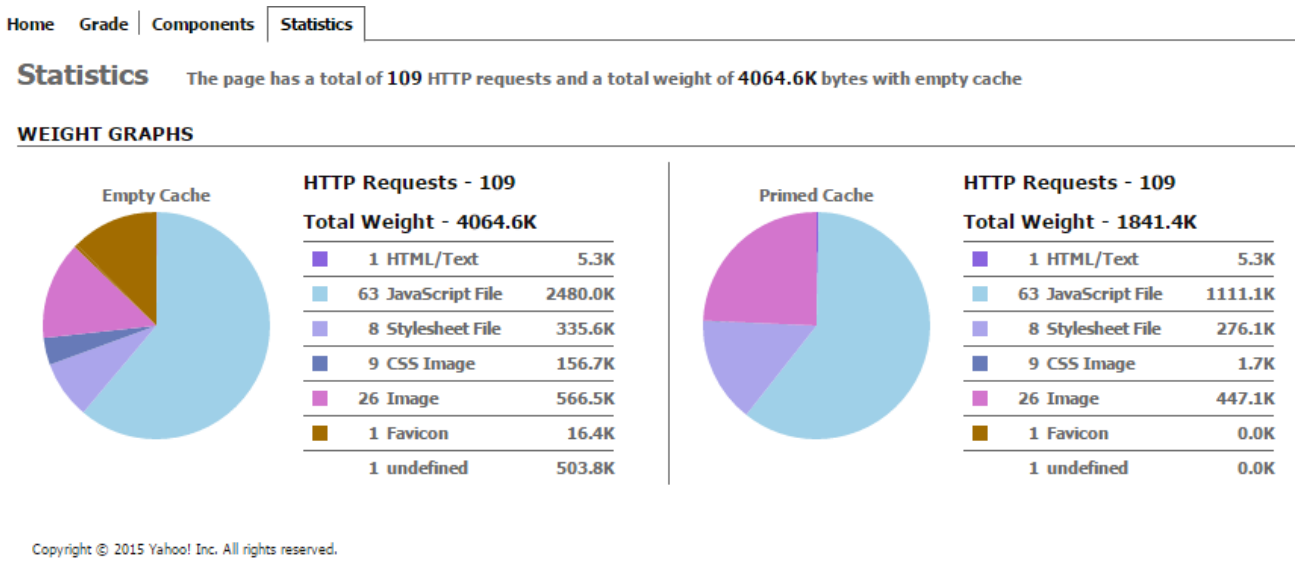

Figure 31 - YSlow Statistics Overview

Chapter 4 – Conclusion

10. Completed Goals

To begin with HazApp was only a simple idea with infinite possibilities. Its one goal was to be an innovative online hazard reporting system which would eliminate the hassle of current paper based hazard reporting systems. Before any development had started I worked closely with the team at Opus to plan out how we would solve this problem. After the long and thorough planning phase (in which we decided on what HazApp would eventually become and how it would go about doing so) I made some personal goals which I considered realistically completable within the timeframe I was given for this project. The overarching goals that I had decided on at the start of our development phase were as follows:

1. Create the hazard reporting side of the application and get it running on-line so that it may be used by real Opus employees - We have already deployed a stable release of HazApp's reporting side online at https://apps.opus.co.nz/hazapp. Although only a small number of employees are currently using the system (those participating in the User Acceptance Testing), it should be very easy to increase the numbers of available users. This is great as having more users that are using HazApp will increase the numbers of hazards that are reported and it will increase the awareness of those hazards.

2. Create the management portal so that the real reported data could be analysed in some way - While this is only in a basic state at the moment, the management portal still offers what I consider very useful information on the reported hazards. As more people begin to use HazApp the information reported will carry more and more weight, to the extent that real business changes may be based upon the findings. There is a lot of room to expand and many possibilities for further statistical analysis.

3. Complete some form of usability study on the working application - While I did not carry out a full blown usability study, it was still very important to see how users who had used Opus' paper based hazard reporting system felt about the application we have created to combat it. This was done through a User Acceptance Test completed by a small group of Geotech Site Managers. Their input was very interesting and very useful in regards to making improvements of our current system.

4. Have all of my development code in a working state in which is up to Opus' software development standards. Meaning that any members of the software development team could continue further development and maintenance on the HazApp after the completion of my project – This took a lot of work and required me to learn a completely new way of structuring code. However it was very important for me to do this well and I am happy to say

that two software developers at Opus will be continuing work on HazApp after my completion.

Each one of these points had challenges within itself but I am very happy to say that all four of these goals have been completed to a standard I am very proud of.

11. Lessons Learned

Throughout this whole process I have come across many challenges and I have also learnt a lot. I believe that the knowledge and experience gained through this project will definitely help me in the future as I commence working as a software developer. The most significant lessons I have learned from this project are:

- Using frameworks and design techniques that real software professionals use in the real world - Whether or not I will be developing HTML based web applications and/or using any of the frameworks I have used for this project in my career is unknown, however I believe that regardless it was a very useful experience. Even if I never use any of these languages or API's again the mere exercise of learning new languages and frameworks is useful in itself. As the IT industry is so fast moving it is a real skill to be able to adapt. One of the most vital ways to keep up is to be able to learn.

- How to work with a real team of not only software developers, but also business and health & safety managers - The interactions between the different groups made me realise that to get the information you want you must be ready to cater communications for each separate group. Communication is a vitally important aspect of working in the real world, and understanding how to communicate with different technical skill groups in a company is a very useful skill.

- Getting a small taste of agile software development and what software development will be like in a professional setting – Agile software development is common among many IT industry companies. After getting to work in a semi-agile software development environment I can see why it is so popular. Being able to quickly change course to best improve an application seems to me like a very smart and efficient way of developing systems. There is no doubt that I will be working in some form of an agile team at some point in my software development career.

- Getting to work closely with a small team of real users undertaking a user acceptance test - Being able to properly understand your user base is also incredibly important, and whether or not I will be conducting any forms of usability studies as a software developer, it is always great to have the end user in mind when developing any system. At the end of the day, if an application has poor usability or is difficult to understand it will simply not be used. This is why usability is so important in this line of work.

12. Future Work

While I am very pleased with the overall progress of the project and what I have learned from it, it was always inevitable that due to the scale and freedom of the project there would always be more that could be done. However with that in mind, Taylor Carnell and Mitchel Bennett at Opus will be fully taking over the project after my completion and we have discussed their plans on further development in the coming months.

First and foremost, they will be focussing on filling out the management portal side of HazApp. This will primarily be through the addition of more statistical analysis functionalities (several of which have been mentioned in the 'Data Analysis Using HighCharts' section). All of these will be used to highlight any areas of Opus' business and health & safety practices which may be below standards. Opus is working towards a safety objective of having a "zero harm workplace" and the management portal in conjunction with the hazard reporting portal of HazApp will hopefully greatly help the cause.

Within the management portal there will also be functionality for managers to work on data Quality Assurance/Quality Control (QA/QC). Which means that managers will be able to observe what other Opus employees have submitted to evaluate whether it is appropriate and/or whether it is how the application is intended to be used. Similarly, this may be used as a basis to improve health & safety trainings.

There are also several minor improvements planned for the hazard reporting side of the application. These will need to be completed before expansions to a larger user base. Some of the more important improvements are:

- Offline support for the application, which will be very useful for any employees wishing to report hazards while working on a project site which may have little to no internet access capabilities.
- Integrating HazApp with mapping systems regularly used by site managers, which will mean that they will only have to have one application for all of their managerial needs.
- Further expansion of the hazard type list within the database to match the required health and safety standards.
- Any additional user interface improvements that may arise through further User Acceptance Testing.
- Any additional functionalities suggested by users and/or managers which are deemed as useful.

They will also be continuing to work with small user groups to improve things such as minor bugs and user interfaces in preparation for a notional rollout. When HazApp is at a completed state that the business and health & safety managers are happy with, the user base will be expanded to reach the major cities within New Zealand the eventually all of New Zealand. Assuming the success of nationwide rollout, HazApp will be considered to be expanded to all Opus bases globally. If this does go ahead several more refactors may need to be taken into consideration. For example language boundaries.

13. Concluding Thoughts

As stated earlier I am very happy with what I have learned and how I have gone about completing this project. I genuinely believe that the skills and experiences I have gained through this project will be greatly beneficial as I begin my career as a software developer. I really hope that HazApp will be completed and fully adopted by Opus so that I may know I have made a positive impact for my work over the year. Finally, as my project comes to a close I would again like to thank all those that helped me along the way, the entire experience was truly enjoyable.

Appendix

Item 1 - Simple Hazard Reporting Template

Area/Locality of hazard | Date

Name.....

(Name of person preparing report)

DESCRIPTION OF HAZARD (Include area and task involved, any equipment, tools, people involved. Use sketches if necessary.)

POSSIBLE REMEDIES (List any suggestions you may have for reducing or eliminating the problem, e.g. re-design mechanical devices, procedures, training, maintenance work, etc.)

To be submitted to the Manager

Signed……..

ACTION TAKEN

Date……

Manager……

CONTROL IMPLEMENTED & EVALUTATED

Date……

Manager……

Item 2 – Site Hazard ID Risk Assessment Form

OPUS

Daily Hazard ID FORM for uncontrolled sites

Contract: _____ **Site:** EXAMPLE

Project Manager: **Job Number:**

ALARP	As Low As Reasonably Practicable (**ALARP**)
Hazard	A hazard is an activity event, arrangement or occurrence that can cause or potentially cause harm to ourselves or others this includes a person's behaviour and/or impact on the environment
Significant Hazard	Significant hazard means a hazard that is an actual or potential cause or source of Serious harm. If it is identified as a significant hazard you must follow control s **E.I.M**"". = Eliminate, Isolate, Minimise
"**Inherent Risk**	Risk in the existing working environment, in the absence of any action to control or controls implemented.
"**Residual Risk**	After the controls have been put in place, factored in, or eliminated.

Likelihood \ Severity	**High** Fatality Major Illness Long term disability	**Medium** Injury/illness causing Short term disability	**Low** Minor injury/illness
High Certain/nearly certain	H	H	M
Medium Reasonably Likely	H	M	L
Low Seldom	M	L	L

High Risk is undesirable. Verify, and where possible quantify the accuracy and certainty of the existing risk level. Implement control measures to ensure risk level is reduced to or is confirmed to be **ALARP**. Operation at this level requires management approval.

Medium Risk is only tolerated if examination proves them to be **ALARP**. Implement management plans to prevent the occurrence and monitor for changes. Reduce to Low Risk if the benefits outweigh the cost.

Low Risk is acceptable. Review at next interval.

What can harm me? What can harm others?	Date Identified	Inherent Risk Rating*			Significant hazards Yes/No		HAZARD CONTROLS (How can we reduce personal harm here?)	Does the Control Measure"" Eliminate, Isolate, Minimise			Residual Risk Rating*			Person or team responsible	Review by and date
		H	M	L	Y	N	List Control Measures	E	I	M	H	M	L		
(Example) Heavy Machinery	7 March 2014	X			X		1. Vehicles to have: • Flashing beacons			X		M		1. (just an example)	Weekly

What can harm me? What can harm others?	Date Identified	Inherent Risk Rating*			Significant hazards Yes/No		HAZARD CONTROLS (How can we reduce personal harm here?)	Does the Control Measure** Eliminate, Isolate, Minimise			Residual Risk Rating*			Person or team responsible	Review by and date
		H	M	L	Y	N	List Control Measures	E	I	M	H	M	L		
Workers hit/run over **Outcome Fatality**							• Audible reversing alarm • Cameras 2. Vehicle Operator • Trained operator o Training records available 3. Staff on site: • Correct PPE as per site rules • Trained in approach to heavy vehicles o Training records available • Staff onsite not to work or walk while using telecommunications devices or listening devices (mobile phones, i-pads, i-pods etc) 4. Designated work areas: • Understand where the no-go zones are • Stay on designated pathways 5. Spotters: • Know where the spotters are and follow their instructions 6. Barriers: • Hard Plastic Barriers defining site office work areas, walkways			X X X X X X X X		M M M M M M M M M M			L	company in charge 2. Project/Site Manager 3. Project/Site Manager 4. Project/Site Manager 5. Project/Site Manager 6. Project/Site Manager	Weekly Weekly Weekly Weekly

*Only add/edit hazards to this document, do not change any other part of this document without prior consent by the H&S Officer NZ

Bibliography

[1] Opus International Consultants Limited, “HazApp: The Opus Geospatial Hazard Management System,” Auckland, 2015.

[2] Cloudsource Limited, “Health and Safety Mobils Apps | ThunderMaps,” 2015. [Online]. Available: https://learn.thundermaps.com/. [Accessed 20 April 2015].

[3] R. Ghatol and Y. Patel, Beginning PhoneGap, New York: Apress Media, 2012.

[4] C. J. Date and H. Darwen, A Guide To SQL Standard (Vol. 3), Reading: Addison-Wesley, 1997.

[5] R. Cattell, “Scalable SQL and NoSQL data stores,” *ACM SIGMOD Record,* vol. 39, no. 4, pp. 12-27, May 2015.

[6] J. Farrar, KnockoutJS Web Development, Packt Publishing, 2015.

[7] P. B. Darwin and P. Kozlowski, AngularJS web application development, Birmingham: Packt Publications, 2013.

[8] A. Osmani, Developing Backbone js Applications, O'Reilly Media, Inc., 2013.

[9] B. LeRoux, “Lawnchair simple json storage,” 10 March 2015. [Online]. Available: http://brian.io/lawnchair/. [Accessed 14 May 2015].

[10] Mozilla, “Local Forage,” April 2015. [Online]. Available: https://mozilla.github.io/localForage/. [Accessed 14 May 2015].

[11] M. David, Developing websites with JQuery mobile, Taylor & Francis, 2015.

[12] A. Shevchenko, R. Van Baalen, K. D. Moore, A. Levicki and D. Netto, Developing an Ionic Edge: HTML5 Cross-Platform Hybrid Apps, Bleeding Edge Press, 2015.

[13] M. Stevenson, Bootstrap: The ultimate beginners guide to Bootstrap 3.0, USA: CreateSpace Independent Publishing Platform, 2014.

[14] Less, “Less - Getting started,” 2015. [Online]. Available: http://lesscss.org/. [Accessed 26 May 2015].

[15] H. Catlin and M. L. Catlin, Pragmatic Guide to Sass, Pragmatic Bookshelf, 2011.

[16] K. Johnston, J. M. Ver Hoef, K. Krivoruchko and N. Lucas, Usin ArcGIS geostatistical analyst (Vol. 380), Redlands: Esri, 2001.

[17] B. Momjian, PostgreSQL: introduction and concepts (Vol. 192), New York: Addison-Wesley, 2001.

[18] JGraph Limited, “draw.io,” 2015. [Online]. Available: https://www.draw.io/. [Accessed 24 May 2015].

[19] pgAdmin, “pgAdmin - PostgreSQL Tools,” 12 December 2014. [Online]. Available: http://www.pgadmin.org/. [Accessed 28 May 2015].

[20] M. Hills, Klint P and J. Vinju, “An empirical study of PHP feature usage: a static analysis perspective,” *ISSTA 2013 Proceedings of the 2013 International Symposium on Software Testing and Analysis. ACM, New York, NY, USA.,* pp. 325-335, 14 May 2015.

[21] D. D. Dvorski, Installing, configuring, and developing with Xampp, Skills Canada, 2007.

[22] R. Molich and J. Nielsen, “Improving a human-computer dialogue,” *Communications of the ACM 33,* pp. 338-348, 3 March 1990.

[23] R. Molich and J. Nielsen, “Heuristic evaluation of user interfaces,” *Proc. ACM CHI'90 Conf. Seattle, WA,* pp. 249-256, 5 April 1990.

[24] J. Nielsen, “Enhancing the explanatory power of usability heuristics,” *Proc. ACM CHI'94 Conference Boston, MA,* pp. 152-158, 28 April 1994.

[25] B. Shneiderman and C. Plaisant, “Designing the User Interface: Strategies for Effective Human-Computer Interaction: Fifth Edition,” *Addison-Wesley Publ. Co., Reading, MA,* p. 606, 2010.

[26] F. F.-H. Nah, “A study on tolerable waiting time: how long are Web users willing to wait?,” *Behaviour & Information Technology,* vol. III, no. 23, pp. 153-163, 2004.

[27] “YSlow,” October 2015. [Online]. Available: http://yslow.org. [Accessed 20 October 2015].