Bridgestone NZ Ltd: Electronic Job Card

Mid-Year Report

Scott Patterson

ID: 1304962
UPI: Spat239

Department of Computer Science
University of Auckland
Auckland, New Zealand

June 2012

CONTENTS

Abstract-**With mobile devices being integrated more and more into the everyday lives of today's people, the area of designing and programming applications for these is one of the more innovative and quickly advancing parts in the current I.T. industry. Bridgestone has seen the potential in these devices and are looking for an application to be created for them in order to help their fleet technicians complete their jobs in an efficient manner.**

The purpose of this report is to detail in depth the first half of Scott Patterson's BTech project for Bridgestone New Zealand Limited in 2012. The project centres around creating a mobile application to replicate the current paper process of writing up a job card, with hopes of reducing redundancy and making the whole process more efficient than it is in the present. This project will deal with queries to databases for current data, and pushing information back to them when new data has been gathered. There will also be additional functionality that I feel is relevant to the scope in helping the fleet technicians complete their jobs in a more effective way.

This report is separated into four main sections; an introduction to the project and the company it is for, giving a look at the motivation and goals behind the project as well as the current state of the company; a research section looking into seeing how the work process is from an end user perspective, as well as understanding the pros and cons of multiple mobile device operating systems, network carrier data plans, and methods of web service design; a project progress section which details all planning, documents and applications developed so far that have helped move the project forward; and finally a conclusion which addresses solutions found to all problems including those researched as well as a plan for the future of this project.

1. INTRODUCTION

1.1. OVERVIEW

One of the major thriving industries in today's world is that of the handheld system. Handheld phones and tablets are constantly being given additional functionality and are becoming a competitive market with a number of participants. iPhones and iPads were one of the frontrunners leading this revolution in 2007 [1], and more recently Android phones have been seeing a rise in popularity after their 2008 release [2], being named the leading platform in Q4 2010 [3]. Seeing the potential in this market Microsoft came out with Windows phones in 2010 [4] and has been developing their Windows 8 platform with touch screens in mind as a central focus, allowing it to be used on Windows phones and tablets. A couple of the biggest draws of these devices are that they are easily portable, being able to be used almost anywhere, and that the applications that they support are very versatile, being able to be used for almost anything. With these two main draws in mind, Bridgestone wants to get a mobile application created for them through The University of Auckland BTech programme.

The goal of this report is to detail the mid-year progress of this project, giving introductions to Bridgestone as a company, the project and the motivation behind it, the goals the project hopes to achieve, and the current systems in place that the project will be dealing with. This report will also detail all research done both inside Bridgestone with the end users, and outside of it for what hardware and software would be best to develop on, different networks that can be used, and different methods of data transfer. The current progress made will also be detailed, showing a variety of planning documents for the project as well as a minor side-project that can be thought of as a minimised version of the overall end product.

1.2. THE COMPANY

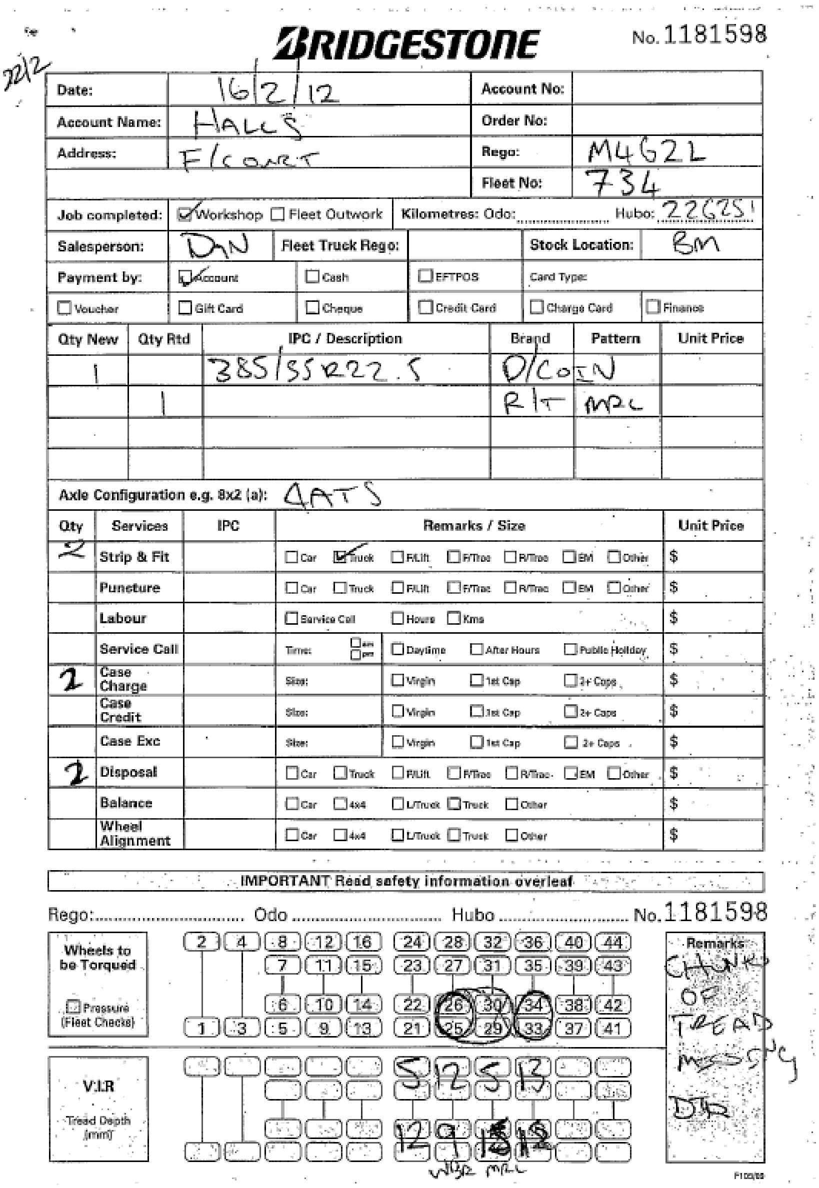
The Bridgestone Corporation was founded in 1931 in Japan by a man named Shojiro Ishibashi. It has since become the world's leading tyre manufacturer for cars, trucks and busses among others [5]. On the path to becoming the well-known corporation it is today Bridgestone bought out and merged with Firestone, one of its leading competitors in the market in 1988 which only helped the two to grow further. The New Zealand section known as Bridgestone New Zealand Limited was formed ten years later in September 1998. The size of the corporation is not all that has led to their success, as they are renowned for producing quality tyres around the world, and being allowed to be a part of the corporation for this year attempting to promote efficiency on their work is truly an honour.

From inside the Auckland offices it is clear to see that the Bridgestone Group stays strong to its roots and bases its philosophy on four goals given by the initial Japanese company:

Seijitsu-Kyocho [Integrity and Teamwork]
Shinshu-Dokuso [Creative Pioneering]
Genbutsu-Genba [Decision-Making Based on Verified, On-Site Observations]
Jukuryo-Danko [Decisive Action after Thorough Planning]

These four goals can be seen in multiple places inside the offices as a constant reminder of what the focuses of the corporation should be. In addition to this Bridgestone New Zealand Ltd is known to be very conscious of the environment, and holds the internationally recognised carboNZero^{CertTM} certification.

The official project description as given by Bridgestone was as follows: *"A mobile application needs to be developed for their electronic job card. This is to emulate the manual recording of information on a job card to a Mobile device with a view linking ultimately to the Advanced Retail POS system."* This gives a pretty general but easy to understand view of the project. Currently there is manual recording of job card information on paper documents like that shown in figure 1, which is later inputted into the Advanced Retail Point of Sales system. The application should replicate this in a way that is simple and easy for fleet technicians to use, but still link with the POS system so that invoices can be produced for each job done. This would help reduce the redundancy and promote efficiency if the data need only be inputted once into a mobile application, and then have that data be sent to databases from which an invoice could be produced with no need to re-enter this information. I also hope to include other functionality outside of simply reproducing the paper system in order to help the jobs be carried out more easily, for example the ability to look up stock and find the closest location to a fleet technician, so they will be able to pick up stock more quickly instead of calling different places around and waiting for the person on the line to find out what stock is available.


BRIDGESTONE No. 1181598

Date:	16/2/12	Account No:	
Account Name:	HALLS	Order No:	
Address:	F/CORT	Rego:	M4621
		Fleet No:	734

Job completed: Workshop / Fleet Outwork — Kilometres: Odo: .... Hubo: 226251

Salesperson: DAN — Fleet Truck Rego: — Stock Location: BM

Payment by: Account / Cash / EFTPOS / Card Type: / Voucher / Gift Card / Cheque / Credit Card / Charge Card / Finance

Qty New	Qty Rtd	IPC / Description	Brand	Pattern	Unit Price
1		385/55R22.5	D/CO	IN	
	1		R/T	MRL	

Axle Configuration e.g. 8x2 (a): 4ATS

Qty	Services	IPC	Remarks / Size	Unit Price
2	Strip & Fit		Car / Truck / F/Lift / F/Trac / R/Trac / EM / Other	$
	Puncture		Car / Truck / F/Lift / F/Trac / R/Trac / EM / Other	$
	Labour		Service Call / Hours / Kms	$
	Service Call		Time: am / pm; Daytime / After Hours / Public Holiday	$
2	Case Charge		Size: Virgin / 1st Cap / 2+ Caps	$
	Case Credit		Size: Virgin / 1st Cap / 2+ Caps	$
	Case Exc		Size: Virgin / 1st Cap / 2+ Caps	$
2	Disposal		Car / Truck / F/Lift / F/Trac / R/Trac / EM / Other	$
	Balance		Car / 4x4 / L/Truck / Truck / Other	$
	Wheel Alignment		Car / 4x4 / L/Truck / Truck / Other	$

IMPORTANT Read safety information overleaf

Rego:........ Odo .... Hubo .... No. 1181598

Wheels to be Torqued — Pressure (Fleet Checks)

Remarks: CHUNKS OF TREAD MISSING DTB

V.I.R Tread Depth (mm)

W/32 MRL

Fig. 1. This is the currently used Job Card in its paper form. This specific job card is already completed so you can see the customer information, invoice number, job information and vehicle information all written, as well as some remarks.

1.4. THE PROJECT GOALS

To successfully complete this project the application primarily needs to be easy to use and must mimic their current processes shown in figure 1 on a smaller and smooth looking interface. The job card replication will need to contain only information that needs to be entered into these Job Cards, no unnecessary extra information which overcomplicate the application and add clutter to the small screen. The application should be familiar to the users using colours fitting for Bridgestone and there should be no scrolling of long pages, as it is to be designed for a mobile device. Any additional functionality should be on separate screens or applications so as to not complicate the job card application. These goals can be separated into the five main parts of functionality; aesthetics; programming; legal; and skills.

The final application needs to have the fundamental functionality of being able to input job card information that can be sent off to a database. The application will need to link closely with the VIR application in order to populate any and all information about the current job that it can, and then after this additional job information will need to be able to be easily entered. There should be validation on input fields to check if they meet SLAs and rules such as having the same tread along all wheels on an axle. After this validation the application should check if the job will push the client (if it is a job for a current client) over their credit limit with Bridgestone, as giving some kind of notification when this happens would be useful. In the middle of the process there should be functionality to allow for stock lookups by location to find out where they can get certain tyres from. Finally after all information has been entered it should be able to be sent back to Bridgestone's databases so that an invoice can be produced.

In terms of aesthetics there are multiple goals that I hope to achieve with the design of this application. I hope to make the content easy to read and easy to view for the target audience. Content should be sized appropriately and be visually recognizable to the users so they can distinguish it from other non-Bridgestone applications. All images and visual content should not only be appropriate but should also be smooth and should fit well onto the screen size of the designated device. As mobile devices have smaller resolutions than PCs and laptops it is a goal to avoid scrolling where possible, so content should be divided between multiple pages that are easy to navigate between naturally instead of simply having one screen replicating the paper form that needs to be scrolled and zoomed to use.

From a programming perspective I hope to make the application run in a way so it does not function too slowly, as this would be inefficient which contradicts the main purpose behind the application. In order to achieve this I will need to look at ways to create mobile applications properly and how to handle wireless data transfer using web services that are quick. I should also be conscious of how I code and indent everything properly with appropriate comments, in case the application needs to be understood and modified by someone else at a later date.

Avoiding copyrights is an important societal - legal factor, if copyrights are ignored and information or images are used without the permission of their creators then it becomes a legal issue. I must make sure all content is used with proper rights and references, as well as make sure that no sensitive information is leaked outside of the company about their business and how it operates. I must also make sure unauthorized users are not able to modify data incorrectly in ways that could disrupt the proper business flow.

There is also a skills factor in that the creator of the application will need to develop the skills necessary to meet the requirements of the project to create a product that is both properly functional and aesthetically pleasing in ways that are appropriate for the end use.

1.5. SYSTSMS IN PLACE

Bridgestone being a well-established corporation has many current systems in place to handle their data in various ways (figure 2). The data warehouse is where most data is stored eventually after coming from various locations; Wherescape RED builds this data warehouse and its fact tables. Figure 3 gives a closer look at where the customer, transaction and vehicle data comes from (indicated by the red circle in figure 2) before it gets sent to a data warehouse, and how it interacts with SAP PI. BPCS deals primarily with transaction information. CRM deals mainly with customer details but also the POS system information which is then sent to the data warehouse via SQL. These systems interact with the SAP PI which has been recently introduced into Bridgestone New Zealand Limited. Vehicle information comes from a vehicle repository where a third party web service allows a registration number to be looked up in order to return comprehensive information on a certain vehicle. The customer and fleet information, including this vehicle information is stored in the data warehouse via overnight processing.

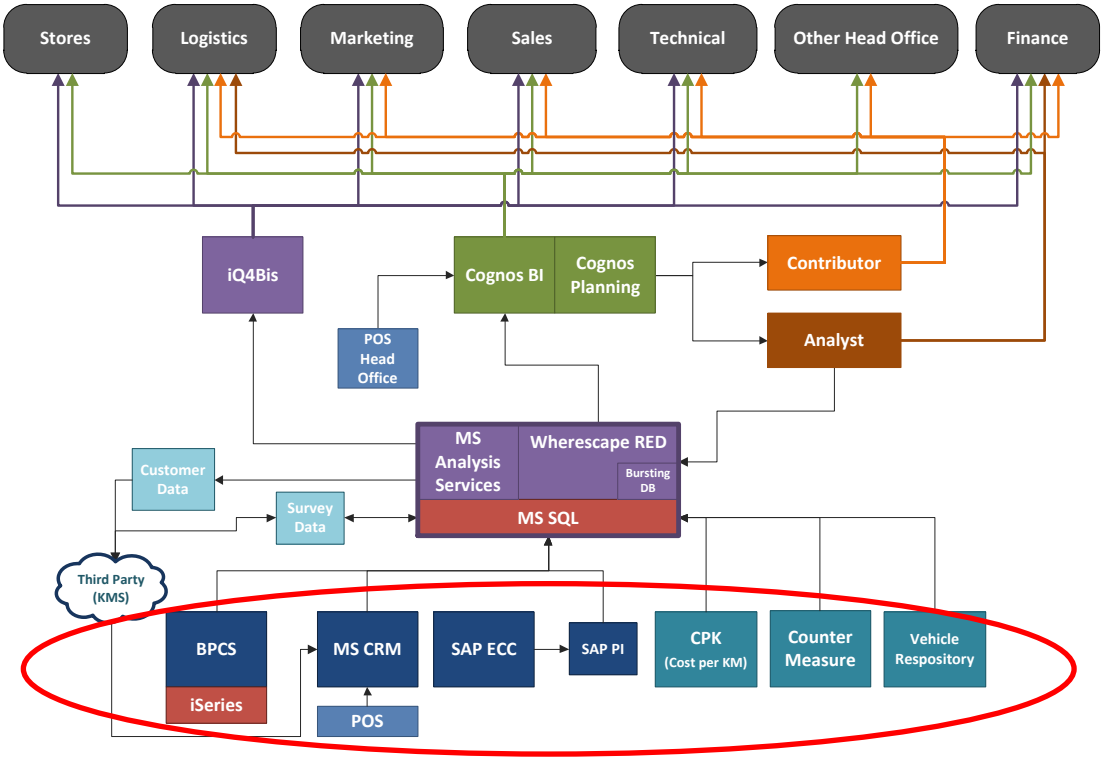


Fig. 2. An overall look at the systems and links in place inside Bridgestone New Zealand Ltd.

The Electronic Job Card will mainly be pulling data from just the VIR application as opposed to from the Wherescape RED data warehouse. However it will be dealing with the customer, transaction and vehicle information that the VIR application has. Customer information will be able to give my application data like SLAs for a customer, their credit limit as well as simple pieces of information such as contact information. For a specific customer I will be able to find transaction information to give outputs like previous transactions to give an overview of what kind of jobs their fleet has been having and how frequently. I will also need information about the vehicles in these fleets and their tyres as these are what the jobs will be carried out on, tyre information can be used to enforce specific rules such as same tread type across an axle.

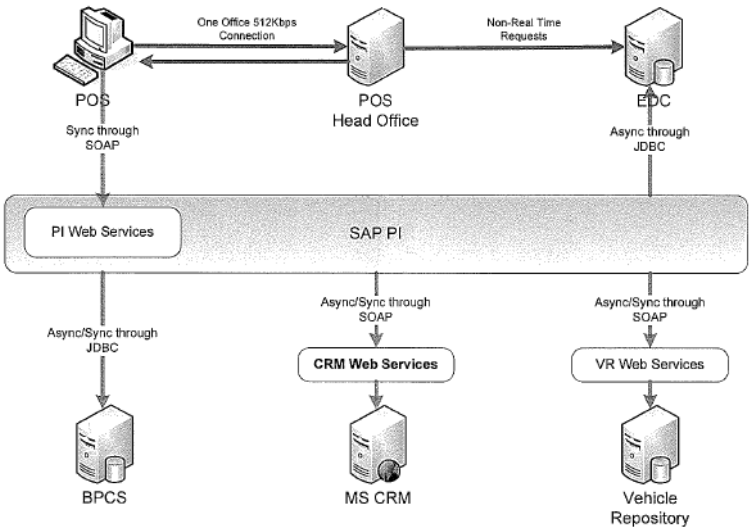


Fig. 3. A closer look at BPCS (transactional data), CRM (customer data) and the Vehicle Repository (vehicle data) and how they link with POS through SAP PI.

2. RESEARCH

2.1. END USERS

On the 30th of April we went on a 'field trip' of sorts to meet the end users of the application. In this case the end users are fleet technicians, workers who go out to various locations to monitor and work on fleets of trucks. These locations can be both dirty and dangerous, and are sometimes in locations out of the way where phones are out of range. The purpose of this trip was to meet end users and see the current work process in action, allowing us to ask questions and identify areas where there is potential for our applications to improve the efficiency of their job. After talking to the fleet technician I was assigned to and asking questions about different situations I was able to come up with the following figure 4 to show their usual work process, however they stressed that it is not a constant thing as there are many different things that can happen, therefore the final application needs to be appropriately flexible to deal with these situations.

It was interesting to see that there were a large number of different fleets that Bridgestone handles, some very well-known over New Zealand. Examples of these were Transrail, Foodstuffs, Komatsu, KiwiRail, Fulton Hogan (a recently acquired customer), Mainfreight, and TR Group (a rental company so there are always different vehicles at the site, but if they are at the site they usually remain there for a while between rentals).

Usually it takes about 30 minutes between start to finish for a job card, but it can take a lot longer depending on the job, such as if difficulties arise or things like changing the larger tyres which are quite expensive and require at least 2 people, 3 or more ideally, and can take about an hour and a half each. Each tyre gets the related job card number written on them, so it is quite important that this information be easily visible on the tablet application, possibly on every page in a corner.

The most important pieces of information on the job card as perceived by the users are the customer name and phone number, these are the things they make sure to have for every job so that they can contact a customer. Each job card is double checked before it is sent off to make sure all required information is present and correct so it may be important to provide functionality to check over recent jobs done so that the inputs can be double checked, however with the field validation I hope to reduce the need for this. Different workers do things differently; some do all the work first then write the paper documents later to get the work out of the way; others check out the vehicle first and note everything down before doing any work, so it is important to allow for the application to be opened and used at different parts of the work flow as opposed to being required to be always before or always after a job is carried out.

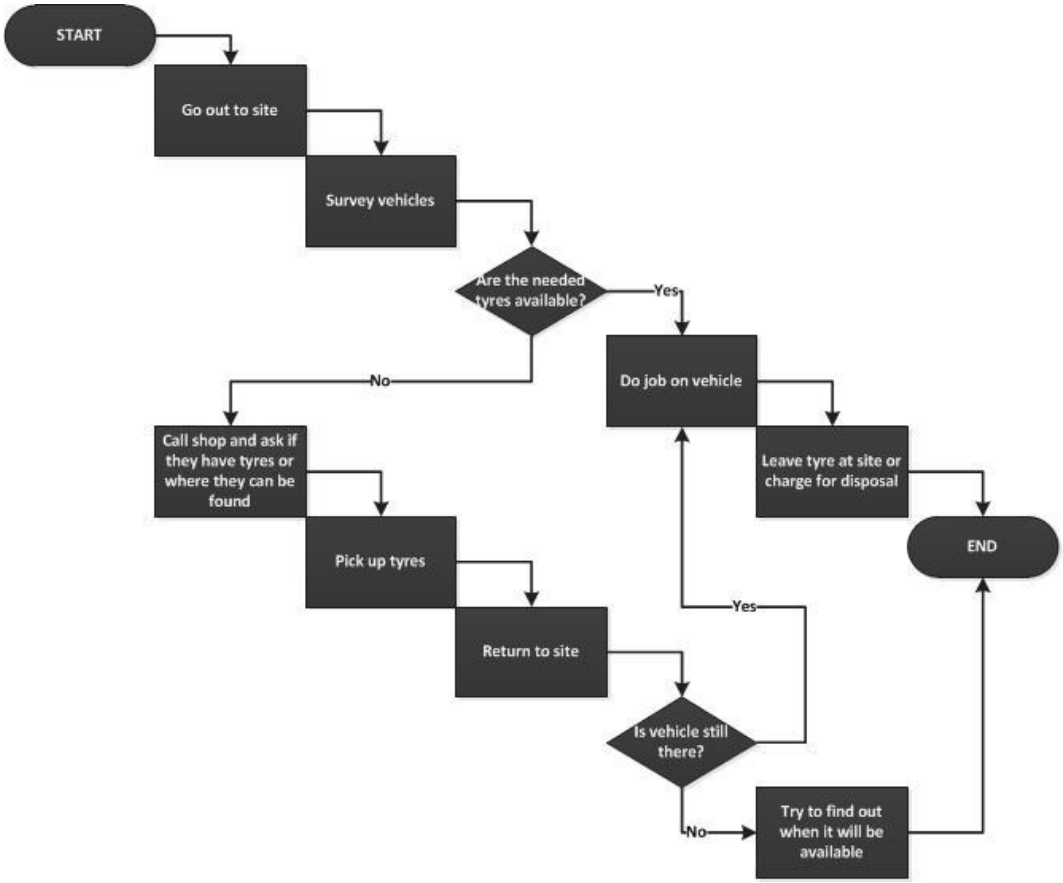


Fig. 4. A common workflow of jobs for a fleet technician and some kinds of issues they may face.

Fleeties often have locations they go to first thing where they stay most of the morning checking out the different fleets for issues. This takes pressure off of the customers with the fleetie going to them instead of needing every vehicle to come to their workplace. Fleeties usually have set hours for locations they check. When checking out a fleet first they survey all the vehicles there which can take 15 minutes to half an hour depending on fleet size, looking for things like punctures, bare tread, or mismatching tread on the same axel (this is not allowed due to balance issues). They then get authorization to work on these. The fleetie I was with often wrote notes about the vehicles and work to be done on his hand and then re-wrote it in a notebook later when he could. There is potential here for reduction in redundancy with portable tablets.

If the fleetie does not have the correct tyres for the jobs they will need to pick these up. This can entirely depend on the vehicle they are driving and how many tyres it can hold. The fleetie I was with was working in a smaller vehicle and could only carry around 8 tyres at a time, therefore he went to make note of all work to be done, and then returned to pick up the needed tyres. Sometimes these tyres need to be picked up from different places than their usual workshop. If a tyre is indeed replaced the old case is usually left with the company and not taken back to the workshop, though sometimes Bridgestone will buy these cases off the companies and then re-tread them to resell later.

Before going back with tyres the fleetie will usually call the company they are going to in order to double check that the vehicles they wish to work on are still there as often they can be in for only a short time before going out again, it can be difficult to catch them. It depends on the company, for example a rental company will usually have vehicles sitting for a while, but other companies with working trucks only have some vehicles sitting at a location for minutes before they are back out, sometimes work needs to be scheduled in advance for these.

One person a week takes the after hour calls, this means they must respond to all calls outside of business hours at various locations which includes both late night and early morning calls, often working all night without sleep. The fleet technicians have a week where they work after hours then at least 3 week off on a rotation with other technicians. After they have worked for 16 hours they are required to get at least a 9 hour

break for sleep. This is particularly stressful with them lacking staff right now and it putting a large workload on these people during these times.

The busiest times for a fleet technician are usually very early morning before people start work before 7am, or late afternoon when work is finished after 5pm. This is due to the fact that during these times fleet vehicles are not as likely to be on the road. During normal work hours is usually the slower time for fleeties and there is more catching up of vehicles and doing whatever work they can do. There are still things to do constantly but there is more moving required as vehicles are not bunched up together in one location as they are at the start and end of a work day.

In addition to going out to fleets, the workshop is open to whoever comes in. This almost always involves people with an existing account but they do get the occasional cash sale. Customers usually have an idea of what needs to be done or a specific request. When the fleet technicians are not at the workshop there is an in house GPS system which exists to see where all fleeties are real time. This is used quite frequently throughout a day. There is a lot of communication and phone calls between the fleeties and the workshop, as well as between the fleeties and the clients, so it is useful to know which fleetie is where.

When taking stock the in store stock-taker must be informed and it must be noted, it is important to keep track of all stock at all times and all changes in stock must go through this worker. Stock has three states being: in the workshop, on a work truck, or used after a job. If a tyre is damaged by a worker it needs to be brought up with the stock-taker and replaced, the worker is charged for the tyre as it is their responsibility. There is a lot of knowledge required for damage reports in terms of identifying cause and effect of damage, they must observe, identify and note this based on what they already know and this knowledge cannot easily be replicated by a computer system. After re-treading the new tread does not necessarily match the case, and thus the tread type noted on the case is not always right; workers must be careful. Customers do not usually know this and can misinform the workers if they are talking through a phone call and the worker cannot see the tread.

Often different fleets are located right next to each other, as different companies own spaces that are in close proximity. This allows fleeties to do multiple sets of jobs back to back assuming there is work to do and vehicles are available which is quite efficient. It is also possible for fleeties to be called out by known customers with large vehicles to work on the customer's personal car. However most of their tools cannot be used on them as they are designed for stronger, larger wheels. For example the fleetie I was with was called out to pump a wheel that blew out on a customer's personal car so that they could drive somewhere where they could get it replaced or repaired.

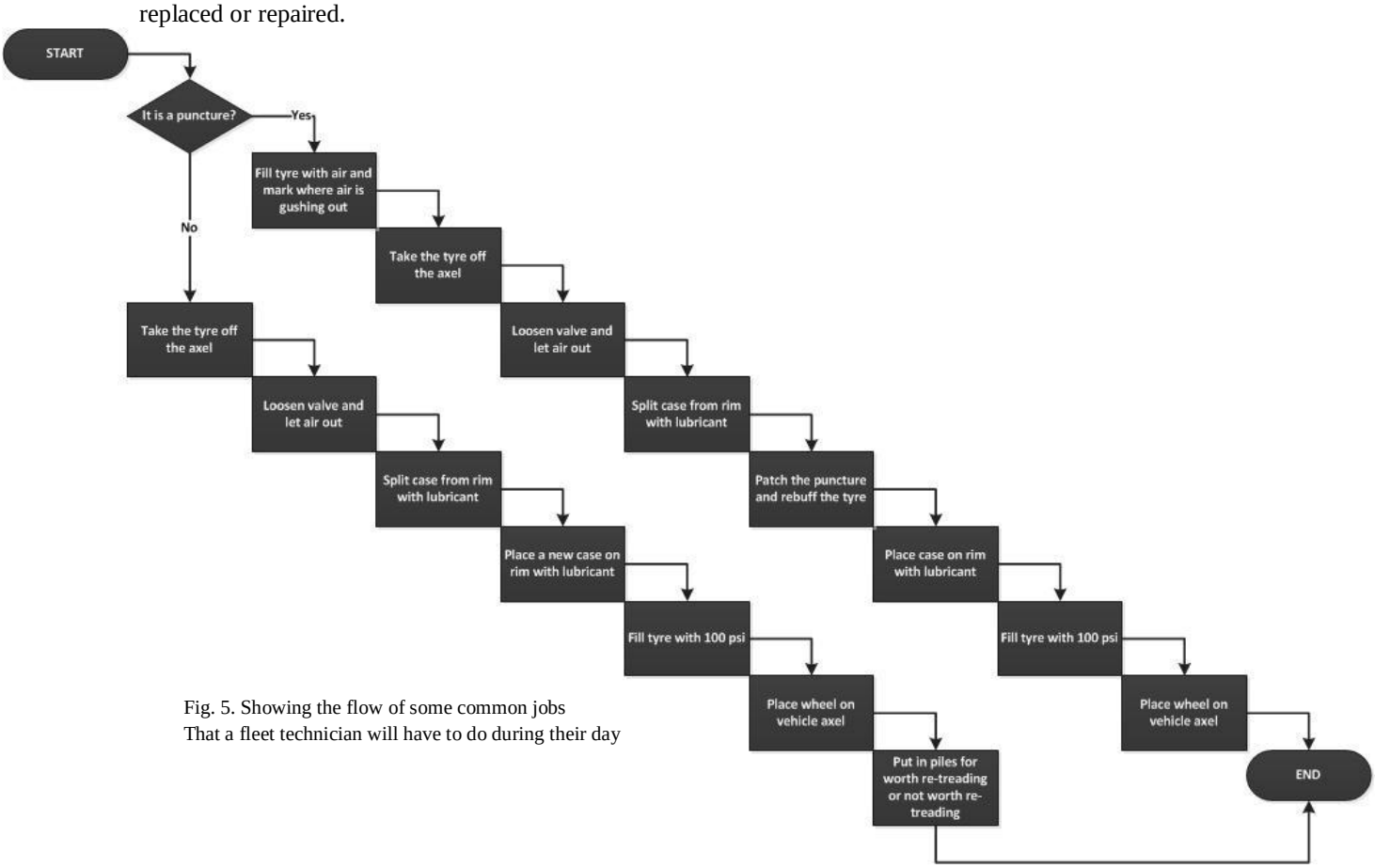


Fig. 5. Showing the flow of some common jobs
That a fleet technician will have to do during their day

2.2. POSSIBLE DEVICES

In today's market when developing for mobile devices there are really two main competitors that come to mind, those being iOS and Android, with Windows attempting to break into the industry in a big way more recently. All three of these OSs have different pros and cons, and different rules and restrictions for them, therefore when deciding exactly which device to build the Electronic Job Card application for I must first research the alternatives to gain a better overall understanding of the options. I have researched both iOS and Android development by looking at various sources and finding relevant information, however there is not as much information available for Windows 8 devices as they are still in beta development, however on the 22nd and 23rd of March I attended a windows camp, with a 9-5 lecture on the first day and a 2-4 interactive lab on the second. There are also current Windows Phones running an earlier version of the OS that can be looked at.

First looking at Apple, iOS is known for being very smooth and consistent for their interfaces on both the iPhone and iPad, including inside of apps. The Apple Store that is the source of applications distributed on iPhones and iPads is the largest app store currently, and it very well maintained. However in order to keep the reputation Apple has with these things, applications must be relatively consistent throughout. This means there are some rules put in place and there are restrictions on flexibility for developing apps, especially if you wish for them to be distributed.

An example of this is in interface design, where Apple has a list of 'Interface Guidelines' [6] that influence all applications that run on their devices. While this can be quite useful in terms of design tips, it is also somewhat restrictive for those that wish to have more freedom with what they develop. These guidelines include such things as the characteristics of the platform and how to best integrate these for a successful app; interface principles which define how well an applications appearance integrates with its function as well as if the app is consistent with iOS standards; strategies for designing apps based on iOS UI paradigms, and guidelines to elevate user experience based on using iOS. Overall while they have the users in mind when giving these guidelines and standards and the end result is a very consistent and clean platform, it can also be restrictive.

One thing that separated iOS devices from others is that the range of hardware that supports iOS is very limited. Official Apple made iPhones and iPads are the only devices which will run Apple's iOS and therefore users have little choice in their purchase and device specifications, instead they only have updated hardware versions such as iPhone 4 and iPhone 5. Again this is somewhat restrictive however the hardware is trustworthy and secure, also while smartphones in general are known for having bad battery life Apple devices are not known to be better or worse than their competitors, they are just a solid and trustworthy middle ground.

Originally in New Zealand iPhones were only available for use with Vodafone as the carrier, however they have since become open for use on Telecom networks as well, and are compatible with 2 degrees micro-sims though they are not an official carriers.

In order to develop for iOS there are specific hardware and software requirements that must be met. Xcode and the iOS SDK are required to develop apps, while free, are only usable on Apple operating systems [7]. In addition to this the programming for these apps must be done in Objective-C, which is a superset of C. This means that being unfamiliar with this language, I would need to learn its conventions in order to program for Apple devices. This should not be too difficult as Objective-C is a simple object-oriented language with easy to learn syntax, but I would need to get familiar with them none-the-less.

Using Xcode and the iOS SDK developers are able to create and simulate their apps, however they are not able to distribute them through the App Store or as in-house iOS apps. For this the appropriate developer programs must be purchased. As the Electronic Job Card is to be created as an in-house app the iOS Developer Enterprise Program would be needed [8]. This would cost Bridgestone $299 USD per annum, which using June 2012 market rates would be an estimated $395 NZD per annum. Without this program we would not be able to test the app on a device and would only be able to simulate it with the simulation tools provided by Xcode and the iOD SDK.

Looking at the same areas for Android devices we can see a clear increase in the flexibility of both the devices and the applications used on them. Android has been developed by the Open Handset Alliance led by Google to be an open source smartphone and tablet operating system, allowing users more visibility and innovation with what they are doing when they create an application.

Because of this flexibility there are less restrictive guidelines on how an application must be created or standards which must be met for the interface than we saw with Apple. Android lists guidelines for creating icons and widgets for the created applications so these may look somewhat clean and consistent [9], however there are less restrictions placed on the look inside the applications themselves. The do provide design principles for user interfaces, but they give more freedom to the users than the iOS principles.

Continuing with the flexible theme there are a great variety of devices that support Android as an operating system. A number of different companies are able to create hardware for Android, leading to a more competitive smartphone market, allowing users to pick a device that has specifications that more suit their needs. Of the companies that create these devices Samsung stands out as one of the leading manufacturers and are known for providing quality products. There are other devices created by other companies that are also more economical options providing the core functionality of a smartphone if this is what a user is looking for. Because of this variety in hardware there is clear variance in the device specifications, including that of battery life. It is difficult to say if Android is more or less battery intensive than iOS due to the fact that different Android devices will be using different batteries that handle stress differently, you can see some Android devices with poor battery lives well below that of an iOS device, while you can also see some with greater battery lives [10].

Android devices have been open for use on the primary New Zealand networks, being Vodafone, Telecom and 2 Degrees so there are no issues with this.

As far as coding for Android goes, it is quite simple to set up the Android SDK on most PCs. One such way of doing so if through the Android Development Tool for Eclipse, allowing for simple programming of Android apps inside Eclipse. Programming for Android with the SDK is primarily done with Java, which is the language I am most familiar with from my studies at the University of Auckland. However with the Android NDK companion tool it is also possible to code in different languages such as C or C++ [11]. While I am less familiar with these languages the flexibility is nice for developers to have, allowing them more freedom to develop in a way that feels more comfortable to them, however some developers are not aware that this NDK is available as Java is the primary programming language for Android.

In order to publish an Android application through Google Play, a store similar to Apple's App Store, you must create a developer profile and pay a registration fee, similar to the developer programs Apple has. However this is not the only way of distributing Android apps as Android also allows for apps to be e-mailed or made downloadable on developers own websites outside of Google Play, and therefore no fee is required [12].

It is slightly more difficult to analyse all these areas for Windows Phones as the Windows 8 platform is still in development, however a lot of information can be obtained from the current Windows Phones running the Windows Phone 7 OS. In addition to this research a lot of information was gathered from attending the Windows Camp held in Auckland. Windows Phone is not open source like Android is, and therefore does have its restrictions. It seems to have a strong focus on consistency aiming for an approach similar to what Apple has done, being solid and consistent as opposed to flexible, so that they are able to promote stability.

The Windows Phone has been putting emphasis on what they call 'metro style apps' for interface design. They have created the OS with touch screens in mind and have attempted to provide developers with all sorts of tools necessary to create consistent apps that take full advantage of their OS [13]. Windows likes to explain their

metro style apps and has guidelines on how to develop them. This restricts users similar to how the Apple apps are handled however it has worked quite well for Apple so it is quite possible that metro style apps will take off with users after the full release of Windows 8 on both PCs and other devices.

It is difficult to find a definite source for the availability of Windows Phone 8 operating systems on various different hardware devices however it seems to be that way. There are multiple different models of phone that run the current Windows Phone operating system such as Nokia devices. So while Windows places restrictions on how their apps are created like Apple does in order to promote consistency, they allow for a variety of devices with different specifications to meet their users' needs like Android. This means there will be options for buyers of different budgets and differing needs in a competitive market. This also means that the battery life of Windows Phones will be varied depending on the device the operating system is placed on as opposed to a constant across one device.

It can also be confirmed based on the current Windows Phones in the market that Windows Phones will be able to connect to the networks all three leading carriers in New Zealand, those being Vodafone, Telecom and 2 Degrees. Current Windows Phones such as the Nokia Lumia 800 support all networks [14].

The current development tools for Windows 8 are still in beta and therefore the development toolkits are available free for use until their release. The windows 8 labs held in Auckland focused on using Expression Blend 5 for interface design for Windows Phones and Visual Studio 11 for the back end programming, allowing for the two to be integrated together to form a final product. These tools include the Windows 8 SDK. Microsoft also hopes to provide developers with the flexibility of deciding how they wish to program their apps and provided guides for their labs at the Windows Camp in both HTML with Javascript or in XAML with C#, VB or C++, they also allow metro style apps to be created with DirectX and C++ [15]. With all these options available to developers from the start Microsoft are allowing developers to feel comfortable with developing apps for their OS, and unlike Android they do not need to find out about and download the extra NDK.

It appears as though in order to distribute applications developed for Windows Phones a Windows Store developer account is required, and is not optional, similar to how Apple handles this. Different prices are required depending on if the registration is for an individual or for a company, and differs by location. For New Zealand the annual company registration fee is expected to be $140 NZD [16].

2.3. DATA PLANS

As all three devices are compatible with the three major New Zealand networks the chosen network can be thought of and decided separately from the chosen device. The main thing to consider when picking a network is the cost of data usage for data sent to and from databases using 3G, however network coverage should also be considered. The data plans given by all three providers have been researched and compared. It is also important to note that the three carriers provide different sections of their websites for businesses than they do for normal users.

Vodafone seems to have a very large number of data plans for businesses; however these plans include all sorts of phone usage such as txts and calls. For specific mobile broadband plans the business section of the Vodafone website redirects to the personal section and therefore there are no discounts [17-18]. Only looking at mobile broadband usage Vodafone provides mainly two options, 512MB of data for $30 per month, or 2GB of data for $49.95 per month.

Telecom provides the same plans for both their personal and business consumers for mobile broadband, however at different prices [19-20]. For each of the three data options the business costs are lower than that of the normal consumer costs. The three options provided for businesses are 750MB of data for $26.09 per month, 2GB of data for $52.17 per month, and 4GB of data for $69.57 per month. They have more plans than Vodafone provided allowing for a 4GB plan is it would be needed. The lower data usage plan of 750MB was actually cheaper than the 512MB plan Vodafone provided, however the 2GB plan was more expensive than the same data usage provided by Vodafone. Also keep in mind that the Telecom rates are reduced for businesses while the Vodafone rates are not.

While 2 Degrees provides two separate pages for their business and personal users' mobile broadband plans, the prices of both are the same [21-22]. 2 Degrees handles their mobile broadband usage a little differently than

the other two providers in that they are not all monthly plans, instead the plans expire after a certain period of time after which (or when you run out of usage) you can purchase more. The data 'packs' available are 1GB of data for $20 which is valid for one month, 3GB of data for $50 which is valid for two months, and 12GB of data for $150 which is valid for six months. It is difficult to compare these prices with those of the other two providers as it entirely depends on how much of your data 'pack' you use and in what length of time.

From this research it looks like for smaller usage amounts Telecom would be a better choice than Vodafone, however for around 2GB per month Vodafone is more economical. If more usage is needed then Telecom offers a larger plan while Vodafone does not. If data usage is low however (averaging less than 2GB per month) and the users do not mind being locked into a plan, the 12GB 2 Degrees plan that lasts for 6 months is the most economical, costing a minimum of $300 per year if less than 12GB is used every 6 months. This is less expensive than even the 750MB per month plan provided by Telecom which would end up costing $313.08 per year for a business. If less usage is planned on being used then the other 2 Degrees plans can also be quite cheap, the 3GB plan allows for an average of 1.5GB per month but only locks the user in to a 2 month period as opposed to a 6 month one, and costs the same price of $300 per year. The lower usage plan of 1GB per month is even cheaper at $240 per year, this is the cheapest possibility over all plans by all providers, however the chosen plan really depends on the amount of data used and the coverage of the networks.

2.4. WEB SERVICE DESIGN

As the Electronic Job Card application will need to interact with the Bridgestone databases I will be creating and consuming a web service for these data transfers, therefore I must first decide on an approach to building web services. The possibilities for this include SOAP or POX protocols, or the RESTful architecture for web service design. As SOAP and POX are protocols while REST is an architecture they cannot be directly compared, however performance and ease of development between them can be [23]. All of these methods have their own pros and cons and I have researched these ways in order to gain a better understanding of my options.

SOAP has been around since the late 1990s and is still the prevailing standard for web service design. It is known to be safe in terms of error handling and is able to transfer attachments without problems. There are set contracts that SOAP must adhere to such as having a compulsory body section with an optional header, giving it more structure than REST and making it very robust [24]. However SOAP can require development tools to produce which can either act as guidelines or complicate things, and can be conceptually difficult to understand.

POX is a similar protocol to SOAP in that both use only POST when invoking web services [25]. This means the XML must be parsed first in order to figure out the web service name and parameters. Both the advantage and the drawback POX has when compared to SOAP is that it does not use the elements SOAP does, what this means is that while this reduces the size of the sent message, it forgoes quality of service. POX may be used instead of SOAP if a message was required to be small and quality of service did not matter.

REST on the other hand can be much simpler to develop than SOAP and has less reliance on tools, everything is simply a resource accessed through a URI [24]. REST also does not encapsulate the messages in XML like SOAP and POX do so the data usage is even lower than that of POX [25]. The messages sent are very simple (GET, POST, PUT, DELETE) and because of these, it is easy for a program to see what the web service is doing. For example from a security perspective if a REST GET message is sent it can only query data and will always be safe, however if a SOAP POST message is sent it is impossible to tell just from the POST flag if it will be just a query or a more potentially destructive instruction, you must look into the message for this [26]. An additional advantage given to REST over SOAP and POX due to not encapsulating its messages in XML is that even web intermediaries will be able to read the messages, allowing for the efficiency of caching which does not happen with SOAP and POX messages as they are protocol agnostic. Because REST has simpler commands and does not require an XML wrapper around every request and response like SOAP does it is more lightweight and uses less bandwidth. REST has smaller request messages than both SOAP and POX and also smaller response messages than SOAP. REST also has much quicker response times than SOAP [25]. The characteristics of SOAP and REST are compared in table I. Some major companies such as Yahoo and Google opt to use REST [24] however there are still some flaws such as it lacking the helpful tools and W3C standards SOAP has and is more difficult to develop sophisticated requirements for.

Table I
Characteristics of SOAP and REST from [23]

	REST	SOAP
Characteristics	Operations are defined in the messages Unique address for every process instance Each object supports the defined (standard) operations Loose coupling of components	Operations are defined as WSDL ports Unique address for every operation Multiple process instances share the same operation Tight coupling of components
Self-Declared Advantages	Late binding is possible Process instances are created explicitly Client needs no routing information beyond the initial process factory URI Client can have one generic listener interface for notifications	Debugging is possible Complex operations can be hidden behind façade Wrapping existing APIs is straightforward Increased privacy
Possible Disadvantages	Large number of objects Managing the URI namespace can become cumbersome	Client needs to know operations and their semantics beforehand Client needs dedicated ports for different types of notification Process instances are created implicitly

This table shows some key differences between SOAP and REST

3. PROJECT PROGRESS

3.1. WORK FLOW SPECIFICATIONS AND DATA USED

After having gone to meet the end users and seeing how they carry out their job, I was able to put together a well-defined proposed work process. The work flow diagram shown in figure 6 can be thought of as showing the scope of the application giving an understanding of what functionality must be implemented and how these will be used. To go over the steps in figure 6 the start of a job will always be after a vehicle inspection has been done identifying work needed, be it a complete VIR or not. Therefore the first thing that needs to happen is to sync information from the VIR application; this will mainly be customer and vehicle information. This information needs to be readily viewable in the new application.

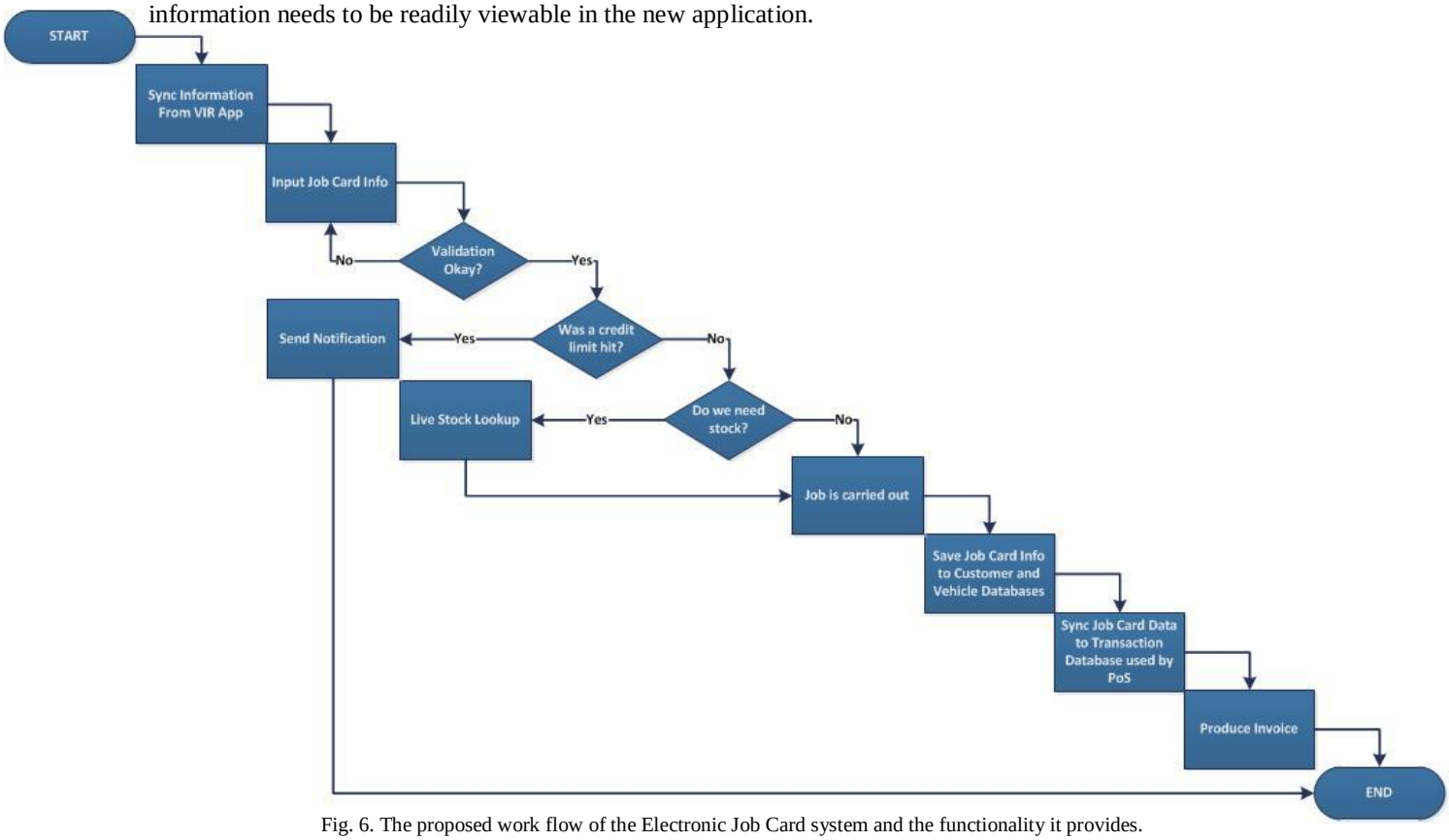


Fig. 6. The proposed work flow of the Electronic Job Card system and the functionality it provides.

After the user can see what kind of job they will be potentially carrying out on a vehicle at this time they will be able to input the job card info. This will be the specifics of the job detailing what was done on what wheels with what products. The job prompt from the VIR should identify a job to be done and on what wheels however the specifics as to what is used to do this job will need to be entered into the Electronic Job Card application. Based on the data received by the VIR application the appropriate vehicle configuration diagram should be displayed for each vehicle and this should be able to be changed or entered if necessary. The user must then be able to add information about the current job, so fields must be properly specified as optional or required. The application should also use dropdowns, radio buttons and checkboxes where appropriate if there are predetermined options for a specific input, this enables the user to both understand what is being done and help reduce the time spent entering information.

All relevant data inputs must have validation to determine if the inputs are allowed for this vehicle under this customer. If these validation rules are not met the job card should clearly tell the user so and ask them to fix any issues before moving on. It should also check the cost of the proposed job and make sure the cost does not push the customer over their credit limit, if they have one. If it will the job should not be completed.

After having the information about the job to be carried out completed we will be able to identify what stock is required for the successful completion of the job, and from this the user will know if they have the stock with them or not. If they do not have the required stock I will provide functionality to perform a stock lookup, checking databases of stocks by location to show where the needed stock may be available to be obtained from, this is in hopes of making the process of locating stock more efficient. This stock lookup will not be compulsory and if the user desires they will be able to find stock on their own by calling around.

At this point ideally the actual physical job will be carried out, as if it were carried out earlier there is potential for a job to be done that does not meet validation or that goes over the customer's credit limit. This step is not done in the application at all and is just shown for where it should be handled in an ideal work flow.

After the job is completed all information used in the job should be saved to relevant databases. This includes customer, vehicle and transactional data. Transactional data should be saved in a way that allows for the POS system to produce an invoice for each job handled in this way. In the event that there is no connectivity to send this data live the application needs to provide functionality to save a job card and send the information at a later time. This must be done in a way that jobs are not left saved and forgotten.

The data used by this application can be separated into two sections, data that it obtains from the VIR application and data that is unique to just the job card as shown in Table II.

Table II
Data available to the Electronic Job Card application

Data From VIR Application	Data Unique to Job Card
Salesperson	Date (from tablet clock)
Customer/Account Name	Invoice/Job Number (using blocks where needed)
Customer Address	Job Completed (Workshop or Fleet Outwork)
Vehicle Registration Number	Payment Method (from a list)
Vehicle/Fleet Number	Equipment Used (Qty New/Rtd, IPC, Brand, Pattern, Unit Price can be auto calculated)
Make/Model	Work Done (Qty, Service, Remarks/Size, Unit Price can be auto calculated)
Odo/Hubo	Credit Limit
Vehicle Configuration (indicated by diagram)	Customer SLAs
Tread Depth (for each tyre)	Stock Location
	Remarks (must be free text field)

This table shows the general idea of different pieces of data used by the Electronic Job Card application

Most of the generic information about the customer and vehicle is given to the job card application from the VIR application, while most transaction information is specific to the Electronic Job Card, this is because a VIR does not result in a transaction. Of the data unique to the job card the Invoice Number has been identified as being particularly important as it is physically written on wheels that have had jobs done on them for identification purposes and should therefore be visible at all times. The payment method must be compulsory as

without it no payment could be received, whereas data like credit limits and SLAs will only be present when possible and therefore may not need to be made visible to the user.

The primary job information will be the equipment used (what kinds of tyres were used on the job, how many and how much they cost) and the work done (what jobs were carried out on the vehicle, how many of each job and other information such as the cost of the job). These are the pieces of data that will be showing up on an invoice and are taken from the central area of figure 1.

3.2. CLASS DIAGRAM

Based on the above information the following is a class diagram that has been made to show how data will be handled as objects in the Job Card application. Do note that the data names and method names in this diagram are not official and will be different from the final product, the diagram is more to give a view of how data is hoped to be sorted in classes.

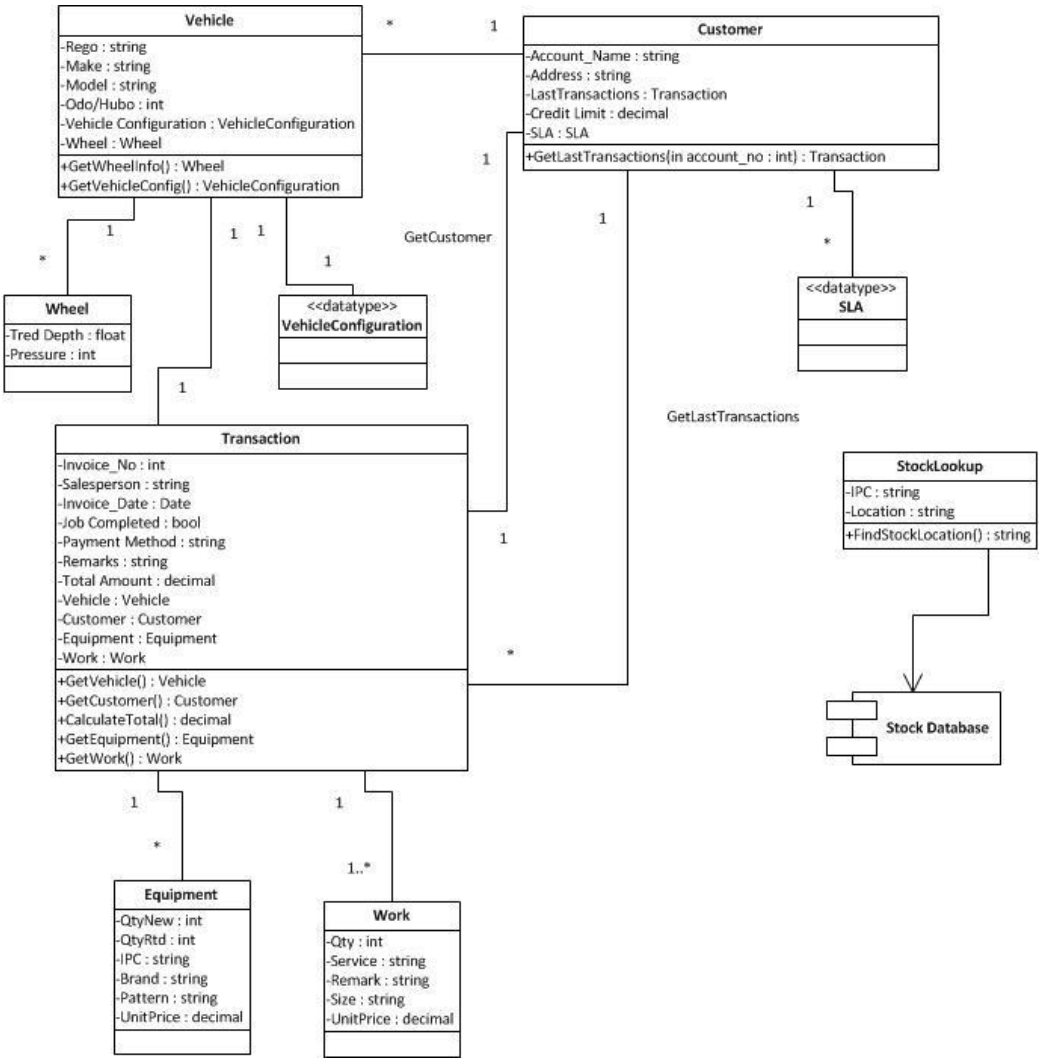


Fig. 7. The proposed classes and methods and how they will interact.

In this class diagram I have separated the data components into three main classes, those being the Vehicle, Customer and Transaction classes. The transaction class will have a corresponding Customer and Vehicle for each transaction. As a Customer can also have Transaction information from past transactions, there are two links between Transaction and Customer, these have different multiplicities also. The names of the methods that cause these links have been written beside the connections to help indicate why there are two.

The Vehicle class will have a number of pieces of data relating specifically to a vehicle, including the vehicle configuration with corresponding diagram which may be handled as its own data type. This will come from the VIR application. Based on the vehicle configuration each Vehicle will have a number of Wheels in different positions, and each of these Wheels will also have tread depth and pressure information if available.

A customer will have standard customer information such as their name and contact address. It will also have the functionality to lookup recent transactions for a specific customer in order to see trends. In order for proper validation to be carried out in a transaction the SLAs will need to be read, these will be different for each Customer and may be handled as its own data type. Similarly each Customer may have credit limit information in order to prompt the application not to go through with a job if this is reached.

The transaction is mostly filled with data unique to the Electronic Job Card application and is the largest class shown on this diagram. There will be basic information about the job such as who carried it out and the invoice number, the payment method and where the job was carried out (JobCompleted, as there are only two options for this it is handled as a Boolean). Remarks are allowed to be written as a String from a free text input without validation. Each transaction will also have Work and Equipment associated with it. As there can be multiple jobs carried out on a single vehicle at once and many pieces of equipment can be used these are both their own classes that can have a multiplicity of many for a single transaction. Based on the work done and the equipment used the Total Price can be calculated as a decimal value.

The Stock Lookup is handled separately from the other information and interacts with external stock databases in order to find stock location based on an IPC.

3.3. USE CASE

The use case shown in figure 8 was created to give a look into the functionality of the application from a user's perspective. Uses cases show what an 'actor' can actually do to an application, the basic functions they have available to them and how these functions may link to database actors.

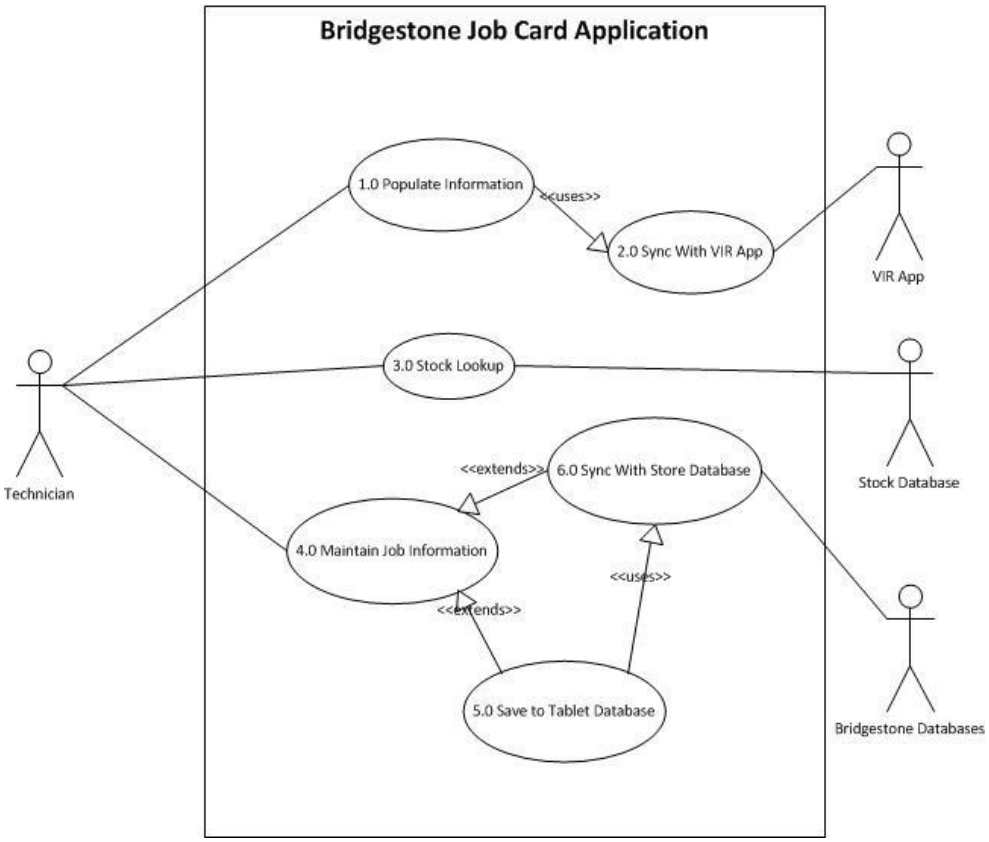


Fig. 8. A Use Case diagram for the proposed Electronic Job Card application.

Going over the use cases shown in this diagram use case 1.0 is populating information, this will be all information that the job card application is able to populate without needing user input. This will use such things as the date from the tablet clock, but will mainly include use case 2.0, syncing with the VIR application. This use case is a compulsory extension of use case 1.0 and must happen in the process. It will extract information from the VIR application which is outside the system boundary for the job card application.

Use case 3.0 is the optional Stock Lookup function that a user may wish to perform to find out where they may be able to find stock. This can be thought of as a lone use case that does not interact with any others, and simply links in with stock databases outside the system boundary.

Use cases 4.0, 5.0 and 6.0 are closely linked in that they are dealing with the job information and sending this to appropriate databases. Use case 4.0 is maintaining the job information. This involves inputting and modifying information about a job. This can either extend use case 5.0 in the case that a connection to external databases cannot be made, or use case 6.0 if a connection can be made. In the case where there is no WiFi or 3G connection the job card information will need to be saved onto the tablet as in use case 5.0. As this is just a temporary storage until it can later be sent off to the appropriate databases this will include use case 6.0 at a later time, when connections are available. Use case 6.0 will be the SQL commands syncing job card information to the Bridgestone databases outside of the system boundary.

3.4. DATA FLOW DIAGRAM

A data flow diagram is used to give an overview of where what kinds of data are flowing through the system. External entities are given in square boxes, processes in circles, and data stores in ovals (often they are over and underlined instead). Data flow diagrams often have multiple levels of abstraction, with the top level being a ‘context diagram’ showing the system as a single process, and the external forces that work on it. As this context diagram is extremely simplistic and can be inferred from the diagram of the next level down, I will not be including it here. Below in figure 9 is the level 0 data flow diagram showing the main processes inside the application and where data will be flowing.

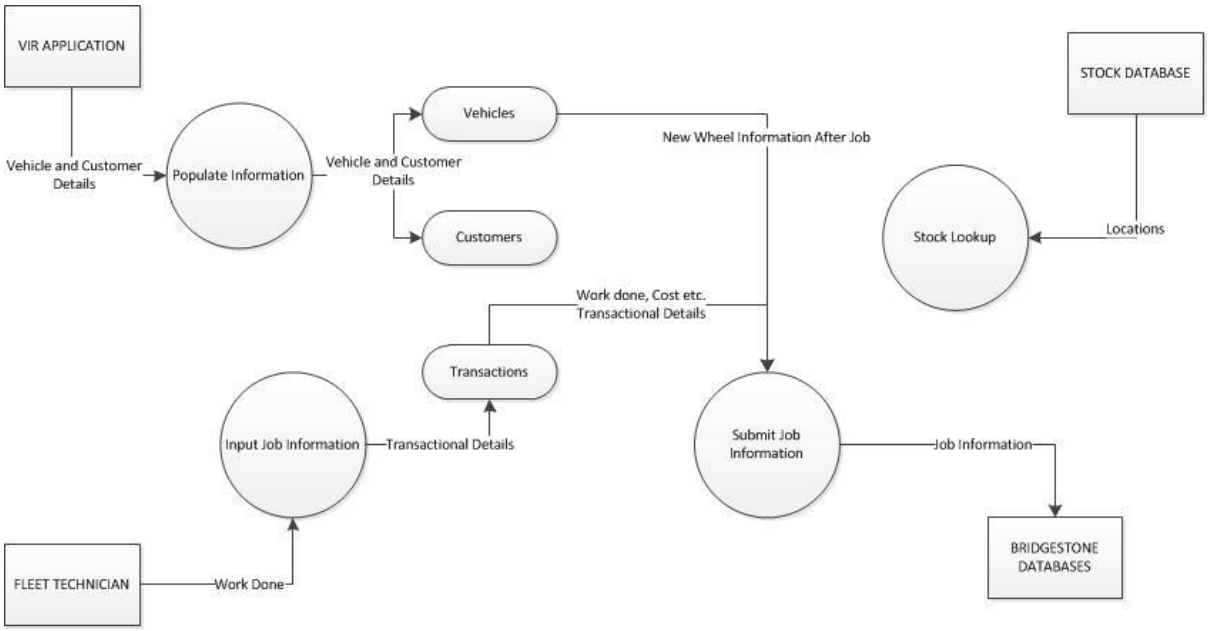


Fig. 9. A level 0 data flow diagram showing how data will move through the application to/from external sources.

From this level 0 data flow diagram you can see that there is one main flow of data in the central application, and then again similar to the last few diagrams, stock lookups are handled to the side separately. The data flowing for the stock lookup is simply the location from the stock databases being fed through to the tablet based on SQL queries. The queries will filter the locations based on a product IPC however no data is actually being given to the stock databases so this is only a one way flow.

Starting from the top left the VIR application will be an external source of data, syncing with my job card application in order to give it customer and vehicle data that the VIR application obtained either from Bridgestone databases or through vehicle inspection. This information is populated into my application through a process, and then stored in their respective Customer or Vehicle classes. The customer information will not be changed at all by my application, however it is still required. Therefore after it is stored it does not need to flow anywhere from that point. The vehicle data however may change depending on what work is done on the vehicle and because of this after a job is completed the updated vehicle information will flow down into the process to submit job information.

The fleet technician will be inputting various pieces of transaction information depending on what work is done, and is therefore an external source of job data. Through the input job information process transactional information is entered and stored into a transaction class. After a job has been completed this transactional information will need to be sent to a database so the data will flow towards the submit job information process.

After the process for submitting job information has complete data from the vehicle and transaction storage, the job information will be pushed towards the external databases at Bridgestone.

3.5. INITIAL APPLICATION

Primarily as a test of our skills as programmers, and to prove our understanding of Bridgestone's databases we were given a small project to create and consume a web service which allowed a user to enter a registration number and then returned various information about the vehicle and owner of that vehicle. We took this application quite seriously as it provided functionality that would be included in our final projects and went so far as developing a scope document, diagrams and conceptual designs of the user interface that was to consume the web service. In addition to this as we were unfamiliar with how to go about creating a web service and how they worked so we needed to carry out sufficient research to boost our understanding of the project. Figure 10 shows the use case diagram we created for the very simple application, only containing two self-explanatory use cases.

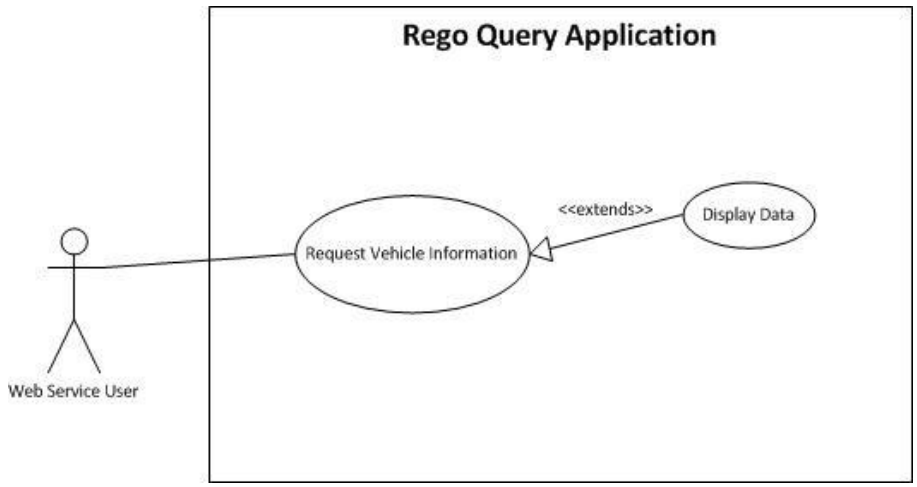


Fig. 10. A simple use case for the initial application

As you can see this is a fairly simple application with only one function a user is able to perform, that of requesting vehicle information with an inputted registration number. Simple as it was, it is still important to understand how web services could be created and consumed, as this will be a major part of the final BTech project.

For interface design we started with concepts on paper, as you can see in figure 11, and then after deciding on a concept to go with we created the interface as a WPF application in Microsoft Expression Blend as you can see in figure 12. For this Blend interface we included multiple pages with navigation, XAML customised hover and click effects on buttons, and a colour scheme which we felt represented Bridgestone.

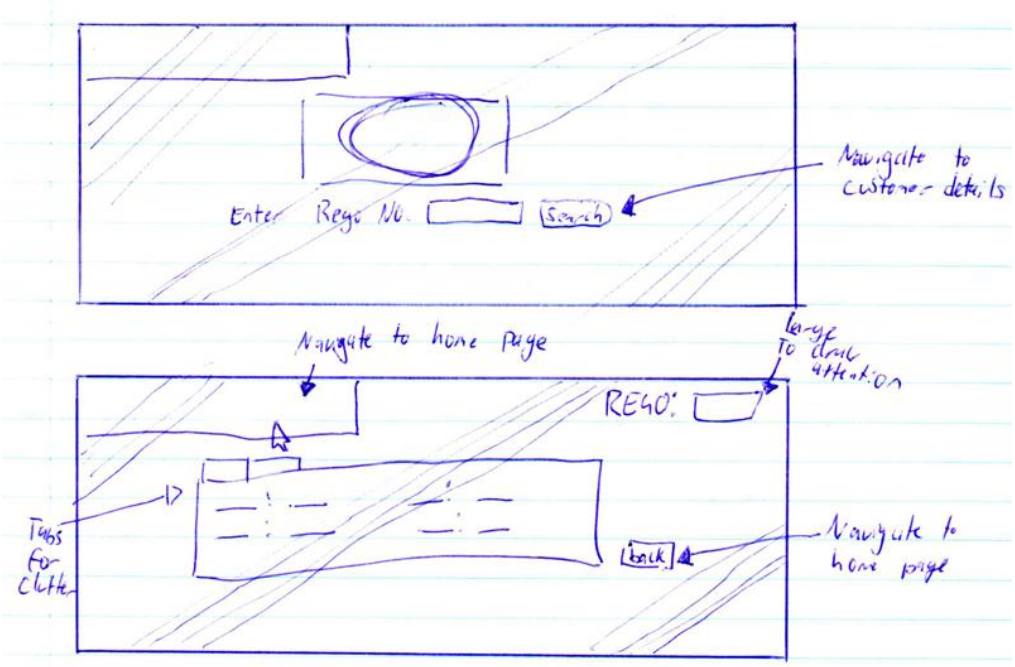


Fig. 11. One of the initial conceptual design sketches of the interface for the initial application

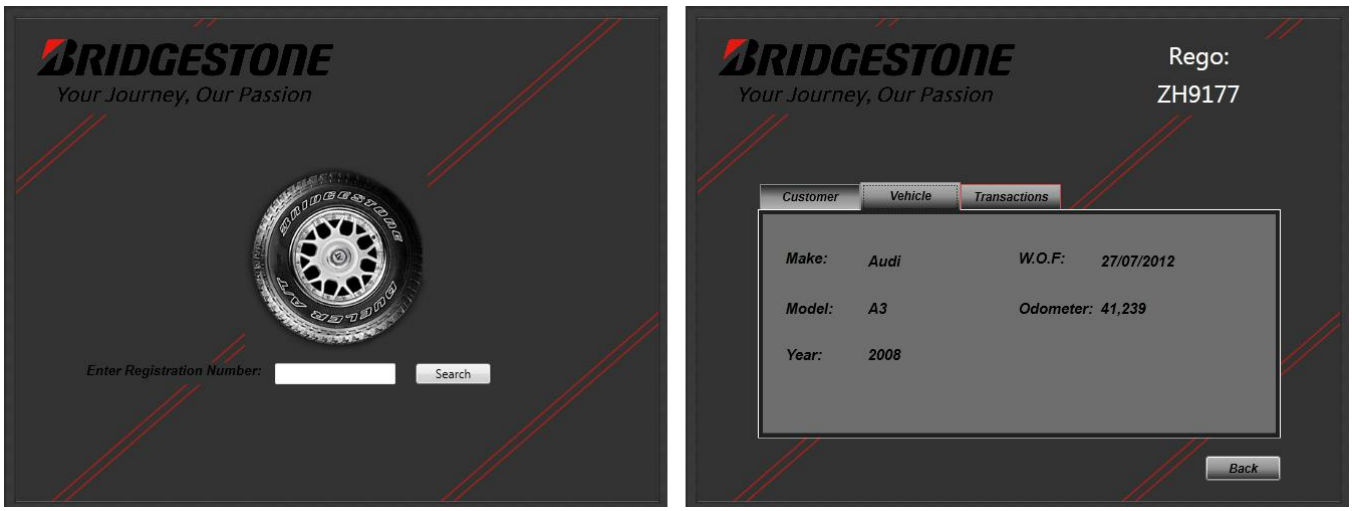


Fig. 12. The completed Blend interface for the initial application

You can fairly clearly see from these that the chosen design translated almost exactly into the final Blend interface. This is mainly due to the fact that Microsoft Expression Blend is a WYSIWYG interface design tool that gives users a fair amount of control over how they wish to customize their interfaces. Blend also allowed you to view your applications in XAML in order to further customize them through programming. We took advantage of this to customize tabs and buttons for specific gradient effects, as well as hover and click animations. Figure 13 shows a small section of this code where dynamic resource styles were created and applied to a button for specific gradient effects. This needed to be done using the XAML section of Blend otherwise these specific brushes could not be properly applied.

```
202     <TabItem Header="Transactions" Background="{DynamicResource Gradient Tab Brush}" Style="{DynamicRe
203         <Grid Background="#FF6E6E6E"/>
204     </TabItem>
205 </TabControl>
206 <Button Content="Back" HorizontalAlignment="Right" Margin="0,0,59,20.04" VerticalAlignment="Bottom" Wi
207     Background="{DynamicResource Gradient Tab Brush Invert}" Style="{DynamicResource ButtonStyle1 Edit
208     <i:Interaction.Triggers>
209         <i:EventTrigger EventName="Click">
210             <pi:NavigateToScreenAction TargetScreen="WpfPrototype1Screens.Screen_1_1"/>
211         </i:EventTrigger>
212     </i:Interaction.Triggers>
213 </Button>
214 id>
```

Fig. 13. Some customized XAML in Microsoft Expression Blend

The front end design process was carried out thoroughly to reach the application shown above, and alongside it the backend was created in Visual Studio as a WCF web service. This kind of process is exactly how we plan to produce the final Windows 8 product for Bridgestone. First we had to create a web service that accessed the Bridgestone data warehouse. This required research on how to achieve this, after which we applied the knowledge we learnt to implement a web service using Bridgestone's database. We then had to connect to this web service and consume it. The web service was created with the default method which was SOAP as we were not concerned with efficiency for this initial application. The code shown in figure 14 is a section that was the main hurdle in connecting to our web service and consuming it, showing the connection string and SQL commands.

```
            string regono = InputTextBox.Text;

            regono = regono.ToUpper();

            ServiceReferenceRego.RegoLookupClient c = new
ServiceReferenceRego.RegoLookupClient();

            Label1.Text = Label1.Text + c.RegoSearch(regono);

            SqlConnection db = new SqlConnection("Data
Source=baksql2 ;Initial Catalog=datawarehouse_UAT;Integrated
Security=SSPI; persist security info=False;");
            try
            {
                db.Open();
            }
            catch (Exception exc)
            {

                Console.WriteLine("Problem opening
connection..");
            }

            SqlCommand command1 = new SqlCommand("select
plate, make, model, year_of_manufacture,
expiry_date_of_last_successful_wof, latest_odometer_reading
from dim_vehicle where plate = '" + regono + "';", db);
            SqlDataReader myReader = command1.ExecuteReader();


            if (myReader.Read())
            {
                OutputLabel.Text = myReader.GetString(0);
                OutputLabel0.Text = myReader.GetString(1);
                OutputLabel1.Text = myReader.GetString(2);
                //OutputLabel2.Text = myReader.Item
(3).ToString;
                //OutputLabel3.Text = (string)
myReader.GetString(4);

            }

            myReader.Close();
            db.Close();
        }
```

Fig. 14. A section of the backend coding on consuming a created web service.

At the top of figure 14 you can see some code for handling different inputs that should all be read the same, as some users would be likely to input a registration number using upper case while others may use lower case, so after being placed into a variable for use in the SQL query all registration numbers were forced to upper case.

The connection string posed a large problem as we had not created a web service like this before and we were unsure of how to structure a connection to a database, let alone to Bridgestone's specific database. However we got it working in the end and were able to execute SQL queries based on an inputted registration number, and then display the results as was required by the application. This whole initial project was a very useful stepping stone in the project for creating the final application as it allowed us to create many sections that our final applications will use, but on a smaller scale. It allowed us to use both Microsoft Expression Blend and Microsoft Visual Studio in parallel to develop the backend and frontend separately, ultimately to be linked together for a final application. It also allowed us to research and develop an understanding of how web services work, how they are created and how they are consumed, and more importantly taught us how to create and use these to interact with databases that Bridgestone owns. This knowledge will be reproduced in our final projects when our mobile applications connect to Bridgestone databases for SQL queries and writes.

4. CONCLUSION

4.1. PROBLEMS AND SOLUTIONS

In the process of narrowing the scope and from the research that was done a number of problems presented themselves, each of which needed a solution to be found. Some of these were simpler than others and required only a question to the appropriate people, whereas others required a decent amount of research.

The first major problem that we faced was which platform to develop the application for, be it an Apple device with iOS, an Android device or a Windows Phone. This was researched in detail as shown in section 2.2. After thorough research and being influenced by the Windows Camp it was decided to go ahead with developing for a Windows 8 platform. This decision was also supported by those at Bridgestone as the whole I.T. infrastructure from top to bottom in Bridgestone New Zealand and Australia is Microsoft based, and the I.T. staff are both familiar with and comfortable with Microsoft products. In addition to this one of the workers had his eyes on a robust tablet designed to use the Windows Phone OS and had previously tested it and felt good about using it. Developing for Windows 8 also means I will be developing for something that is new in the industry, allowing me to stay at the forefront of new technologies, looking towards the future as opposed to devices currently in use that are soon to be made obsolete by new versions.

I have also decided that if I am able I would like to develop the front end for the application for Android and possible iOS as well in order to both become more familiar with those environments and to give Bridgestone other options in case they change their mind at a later date.

The above decision was met with another problem as creating for Windows 8 requires a Windows 8 environment with beta development tools, and while these tools are free to download and use in their beta state there were restrictions placed on us by Bridgestone that meant we were not allowed to legally develop a business application on these without proper MSDN subscriptions. We were able to obtain these subscriptions however and now it is just a matter of setting up the Windows 8 environments.

As seen in section 2.4 a decision also had to be made as to which method of web service was to be used. With REST being simpler to understand, easier to develop and less data intensive, with small message sizes and quick response times, it would be ideal to develop using REST as opposed to SOAP or POX if possible. This is especially true considering the fact that a mobile device will be accessing the web service, so a light weight less data intensive call would be ideal over mobile networks.

As the Electronic Job Card application will be made to ultimately link with the POS system to produce an invoice an invoice number is needed. This number is important to the work of fleet technicians as it is written on each tyre to match it up with a job, therefore blocks of invoice numbers will need to be assigned to the application so that it can be used without overlapping with other job numbers.

It is also required that the invoices produced be printed on paper, and we briefly considered the possibility of a small printer attachment that could be used, however we ended up not pursuing this option and instead will print the invoices with the POS system after the job information has been sent to the database.

There are many different rules for what can and cannot be done to fleet vehicles, so the application will need to handle rigorous validation on input fields. This validation will need to be based on both universal rules and specific SLAs that a customer may have for their fleet.

It was also identified that it was important to the end users that the application be very simple. In order to solve this I plan to use as many visual cues as possible such as representing a vehicle as a configuration diagram like that shown in figure 15, where wheels that jobs will be done on can be naturally selected. I also plan to handle the application as a sort of wizard that will flow seamlessly forward and backwards between steps.

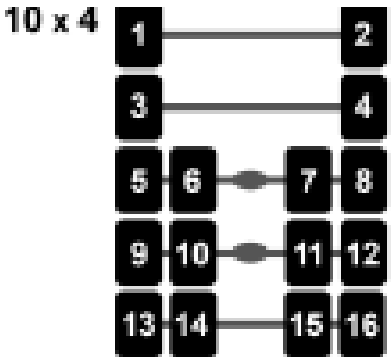


Fig. 15. A simple vehicle configuration diagram that I hope to use as a focus in my interface.

We thought about implementing GPS map systems inside our application to view the location of other fleet trucks however for many of these (for example systems that use Google Maps) there is a monetary cost for every time the map is accessed inside a personally created application. Therefore these maps will have to be handled in separate third party applications such as a Navman system when needed.

Bridgestone has expressed interest in hoping to be able to charge customers for labour for their jobs done based on hours worked, which they are unable to do at present so I hope to add functionality to the application to record the time that a job is initiated and a time it is completed in order to have a specific measurement of how much time (labour) was put into each job.

As locating and picking up stock was identified as a problem area in the 'field trip' I decided to implement a stock lookup function in my application in an attempt to make this process more efficient, hopefully meaning they do not miss as many chances to do a job on a vehicle that will soon be taken out.

As some jobs are carried out in areas without WiFi or 3G connectivity there will need to be functionality to still carry out jobs and store the job data on the tablet so it can be sent off to the database at a later time.

4.2. FUTURE WORK

We will need to get the Windows 8 development environments set up inside the Bridgestone offices so we can create the application using Microsoft Expression Blend 5 and Microsoft Visual Studio 11. This is ready to be done and these environments will hopefully be set up before we return for the second half of the project.

I will need to complete sketches of some interface designs that I can show Bridgestone so I can get feedback on what sort of direction they want me to take the interface in. From there I will refine what I have done and create clean digital mockups of what I hope to produce for the interface. Based on the feedback from these I will be able to develop the final interface in Microsoft Expression Blend 5.

As well as the front end I will also be creating the back end application and web service using Visual Studio 11. I will be using the techniques I learnt in the registration lookup application and web service but will modify it to handle different data and to use REST as opposed to SOAP.

After both the front and back ends have been completed they will need to be tested together and shown to Bridgestone. I will get feedback on what I have done so I can modify parts of it to be more to their liking. I will also have to thoroughly test and re-test every part of the application.

4.3. SUMMARY

Throughout the first semester of this project I have been able to become very familiar with Bridgestone as a company and how they work. I have been familiarised with their systems as well as how they handle work processes. Seeing these in a professional environment has been very beneficial to me as a student and I have come to feel the weight of what this project means. Over these first few months of the project I have carried out extensive research to solve problems that needed to be addressed before the designing of the Electronic Job Card Application, including meeting the end users and seeing how my application would be used by them in their everyday work flows. I have been able to recognise various areas in which my application should be able to promote efficiency in the fleet technicians work and through many discussions with various people inside Bridgestone I have been able to clearly define the scope of the project, which proved to be difficult as it needed to be clearly distinct from but still closely linked to the VIR application that will be developed alongside the Electronic Job Card application. The field trip to meet the end users was the most helpful event in terms of being able to see the cut-off between the two applications and see what actions are required by one or the other, as well as for how they must interact.

From here over the break between semesters I hope to finish the interface designs which I will get feedback on and refine before the second semester starts, allowing me to focus primarily on developing the application for the whole second half of the year undisturbed by documentation or research.

ACKNOWLEDGEMENT

Scott Patterson thanks senior lecturer Dr. S. Manoharan for giving the opportunity to take part in this project and for overseeing the BTech degree, as well as for being the supervisor for my project specifically. I would also like to thank senior business analyst Robert A. Lee for looking over my work at Bridgestone and checking up on what I have done to give me advice on where I should be heading, and business analysts Philip Low and Candy He for supervising my work while at Bridgestone, constantly giving helpful advice and being there to answer questions and discuss the project with me.

REFERENCES

[1] *Apple unveils iPhone | Macworld*. (2007).
Retrieved from http://www.macworld.com/article/1054769/iphone.html
[2] *HTC - Touch Phone, PDA Phone, Smartphone, Mobile Computer*. (2008).
Retrieved from http://web.archive.org/web/20110712230204/http://www.htc.com/www/press.aspx?id=66338
[3] *Google's Android becomes the world's leading smart phone platform.* (2011).
Retrieved from http://www.canalys.com/static/press_release/2011/r2011013.pdf
[4] *Microsoft announces ten Windows Phone 7 handsets for 30 countries*. (2010).
Retrieved from http://www.engadget.com/2010/10/11/microsoft-announces-ten-windows-phone-7-handsets-for-30-countrie/
[5] *Company overview - Bridgestone*. (2008).
Retrieved from http://www.bridgestone.co.nz/corporate/page/company_overview
[6] *iOS Human Interface Guidelines: Introduction*. (2012).
Retrieved from
http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/Introduction/Introduction.html
[7] *Developing apps for iPad – Apple Developer*. (2012).
Retrieved from https://developer.apple.com/ipad/sdk/
[8] *iOS Developer Enterprise Program - Apple Developer*. (2012).
Retrieved from https://developer.apple.com/programs/ios/enterprise/
[9] *User Interface Guidelines | Android Developers*. (2012).
Retrieved from http://developer.android.com/guide/practices/ui_guidelines/index.html
[10] *iPhone 4S Battery Life Testes And Compared With Android Smartphones*. (2011).
Retrieved from http://www.redmondpie.com/iphone-4s-battery-life-tested-and-compared-with-android-smartphones-the-result-may-just-surprise-you/
[11] *Android NDK | Android Developers.* (2012).
Retrieved from http://developer.android.com/sdk/ndk/index.html
[12] *Publishing Overview | Android Developers*. (2012).
Retrieved from http://developer.android.com/guide/publishing/publishing_overview.html
[13] *UX guidelines for Metro style app development | BlendInsider*. (2011).
Retrieved from http://blendinsider.com/technical/ux-guidelines-for-metro-style-app-development-2011-10-21/
[14] *Windows Phone NZ : Review: Nokia Lumina 800.* (2012).
Retrieved from http://windowsphonenz.com/wp7/review-nokia-lumia-800.html
[15] *Getting started with Metro style apps*. (2012).
Retrieved from http://msdn.microsoft.com/library/windows/apps/br211386
[16] *Windows Store markets*. (2012).
Retrieved from http://msdn.microsoft.com/library/windows/apps/hh694064
[17] *All Mobile broadband plans | Vodafone NZ.* (2012).
Retrieved from http://www.vodafone.co.nz/mobile-broadband/all-plans/
[18] *Mobile plans for your Business | Vodafone NZ*. (2012).
Retrieved from http://www.vodafone.co.nz/business/mobile/plans/
[19] *Pay Monthly - Plans and Pricing - Mobile Broadband - Internet - Telecom NZ Ltd.* (2012).
Retrieved from http://www.telecom.co.nz/internet/mobilebroadband/plansandpricing/paymonthly
[20] *Mobile Broadband - Pricing - Telecom Business Hub*. (2012).
Retrieved from https://www.telecombusinesshub.co.nz/Internet/MobileBroadband/Pages/Pricing.aspx
[21] *2degrees - Mobile Broadband.* (2012).
Retrieved from http://www.2degreesmobile.co.nz/paymonthly/plans/mobile-broadband
[22] *2degrees - Mobile Broadband Plans.* (2012).
Retrieved from http://www.2degreesmobile.co.nz/business/plans/mobile-broadband

[23] Muehlen, Michael, Nickerson, Jeffery and Swenson, Keith, “Developing Web Services Choreography Standards - The Case of REST vs. SOAP” (2004).
Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.80.6494
[24] Potti, Pavan Kumar, "On the Design of Web Services: SOAP vs. REST" (2011). *UNF Theses and Dissertations.* Paper 138.
Retrieved from http://digitalcommons.unf.edu/etd/138
[25] Kennedy, Sean and Molloy, Owen, "A Framework For Transitioning Enterprise Web Services From XML-RPC to REST" (2009). *CONF-IRM 2009 Proceedings.* Paper 52.
Retrieved from http://aisel.aisnet.org/confirm2009/52
[26] *Web Services, Part 1: SOAP vs. REST | Ajaxonomy.* (2008).
Retrieved from http://www.ajaxonomy.com/2008/xml/web-services-part-1-soap-vs-rest