VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

A Web-based System for Traffic Management Plans

Jack Robinson

Supervisors: Professor Dale Carnegie and Dr David Pearce

Submitted in partial fulfilment of the requirements for Bachelor of Engineering with Honours.

Abstract

A short description of the project goes here.

Contents

Chapter 1

Introduction

The aim of this project is to design, develop and evaluate Coneheads, a web-based application for creating Temporary Traffic Management Plans (TTMPs). This is achieved by the implementation of core functionality in a minimum viable product, allowing the definition of work sites on a cartographic map, and the automation of portions of an TTMP application. This is an industry project in association with Coneheads.

1.1 Motivation

When a contractor wishes to complete work on a footpath or road in New Zealand, they must submit a TTMP application to a Road Controlling Authority (RCA). These take the form of a ten page document requiring the applicant to consider everything from how the site will be monitored at night, through to the distance between placed cones. Each facet of this ten page form needs to abide by the Code of Practice (CoPTTM), guidelines provided by the New Zealand Transport Authority that must be adhered to in order to have their application accepted. Despite this, a number of applications are denied due to simple mistakes, meaning even a basic application could take up to three weeks to be approved. Atop of this issue is also the information provided by NZTA. Each TTMP application requires the contractor to provide a diagram that outlines the proposed work site, however in guidance, the Code of Practice only provides very generic scenarios that explicitly state they should not be copied. These example diagrams shows how basic sites are meant to be setup, but do not factor in obstacles such as driveways, or curves in the road. Combining this with the complex submission process, some applicants provide diagrams that dont accurately describe the work proposed, and at times take the example scenarios from the CoPTTM. Lastly, because of the variable nature of the work done on roads, such as the location, speed limits and nature of the work site, each TTMP is vastly different from others, requiring substantial work from both ends of the process to complete a simple plan.

1.2 Proposed Solution

The proposed solution to the above outlined problems is to develop Coneheads, a web-based system that can aid in the TTMP application process. The system will provide functionality that allows users to contextually specify a proposed work site by defining one on a cartographic map. By allowing this, applicants can factor in the surrounding environment to a work site, such as the aforementioned driveways, and make amendments dependent on the situation. Portions of the system will also allow for automation in the completion of the TTMP form due to the information provided by the contextual definition process. This

automation will be supported by logic extracted from the Code of Practice to allow for easier adherence to the guidelines. Consequently, by placing the application process onto a form, TTMPs produced from the system can provide a standard form for RCAs to enable them to easier approve applications by contractors.

1.3 Contributions

Through the course of the project, the following contributions have been made:

- The development of a cartographic map based work site definition and editing editing screen, formulating a basic work site atop satellite imagery based on user input
- The automated creation of a basic plan on the proposed site adhering to CoPTTM guidelines.
- Enabled the ability to edit work sites based on contextual information provided by the map.
- Enabled the partial completion of a TTMP form automatically based on information extracted from the proposed work site.
- Development a Minimum Viable Product that allows for the completion of a TTMP online.
- Development of a document generator allowing for a standard form on the output of the process.

Chapter 2

Background and Related Work

2.1 Temporary Traffic Management

Temporary Traffic Management (TTM) is the method in which contractors, or related parties, manage road users such as vehicles and pedestrians for a non-permanent period of time (CITE). The goal of TTM is to guide traffic through, around or past a road reserve with minimal risk to both the public, and the workers at the site. It is also imperative to manage traffic concerns, not only at the work site itself, but surrounding roads as well due to the impact the work may have on traffic elsewhere (CITE). TTM measures are generally processes that edit the normal functions of a road, such as redirected lanes through the placement of cones, or temporary speed limits (CITE). Within New Zealand, whenever a contractor wishes to enact measures of TTM in order to complete work on a road, they must submit the aforementioned Temporary Traffic Management Plan to a local authority. These plans detail the design, implementation, maintenance and removal of TTM measures, and ensures compliance to national legislature by requiring them to be approved before work can be carried out on a work site.

2.2 Code of Practice

The regulations on how TTM and TTMPs are carried out are defined in the Code of Practice for Temporary Traffic Management (CoPTTM) provided by the New Zealand Transport Authority. The application process requires full compliance to the guidelines set out in order to be accepted, and no work is able to commence until a TTMP has been approved. By nature, roads and work that is intended to be completed on them are highly variable, with considerations such as speeds, widths and volume needing to be considered in order to effectively minimise potential hazards on the work site. Due to this complex nature, the Code of Practice attempts to provide base variables in order to determine a valid site

The first of these variables relates to volume, where the annualised average daily traffic (AADT) determines the level of a road. A road with less than 10,000 vehicles per day (determined by the AADT) is classed as Level 1. Over 10,000 is either a Level 2 road, or Level 3 dependent on the permanent speed limit of the road. Level 3 is primarily designated on Auckland Motorways, and Level 1 is the most common road level in New Zealand (CITE). A fourth designation exists, LV or Low Volume is used when the daily traffic is below 500 vehicles per day. Road levels are useful as it determines the level of TTM required if work needs to be completed on one. Level 3 roads may require more signage, both in terms of the size of the sign as well as frequency, whereas an LV rated road may just require the minimal in terms of signage.

Together, speed and road width are considered in safety zones. On a road Level 1 or above, a taper of cones is required leading into a worksite. The distance this taper is from the work site is determined by the longitudinal safety zone. The higher the permanent road speed is, the longer the longitudinal safety zone is on the oncoming lane to the work site. An example of why this is better than no safety zone is if a vehicle enters the work site through the taper, it has enough time and distance to place the brakes on or avoid the work site. The opposite is the lateral safety zone, or the distance width-ways that is required between the work site and the placement of cones. If the work required the shut off of all or most of a lane, the lateral safety zone is required to push into the next lane, requiring either a traffic control (such as a worker with a sign), or a contraflow (having the left lane traffic use part of the right lane to pass the site).

2.2.1 Roles and Responsibilities

Due to the strict nature of TTM, there is a rigid framework in place that designates certain responsibilities to a number of qualified roles. Key roles that are defined in CoPTTM are described in detail below:

- **Road Controlling Authorities (RCA):** RCAs are the bodies who administer every road within the country. These are generally local councils at a local level, however all state highways are controlled by NZTA. It is the RCAs job to give consent to contractors to begin work on a road reserve, and are the ones who ultimately accept or deny TTMP applications.
- **Site Traffic Management Controllers (STMS):** STMS' are those that are qualified in TTM, and are the ones who initially draft and apply for the TTMP approval from an RCA. In the cases of Level 1 or LV roads, the STMS can also approve work on these roads, depending on whether the RCA has delegated authority to them. STMS are also in charge of ensuring compliance to the CoPTTM during the course of work, having authority to postpone, cancel or modify work due to safety or traffic concerns, allow visitors on a site, and even order people off a site if non-compliant to the guidelines.
- **Contractors:** Contractors are the ones who are undertaking the work that requires a TTMP. It is their job to ensure they have approval from an RCA to carry out the work, and ensure that TTMP applications are completed by an STMS. They are required to uphold all TTM measures outlined in their TTMP application to ensure safety of workers and the public that wish to travel through the outlined area.

2.3 TTMP Process

The process of completing a TTMP is two fold; a textual form is required to be completed by a STMS, and a diagram giving a graphical overview of a work site provided. Examples of both are given in Appendix 1, as well as a diagram outlining the process in which a TTMP is applied and approved.

2.4 Related Work

Coneheads is an interesting case, as its concerns in the TTM area has not previously been entered by any discernible company before. However, applications built upon cartographic maps are relatively prevalent in Geographic Information Systems. In this section I discuss related work to the Coneheads system.

2.4.1 Geographic Information Systems

A Geographic Information System, or GIS, is an information system that enables the viewing, storage and manipulation of geographic and spatial data information. GIS has a wide use in both research and consumer based products. In research, statisticians and data scientists adopt map based visualisations to relate information to a particular location. An example is researchers in medicine can build a relationship between health and a location, adding layers of information that may attribute to certain diseases such as air pollutants (CITE BERKE). In New Zealand, GIS has been used to help rebuild Christchurch infrastructure in the wake of the 2011 earthquake. Maps detailing underground infrastructure has enabled both the public and private sectors priortise and assess work that needs to be done, while also providing contextual spatial data to elaborate on why certain work needs to happen before others (CITE LINZ).

2.4.2 MobileRoad

The closest similar product to Coneheads is known as MobileRoad. MobileRoad is an application developed by the Auckland Motorway Alliance, a subsidiary of NZTA in charge of operating and maintaining the Auckland motorway network. It's primary use is to aid the road maintenance industry determine locations in GPS coordinates to a the format that their road asset maintenance and management (RAMM) system can interpret. However, as a secondary feature, it also provides information such as AADT and Road Widths to a simple GUI that can also be downloaded to local browser history for use externally. Users load the data for their desired RCA, then are presented with a full screen map. Hovering over streets present the information in an information box overlay on the screen. Despite this, there is little functionality in MobileRoad that allows a user to edit and manipulate the data provided. Of note, however, is this applications relatively low-publicity. It was evidently developed for internal use and it has slowly spread for use across the maintenance industry. This could indicate that the information provided is not technically free and open as it currently is, therefore meaning that use of its information could be unethical.

Chapter 3

Requirements

Upon commencement of the project, a number of requirements were created through a brief supplied by Coneheads, as well as numerous discussions between each party. This section defines the core functional requirements of the Coneheads application in detail

3.1 Previous Work

This project is the second attempt at developing a web based systems for TTMPs. Coneheads previously had a developer creating a system for them to use, however it was deemed too complex for the targeted audience, and not what Coneheads were looking for. While there was the option to utilise this existing and incomplete project, it was decided that a new application should be built, taking from the first project the aspects that are good, and applying them to a much simpler application.

3.2 Minimum Viable Product

A key influence on the definition of requirements for the project is the fact that this application is intended to be a Minimum Viable Product (MVP). An MVP is a level of completion, slightly more concrete than a simple prototype, that a person or business can use to validate the ideas of their proposed product [CITE].This has numerous benefits over a complete application, as the small level of development required can confirm or disproves assumptions that could have been made about the intended market. MVPs are initially released to early adopters to acquire feedback from before new features are added or it is released to a wider market. An MVP is also a good indication of the direction a product or a service is intending to take [CITE], therefore they can also be used to show potential investors to secure funding for the expansion of features. This directly influences the functional requirements of the Coneheads system, as certain features can be ignored in order to complete other portions of the system.

3.3 Requirements

After analysis of the brief and discussions with Coneheads, the following functional requirements are defined as key indicators of completion in the project.

1. **The system must be able to produce a correct work site on a geographic representation of a work site.** While placing the TTMP process online is a helpful feature to have,

the defining nature of Coneheads is that a user is able to produce a geographically accurate diagram of a proposed work site, allowing them to be aware of the environment surrounding it.

   (a) **The work site plan must be editable.** With a better representation of the environment around a work site, an applicant should be able to make edits to the generated plan to factor in obstacles such as driveways and bridges.

2. **The system must be able to produce at least a single plan.** Being an MVP, a lot of work needs to be done on validating the application. Rather than spend time implementing the large number of plans into the system, the system must be able to produce a single common plan well.

   (a) **The system must be able to produce a TTMP application adhering to the Code of Practice for Traffic Management Plans.** If a generated plan does not adhere to the Code of Practice, then it will fail submission, defeating the purpose of taking the process online.

   (b) **The system must be able to complete portions of the form automatically.** The current process contains a large amount of repeatability between applications, so the system must be able to complete the common sections automatically. As well as this, geographic and road information, such as AADT and Road Speeds are available to be automatically filled out in an application. There also has to be consideration of the opposite occurring - users should not become complacent with the automated system and relying on it to always be complete.

   (c) **The system only needs to support one role, a contractor** As an MVP, the Coneheads system only needs to support one role, that of a TTMP applicant.

3. **The system must be easy to use.** The primary users of Coneheads are expected to be qualified in the completion of a TTMP, however their level of familiarity of the use of computers is assumed to equivalent to comfort using Microsoft Office products.

4. **The system needs to show existing work sites.** By having Coneheads online, we can now have a centralised repository of current work sites in an area. By showing all current work sites, we can minimise the amount of duplicate applications provided.

5. **The system must keep record of all applications.** Data filled into Coneheads needs to be persisted, even after the submission has been completed to be accessed for potential auditing concerns.

Chapter 4

Application Design and Architecture

This chapter introduces and discusses methods used to design the Coneheads system, followed by an exploration of the system architecture behind it. It will also reflect on the technologies that were available, and why particular ones were selected over others.

4.1 User Stories and Use Cases

As per Requirement 2c, the system is intended only to serve a single actor, however, there are multiple scenarios this user may take through the system. Gonzalez showed that information visualisation systems in the workplace are used in a direct fashion, with the user knowing exactly what task they wish to complete before interacting with the system [CITE GONZALEZ]. While Coneheads is not primarily an information visualisation system - it borders on an intersection of visualisation and a number of other fields - the research provides useful guidance in the design of the system. Creating convoluted paths through an application might deter a determined user from Coneheads, especially one who is not overly familiar with the technology. To target this, user stories were formulated to aid in developing what is required from the system. In order to conceptualise these in depth, use cases were formulated based around a persona of a single actor. Whereas a user story can succinctly describe what a user would be wanting from a system, Use cases provide more contextual scenarios that can be used to empathise with potential users. The persona, as well as the user stories and use cases can be found in Appendix A.

4.2 User Interface

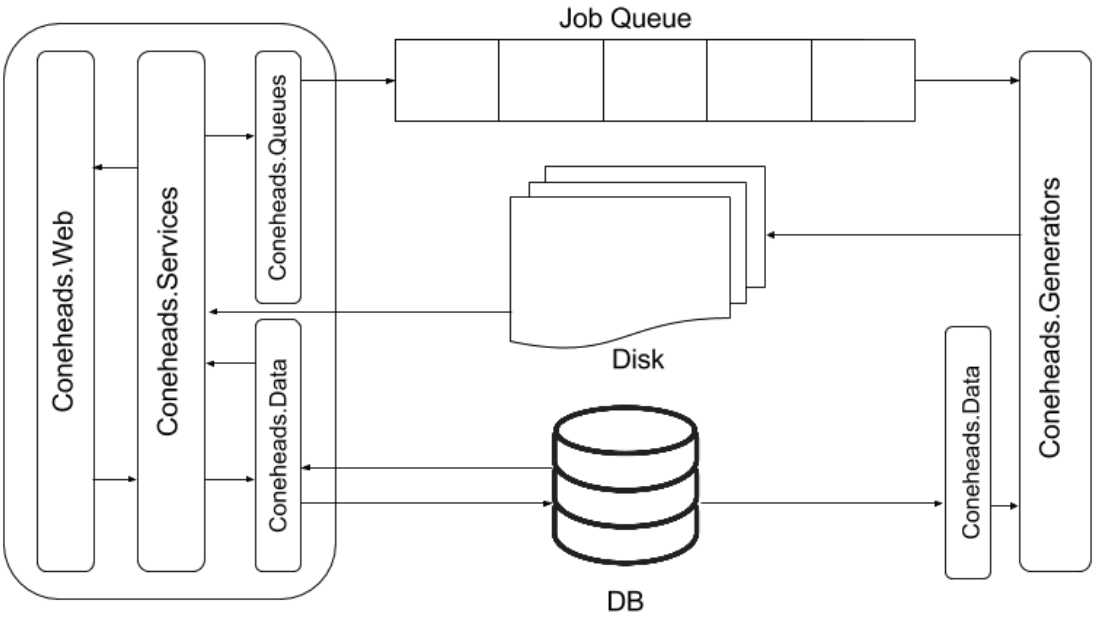
The user interface is not a primary requirement of the application, but in order to capture the Requirement 1, 1a and 3, consideration of how a user can interact with the system is still important. The system is inherently complex behind the scenes, something described in detail in Chapter [x - Implementation Chapter], but in order to make it simple for a user with basic technical knowledge, the design of the definition and editing pages tries to resemble a simple paint program. Programs like Microsoft Paint have been used in numerous settings due to its simplistic nature, especially in primary school level education [CITE LAZAROS & SPOTTS]. In Paint, you are offered a canvas filled with graphics, while to the side is a set of controls a user might use to modify and add to the current pane. The editing page was designed to be similar, with nothing but the canvas (the map) and a set of controls that can edit the default work site template. The desire is that a user with little familiarity with the system can intuitively interact with the controls with next to no guidance on how to use it.

4.3 System Architecture

Being a web-based application, Coneheads is required to be able to be run via a web browser. Many modern web browsers have restrictions on how much software on the web can interact with the clients computers, such as saving to disk for security reasons. Because of this, the most logical architecture for Coneheads to have was a classical server-client based one. As well as enabling the development of the application, isolating each component allows for a clear separation of concerns, a desirable characteristic of many modern systems, and even more desirable for this MVP. By separating apart the logic, future developers can modify certain sections of the code base without prior knowledge to another, meaning, should the code be revised in the future, it is simple and safe to do so.

Within each component of the system, they adhered to their own inner model. This section will explore the design of both the back end server side of the application, and the front end client side.

4.3.1 Back End Architecture



The back end is defined by three layers of operation; the **Controller layer**, the **Services layer**, and the **Adapters layer**. This resembles an Onion based architecture, where a layer can only communicate with its immediate neighbouring layers.

Controller Layer

At the top level is the Controllers, represented in the model above by Coneheads.Web. The primary role of the Controller layer is that it exists solely to transport data from an endpoint to the service layer of the application, ensuring that requests have been formed correctly. It strongly adheres to the HTTP verb system of `GET`, `POST`, `PUT` and `DELETE` to ensure that the caller knows exactly what their request will action. Each logical boundary is separated into its own controller; document related calls go to `DocumentController`, for example. Strictly no logic exists in these, and instead they pass this onto the Service layer.

Service Layer

The Service layer consists of a group of classes that capture the logic of a single action. This design allows for high maintainability and modularity as it cohesively encapsulates a type of work within a single file. It also ensures that there is only ever one reason to change a file - if the class has multiple responsibilities, a change in this file may result in numerous changes around the system. In `Coneheads.Services`, these services handle taking requests from the Controller layer, and handing them onto the Adapter level.

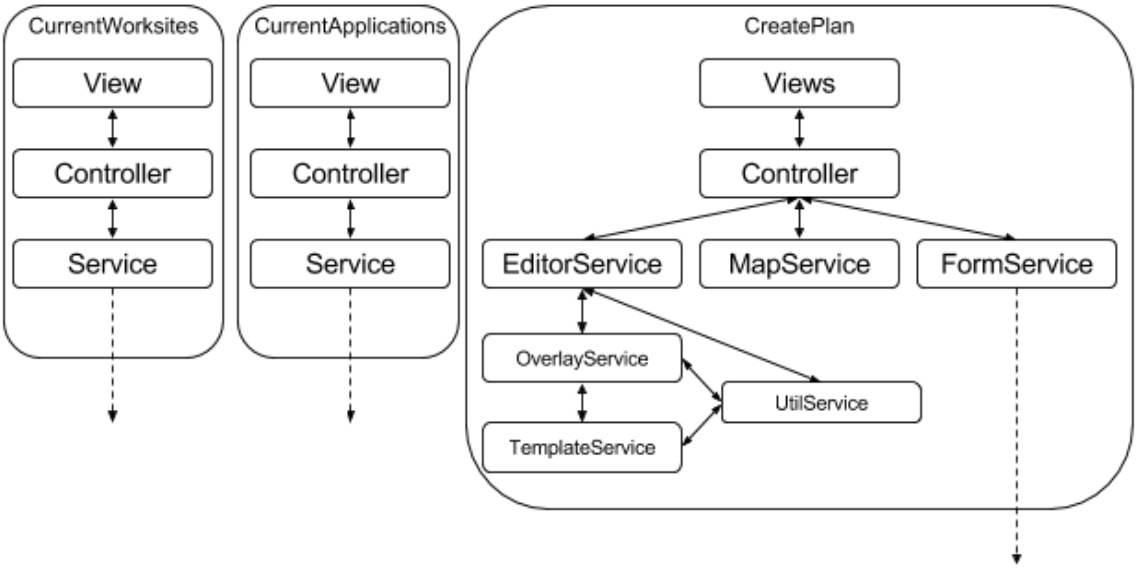
Adapter Layer

Thirdly is the Adapter level. These operate as adapters to other systems that the web application may need access to. The primary uses in this system are connections to the Queue and Database platforms that are required by the Service layer. By encapsulating the logic of connecting to these outside services, the backing technologies can safely be modified or changed with changes only required within the Adapters layer of the system.

Generators

Not adhering to this model is are the Generators. These are standalone background services that run complex operations separate from the Web projects, such as the rendering of documents and images of a work site. While these arguably could exist somewhere in between the Adapter and Service layer of the back end model, it is more logical to treat them as not part of the architecture, and instead as an outside service.

4.3.2 Front End Architecture



The front end architecture is is a different model to the back end, yet it still is designed to achieve a clear separation of concerns. It adheres strongly to the Model-View-Controller model in seperating the presentation logic from the base application logic. In this model, a module is split into three components. The first is the **View**, an HTML template that is rendered by a **Controller**. The Controller acts as a bridge between the Model, which in Coneheads is represented by singleton class known as a **Service**. When a user interacts with the View, an event is triggered, and sent through the Controller, which in turn is passed

to the Service. The Service enacts on this event, and passes it back through the stack to be rendered onto the View by the Controller.

The system itself split into three modules, `CurrentWorksites` to capture the Requirement 4, `CurrentApplications` to capture a in part Requirement 5 and `Create` which covers a majority of the remaining requirements. In comparison, The first two modules are relatively basic components, building off data and information derived from the `Create` component, and therefore have a simple three-component hierarchy. In the model above, `Create` has significantly more logic behind it. In order to keep complexity down, some services have services themselves that handle their own responsibilities. An example of this is the `TemplateBuilderService`, which loads in the JSON template of a work site and calculates locations on the environment selected. This is passed to the `OverlayService` which then renders it on the map.

4.4 Technology Decisions

The selection of technologies is important in the early stages of a system. Choosing older technologies may in the future become unsupported and eventually unmaintainable. On the other hand, choosing a new technology brings with it issues such as stability and support for when something goes wrong. In designing Coneheads, a number of technologies were considered for various components in the system. In this section, the back end and front end technologies are discussed, specifically, the Web Framework and Database for the back end, and Map software and Framework for the front end.

4.4.1 Back End Technologies

Web Framework

At the core of the back end was the framework used to power the API that was designed. In research, three possible technologies were considered: Django, a Python based framework, Iris, a Go based framework and WebAPI 2, a .NET technology. The decision on which framework was chosen would have a large effect on the remaining software selected. A key attribute sought after in this regard was the simplicity of setting up a basic API and have it deployed quickly. Django has been around for a long time, and with Python being a mature language, has a large community supporting it. Iris, on the other hand, is reportedly the fastest framework available [CITE], and its language, Go, was specifically built for web-based applications. However, it was decided to develop on WebAPI 2 for Coneheads for multiple reasons. Firstly, it was selected ahead of Go due to the age of the ecosystem; Go is a relatively young programming language, and the support is sometimes difficult to find. Its dependency management is also somewhat confusing, with the package manager behaving radically different to .NET or Django. When considering the remaining frameworks, Python, the language in which Django is written in, is what ultimately deemed it not suitable for use in the back end. Python is a dynamically typed interpreted language, whereas C#, the language of .NET, is statically typed and compiled. This allows for a number of benefits when it comes to debugging and static analysis, with unnecessary code and possible operation paths discovered at compile time. Lastly, .NET has wide support for document manipulation. The format for word documents, `.docx` was proposed by Microsoft, and their platform is well suited to the editing of documents programatically. Choosing WebAPI 2 has its tradeoffs, however. Unlike Iris and Django, WebAPI 2 is restricted for use on Windows based systems, and the web server requires some more configuration to set it up. Despite this, it was deemed the benefits outweigh the negatives, and WebAPI 2 was chosen.

Database

Three database systems were considered: MySQL, PostgresSQL (Postgres) and SQLite. By nature, the TTMP process is heavily structured, which was why NoSQL databases such as key-value stores were deemed unsuitable for use. In determining which database system to use, planning first had to be done on the type of information required to be stored into the database. Postgres has an add on known as PostGIS that enabled spatial and geographical information to be stored natively, and was considered as an option. However as the design matured, it was determined that the features were not required, as the front end dealt with object conversions, meaning the main concern of the database was data persistence. MySQL and SQLite are similar in how information is stored and retrieved, however SQLite is a more portable and easy to set up database system. Consequently, SQLite was chosen based on the fact that this was an MVP application, and it is suited to the idea of quick development. As mentioned above, the back end is designed so that if an adapter technology is wished to be changed, this can be done in isolation with little or no impact.

4.4.2 Front End Technologies

Framework

Initially, the system was design to be built using 'vanilla' JavaScript without the aid of any Framework, however this was quickly realised to be unsuitable. The system itself has a complex front end, and without the aids of a framework, the system concerns could easily spill into other responsibilities, rendering it unmaintainable. Because of this, two frameworks were considered: AngularJS and ReactJS. Both act as a frameworks that offer superior benefits over ordinary JavaScript in terms of flexibility and organisation. However, ReactJS is more of a library to facilitate better UI design, while Angular is a fully fledged framework offering inbuilt functions to deal with operations as routing and cookie management. Angular also forces the use of a particular architecture (MVC), whereas React has no set architecture. In order to ease the burden of managing a large JavaScript codebase, Angular was chosen due to its rigidity it enforces on the developer, helping to achieve a modular and easy to maintain system.

Map Software

It was immediately apparent that in order to capture Requirement 1 that the system needed a Map based editing and work site definition pane. This map pane also requires the ability to access the underlying Road data so that calculations can be made based on the desired work site. In online mapping, there are two providers of this type of information; Google Maps and Open Street Map. A distinction needs to be made between the services each of these systems provide. Open Street Map is more an open source database of road and related information, while Google Maps is a proprietary stack of technologies that create a map. On top of this, the road information is available to Google Maps users through a paid-access API. The issue with the proprietary stack is that Google Map services are all provided through an API, rather than a downloadable and freely editable library. Open Street Map, however, only provides the data for drawing the map, therefore a library would be required to render this information. This library is called Leaflet. Leaflet acts more as a framework that allows a map source to be added, decoupling the desired Map tileset from the library. Coneheads requires a heavily customised map system, including the ability to edit the underlying mapping framework, which is why having an Open Source library was desirable so that such edits could be made. The benefit of using Leaflet was also the community back-

ing it; there are a large number of plugins that provide some of the functionality required in Coneheads, such as overlay interaction. Because of its open and entirely customisable nature, Leaflet, backed by Open Street Map data, was chosen as the map system.

4.5 Chapter Summary

In this Chapter, the design and architecture of the Coneheads system was discussed. User stories and use cases were used to empathise with users of the system, and aid in developing an intuitive user interface on the complex portions of the site. At a more technical level, the application adheres to a classical Server-Client model, running .NET Web API and SQLite on the back end, with AngularJS on the front. The Map library used, Leaflet with Open Street Map, allows for far superior customisation compared to its counterpart Google Maps, which was why it was chosen for use in this project.

Appendix A

User Stories and Use Cases

A.1 Persona: Joe Murton

Joe Murton is a 27 year old worker at a local contracting company. The company is relatively small, and therefore cannot afford a team of administrators to fill out numerous forms that are required in such an Industry. Joe is nt the most comfortable with computers, but courses taken at polytech have taught him the basics - primarily word processors and the like. Despite this, he has seemingly become the de-facto TTMP application submitter at his company. Because of this, Joe has qualifications in Traffic Control and as a Site Traffic Management Supervisor, meaning he is well versed in the Code of Practice.

A.2 User Stories and Use Cases

Each following scenario is started off with a brief user story, describing the 'who,' 'what,' and 'why' aspects of a given action. Following the user story, more in-depth use cases are defined to give a more detailed flow through the application.

A.2.1 Scenario 1: Viewing Current Work Sites

User Story: `As a contractor, I wish to know of any conflicting or nearby work sites underway so that my company does not file an unnecessary claim`
Use Case: Joe logs into Coneheads and is met with the TTMP Dashboard He has been asked to check if the contract for the council in Taupo has been applied for by a rival company. There is no point for their company to apply for work on a street if it has already been carried out by a rival competitor.

1. Joe clicks on the *"Current Work Sites"* tab on the toolbar.
2. He is met with an interactive map. A legend shows that all work completed by another company is company is displayed as a grey outline on the street. However, this map is currently zoomed out at a level to show New Zealand, not just Taupo. All that is currently shown is markers where the work sites should be.
3. Joe zooms in to Taupo closer, and sees that the markers spread out the closer he zooms in. When he is close enough, he begins to see the markers turn into lines signaling the existence of a work site.
4. He navigates to the road he has been asked to investigate to find it has a grey line. Hovering over the line shows that their competitor has filed for work on this road. This means they do not have to apply.

A.2.2 Scenario 2: Loading Current Applications

User Story: `As a contractor, I wish to load a previous plan that has been started to either revise or complete the application process, so that we can successfully send a submission to the RCA`
Use Case: Another member of the company wishes to upskill and learn the Coneheads system. That way, if Joe was sick and one needed to be filled out, it could be done. She asks Joe if he would be able to double check her progress before submitting it to the RCA. Joe logs into Coneheads, and as before, is met with the TTMP dashboard

1. Joe clicks on the *"Current Applications"* tab on the toolbar.
2. After the plans are loaded, he notices three levels of applications laid in front of him; Partial, Completed and Submitted. It is a completed plan, as his workmate said, so he looks for the one that she completed.
3. Joe clicks load on the correct plan, and is taken to the form completed by the workmate, as she had left it, able to review all sections of it.

A.2.3 Scenario 3: New Application

User Story: `As a contractor, I wish to load a previous plan that has been started to either revise or complete the application process, so that we can successfully send a submission to the RCA`
Use Case: Joe has been tasked by his managers to fill out a TTMP for a local road to pursue work on it. He is familiar with the road, so he knows roughly where it is. He is also told that it is to be a T̈wo Lane, Two Way Single Alternating Flowp̈lan, the shutting off of a single lane to be manned by a manual traffic controller. As before, Joe logs into Coneheads and is met with the TTMP dashboard.

1. Joe clicks on the *"New Plan"* tab on the toolbar.
2. He is met with a popdown that asks what name this plan should be given. He fills in the textbox and presses the next button.
3. He is then taken to a page that presents a map. As Joe hovers over each road, information on the average daily traffic, the speed limit, road widths and RCA appears on a box to the side.
4. He zooms in and finds the street his manager asked him to fill out.
5. Joe then clicks in two places to outline the proposed work site. The distance between the two clicks is provided in the information box to the right.
6. Once the second click is made, the work site is emboldened to allow the user to confirm that this is the desired work site
7. He is off by a couple metres, so he drags the point such that the metres are accurate to the proposed work.
8. He clicks next, then is presented with another map. Another popdown appears, asking a couple calibration questions so that the system can accurately determine where it is.
9. Once calibrated, a default plan appears on the window in front of him. It isnt the plan he was looking for, so he looks to the control panel, and selects Two-Lane Two-Way, Single Alternating Flow.

10. The map is redrawn, and is accurate to the work site that was requested. Joe sees that some cones are in the way of a residents driveway, so Joe clicks on the cones, greying them out, showing that they have been removed from the base plan.

11. Happy with the plan, he continues on and fills out the proceeding form. He notes that a number of boxes have been automatically filled out, both since the information from the map has flowed through the form, as well as personal settings being included, such as the contact information of the company.

12. Joe then decides to print off a copy to send to the RCA for approval.

Appendix B

TTMP Form and Diagram Examples

B.1 TTMP Form Example

RCA consent (eg CAR/WAP) and/or RCA contract reference	*Add RCA consent reference, for example the corridor access request (CAR) or work access permit (WAP) and/or any RCA contract reference.*

TRAFFIC MANAGEMENT PLAN (TMP) – FULL FORM

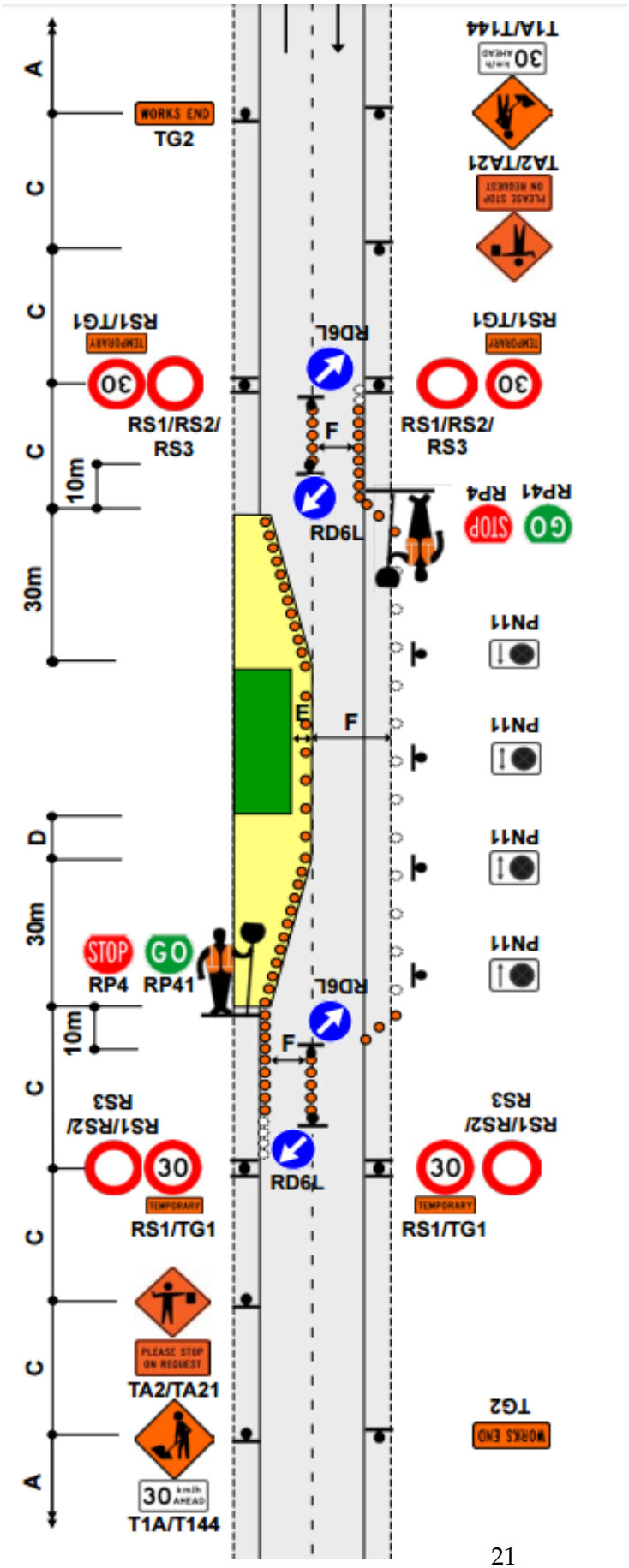
Use this form for complex activities. Refer to the NZ Transport Agency's Traffic control devices manual, part 8 Code of practice for temporary traffic management (CoPTTM), section E, appendix A for a guide on how to complete each field.

Organisations /TMP reference	**TMP reference:** *Add the RCA's and contractor's reference number*	**Contractor (Working space):** *State the name of the contractor responsible for the working space*	**Principal** *(Client): State the name of the principal or client for this project (eg NZTA or Chorus)*		
		Contractor (TTM): *State the name of the contractor responsible for the TTM*	**RCA:** *State the name of the RCA who controls the road that the worksite will be on. Note: There can be more than one RCA.*		
Location details and road characteristics	**Road names and suburb**		**House no./RPs** *(from and to)*	**Road level**	**Permanent speed**
	Include the road name/s and any affected intersections. Also include the suburb		*Enter house numbers, route positions or power pole numbers where applicable*	*Enter RCA designation*	*Enter highest permanent limit*
	As above		*As above*	*As above*	*As above*
Traffic details ***(main route)***	**AADT** *Include AADT where available.* *The RCA or engineer must provide this information if available.*		**Peak flows** *Include peak hour and heavy vehicle counts where available.* *The RCA or engineer must provide this information if available.*		

Description of work activity
Briefly provide an accurate and complete description of the work or activity eg repairs to median barrier

Planned work programme							
Start date	*Enter earliest date activity may start*	**Time**	*Enter earliest time activity may start*	**End date**	*Enter latest date activity may finish allowing for unforeseen issues*	**Time**	*Enter latest time activity may finish allowing for unforeseen issues*
Consider significant stages, for example: • road closures • detours	*Provide details of any significant stages*						

B.2 Work Site Diagram



Bibliography