The Fleet App

Final Year Report

BRIDGESTONE NEW ZEALAND LTD

Authored by: Vishal Naidu

Department of Computer Science
University Of Auckland
Auckland, New Zealand

ABSTRACT

With the production of mobile devices and mobile applications, more and more workers are performing tasks to progress the current workflow in businesses more proficiently. People have become accustomed to accessing information when they want it, wherever they are, and mobile device is one way to stay connected. A mobile application that allows fleet technicians to complete the form filling aspect of their job more efficiently is what Bridgestone requires.

This report illustrates the work Vishal Naidu completed for the final year of his BTech project at Bridgestone New Zealand Ltd in 2012. The main purpose of the project is to create a mobile application that mimics the current Vehicle Inspection Report work flow process and helps Bridgestone's fleet technicians in completing Vehicle Inspection Reports efficiently. By doing this I hope to improve the effectiveness of this process by reducing data redundancy.

Table of Contents

INTRODUCTION

Overview

The swift progression of handheld devices has led to the resurgence for mobile services. Smartphones makes lives a whole lot easier with the addition of applications. In turn this has led to almost half a billion sales worldwide at the end of 2011 and 145 million sales alone in Q1 of 2012 [12]. The handheld devices biggest strengths lie in its portability and interactivity with the user. Bridgestone sees potential in these devices and requires a mobile application that efficiently carries out the current work flow process of a Vehicle Inspection Report.

This purpose of this report to detail the work completed up till the end of semester 1 2012 for the final year BTech project at Bridgestone. This report will contain a research section which will detail all the research being done in order to move forward this project. It also contains a planning section which details all the modelling, documentation and preparation required before developing the application and finally addresses solutions found to all problems including those researched as well as a plan for the future of this project.

The BTech degree is a 4 year honours course which requires students to complete a project in the final year for a company. This project will be carried out by me, Vishal Naidu. This project requires me to work at the Bridgestone headquarters for 10 hours a week.

The Company

In 1931, Shojiro Ishibashi created Bridgestone Tire Company Ltd in Kyushu Japan. Today the Bridgestone Corporation is the world's leading tyre manufacturer, supplying quality tyres for cars, trucks, aircrafts and others. Bridgestone NZ Ltd was formed in September 1998, followed a merger between two world premier tyre companies: Bridgestone and Firestone [3]. In New Zealand, Bridgestone, Firestone and Supercat branded tyres are available throughout the country at a number of retailers including; Firestone, Tony's Tyre Service and other Bridgestone stockists.

Figure 1: Bridgestone Logo

The Problem

Currently Bridgestone fleet technicians fill out Vehicle Inspection Reports for every Vehicle in the fleet which takes a lot of time. Redundancy is also an issue because information has to be noted multiple times before being keyed into the system. Also, there is no structure in the way they fill this form, and key information to the Company are being missed out.

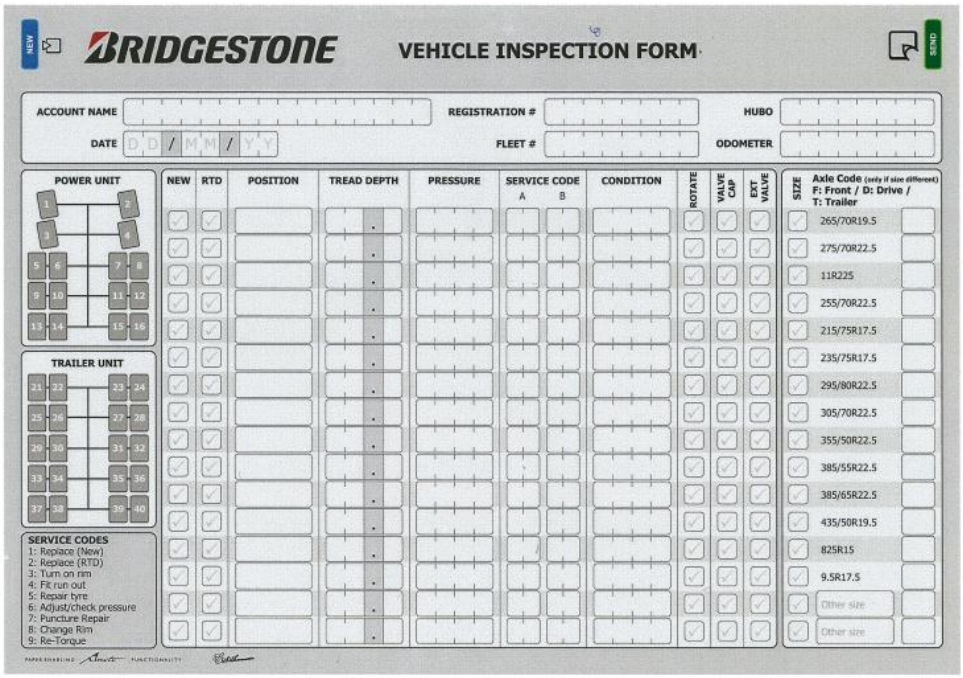


Figure 2: Current Vehicle Inspection Report

My Objective

A mobile application (likely for Android tablets and/or phones) needs to be developed for Tread Depth and Tyre pressure recording.

Figure 3: Official Project Description

The idea is to build a digital alternative of a Vehicle Inspection Report. A mobile application on a tablet PC would help quicken up the current workflow process of the filling out of VIR reports. It is designed also to reduce the redundancy of having the employees manually filling out each form and entering into the database.

Tread Depths and Tyre Pressures are important to Bridgestone so that they can analyse the state of the vehicle and determine what work needs to be done. The application needs have the important functionality of entering Vehicle Inspection information like Tread Depths and Tyre Pressures as well as a Damage report if needed. This data would then be stored into the Bridgestone databases via the mobile application.

In this mobile application, it is very important that the application validates everything, every time the fleet technician adds/modifies/ deletes. It has to validate based on what information is suited to that field and the tables corresponding with that field in the data warehouse itself. Also, the recorded information that the fleet technician entered must be validated against a set of simple rules to allow the fleet technician to make a decision on what job requires to be done. These rules are going to be applied as soon as the fleet technician saves the Vehicle Inspection Report; the mobile application is set to validate the Tread Depth and Tyre Pressure against these rules.

Visually the mobile application content must be clear and easy to read from the user's perspective. The Content, Images and Diagrams should be sized accordingly so that users can recognize it from a distance. Navigation in the applications should be smooth and simple avoiding scrolling.

RESEARCH

The Fleet App

A major defining point in the process of developing the application happened midway through July when a decision was made to combine the Vehicle Inspection Report Application and the Job Card Application. We decided to call the resulting application, the *Fleet App*. The decision to combine the apps was due to the fact that the user does not need to switch between apps and also makes it easier to pass data between stages of the work flow.

The old scenario had my mobile VIR application synchronize with the Job Card application at the end of my workflow. This required a lot of research on the implementation and would take a lot of time. Combining the applications into the *Fleet App* is more advantageous as the workflow would be far smoother than the previous design, and therefore the passing information across to the Job Card part of the application would be far simpler. This decision involved my Course co-ordinator Mano, my fellow developer Scott and Bridgestone, all agreeing that this path was the best option to take in order for a crisp solution.

Dealing with Bridgestone Systems

The *Fleet App* is designed to help the fleet technicians at Bridgestone in completing Vehicle Inspection Reports quickly and efficiently. The application helps with redundancy issues, as you only have to enter the information once, and that information is stored in the database. That way if the fleet technician wants to access a particular piece of information, the application sends a request to the database and all the information is populated on the application.

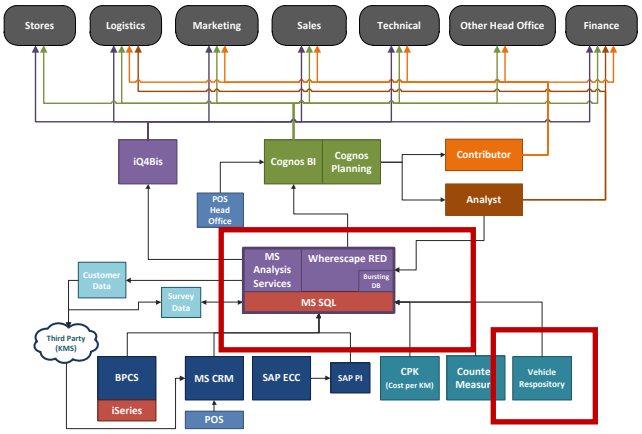


Figure 4: Bridgestone's Systems

Data Warehouse

The systems that the *Fleet App* primarily deal with, is the data warehouse itself. This is where majority of the data is held extending from several other systems like BPCS and CRM. The Wherescape RED tool helps build, manage and update the data warehouse. Other systems that are loosely going to be related to this application are the BPCS and the CRM systems. BPCS mainly deals with the transactional information of Bridgestone while CRM deals with the customer information. Both these systems information are stored in the data warehouse.

The Vehicle repository

Vehicle information comes from a vehicle repository database and if in fact the vehicle is registered, information such as vehicle configuration data and characteristics are returned. We access this information by using a web service which uses a registration number as a key, to look up and return information about the vehicle. Business rules are needed to be defined for refreshing the vehicle repository for those vehicles we haven't serviced to ensure WOF's (Warrant of Fitness) and COF's (Certificates of Fitness) are up to date.

Choosing the Operating System

The most important feature for the tablet 'experience' is the Operating System. The Operating System is the heart of the tablet PC as it has all the features the device can support including the interface, application support and external device compatibilities. Looking at the smartphone market today, there are three different competitors that come to mind when it comes to development. The three OS's are all unique in their own way,

having different advantages and disadvantages. Therefore I have researched the alternatives in order to gain a better understanding of each of the OS's before deciding the OS I choose to develop on for the Bridgestone's Fleet Application.

Apple

The advantage that the Apple tablet, otherwise known as the iPad, has over the operating systems is that it's one of the easiest tablets to pick and use. Its interface isn't cloggy and since it's been in the market for the longest, it has the largest number of applications available to the market.

The iPhone OS uses Apple's Objective-C language, which can be programmed by developers who are familiar with C and C++. I have limited knowledge about the programming aspect of Objective C. Although it's not a big drawback as I would have to get familiar with its conventions and syntax, but I would like to finish the mobile application of at the end of this year and learning the language could be a time constraint in achieving the project goal.

Apple is pretty restrictive with developing especially with their developer guidelines [15]. Developers have a standard set of tools to develop applications and cannot use anything outside of those which restrains the creativity of the programmer.

Apple devices such as the IPad's are the only devices that can run Apple's iOS which allows the users to have a little choice to the specifications of the device that they would be trying to make a purchase [7]. On the other hand the hardware that Apple uses is very secure and has great battery life.

Android

The Android platform has been developed by the Open Handset Alliance and Google to be an open source OS for mobile and tablet devices which allows users more prominence and improvement with what they are doing when they create an application.

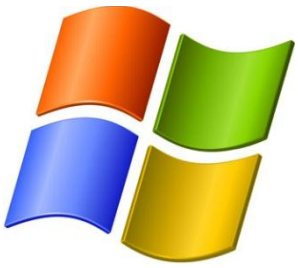
The Android OS mainly uses Java, which is one of the most common programming languages used by developers. The language is familiar to me as it is one of the main languages being used for assignments at my University, and I can easily develop on this tablet. It has an open development platform allowing programmers to use third party tools for the application development. Adding more functionality to applications in this manner is one of the reasons why this platform is very successful. The flexibility of the Android OS can be carried out to the amount of operating systems it supports[7]. Android mobiles in terms of hardware are very versatile, there are many different mobile devices running the Android OS. This provides the user a choice on the internal hardware if he wishes more or less compared to the iOS who only have one choice of device. One very attractive feature of the Android platform is that it lets developers skip the Android Market completely and distribute apps directly to users [4]. Android has major flaws when it comes to security. Its current permissions scheme is very comprehensive allowing for applications to gain more access than they actually require. This is one of the many security flaws, which made Bridgestone decline developing on this OS as it is not ideal for a large business.

Windows

When we are talking about Microsoft's Windows 8 OS, one of the main benefits Microsoft is putting out there is that the OS will run on your desktop, laptop and tablet. This obvious advantage would mean that all your applications can work across all your hardware and you would not have to re-learn everything when you buy a new piece of equipment. This also fits into Bridgestone's current and future plans as the company's primary and only OS is Windows. Windows 8 is not an open source operating system therefore takes a similar attitude to what Apple is implementing, by creating a robust and consistent OS to promote stability as opposed to the Android's approach of being flexible.

Microsoft has been emphasizing the new 'Metro-Style' approach in order for the design of their applications. For e.g. a Metro-style app is touch-enabled, and it displays in a single, plain, borderless window lacking of any decoration such as caption bars or icons [2]. At the same time, a Metro-style app adapts to the effective size and shape of the screen and offers a fluid rendering experience because the content adapts smoothly and intelligently to whatever layout the physical screen may have [2]. The design principles have been explained by Windows so there is a restriction, but not enforced like how Apple does.

Currently the development toolkits are accessible for free until the official release of the OS because the development tools for the Windows 8 are still in beta. I attended a Windows 8 camp at the start of the year to get some experience in using the beta tools like Microsoft Expression Blend 5 and Microsoft Visual Studio 2012. Microsoft has given developers more flexibility in how they wish to program their applications. The Metro-style apps are developed with the new Windows Runtime platform using various programming language's which includes the decision of designing it on: C++, Visual Basic, C#, or HTML/JavaScript [2].



The reason why I chose the Windows 8 OS to be the platform of my mobile applcation was mainly due to the fact that Bridgestone's whole software architecture is Microsoft based. The management of devices with the same OS would be far easier for Bridgestone as they already have expertise in Windows. If I were to develop on a different OS other than Windows, Bridgestone would need to understand how to develop and manage that device or hire expertise in that field. Also C# is a very familiar language to me, so I would not have a time constraint in understanding the language.

Synchronization

Currently the mobile application requires internet connectivity in order to use web services to extract data and populate the fields of the mobile application. There is a possibility that fleet technicians travel to areas which don't have 3G connectivity therefore having no access to the Web Service. I present an offline scenario where the application can have still work through its regular workflow without Internet access.

The offline scenario requires the fleet technicians at the start of every work day to download an image of the data warehouse directly into the Tablet PC. This can be done via Wi-Fi at the Bridgestone warehouse itself. Fleet technicians then can head out to the field with their tablet offline out onto the field. Once at the field, the fleet technician enters the registration number of the Vehicle is inspecting upon. Most of the customer fields will be filled up with information linked to that registration number through querying the image.

The fleet technician has to complete the VIR form on the tablet that would contain a vehicle configuration diagram. A damage report option would be available as well to the user if that possibility would occur. Once the VIR form is completed, the *Fleet App* should be able to validate the Tyre Pressures and Tread Depths and give suggestions to the users about the work that should be done. Once the fleet technicians has returned back to the warehouse, he would be able to synchronize the all the information back onto the data warehouse.

Understanding Synchronization

There are three mechanisms in order to synchronize remote databases locally: 1) Remote Data Access Synchronization 2) Merge Replication 3) Data Synchronization Services [14].

Remote Data Access Synchronization – is the most simplest and fastest synchronization method out of the three but lacks in not have a great functionality that includes an under par interface.

Merge Replication – is set up by a database administrator and allows for the establishment of database update rules which includes rules for conflict resolution when synchronizing new data into the database [6].

Data Synchronization – is more developer/programming orientated compared to Merge replication when synchronizing the database data with data stored in the mobile device locally [1].

There are mechanisms have been provided to move data and schema between SQL Server CE and SQL Server. All are used in combination with a Web server based on Microsoft's Web server technology, Internet Information Services (IIS).

Synchronization of remote data with local data

While IIS, Internet Information Services, is used for data communication, SQL Server CE components are used at both ends of the transmission for database connectivity. SQL Server CE has two components for this: the SQL Server CE Client Agent runs on the Tablet PC, and the SQL Server CE Server Agent runs on at Bridgestone [14].

The server-side components can be deployed with some amount of flexibility. In particular, IIS and SQL Server can be deployed either on the same server system or on separate machines. The SQL Server CE Server Agent component, however, must be deployed on the same system with IIS.

The SQL Server CE Client Agent is a device-side component that tracks changes made to data on the tablet PC and, when instructed to do so, sends them to the SQL Server CE Server

Agent. The SQL Server CE Server Agent is a server-side component that receives changes from the Client Agent and connects to the SQL Server database on the Tablet PC [14].

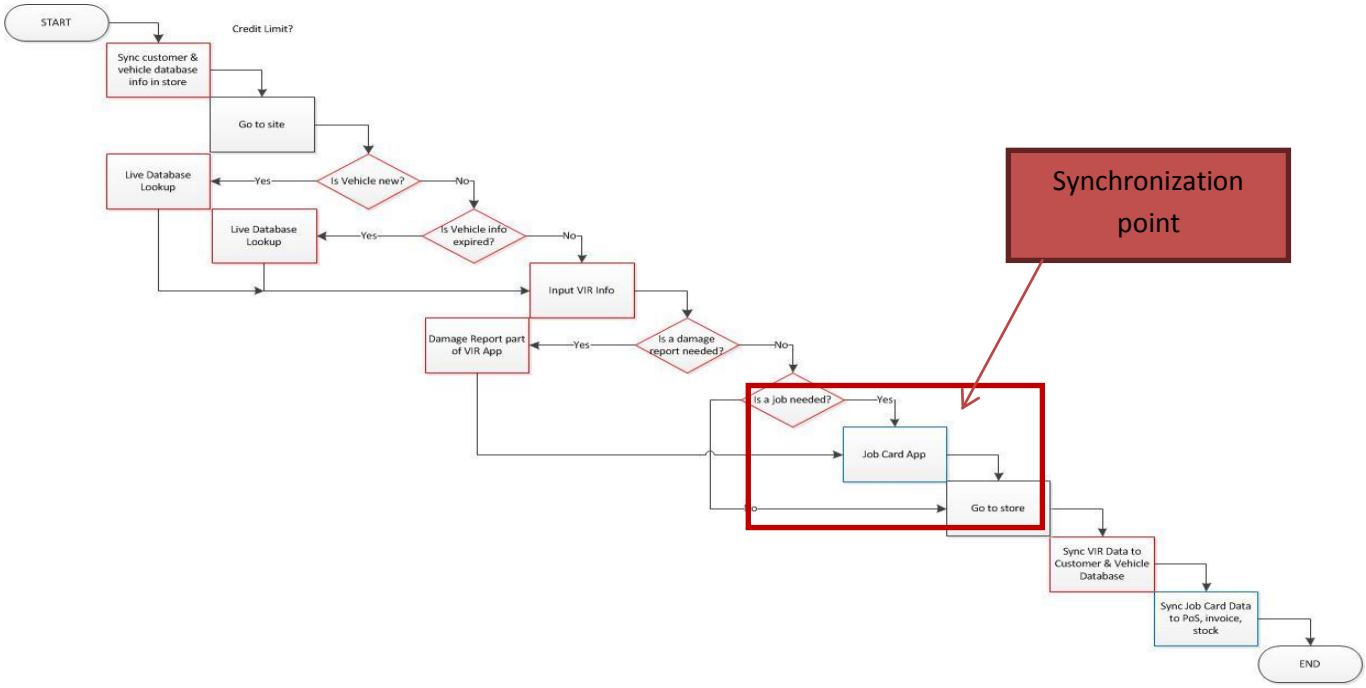


Figure 5: Ideal Synchronization Point in Workflow

Mobile Application Development Approach

There are multiple ways of developing mobile applications, such as waterfall and agile practice. They both encompass good points and they all have their own situations where they should be used. For this project I have tried to research and choose a methodology that fits with a single person project development and fits with the development of the *Fleet App*.

Waterfall Methodology

The waterfall method states that each stage should be completed before moving to the next one. The problem occurs when developing the mobile application; there is a lot of back and forth development between the stages [11]. Generally you would have to perfect each stage before moving onto the next stage, which is not ideal for developing the *Fleet App*.

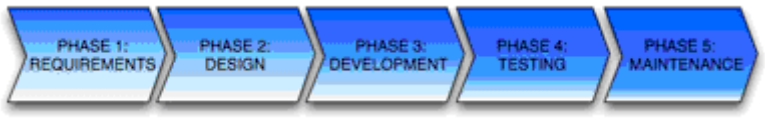


Figure 6: Waterfall Methodology

Agile Methodology

Agile development methodologies are a natural fit for mobile app development. The commitment to simplicity, low overhead, business adaptability, rapid delivery, and customer feedback complement the needs of mobile development [5]. The simple ability to revisit the "phases" of development dramatically improves project efficiency. The methods are lightweight as opposed to the continuous waterfall planning methods because when developing for a mobile application, regular changes are bound to happen. Also the methods commitment to delivering working deliverables at different intervals through the project timeline ensures greater reliability and opportunity to incorporate the user in the process [11].

Agile Method	**Waterfall Method (MSF Process Model)**
Iterative Process	**Procedural Process**
Use for Internal Development Conceptual Specification Document (CSD) - 2 - 3 Pages, whiteboard picture(s) to start Storyboards for mobile device(s) and Web by Dev. Frequent Review by All Frequent Updates by All Heavy Collaboration Plan as we go Welcomes change during process	Use for External Development Conceptual Specification Document (CSD) Functional Specification Document (FSD) UI Mockups Most Planning up front Does not welcome change

Figure 7: Comparing Methodologies

Web Services

A web service is a method of communication between two electronic devices of the Internet. It is clear to me that I would require to create and consume a web service in order to transfer data through and from the data warehouse. Before designing the web services for the mobile application itself, Bridgestone decided that I design a web service that interacts with the vehicle repository system through a simple interface.

My Initial Web Service Assignment

The front end part of this application was not really important, but was needed in order to display the returning values obtained from our web service. It was a simple plain interface which had just the one screen. It contained a Registration number text field and button. When the button was clicked, information such as Model Name, Make and latest odometer reading all appeared. When testing, if the correct information corresponding to that registration number appeared on the screen, the web service we have created was believed to be correct.

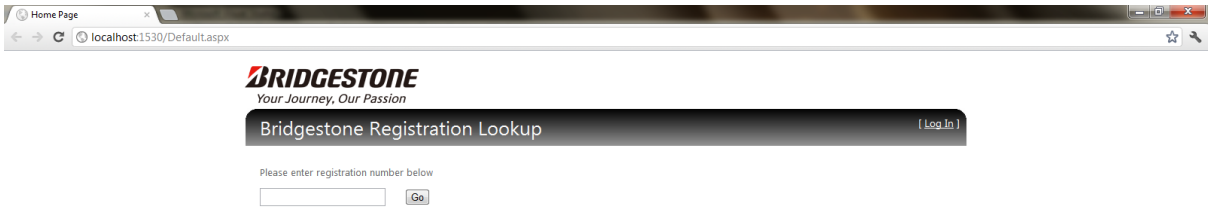


Figure 8: Test Interface

I also designed a simple Microsoft Expression Blend Interface that works the same way as the Web Interface created above as Figure 7. I felt that designing it on Blend would show Bridgestone some of the initial design ideas that I had in store when designing the *Fleet App*.

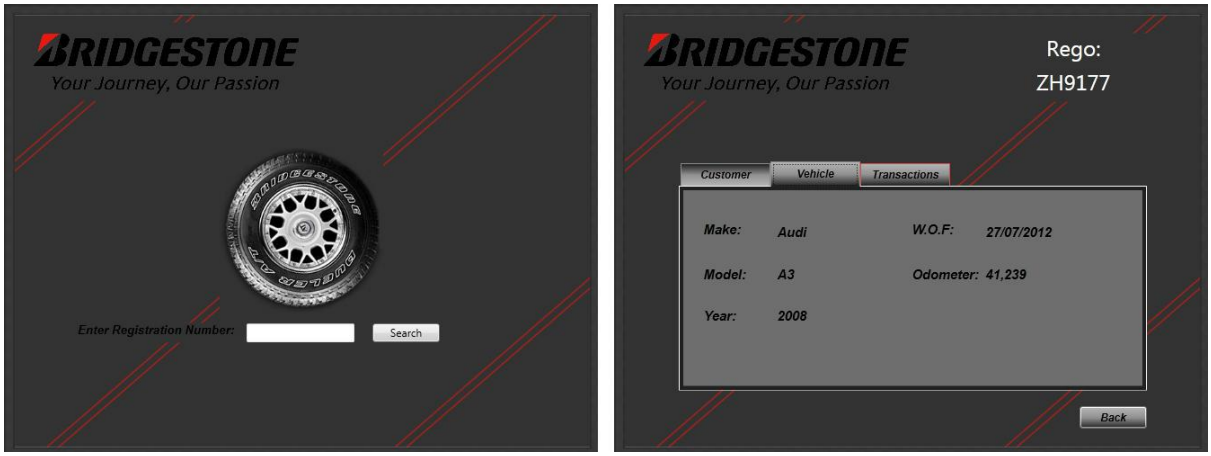


Figure 9: Blend Test Interface

The backend part of the application was the most important part of the assignment. Basically I had to create a web service that obtained certain information from the vehicle repository system. This was done through simple queries at first, just to test to see if the right information was being obtained from the corresponding registration number. I used Wherescape RED to test our queries with the data warehouse. I also utilized SOAP UI to test which methods are needed to use in the Web Service in order to return the appropriate data. WSDL files are central to testing SOAP Web Services, exposes the number services defined in the operation contract and are required by SOAP UI in order to generate tests.

After that was done we had started designing the web service. SOAP was being used by default as our communication protocol as we were not concerned with effectiveness of the process at the moment. The experience of designing this task, helped me understand how Web Services worked, how they are created and consumed and the networking of the Application and Bridgestone's databases. This interaction helped me understand Bridgestone's Systems a whole lot more in understand where data is located and the highlighting the potential data locations within the database.

```
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 whe
SqlDataReader myReader = command1.ExecuteReader();
```

Figure 10: Web Service Code Segment

SOAP VS REST

There are two approaches for communicating to the web with Web Services, SOAP and REST. Both methods work, both having different advantages and disadvantage but it eventually a decision must be made based on the research I have conducted on which method may be best for the *Fleet App*.

SOAP

The Simple Object Access Protocol (SOAP) provides an XML-based messaging structure to exchange data between peers over computer networks [13]. However the general structure that SOAP defines is not sufficient enough to build Web Services. A corresponding requirement, the Western Services Description Language (WSDL), is needed to describe network services as a set of endpoints operating on messages containing either document orientated or procedure orientated information [7].

REST

REST-compliant Web Services are services where resources are manipulated using a uniform set of “stateless” operations (i.e. GET, PUT, POST) [13]. REST is not a protocol, but an architectural style that assumes that the use of existing principles and protocols on the Web are enough to create robust Web Services. This means that developers who understand HTTP and XML can start building Web services right away, without needing toolkits beyond what they normally use for Internet application development [7].

SOAP seems to demonstrate its potential for sophisticated Business to Business interactions compared to large scale e-commerce applications [15]. Its broad scope inherently increases its complexity as SOAP must account or varied environments with different requirements. SOAP is more suitable these interactions because they typically demand:

- Support for transactions, reliably, and asynchronous communication;
- Longevity by standards backup, vendor backup, and maximum platform-independence
- Maintainability due to encapsulation
- Rapid development by tools facilitating the development process

In contrast, REST is designed for Internet-Scale hypermedia systems rather than the sophisticated Business to Business interactions that involve with cooperating with untrusted parties [15]. REST Web services can draw from a large share of Web developers because it is very easy to understand and can be used with any development tool it is similar to which makes it ideal for e-commerce applications.

SOAP VS REST for mobile applications

Now the question that arises is to which type of web service I need to implement and consume for the *Fleet App*. I will be assessing certain specifications that are crucial for mobile applications.

REST is meant for lightweight data transfer over a URI. URI is the Web address of the resource, and makes the resource available online [7]. A resource can be accessed/referenced by its URI. Implementation is one of the plusses of using SOAP. SOAP surpasses REST in terms of implementation and maintaining the application as there are established development kits when using SOAP.

SOAP requests use the POST operation and require a sophisticated XML request to be created which makes response-caching difficult [10]. On the other hand, REST APIs can be consumed by using simple GET messages that can cache responses easily.

REST Web Service calls also go over HTTP or HTTPS, but with REST the firewall can distinguish the intent of each message by analysing the HTTP command used in the request. For example, a GET request can always be considered secure because it only queries data not modifies data [10]. SOAP states that sending remote procedure calls (RPC's) through standard HTTP ports is a good way to ensure Web services support across organizational boundaries. A SOAP request will use the POST operation to communicate with a given service. There is no way to determine what a POST request wants, whether it is a simple query for data or deleting tables from the database [10].

SOAP VS REST Response times

Performance evaluations of response times theoretically should favour the RESTful Web Services because of its high flexibility and less overhead [4]. I decided to test the performance for both REST and SOAP.

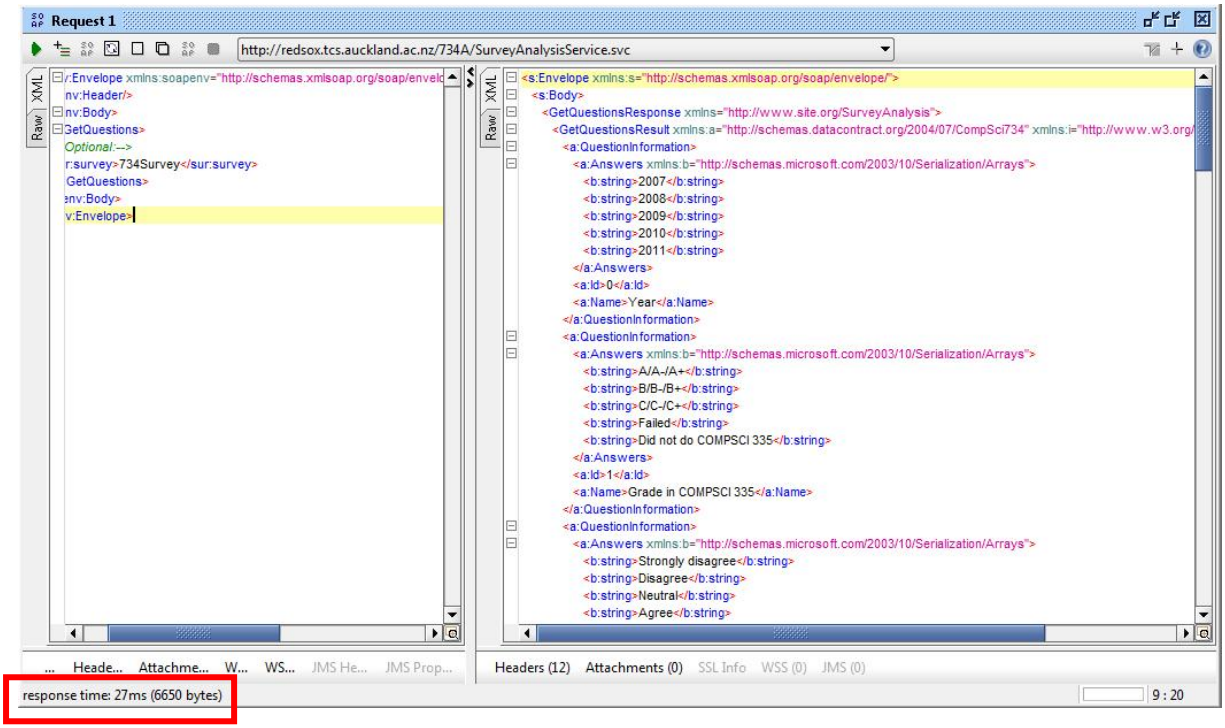


Figure 11: SOAP UI

SOAP Response times	REST Response times
• 37ms	• 24ms
• 20ms	• 26ms
• 35ms	• 22ms
• 25ms	• 18ms
• 27ms	• 20ms
Average: 28.8ms	Average: 22ms

The Load of the message being passed was approximately around 6500 bytes. Clearly, I can determine that the Restful approach is much faster compared to its counterpart because of the less overhead it takes on every message.

The decision of choosing SOAP over REST for the *Fleet App* was solely it was easier to visualise due to the structure, and because Bridgestone's current services were SOAP so it is more compatible with them and does not require them to learn anything extra if they are to deal with our app in the future

PLANNING

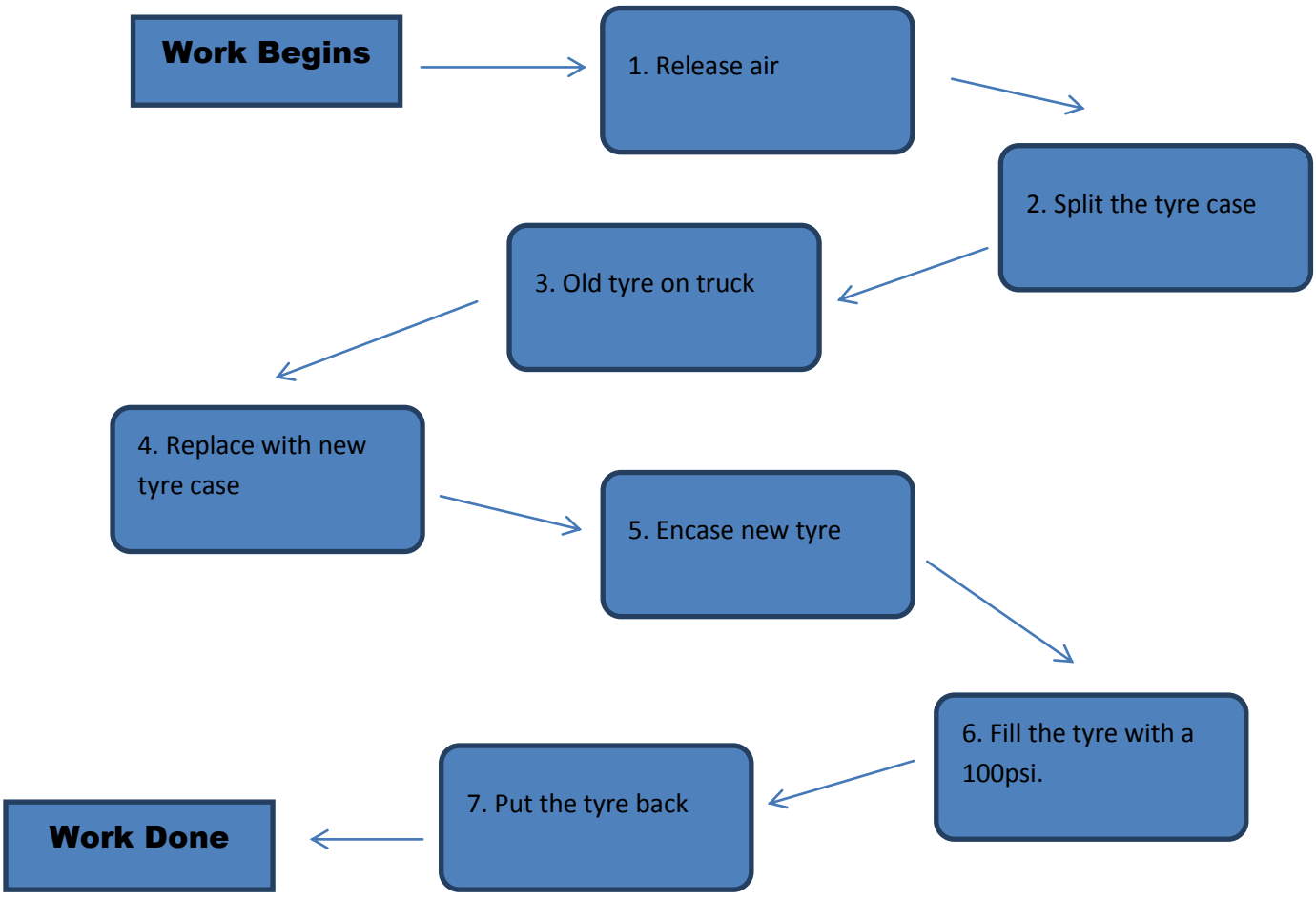
There are many planning stages that I have to go through before starting on developing the *Fleet App*. I firstly have to understand the physical workflow process currently being used at Bridgestone, planning the functionality of the *Fleet App* and the interaction it has to Bridgestone's systems and finally the actual workflow of the *Fleet App* mobile application.

The Bridgestone Field Trip

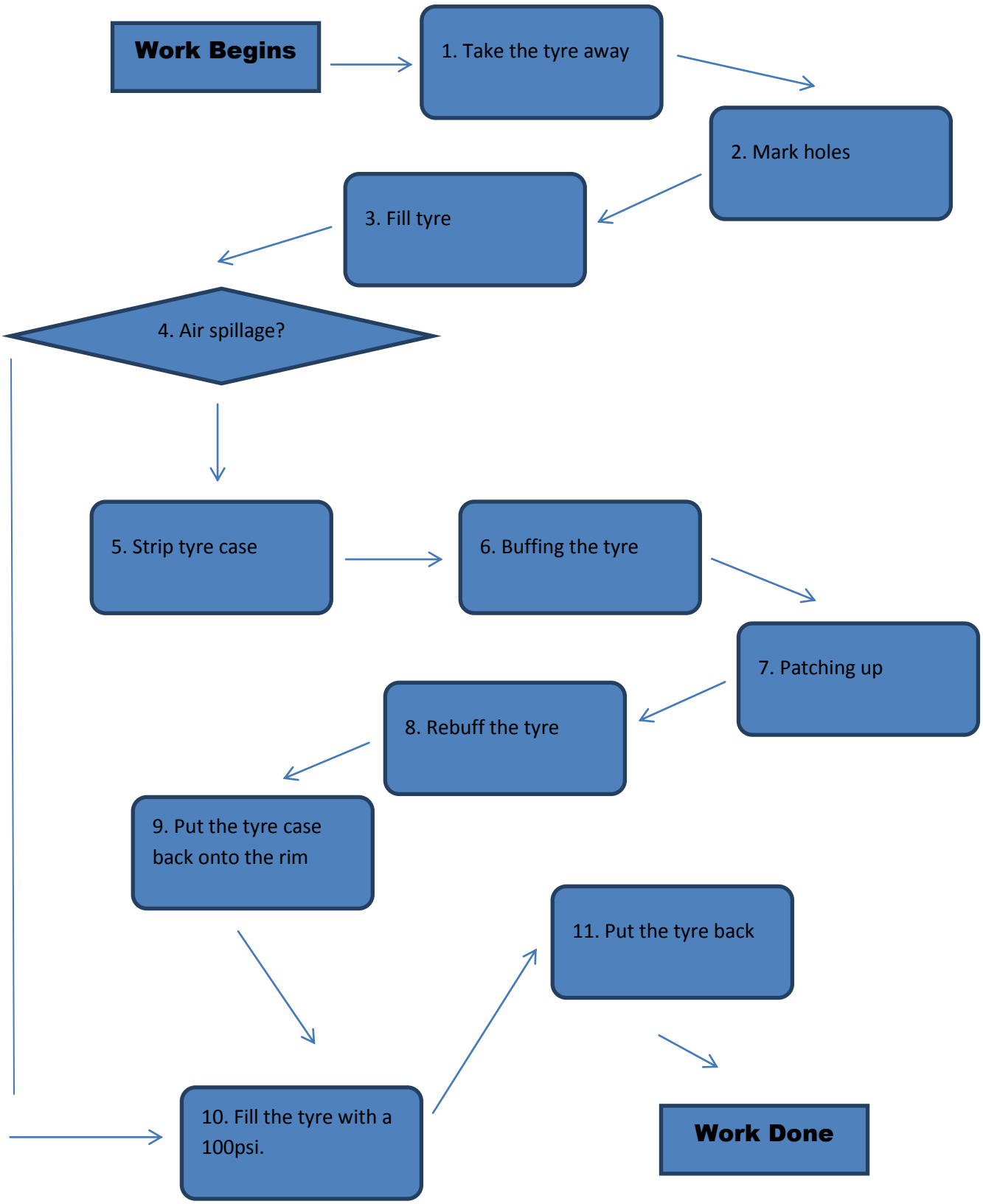
In search of understand and defining the scope even further in this project, Bridgestone decided that a field trip out into the actual workspace was imperative. This would help me by understanding the workflow process of each job being undertaken by a fleet technician and the completion of a Vehicle Inspection Report (VIR).

There were two different common scenarios which were being undertaken at jobs: Tyre Replacements and Tyre Punctures.

Tyre Replacement



Tyre Puncture



By learning the physical workflow process of what the fleet technicians do at the job, we were able to find out what the flaws are in this process. It is much a disorganized process, and by me going out onto the field, I was able to put forth an organized and flexible scheme which should be done at every job. By following what the application does, the fleet technician can be doing the job quickly and efficiently solving the issue of the client efficiently and completing the VIR form simultaneously.

Use Case Diagram

The Use case diagram was created to describe the functionality of the proposed application from the fleet technician's perspective.

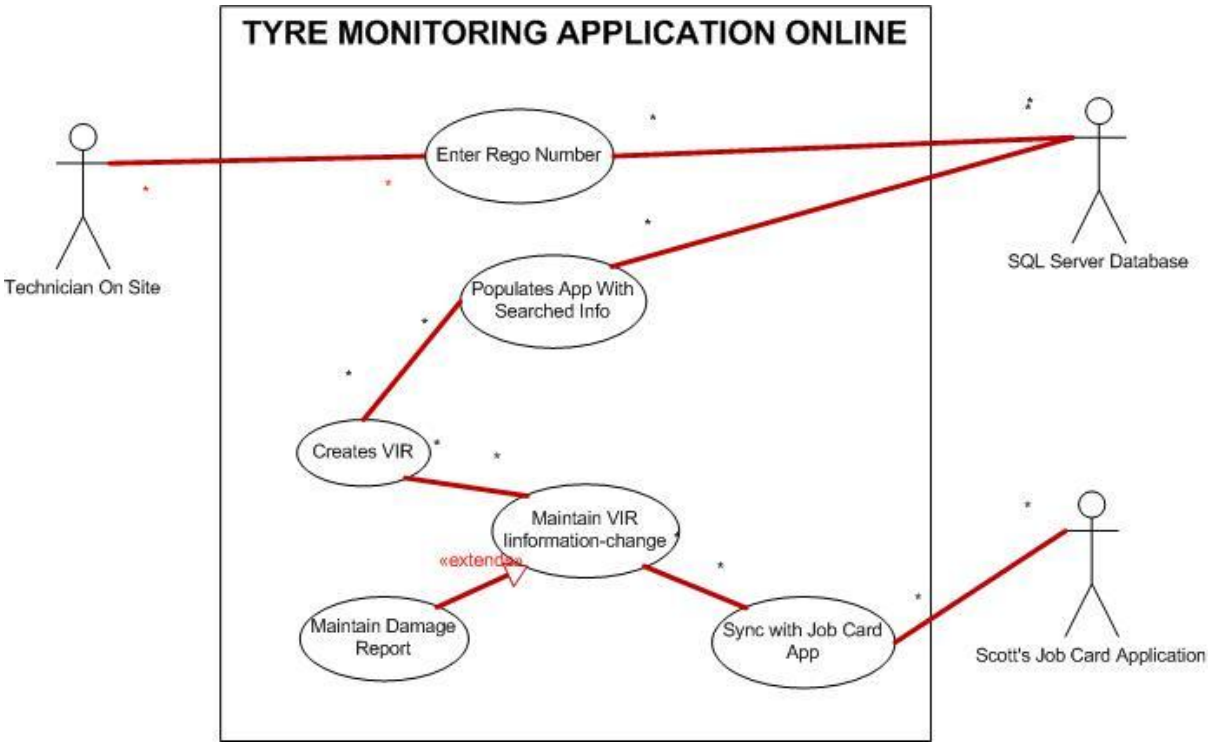


Figure 12: Use Case Diagram

The fleet technician would have to enter a registration number on that tablet in order for the tablet to populate the relevant fields with the appropriate information. This can be done with a Web Service call to Bridgestone's CRM Database. The technician would then be allowed to add information like the current tread depth and tyre pressure of the vehicle, to the already populated information that has been supplied to him. A Damage report is an option given to the technician, if there is any damage done to the vehicle. After completing the Vehicle Inspection Report, the application would proceed to sync the information recorded by my Application onto the Job Card Application.

Software Architecture Diagram

A Software Architecture diagram was created to document the communication between the databases and the *Fleet App*.

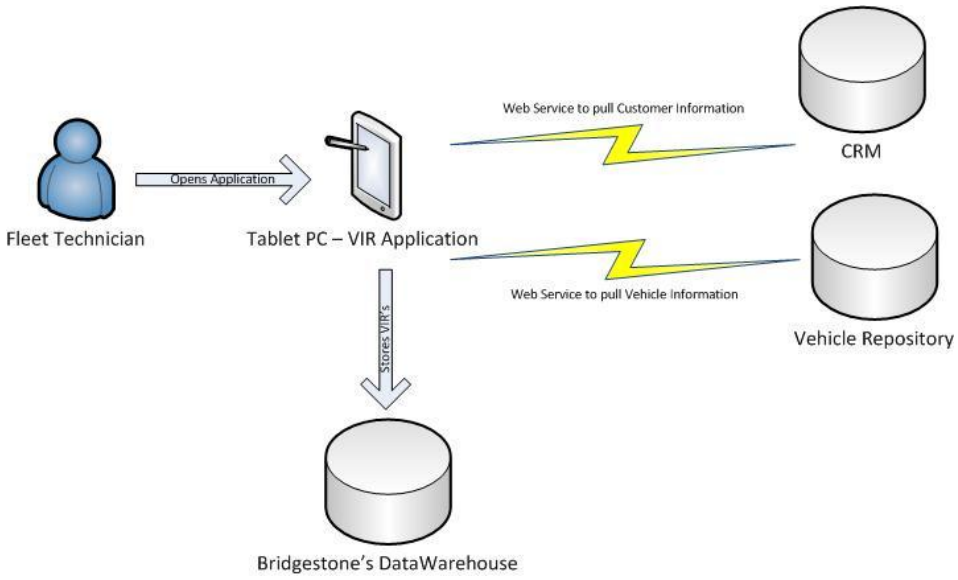


Figure 13: Software Architecture Diagram

The *Fleet App* when booted up gives the fleet technician the option to search a vehicle by registration number. When searched, the results are concurrently being pulled from the Vehicle Repository system and the CRM system, via web services that we have created, and displayed to the technician on the Tablet PC. Vehicle Data is located at the Vehicle Repository and the Customer Information is located at CRM. When the fleet technician has completed the Vehicle Inspection Report, the VIR would write all the recorded information

Class Diagram

A Class diagram was created to describe the structure of the *Fleet App* by showing the application classes, their attributes and methods, and the relationship amongst these classes.

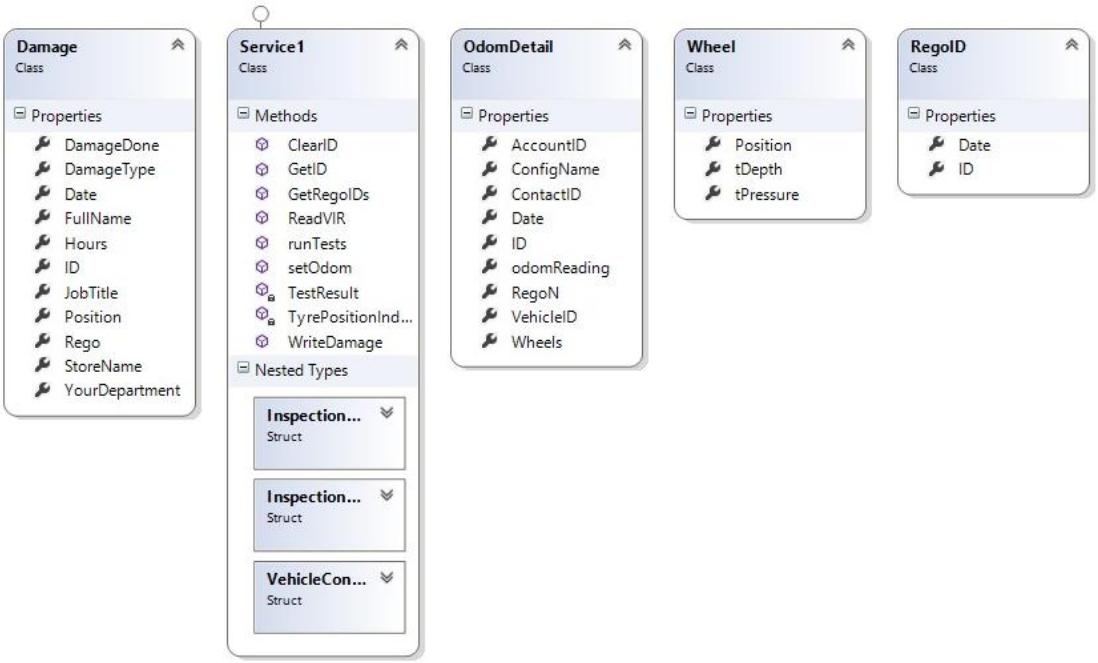


Figure 14: Class Diagram

Currently 4 of the 5 classes get updated as you navigate through the application. The Wheel Class is very important as it stores the Tyre Pressure and Tread Depth and its Position on the vehicle for each Wheel associated with the Registration number of the vehicle. Also to note, I created 3 structures within the Web Service in order to validate the wheel information. A Wheel Structure which has a Tread Depth, Tyre Pressure and Position attributes in it. Also a Rules Structure has been created to incorporate all possible rules for all possible configurations. This is important because my runTests () includes these structures as parameters in order to run the relevant tests against each Wheel for the passed Configuration.

Workflow Diagram

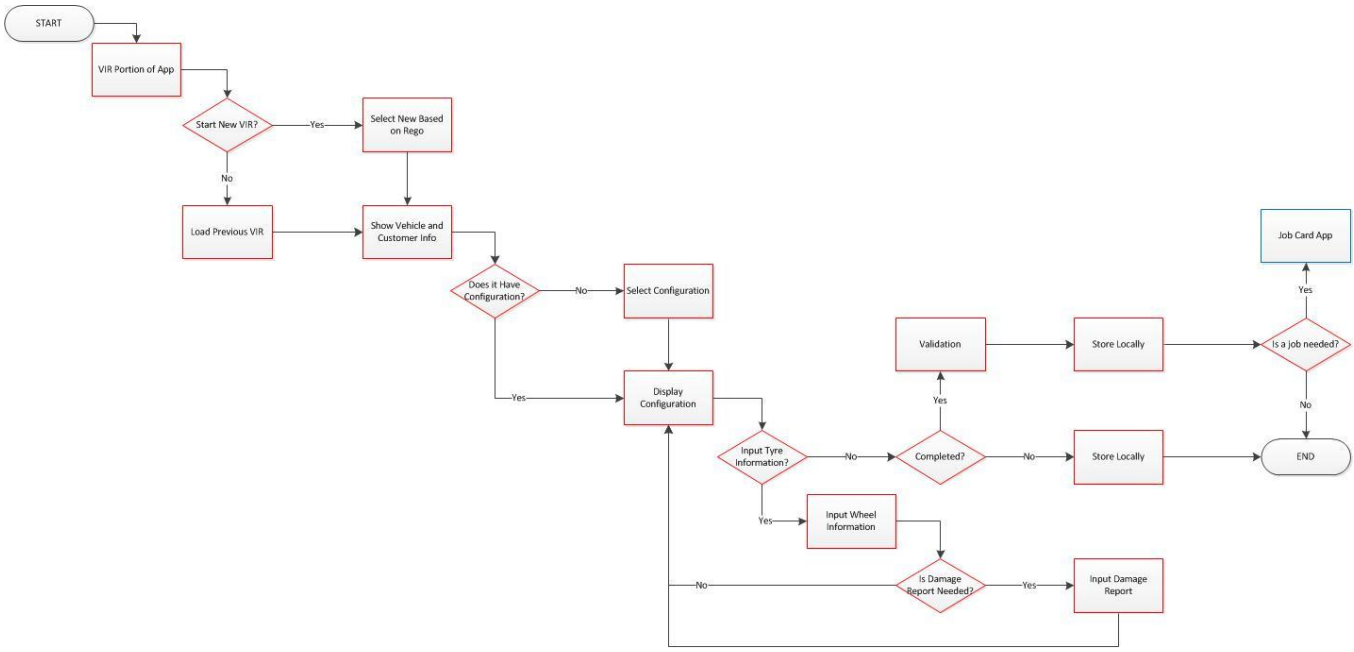


Figure 15: Current Workflow

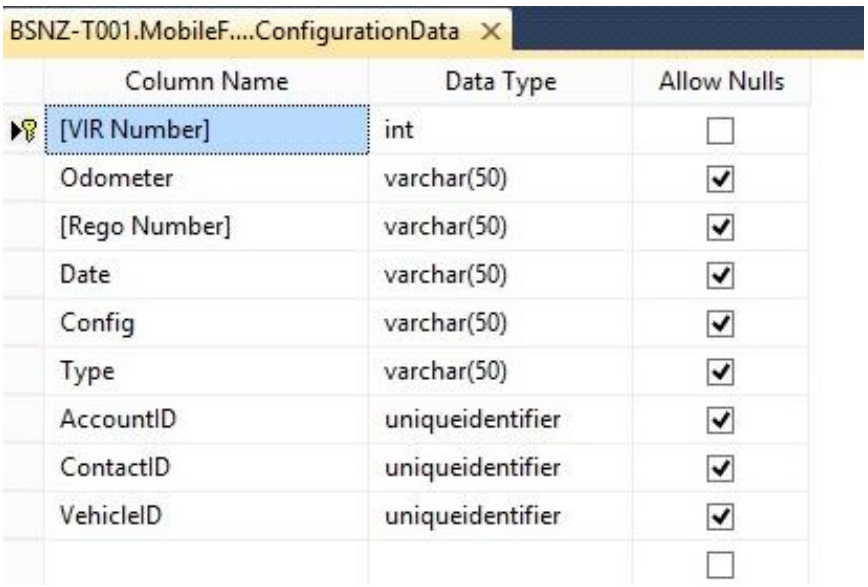
When the fleet technician leaves for the job, all he has to do is to enter the registration number of the vehicle he would be working on, and the web service would populate all the customer and vehicle information in order to complete the report. It displays a configuration diagram of the vehicle and within each wheel position a previous tread depth and tyre pressure is displayed to the technician. A damage report form is included with each wheel position if needed. At the end of the report, the *Fleet App* is going to validate the recorded Tread Depths and Tyre Pressures and give a suggestion back to the fleet technician on what kind of work is required on the vehicle. After the validation has been shown to the fleet technician and takes the suggestions into consideration, the *Fleet App* stores the Vehicle Inspection Report data into the appropriate tables.

Also if the fleet technician decides to park the application, all the classes of information will be stored in the table with the primary keys being the Unique VIR number and the date of when the technician parked the application. The technician can pull back the previous half-filled form by searching it up using the ‘Existing VIR’ option. This will populate the Vehicle Inspection Report with all the data that was previously written and stored in the classes.

SETTING UP THE LOCAL DATABASE

I decided to use SQL Server Standard because all of Bridgestone's Systems are using the same server and it would be best to stay consistent with their systems. We also used the Microsoft SQL Server Management Studio 2012 because it has one of the best GUI's around giving users a more visible understanding of the database and allows easier manipulation and modification of data. We downloaded both the software's into the device. In order to access and view all the tables that currently exist with Bridgestone, we took an image of the current database set up at one of the local machines at Bridgestone and imported it across to the device. Now the tablet can freely interact with the database, and I can view the changes made on the tables through the SQL Server Management Studio

The *Fleet App* currently deals with 2 main tables in the Database: ConfigurationData and Wheel. The ConfigurationData table consists of columns of data relating to information being written about the Vehicle in general. The Primary key of this table is the Unique VIR Number that is distinctive to each Vehicle Inspection Report.


BSNZ-T001.MobileF....ConfigurationData

Column Name	Data Type	Allow Nulls
[VIR Number]	int	☐
Odometer	varchar(50)	☑
[Rego Number]	varchar(50)	☑
Date	varchar(50)	☑
Config	varchar(50)	☑
Type	varchar(50)	☑
AccountID	uniqueidentifier	☑
ContactID	uniqueidentifier	☑
VehicleID	uniqueidentifier	☑
		☐

Figure 16: ConfigurationData Table

The Wheel Table consists of data relating to each Wheel of a vehicle. A Vehicle can have multiple Wheels, so therefore there is a One to Many relationship between the

ConfigurationData table and the Wheel Table. In the Wheel Table, you can you can have multiple wheels. Therefore the primary key of the table would be a combination of the Unique VIR Number and the Position of the Wheel on the Vehicle. This table is where the Tread Depths and the Tyre Pressures of each Wheel are stored. The process of inserting this data into these particular tables will be discussed in the Developing of the App section.

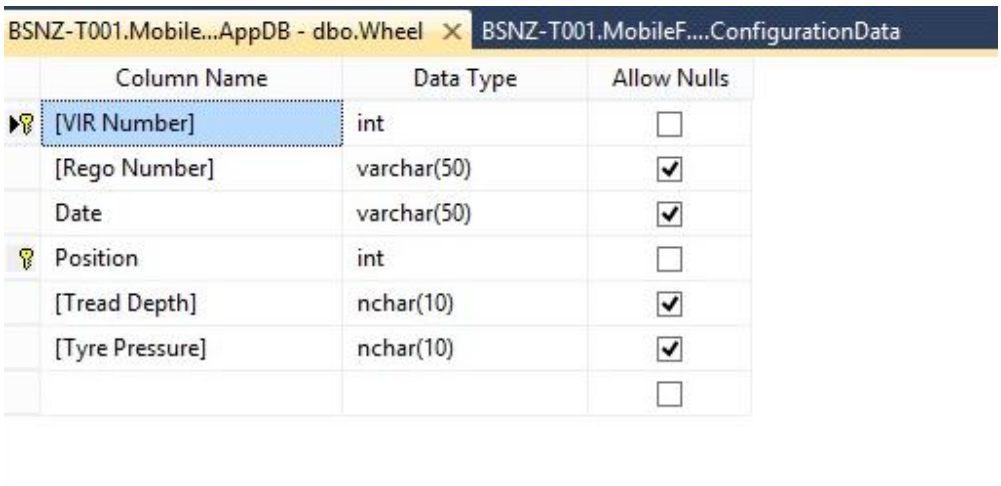


Figure 17: Wheel Table

DEVELOPING THE MOBILE APPLICATION

Microsoft Visual Studio 2012 for Windows 8 is the environment I will be doing most of development of the *Fleet App*. Visual Studio gives me everything I need in terms of creation of code, debugging, packaging and deploying a mobile application. It interacts with Blend really well as you can seamlessly move back and forth in order to develop the Application. Blend is basically used to design a great looking user interface for the mobile application.

On starting the application, the user sees the start-up Main screen displaying the 4 possible options of the *Fleet App*. My part of the application is the Vehicle Inspection Report

Figure 18: Startup Screen

Displaying Information

When it comes to displaying information, one of the specifications was to display vehicle information as well as customer information to the fleet technician. This is needed so that the technician can verify with the customer about the vehicle he is going to be working on.

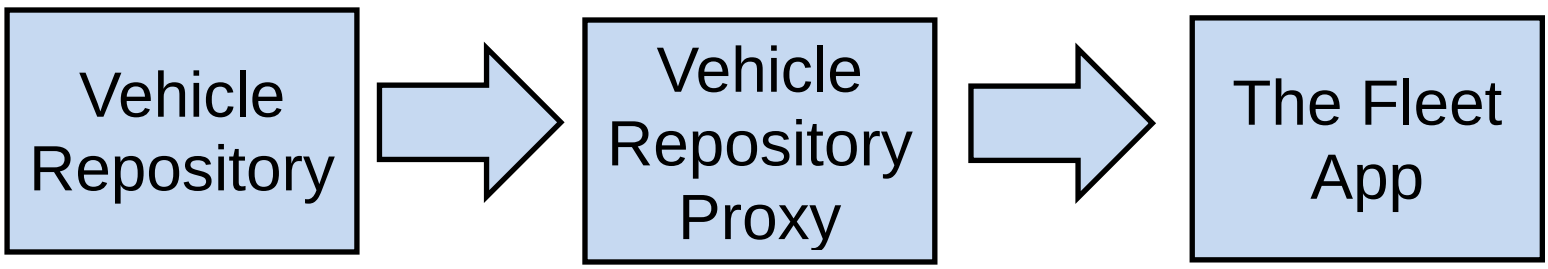
The displaying information of customer and vehicle information to the user is done by using two Web Services: 1) CRM Vehicle Web Service 2) Vehicle Repository Web Service. The CRM Vehicle Web Service access the Vehicle information located in the CRM and returns

information like the Account Number and Account Name to the *Fleet App*. It uses the same concept as done previously in the Web Service I created earlier but dealing directly with CRM.

```
try
{
    VehicleServiceClient v = new VehicleServiceClient();
    System.Threading.Tasks.Task<VehicleInfo> test = v.GetVehicleDataAsync(rego, "RpKU5468OoI05925E4", "naidu_v");


    ModelT.Text = test.Result.model;
    MakeT.Text = test.Result.make;
    RegoT.Text = rego;
    YOMT.Text = test.Result.year_of_manufacture;
    LORT.Text = test.Result.latest_odometer_reading + " " + test.Result.current_vehicle_odometer_unit;
    WOFET.Text = test.Result.expiry_date_of_last_successful_wof;


    CRMVehicleService.VehicleServiceClient c = new CRMVehicleService.VehicleServiceClient();

    System.Threading.Tasks.Task<System.Collections.ObjectModel.ObservableCollection<CRMVehicleService.VehicleHeader>> test


    AccNameT.Text = test1.Result[0].Account.AccountName;
    AccountNoT.Text = test1.Result[0].Account.AccountNumber;
```

Figure 19: Code Snippit on retrieval

Dealing with the Vehicle Repository Web Service was a little bit more complicated. Information directly relating to the vehicle is located in the Vehicle Repository system which is designed to hold the vehicle information. We initially tried the same way to return vehicle information by using the registration number but this led into a major problem. The Vehicle Repository Web Service returns a dataset of vehicle information. The System.Data namespace is not available for use in a Metro-style application and therefore I could not get access to the returned dataset. A solution was designed to overcome this situation through the means of a proxy server.



The proxy server was designed to manipulate the dataset and converts into a class so that the Metro style application can read the class and therefore populates the fields with the Vehicle Information.

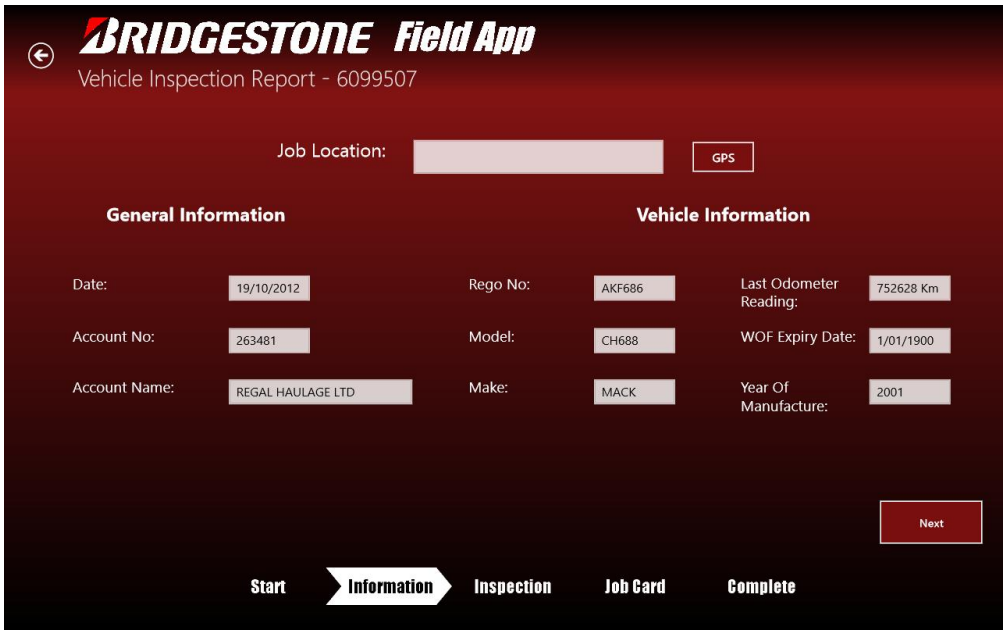


Figure 20: Customer/Vehicle Info Screen

The Interactive Vehicle Configuration Diagram

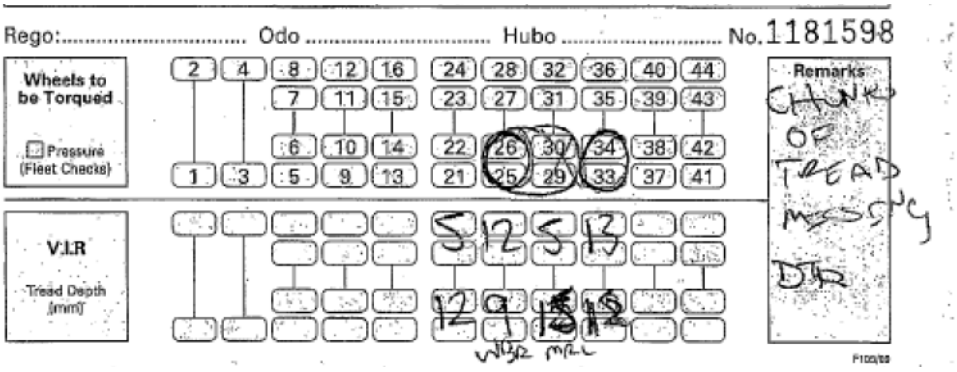


Figure 21: Current Configuration Diagram

An interactive vehicle configuration diagram was designed for the fleet technician to go about entering information about each tyre normally in an organized fashion. I tried to mimic the Configuration diagram above as closely as I can when designing the diagram. I used buttons, as the shape represents a wheel pretty well. We also clearly labelled each

wheel position with the number, referencing the same number scheme as used by Bridgestone. Also an additional textbox was added to the Configuration diagram to record the current Odometer/Hubometer reading.

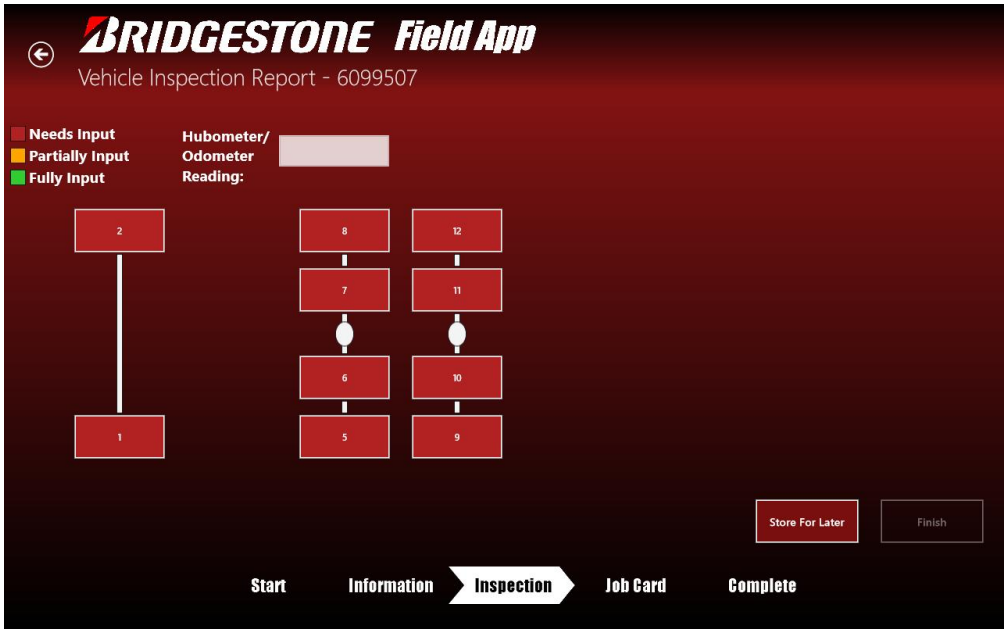


Figure 22: Vehicle Interaction Configuration Diagram

In the backend, we designed all possible configuration diagrams for vehicles at Bridgestone. We then match the diagram with the Configuration Name obtained from the Vehicle Repository Web Service to get the display of the interactive vehicle configuration diagram on the next page.

```
if (configType == "4 X 2 (a) FWD")
{
    Config_Type.Text = "Car | 4x4 | Van";
    Rec_1.Visibility = Visibility.Visible;
    Rec_2.Visibility = Visibility.Visible;

    Ell_1.Visibility = Visibility.Visible;

    Pos_1.Visibility = Visibility.Visible;
    Pos_2.Visibility = Visibility.Visible;
    Pos_3.Visibility = Visibility.Visible;
    Pos_4.Visibility = Visibility.Visible;

    Pos_5.Visibility = Visibility.Visible;
```

Figure 23: ConfigTest

The wheels on the configuration diagram are represented by buttons. Clicking a wheel opens a popup; the app returns values of the vehicles previous tread depth and tyre pressure and can also enter the current tread depth and tyre pressure.

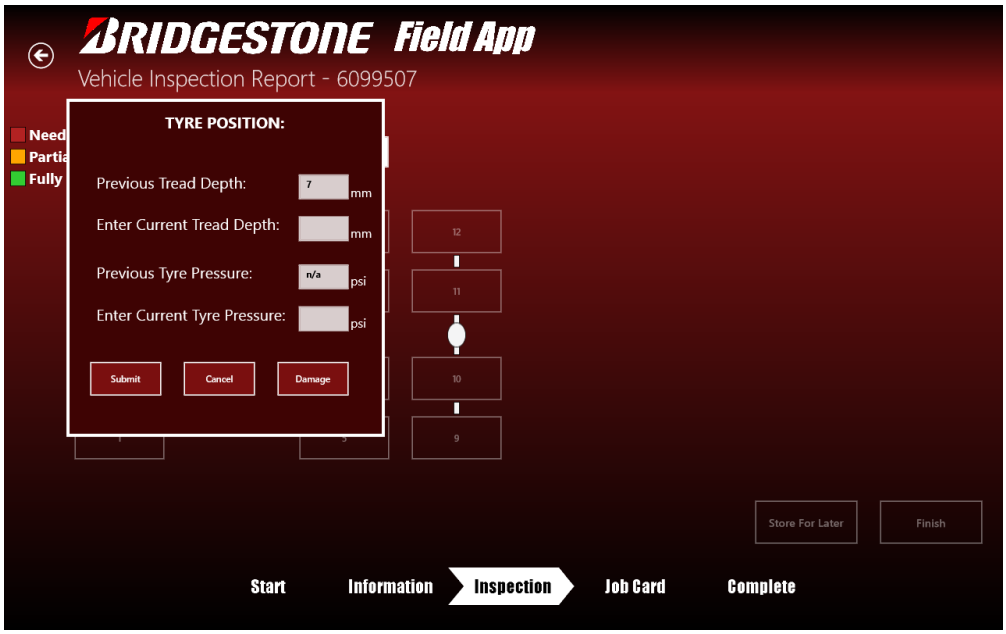


Figure 24: Popup Input For Tread Depth and Tyre Pressure

Giving the fleet technicians a visual aid would be very useful for them to understanding quickly what needs to be done in order to finish up a Vehicle Inspection Report. I have

designed a colour indication of data that needs input, has partial inputted information or has been completed for each wheel. This is very important because Bridgestone would like to record all pieces of information regarding the vehicle before determining what work needs to be done on it. It is also equally important for validating the vehicle's information which I will cover in the Validation section below.

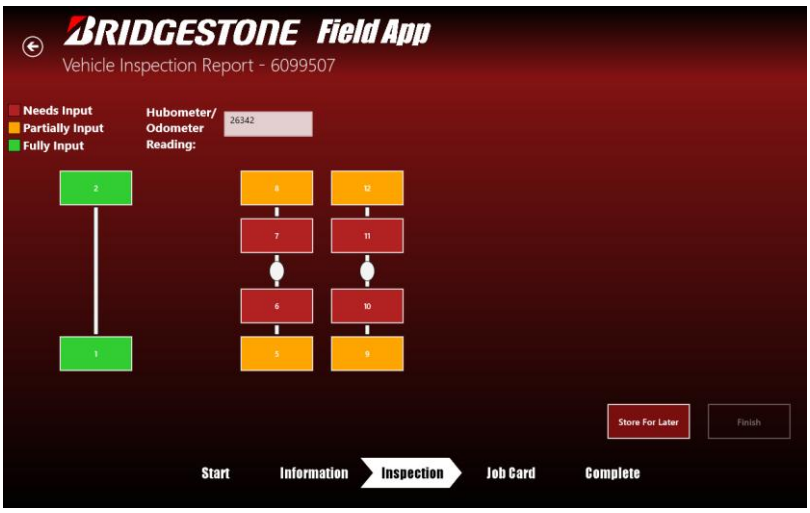
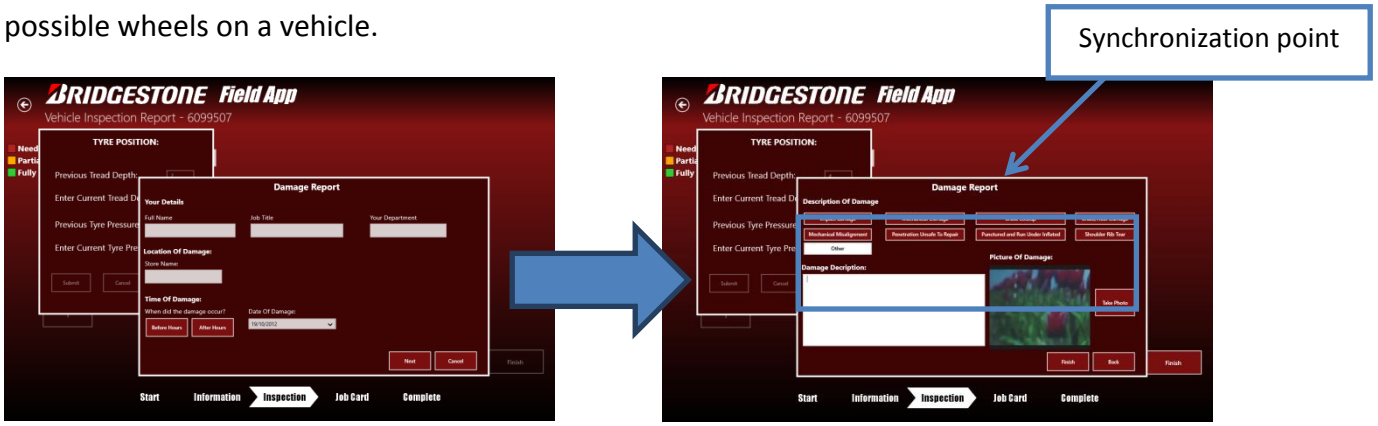


Figure 25: Color Indication

Damage Report

The Fleet Technicians are required to fill out a Damage Report, to report any instance of tyre damage. I have designed the application to have a possibility of a damage report for all possible wheels on a vehicle.



Camera functionality was added to the Damage Report so that the fleet technician can take a picture with the damage along with the written description that he would write. When the

Fleet App writes to the database, the compulsory key's it requires is the Unique VIR number, the tyre position and the Damage type.

Validation

Validation is one of the most crucial parts of the *Fleet App*. It gives the fleet technicians an indication of what the next step in the work flow should be based on the tread depths and tyre pressures they have entered into the application. When the fleet technicians have completed the Inspection Report, a suggestion should be given on what work should be done so they don't have to logically infer it and remember it themselves.

Each vehicle has generic rules that it must follow in order for it to be road worthy. A Rules engine has been placed behind the *Fleet App* to test tread depths against a list of rules based on the configuration on the vehicle. Currently there are 3 rules that will be tested for each Vehicle that is being inspected via the *Fleet App*.

- Tread Depths recorded are less than 3mm.
- Tread Depths variance between two wheels is greater than 5mm.
- Tread Depths variance between four wheels is greater than 7mm.

The rule engine has been previously implemented as a CRM plugin. I used this as template to design a runTests() method which runs the tests against each Wheel Object which contains a Tread Depth and Tyre Pressure instance.

There result of the runTest() method will output a string stating all the failures the method tested and a brief suggestion indicating what kind of work needs to be done on the vehicle.

Figure 26: Validation Result

PROBLEMS/ SOLUTIONS

This chapter describes the problems that I have encountered throughout the year that I haven't detailed already in the report. The problems are going to be categorized as Technical and Non-Technical.

Technical

Unique VIR Number: How do you uniquely identify each Vehicle Inspection Report? I designed a way so that when a New Vehicle Inspection Report is created, a unique number is assigned to the VIR. This Unique VIR Number is also important as it is the primary key for the ConfigurationData table.

Camera: Is a written description sufficient for the Damage Report? It was decided that a camera function was needed as it provides a unique representation of the damage compared to the options given to the fleet technician.

```
private async void NextD1_Copy_Click(object sender, RoutedEventArgs e)
{
    var ui = new CameraCaptureUI();
    ui.PhotoSettings.CroppedAspectRatio = new Size(4, 3);
    var file = await ui.CaptureFileAsync(CameraCaptureUIMode.Photo);

    //await file.MoveAsync(ApplicationData.Current.LocalFolder);

    if (file != null)
    {
        var bitmap = new BitmapImage();
        bitmap.SetSource(await file.OpenAsync(FileAccessMode.Read));
        Photo.Source = bitmap;
    }
```

Figure 27: Camera Implementation

No Configuration: What happens when a vehicle doesn't have a configuration registered in the database? To make the *Fleet App* more flexible, I added a drop down box when the vehicle has no configuration set in the database. This allows the fleet technician to make the decision on what configuration the vehicle is, and selects the appropriate configuration and then the appropriate configuration diagram will appear on the screen. This would also store the set configuration diagram in the database, so when the fleet technician wants to deal with this vehicle again, the last set configuration will appear in the *Fleet App*.

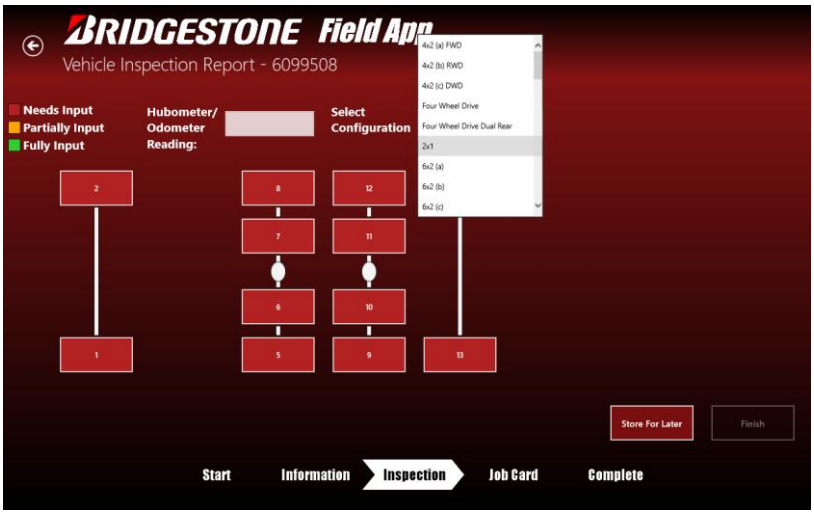
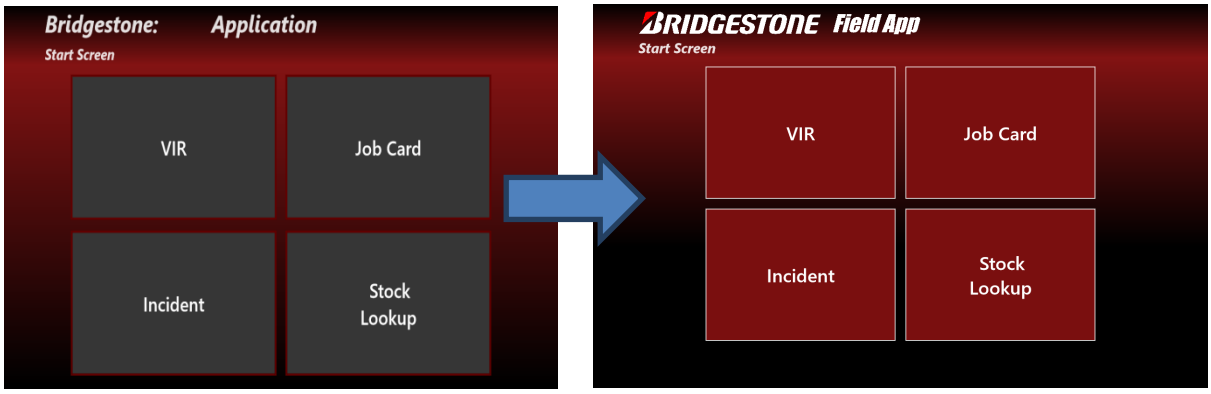


Figure 28: Option Configuration Choice

Non-Technical

Access to Devices with Windows 8: We did face an issue in getting a setup at Bridgestone to work on developing the *Fleet App*. It took quite a while to get permission from the Bridgestone head IT staff to install the Windows 8 OS on a system for me to develop the *Fleet App* on. Eventually after 5 weeks we did get permission to develop the application on a Bridgestone system.

Colour Scheme: The Colour Scheme of the *Fleet App*? I gave Bridgestone options on colour schemes of what the *Fleet App* could look like and got very useful feedback on the background colour of the whole application and buttons/popup colours. They stated that they want the buttons to stand out rather than fade into the background. So I just brightened the buttons and darkened the background to bring the focus of the user attention on the content of the page.



FUTURE WORK

Based on the current status of the *Fleet App*'s development, there are a few parts of the *Fleet App* that could not be completed. Also, there are a few things that are out of the scope of this project that should be added to the system.

Tyre Vigil – Tyre Pressure Monitoring

This tracking system is currently being trialled on vehicles in a customer fleet and uses devices attached to the valve system base that record tyre pressure and feeds this data this to an in cab unit, which in turns sends this to a service in Canada. Currently Bridgestone are able to extract this data and update the data warehouse with the readings and uses the data in Fleet Management vehicle service scheduling and reporting.

In the future, Bridgestone would like to use these trackers with all their customers to get live readings of tyre pressure. Bridgestone would analyse this data and use certain algorithms to determine what kind of work needs to be done ahead of time. Bridgestone could take a more proactive approach when it deals with work for its clients by analysing the live data from Tyre Vigil and running algorithms against this data. The algorithms would then produce an update to notify Bridgestone that a tyre pressure is too low and needs a job to be done on it. Bridgestone could then inform the technicians via the *Fleet App* that certain work needs to be done on a client vehicle.

GPS Implementation

Implementing GPS within the application to find out where exactly the Job is being done and returns a string of the address location to the *Fleet App*.

Less Writing?

Ideally we would like the Bridgestone *Fleet App* to have features where the fleet technician doesn't have to type in much information. We would like to implement more features like toggle buttons and radio buttons giving the fleet technicians options allowing him to quickly make a decision. This would improve the efficiency of the workflow because it increases the speed in which the Vehicle Inspection Report can be completed.

Service Level Agreements

All Service Level Agreements for the relevant client should be displayed on a PDF Viewer when the fleet technician is dealing with the vehicle. Currently we had trouble opening pdf formats within the application.

CONCLUSION

The project started of slowly and quickly gained pace as the year went along. I have successfully achieved many goals involved in both themes. The major accomplishment and happiness was when my company supervisor appreciated me for the end product that is useful for commercial purposes.

This project has helped me to gain knowledge and understanding of work in a professional environment. This will be very useful for my career and future study because it taught me very valuable lessons. I have come across many problems which helped me learn new skills, both technical and non-technical and gave me confidence that enabled me to achieve positive results that are much needed by Bridgestone New Zealand Limited.

ACKNOWLEDGEMENTS

This report is based on inputs from software developers, business analysts, and fleet technicians which include work conducted by Vishal Naidu, under the supervision of Dr. S. Manoharan. Valuable input, guidance, and support at various stages of the project were received from Philip Low (Business Analyst) and Robert A. Lee (Senior Business Analyst). I would like to thank everyone for their backing and direction in the progression of my project.

BIBLIOGRAPHY

[1] Artail, H., Itani Z., Diab, H.: "Efficient pull based replication and synchronization for mobile databases" 2005. [Online]. Available: http://ieeexplore.ieee.org/xpl/login.jsp?reload=true&tp=&arnumber=1506554&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1506554

[2] Asthana, A., Asthana RGS.: "IOS 5 Android 4.0 and Windows 8 - A Reveiew" 2012. [Online]. Available: http://www.ieeebeacon.com/Archieves/March2012/Download/PDF/33_43_IOS%20Windows_Beacon%202012.pdf

[3] Bridgestone New Zealand Ltd.: “Company overview.” [Online]. Available: http://www.bridgestone.co.nz/corporate/page/company_overview

[4] Hamad, H., Saad, M., Abed, R.: “Performance evaluation of RESTful web services for mobile devices” 2009. [Online]. Available: http://www.iajet.org/iajet_files/vol.1/no.3/Performance%20Evaluation%20of%20RESTful%20Web%20Services%20for%20Mobile%20Devices.pdf

[5] Jeong, J.Y.: “Development process of mobile application SW based on agile methodology” 2008. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4493779&tag=1&fb_source=message&tag=1

[6] Johnson, R.: “Data distribution with SQL server replication” 2012. [Online]. Available: http://rjssqlservernotes.files.wordpress.com/2012/09/data-distribution-with-sql-server-replication.pdf

[7] Potti, K.P.: “On the design of web services: SOAP vs. REST.” 2011. [Online]. Available: http://digitalcommons.unf.edu/cgi/viewcontent.cgi?article=1139&context=etd&fb_source=message

[8] Teng, C.C.: “Mobile application development: Essential new direction for IT” 2010. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5501730&tag=1

[9] Tracy, K.W.: “Mobile application development experiences on Apple’s iOS and Android OS” 2012. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6248786&fb_source=message

[10] Web services.: "SOAP vs. REST" 2006. [Online]. Available: http://www.sci.usq.edu.au/courses/CSC8417/Resources/Examples/CSC8417_SOAPvsREST.pdf

[11]Executive Brief. (2008). Which lifecycle is best for your project? Retrieved from http://www.projectsmart.co.uk/which-life-cycle-is-best-for-your-project.html

[12]Ios and Android accounted for 82% of smartphone sales in q1 2012. (2012, 05 25). Retrieved from http://www.iphonehacks.com/2012/05/ios-android-82-percent-smartphone-sales-in-q1-2012.html

[13] 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

[14]Programming 4 Us. (2010). Understanding SQL Server CE Synchronization. Retrieved from http://programming4.us/mobile/930.aspx

[15]User Interface Guidelines | Android Developers. (2012).
Retrieved from http://developer.android.com/guide/practices/ui_guidelines/index.html