Chapter 1

1.0 Introduction

1.1 Overview

Vehicles today are much more intelligent than they were years back. The traditional vehicle timed the ignition of the spark using mechanical distributors. This method of fuel and air mixture did not meet a better and efficient composition to burn inside the cylinders to squeeze the most power output during higher engine revolutions, due to the fixed nature of the mechanical setup, it was. In the current era most modern engines are controlled electronically with the assistance of Electronic control unit (ECU) [1] which consists software to analyse state of the engine efficiency and vary inputs accordingly. The ECU controls many other sub systems of the engine, for example, the anti-lock braking system (ABS) , certain air bag systems, electronic power steering and various drive settings which demand various throttle, steering and gear shifting aggressiveness. All decisions made by the ECU are based on the states of sensors that are placed at various places throughout the vehicle primarily around the engine bay.

As technology evolved, the ECU became more advanced and intelligent to a level supplying diagnostic and sensor data to help mechanics identify the source of problems. This diagnostics span from an array of simple detection of the Carbon dioxide level in the exhaust to identifying misfires in the engine and complex tasks such as measuring engine-load. As the trade is competitive, once the introduction of such ECU [1] in one brand demanded others to manufacture similar components and integrate them to keep the competitive edge where manufacturers had their own errors and diagnostic notations. As this was eminent to become a very complex task for mechanics as all the error codes would be different from the others. This urged for a standardisation and it became as On Board Diagnostics(OBD) [1]. This was then upgraded and made mandatory for all manufacturers namely OBD ii.

Our project focuses in making the OBD messages easy to understand to the owner of vehicle and make it meaningful and useful in their daily life. At present the OBD port does not serve the owner or the driver in day to day life. Our project gets the maximum data out of the port

that is generally untapped, to make the life with vehicles easier and profitable for both users and to mechanics when it comes to repairing or diagnosing errors. We also extract some additional technical details via the port to provide sensible economic and ecological details to the driver to make a greener and sustainable world.

Cardoc connects to the ECU using a scan-tool which consists, the ELM327. This chip or IC is responsible for the low level timing and signalling to and from the ECU's communication bus. This scan tool is then connected to an android driven mobile phone via Bluetooth pairing for data and error code retrieval and further processing.

Cardoc is a combination of software that start from the Android based mobile phone, analytical engine and a web based data representation and fleet control. Initially the android driven mobile phone pairs with the OBD scan tool which starts extracting various processed data that are collected from various sensors and then forwarded to a server. Once this has reached various processes occur according to user type such as personal or fleet manager

1.2 Background and Motivation.

Vehicles today are advancing technologically and are equipped with more and more software which is becoming responsible for powering them and opening up new exciting services to the driver. This is especially true for the next generation electric or hydrogen cell powered cars. BMW are even talking about developing an open-source in-vehicle platform that allows software developers to interface with the vehicle and provide a better journey experience for the driver. Gasoline is running out quick and vehicles will start moving away from the conventional combustion engine.

Though the OBD ii port is designed to suit the traditional and current day hybrid engine it would undergo changes there is a necessity to change the source of power for vehicles. Though there is no current success in finding an alternate energy source that is efficient or effective enough to replace the internal combustion mechanism the CarDoc application would sustain its place in the market. Our application pairs with the GPS system if it is available in the car or else it would use its GPS that is provided in the android phone itself. This would

facilitate by providing details for geo fencing function and monitoring of the vehicles in the fleet management system. [2]

The android platform is current trend which free and easy to work with as it is free and open source in nature. The platform is very stable and easy to look up for any coding instructions as the vendor is Google. The use of Android based mobile phones helps us as it usually comes with GPRS and GPS which are resources that our application is looking forward to communicate in and extract information from.

1.3 Aims and Objectives

The aim of this project is to get a fully functional android application, Facebook application and a fleet management application working with android application that communicates with all modern vehicles. It should be capable of extracting the necessary data from the vehicle's engine control unit (ECU) in order to use it in a meaningful and useful way. Communication to and from the ECU will be done using the Onboard Diagnostics two (OBD-II) standard. The software that will run on the device will have to be able to work with the hexadecimal replies that the ECU returns on request of data. The communication with the ECU will have to be handled using a polling type method as data interrupts or automatic updating of data from the ECU cannot be done sporadically. Instead a cyclic process or thread will have to run continuously to do a query to the ECU followed by the reading of the reply. The software will have to work with the returned hexadecimal data in a way that provides the user or driver useful functionality.

The objectives of this project are as follows:

To communicate with the ECU of a vehicle indirectly with the help of Bluetooth OBD ii scanner which will handle all the low level bus communication with the ECU using whatever signalling method the vehicle uses. There are five OBD-II signalling protocols in total and the OBD II scanner supports all 5 including CAN. The communication with the Bluetooth OBD II scanner will be done wirelessly with the paired android mobile phone. Then the data would be transferred to the main server where it would be saved by the name of driver and then used to rate him according to his driving style and vehicle's state, in the social media in order create awareness and attention towards the contribution to the green factor through vehicle usage.

To implement a design in which multiple vehicle sensors such as engine RPM or engine load can be monitored simultaneously. Every sensor has its own equation or formula [1,2] that is applied to the returned data bytes from the ECU. An important design consideration is to provide a convenient way of adding new sensors to CarDoc with the minimum amount of number of lines of code. This will make Cardoc extensible by providing easy addition of new sensor types when they become available in the future. As a bonus, a priority based system should be implemented where some sensors get updated more frequent than others. For example the engine RPM is a high priority sensor as it changes more frequently than the engine coolant temperature.

To create applications which is interactive and useful in daily life which would provide useful and comprehendible details in a non-technical manner via a user friendly GUIs and in certain aspects enabling social media interaction. The strength of the application also lies in the Mongo data which enables big data analysis which would make the social media aspect of comparing drivers very effective and powerful tool to make a greener world from transport sector.

1.4 Report Structure

The remaining of this report is organized as follows. Chapter 2 is a short chapter that describes how project management was handled. Chapter 3 will describe background information on the techniques and areas worked on in this project. This is information that is required to be read in order to have an idea of what the chapters that follow refer to. Included in this chapter is the extra research that was carried out. Chapter 4 discusses briefly the design of CarDoc. Chapter 5 describes how the implementation and testing was performed and what types of test cases were run with the results as well. Chapter 6 finalizes the report with my conclusion of this project any future enhancements that may potentially be implemented.

Chapter 2

2. Project Management

2.1. Functional and Non-Functional Requirements

1) Real Time View Safety and Efficiency Measurements: The application must provide a minimum of three of the following measurements to track safety and efficiency listed in Table1.

Table 2.1

Measurement	**Description**
Acceleration	Positive change in velocity over time. Accelerating too quickly can be considered inefficient and unsafe.
Deceleration	Negative change in velocity over time. Decelerating too quickly can be considered inefficient and unsafe.
Speed of Vehicle	Measured in MPH. Distance divided by time. Driving too fast can be considered inefficient and unsafe.
Detection of Safety Equipment Issues	Determining faults in the vehicle's safety equipment. Driving with faulty safety equipment is dangerous
Detection of Mechanical Issues	Determining faults in mechanical equipment. Mechanical faults can impact safety and efficiency.
MPG	Miles Per Gallon. Driving style can have a positive or negative impact on the rate that a vehicle uses fuel.
Location	Geographic location. Can be used in conjunction with other data to pinpoint location of inefficient or unsafe driving.

These measurements should be displayed to the user in real time alerting the user of unsafe and inefficient behaviour.

2) Logging of Data: When an issue is detected, the application will log a summarized version of the issue on a five second interval. The five second interval is required because in real time mode, these measurements may be collected and displayed on a much more frequent basis. This interval allows the data to be summarized in a more meaningful way.

3) Reflective View: The application will provide a reflective view that displays unsafe, inefficient, or questionable driving behaviour from the summarized data collected in

requirement #2.This view must display the data in a meaningful way that allows the user to reflect on what trends or environmental factors may have triggered the undesired behaviour (e.g. Merging on to a highway). Most importantly, this view should be available in the historical sense, not requiring any interaction while driving.

4) Platform: The application will run on a modern smartphone device.

5) Performance: In real time mode, vehicle measurements displayed to the user must be updated at a minimum of once per second.

6) Performance: In real time mode, all time critical measurements dependent on third-party devices (e.g. Network communication) must attempt to update once every minute.

2.2 Project Schedule

The following are the major milestones that the team decided to be met within the defined time frames as for the success of the project.

Milestone 1: web site, user login privileges, diagnostics data retrieval from the database and displaying the functionalities of our product in the web as a marketing perspective.

The success from marketing perspective of the project relies on the GUIs and the strength of complex analysis and functionalities of the application. This will achieved by the development of the stable website with proper privileges as the application provides the fleet management system along with the general use. The simple pairing mechanism to ensure the customers comfort to user it with an android device in the day to day life. The connection with the database of all the Front ends transfers all the valuable and analysed data.

Milestone 2: Blue tooth pairing and retrieving data and sending to the server

The success of the project from the technical aspect mainly relies on communicating to the OBD ii scanner with the android device and retrieving the necessary data from the ECU. This is the starting point of all the data that is displayed to the user in the entire GUI provided by the application.

Milestone 3: Development of the android interface and functionalities.

The success of making the customers use it on a daily basis willingly without making it a complex task depends on simplicity the pairing mechanism and usability of the android application inside the vehicle. The application will contain simple steps to pair the device and make the simples selections regarding the vehicle and the rest will be done by the application itself. The notification and additional details provided in the application will very simple and appear in a non-technical terminology.

Milestone 4: Analytical Engine and Fleet Management.

The applications market factor and the empowerment and implementation of green concept rely on the fleet management application and the social media applications respectively. The fleet management application will provide the details of the vehicle details along with its coordinates which is obtained from the GPS of the vehicle along with some valuable details such as the engine load and the vehicle speed to the fleet manager at his computer screen. This will take fleet monitoring to very precise and intensive level. Whereas the social media application will make the environmental impacts by monitoring the drivers driving patterns and the vehicle condition he drives and then rate him among his friends which would indirectly push the user to be more concerned about the vehicle and the way he drives which would ultimately result in a sustainable and greener world.
[3]

Chapter 3

3. Background and Similar products

3.1 Background of On Board Diagnostic system

On Board Diagnostics version II is a computer system built into all light and medium-duty vehicles sold from 1996 onwards. OBD systems are designed to monitor the performance of some an engine's major components, including those responsible for controlling emissions, though automobile manufacturers often include proprietary information (called EOBD2, for Enhanced on board diagnostics 2) over the standardized OBD II connector. The system reports and identifies malfunctions of critical systems, and also reports live readings from selected sensors. OBD II supports limited storage for "freeze frame" events when a failure is first detected, providing and understanding of why the failure may have occurred.

OBD II was developed by the State of California Air Resources Board(CARB) and the United States Environmental Protection Agency (EPA) and is standardized in the US by the society for Automotive Engineers (SAE) and worldwide (as E-OBD) by the international organization for standardization(ISO). Originally multi-protocol, as of 2008, all vehicles sold in the U.S, were required to utilize a protocol; called CAN Bus to reduce the complexity of interfacing with the vehicle.

[5] Rahul Mangharam. Autoplug: Common Automotive Interfaces for Plug-n-Play Servies, from, http://www.autoplug.org/docs/AutoPlug.pdf

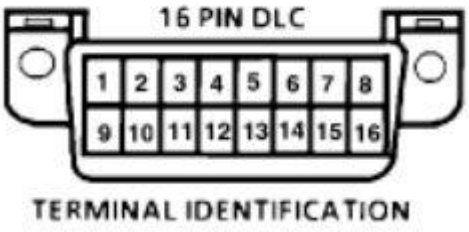


Figure 3.1 OBD Port Terminal Diagrams

Table 3.1 Pin Usage on OBD-2 Port by Standard

Pin Usage on OBD-II Port by Standard		
Standard	**Usage**	**Pins used**
Pulse Width Modulation	Ford	2,4,10,16
Variable Pulse width	Chevy	2,4,16
ISO 9141-2	European and Asian Models	4,7,15,16
KWP200		4,7,15,16
CAN	Mandatory Standard for 2009 and beyond	4,6,14,16

[6]

3.2 Literature Review

The development of an OBDII onboard peripheral is not a novel concept. In fact, many OBDII adapters are available on the market today. These products cover a wide range of feature sets including auto-enthusiast levels of detail up that of a professionally trained auto mechanic with an elaborate array of complexity. These peripherals typically form a wired connection to a vehicle‘s OBDII port. It is then standard fare for these OBDII readers to be connected via wire to a PC where an individual may conduct further analysis given the diagnostic codes pulled from the vehicle. Alternative solutions also provide wireless PC – OBDII peripheral connections over Bluetooth or Wi-Fi.

These stand-alone peripherals are not the solution to consumer education. In fact, one of the underlying problems with these product offerings is the small monochromatic screen used to display the OBDII code. These devices have been designed in a less than consumer-friendly fashion with the intention to target auto mechanics and enthusiasts. Furthermore, these modules are crafted from non-standardized components. Many companies employ different microcontrollers and algorithms in order to analyze and interpret communications from a vehicle‘s OBDII port. The small size of the target market and the cost

of these unique and often proprietary components contribute to making existing solutions prohibitively expensive. [5]

Large automobile manufacturers such as Audi and GM have designed proprietary systems to seamlessly sync with their automobiles. Obviously these systems are closed and accessible only by Audi and GM vehicle owners. While these solutions are a step in the right direction for driver education, the closed nature of these platforms limits the features of these tools to merely —a great addition to the already impressive interior features.

Alternatively, small start-ups and app developers such as Go-Point Technologies have created consumer-friendly products that are designed to integrate with a wide variety of vehicles. The Go-Point BT1 is a Bluetooth-enabled OBDII adapter that communicates vehicle diagnostic information to iPhones, iPads and iPod Touches. Yet even this solution has multiple shortcomings, the first of which is the use of an inferior data transfer protocol.

Chapter 4

4.0 Design and Analysis

Designing a system that accomplishes the three goals outlined in this section. It consists of three interrelated components. The first component is an application that runs on a mobile device such as a smartphone. The mobile application features an interface to relay error/diagnostic information to a user and to transmit the GPS position to the remote server. The second component is the OBD II Bluetooth adapter that attaches directly to the automobile and transmits information to the mobile application. The third component is an online service that maintains up-to-date error code definitions.

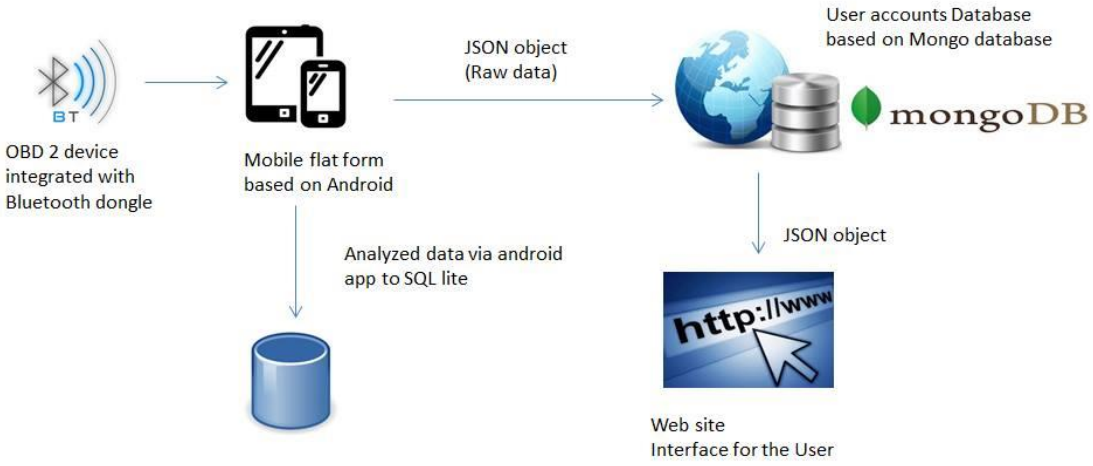


Figure 4.1 CarDoc's System Flow Design

The mobile application and the web are the heart of the Team Hello World's solution. It is responsible for pulling the most recent OBDII code definitions from the web service component. It is responsible for looking up any reported error codes in an up-to-date

definition table. It is also responsible for presenting visual notifications (warnings) to a user in the event that an error is detected.

- The OBD II Bluetooth adapter, listens for information passing over the automobile‘s controller network. The hardware component must collect this information, extracting from it the relevant, useful data. It also must maintain a wireless connection with the mobile device. It then communicates the raw, unformatted diagnostic information to the mobile component.

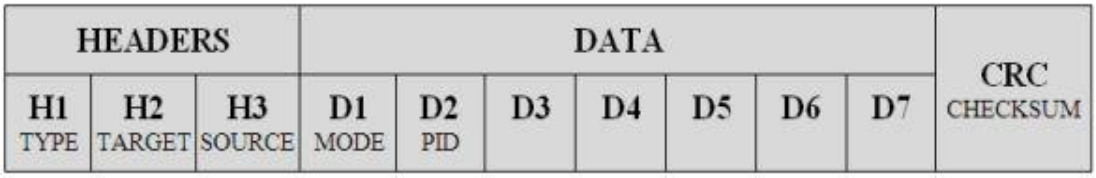


Figure 4.2 OBD Frame Common Structure

- The web service is responsible for maintaining an up-to-date database of error codes mapped to community-provided solutions. It is this database that the aforementioned mobile application references to communicate OBDII code solutions to the user. Through a Forum-like interface, the user community is able to contribute to the existing database of OBDII codes.

For example, instead of having a single base of users who interact exclusively with the mobile application, the CarDoc additionally provides a forum for more professionally oriented data providers (auto mechanics, automotive technicians, hobbyists, performance tuners, car enthusiasts, software developers, etc.) to aid one another and to share their automotive diagnostic knowledge.

The flexibility of the system ensures that people who do not necessarily have the expertise to contribute to the forum can still benefit from its aggregated information. Additionally, those people that do not carry a smart phone, or have no use for a simple, onboard diagnostic tool are still able to the vast database of knowledge that is provided by an active community of interested parties.

4.1 Analytical Implications

As has been previously illustrated in great detail, the real-time performance metrics are an integral part of the application. Because many of these metrics allow users to see how efficiently or aggressively they are driving their vehicles, these statistics can be integrated into new and existing social networks where participants can share and compete on the basis of vehicle performance and efficiency.

4.2 System Specifications

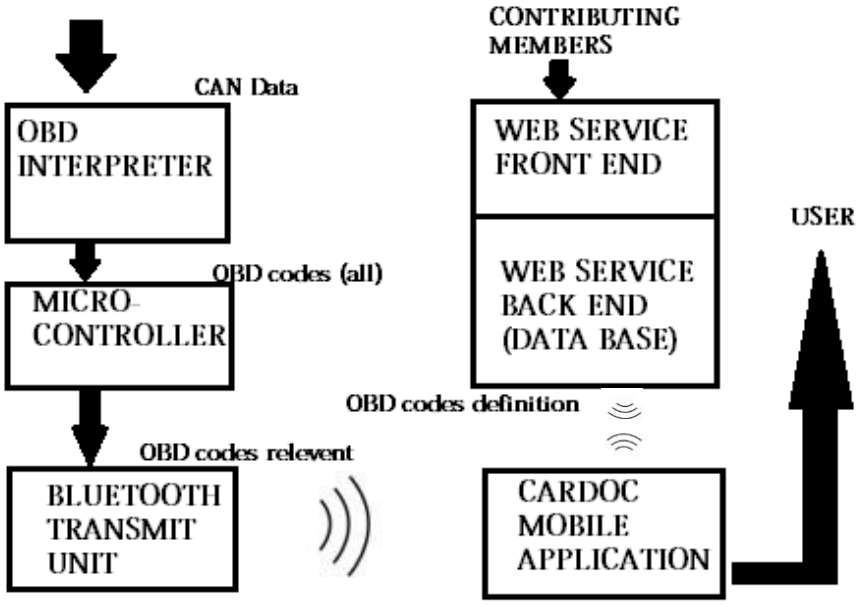


Figure 4.3 System Subcomponents

In addition to the specifications of the OBD Bluetooth adapter, it is worthwhile to define the basic content and structure of the communication between the mobile application and the remote server. Unlike the adapter component with utilizes a stable, high-speed 802.11

connection, the remote server component will be accessed primarily over 3G or EDGE cellular connections. These connections are guaranteed to be neither high-bandwidth nor reliable. Due to these characteristics, it is desirable, even pertinent, that the amount of information being transferred be kept to an absolute minimum. To resolve this issue, information being passed to the server as part of a request must be able to be represented as an HTTP query string. This will keep the upstream network requirements to an absolute minimum. On the downstream side, responses will be passed as pre-processed JSON files. One alternative was returning full HTML pages as they appear on the Cardoc's website. Although this solution is simple to implement it is expensive in terms of bandwidth. Additionally, by performing the data parsing routines on the server, it is possible to reduce the amount of work needed to do by the less-powerful mobile device.

Another goal of the system is to make data interpretation easy and effective. This necessitates an efficient and intuitive user interface. In order to make the interface as clean and accessible as possible it cannot possesses more than four distinct tabs. Furthermore, in order to keep the interface clean, organized and ensure that the data being presented to the user in an uncluttered manner, the graphical interface must not contain any information that the user will not understand. Likewise, it must not contain any more than five distinct interactive elements per page. The implications of this decision are twofold. First it will ensure that a user does not become confused. And second, it will ensure that unnecessary, overly ambitious components are not added during the development process.

The specifics for the web service component are more relaxed. Usability of the site must not depend on the user-specific variables such as browser choice.

Obviously, for the initial implementation of the project the scope of the user community will be limited. This also relaxes the requirements of the web service's back-end as well. A non-relational database will be utilized and it must be able to serve up data at a reasonable speed.

4.3 Choice of programming and Tools

4.3.1 Android SDK

Google provides Android Software Development Kit to aid developers in producing Android applications. It is consists of APIs (Application Programming Interfaces) that enable easy interaction with device's hardware (buttons, touch screen, compass, GPS etc.) and operating system. It also includes a plug-in for Eclipse IDE that enables easy debugging and simulation of the written applications. At the time of writing the report, the latest version was the Android 1.0 SDK release 2.

4.3.2 Eclipse IDE (JSP and Servlet)

It is widely used in industry and highly customisable, especially with the use of plug-ins (such as the one developed for Android). Java Server Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI). But JSP offer several advantages in comparison with the CGI.

Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having a separate CGI files.

JSP are always compiled before it's processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested. Java Server Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP etc.

JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

4.3.3 Maven

Maven is used as a build tool for the development of the web site. Maven promotes modular design of code. by making it simple to manage multiple projects it allows the design to be laid out into multiple logical parts, weaving these parts together through the use of dependency tracking in pom files.

4.3.4 Mongo DB

The OBD II diagnostics codes and the user details database is handled through the Mongo DB. Mongo DB is a document-based data model. The basic unit of storage is analogous to JSON, Python dictionaries, Ruby hashes, etc. This is a rich data structure capable of holding arrays and other documents. This is especially useful because the data is immutable. And MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.

4.4 Use Case Development

Projects with weak requirements can end up with scope creep, rework, delays, or even abandonment. Use case modeling is an approachable way to describe what the system will do. Because use cases are written from an outside perspective, they're easier to understand by the stakeholders who understand the problem you're solving. The team hello world developed a series of five main use case scenarios with the view of applying the main functionalities to the mobile and web applications. Identifying exceptions to a successful scenario early in the project saves a lot of time by finding subtle requirements. Detailed use case tables have been attached in the appendix A.

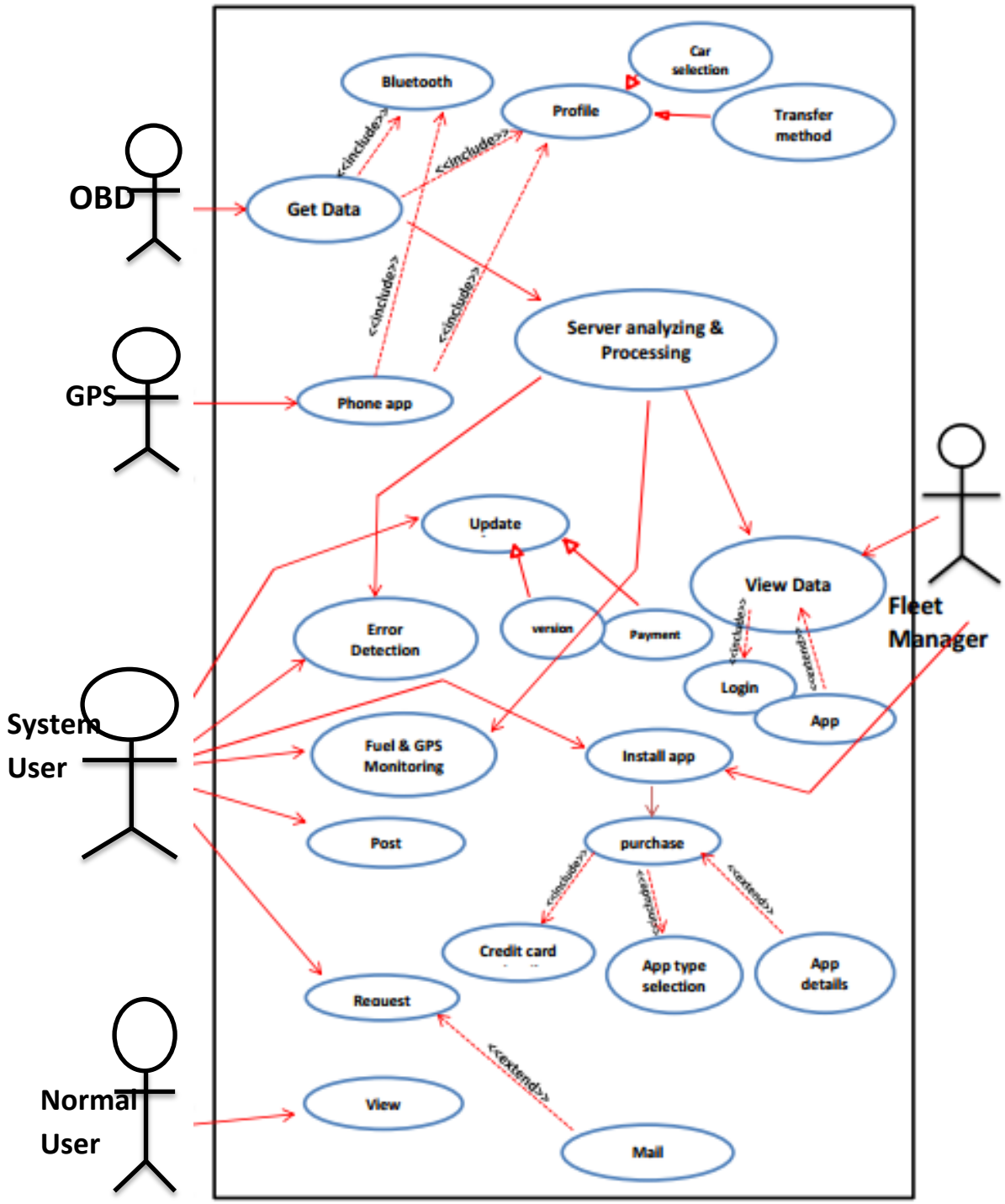


Figure 4.4 Use Case Diagram

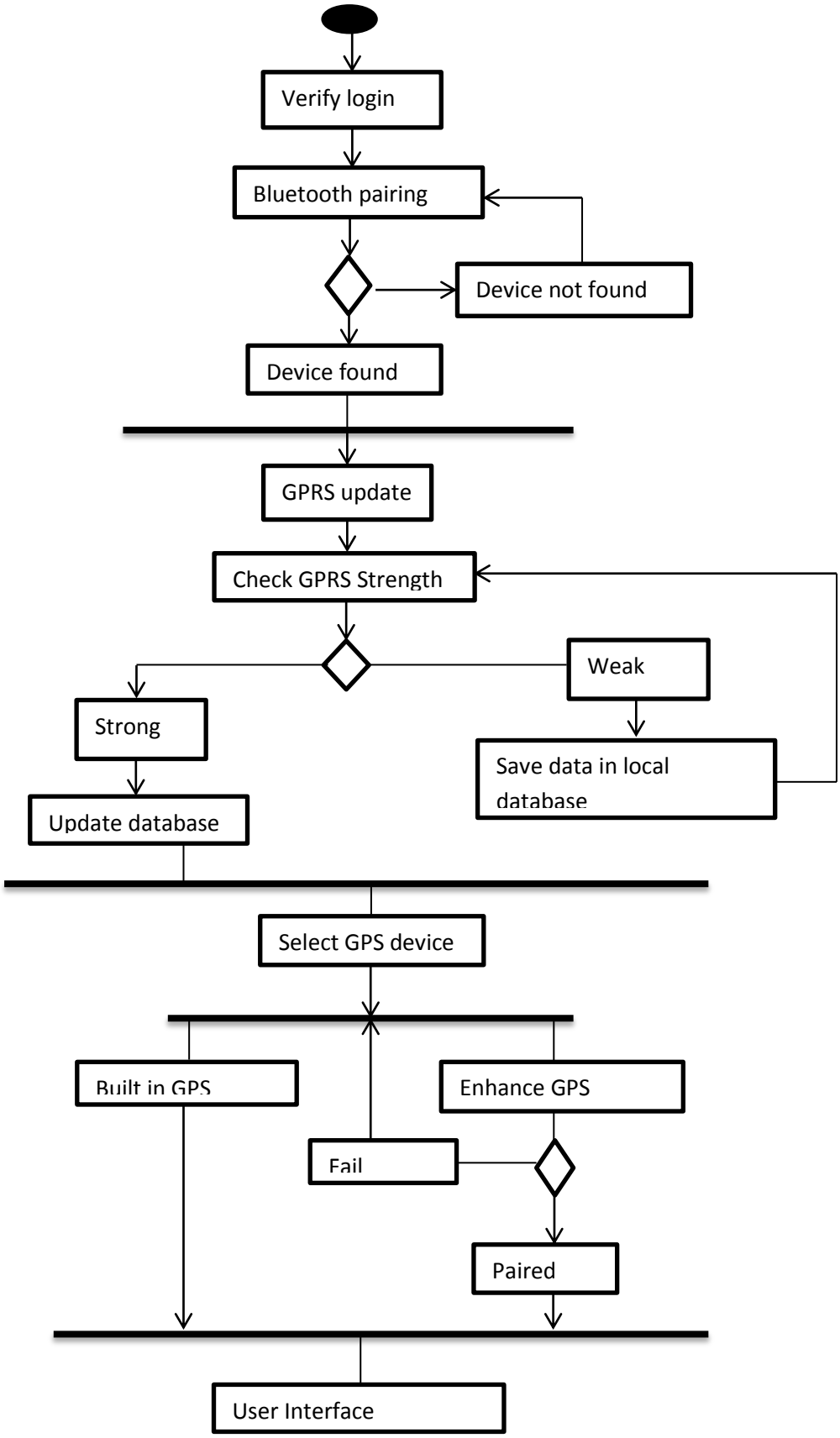


Figure 4.5 Activity Diagram

Database Design

SQL Lite has been used to maintain the database in the smart phone. SQL Lite is a relational database management system. Android provides several ways to store user and app data. SQLite is one way of storing user data. SQLite is a very light weight database which comes with Android OS. And Mongo DB has been used to utilize the system in the server backend. The OBD blue tooth adapter's constant dataflow will be mapped to the SQL lite's tables.

The Database was designed using the entity relationship model in the analysis part of the project. And normalization rules were applied to normalize the database.

Data Base Schema

Following figure 4.6 shows the database schema used in the application. It was based on a relational model

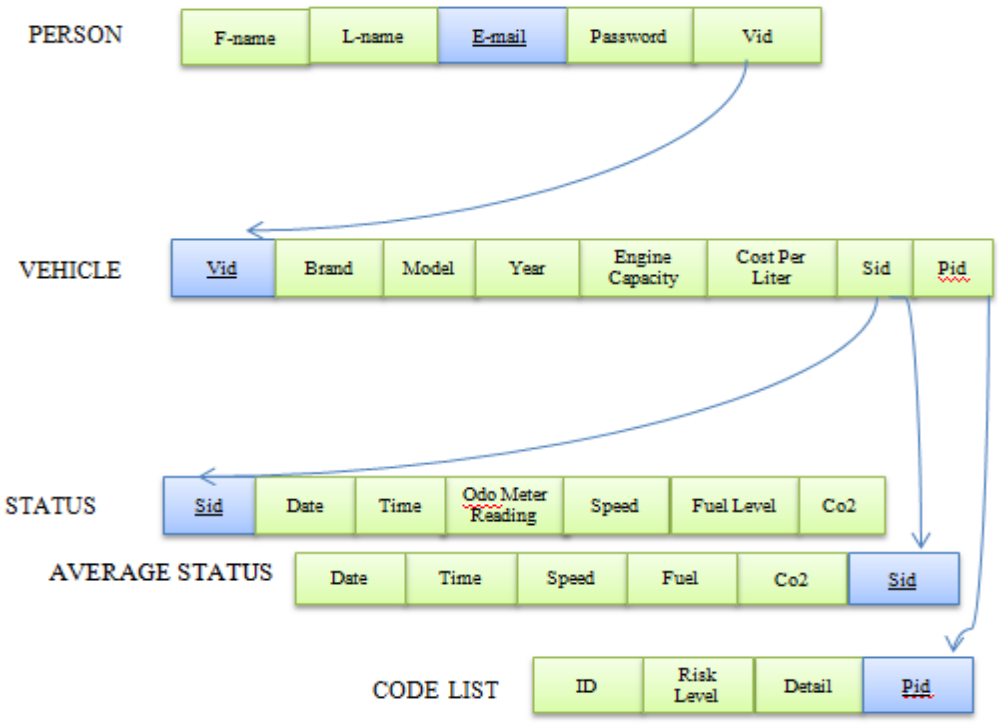


Figure 4.6 Database Schema

Figure 4.7 shows the ER diagram that has been used in constructing the relational database model for the database

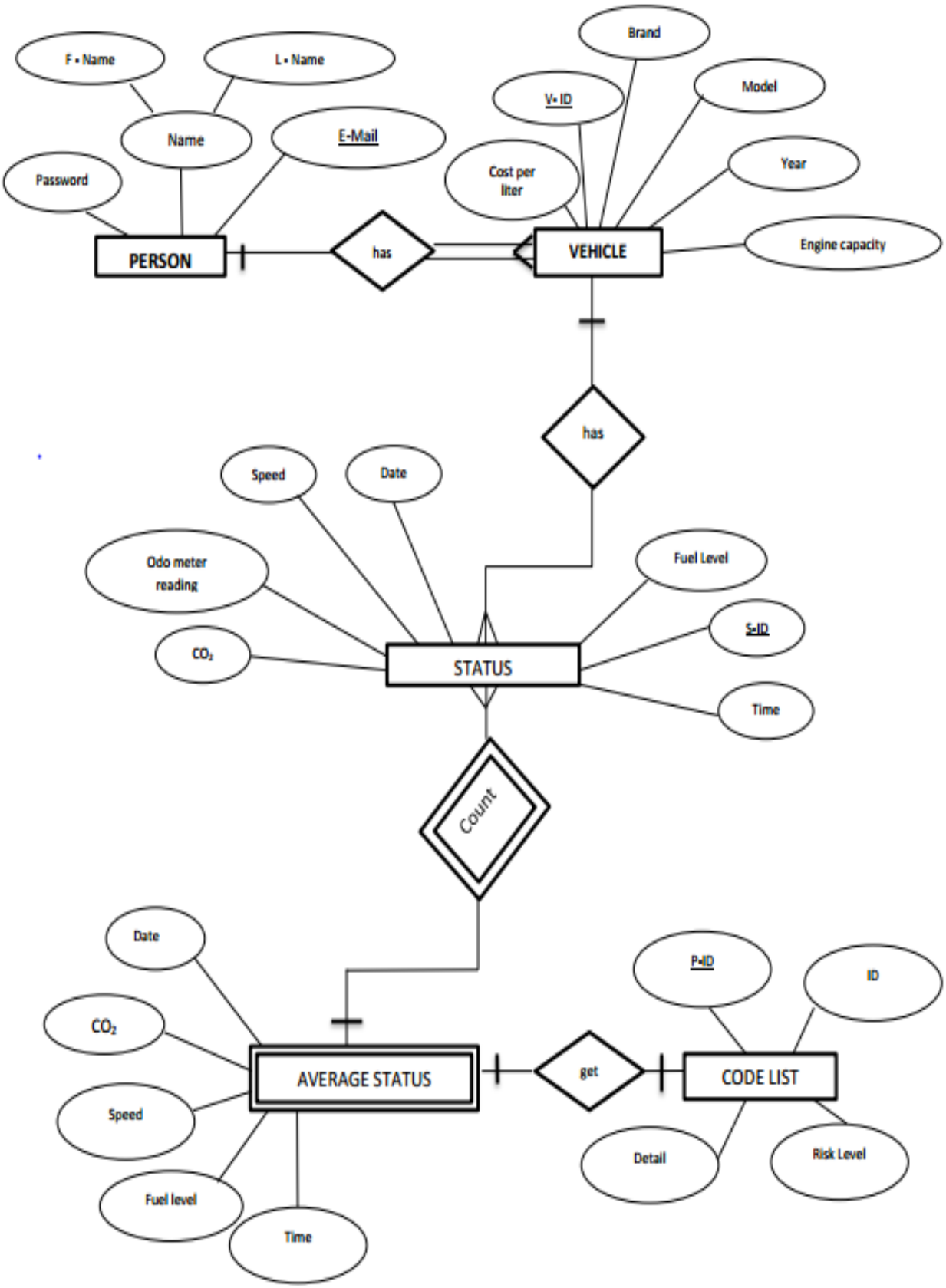


Figure 4.7 ER Diagram of the SQL Light

```
public class Contact {

        //private variables
        int _id;
        String _name;
        String _phone_number;

        // Empty constructor
        public Contact(){

        }
        // constructor
        public Contact(int id, String name, String _phone_number){
                this._id = id;
                this._name = name;
                this._phone_number = _phone_number;
        }
```

Figure 4.8 Codes for The Contact Class

Sample code for the contact class with all the getters and setters method to maintain the single contact as an object

Circular Logging with the SQL Lite.

Circular logging is a method of conserving the database space. It works by overwriting individual log files to keep the transactional log (the set of all log files) from expanding without limit on the database space. Circular logging is commonly used with Exchange native data protection, because in that mode, backups are not made so a detailed transactional log is not necessary.(The term "circular" arises from the fact that the set of log files starts to "rotate" once the disk space limit is reached, something like a LIFO (last-in, first-out queue.) Due to the magnitude of data that will be fed in to the system a circular logging was implemented to maintain the SQL Lite dB as to store only the data that will be used in the runtime analysis, and the RAW data for further analysis will be sent to the mongo DB via the REST API.

```
// Data Entry method to STATUS table
public long enterStatusData(int vID, String date, String time, String fuel,
        String co2, String odo, String speed, String torque, String gpsX,
        String gpsY, String PID) {
    // TODO Auto-generated method stub

    if (rowCount(STATUS_TABLE) > MAX_ROW_STA) {

        deleteFirstEntry_Status();
    }
    if (rowCount(STATUS_TABLE) > 0) {
        if (getdata_sta_last() % AVE_PER == 0) {

            enterAveStatusData();
        }
    }
    ContentValues cv = new ContentValues();

    cv.put(KEY_VID, vID);
    cv.put(KEY_DATE, date);
    cv.put(KEY_TIME, time);
    cv.put(KEY_FUEL, fuel);
    cv.put(KEY_CO2, co2);
```

Figure 4.9 Circular Logging Implementation

The CarDoc's website was utilized by using the mongoDB. MongoDB was chosen as the database because of its simplicity in handling big data analysis.

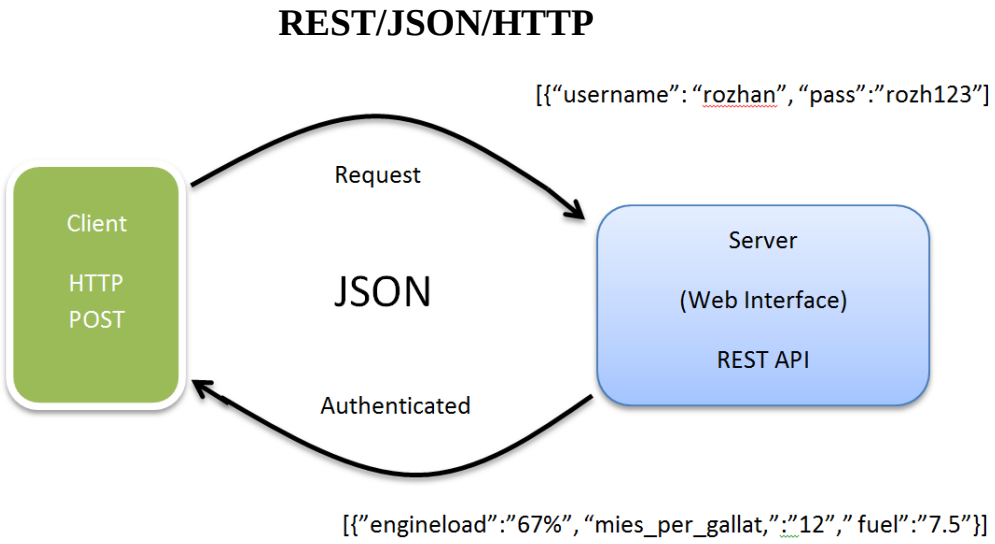


Figure 4.10 REST API and JSON

The REST API was used to send the raw data to the servlet. As a programming approach, REST is a lightweight alternative to Web Services and RPC.

Much like Web Services, a REST service is:

- Platform-independent
- Language-independent (C# can talk to Java, etc.),
- Standards-based (runs on top of HTTP), and
- Can easily be used in the presence of firewalls.

JSON format was chosen to send data over the REST API. The most well-known means of serializing data for transmission to and from applications at present is XML. Yet, XML is a rather cumbersome means of serialization. First, the sender must encode the data to be serialized based on a document type definition that the recipient understands. Doing so creates a great deal of extra padding around the actual data no matter which DTD is used. So, the size of XML documents is often fairly large in comparison with the actual set of values they contain.

Second, the recipient must receive the stream of XML and decode the data in order to then put that data into memory. In comparison, the serialization of data using JSON by the sender is relatively quick and compact because the structure of JSON reflects the structure of standard programming data types and the encoding mechanism adds
only the minimum number of characters required to indicate the structure and value of the data. Once the recipient receives the JSON serialized data, then, the only processing needing to be done is to evaluate the text of the string using either JavaScript's built-in eval function or a compatible function in another language.

```
try {
    HttpClient httpClient = new DefaultHttpClient();
    HttpPost request = new HttpPost(url);
    JSONObject object = new JSONObject();
    try {

        object.put("Fname", fname);
        object.put("Sname", sname);
        object.put("Email", email);
        object.put("Pword", password);
        object.put("Bday", bday);

    } catch (Exception ex) {

    }

    try {
    String message = object.toString();


    request.setEntity(new StringEntity(message, "UTF8"));
    request.setHeader("Content-type", "application/json");
        HttpResponse resp = httpClient.execute(request);
```

Figure 4.11 JSON Data Sent Over Rest API

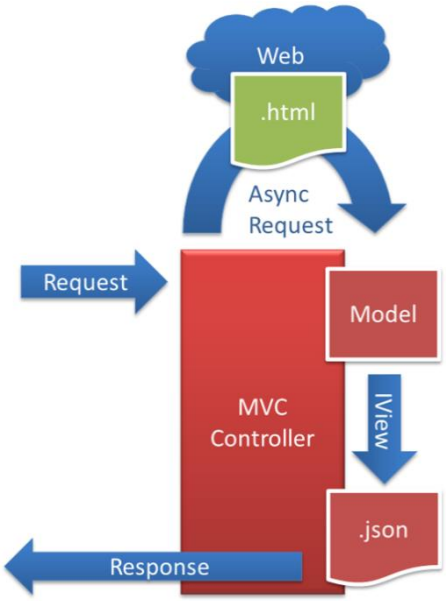
CarDoc Site Analysis.

The software has been designed to accommodate change. The high level architecture for this application is based on the Model-View-Controller (MVC) design pattern. The easiest way to make code overly complex is to put dependencies everywhere. Conversely, removing unnecessary dependencies makes delightful code that is less buggy and easier to maintain because it is reusable without modification. The primary advantage of the MVC design pattern is:

MVC makes model classes reusable without modification.

The purpose of the controller is to remove the view dependency from the model. By removing the view dependency from the model, the model code becomes delightful.

The MVC design pattern inserts a controller class between the view and the model to remove the model-view dependencies. With the dependencies removed, the model, and possibly the view, can be made reusable without modification. This makes implementing new features and maintenance a breeze. The users get stable software quickly, the company saves money,



The model represents the data, and does nothing else. The model does NOT depend on the controller or the view.

Figure 4.12 MVC Architecture

The view displays the model data, and sends user actions (e.g. button clicks) to the controller. The view can: be independent of both the model and the controller; or actually be the controller, and therefor depend on the model.

The controller provides model data to the view, and interprets user actions such as button clicks. The controller depends on the view and the model. In some cases, the controller and the view are the same object.

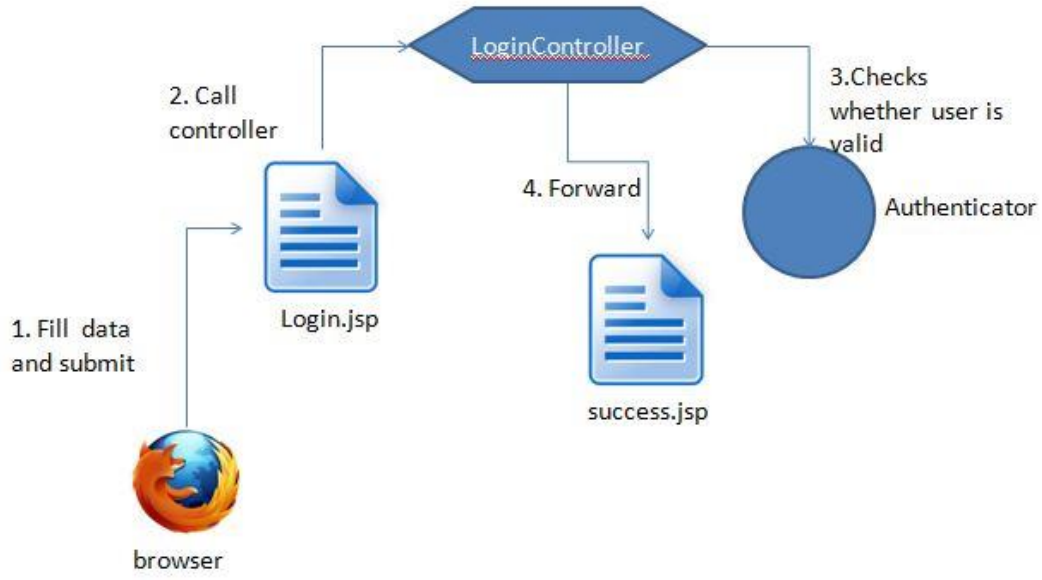


Figure 4.13 MVC Login Model

- First, user visits login.jsp page and fills out data and submits form.
- This causes LoginController to be invoked because of form action="LoginController" In LoginController, following code causes the model to be invoked and set the User properties.
    - Authenticator authenticator = new Authenticator();
    - String result = authenticator.authenticate(username, password);
    - and User = new User(username, password);
- rd.forward(request, response); causes the servlet to forward to jsp page.
- success.jsp page displays the username with expression language

In addition to the Model, Controller classes a web.xml have to be created. A web application's deployment descriptor describes the classes, resources and configuration of the application and how the web server uses them to serve web requests. When the web server receives a request for the application, it uses the deployment descriptor to map the URL of the request to the code that ought to handle the request.

The deployment descriptor is a file named web.xml. It resides in the app's WAR under the WEB-INF/ directory. The file is an XML file whose root element is <web-app>. Figure 4.14 Shows Servlet mapping function

```
<servlet>
  <description></description>
  <display-name>LoginController</display-name>
  <servlet-name>LoginController</servlet-name>
  <servlet-class>mvcdemo.controllers.LoginController</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>LoginController</servlet-name>
  <url-pattern>/LoginController</url-pattern>
</servlet-mapping>
```

Figure 4.14 web.xml Servlet Mapping

Asynchronous Task and UI Threading

AsyncTask enables proper and easy use of the UI thread. This class allows performing background operations and publishing results on the UI thread without having to manipulate threads and/or handlers.

AsyncTask is designed to be a helper class around Thread and Handler and does not constitute a generic threading framework. AsyncTasks should ideally be used for short operations (a few seconds at the most.)

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called onPreExecute, doInBackground, onProgressUpdate and onPostExecute. CarDoc implements asynchronous thread to send raw data to the mongo server in the meantime the UI thread represents the real time analysed data to the user.

```
public class HttpPostdata extends AsyncTask<String, Void, String>
{

    @Override
    protected String doInBackground(String... params)
    {
        BufferedReader inBuffer = null;
        String url = "http://10.0.2.2:8080/Test";
        String result = "fail";
        try {
            HttpClient httpClient = new DefaultHttpClient();
            HttpPost request = new HttpPost(url);
            List<NameValuePair> postParameters = new ArrayList<NameValuePair>();
            postParameters.add(new BasicNameValuePair("email", params[0]));
            postParameters.add(new BasicNameValuePair("asid", params[1]));
            postParameters.add(new BasicNameValuePair("vid", params[2]));
            postParameters.add(new BasicNameValuePair("date", params[3]));
            postParameters.add(new BasicNameValuePair("time", params[4]));
            postParameters.add(new BasicNameValuePair("fuel", params[5]));
            postParameters.add(new BasicNameValuePair("speed", params[6]));
            postParameters.add(new BasicNameValuePair("c02", params[7]));
            postParameters.add(new BasicNameValuePair("odo", params[8]));
            postParameters.add(new BasicNameValuePair("torque", params[9]));
            postParameters.add(new BasicNameValuePair("pid", params[10]));
            //postParameters.add(new BasicNameValuePair("co2", params[2]));
            UrlEncodedFormEntity formEntity = new UrlEncodedFormEntity(
                    postParameters);
            try {
                  Thread.sleep(5000);
                } catch (InterruptedException e) {
                  e.printStackTrace();
                }
            request.setEntity(formEntity);
            HttpResponse hr= httpClient.execute(request);

                    result=hr.toString();

        } catch(Exception e) {
```

Figure 4.15 Code for Asynchronous Thread

Chapter 5

5.0 Testing and Evaluation

5.1 Software Implementation

The main purpose of the software implementation was to create an Android application which established a wireless communication link via the OBD Bluetooth adapter. This will be achieved by using the Eclipse IDE, Android Software Development Kit (SDK), and Android Virtual Device (AVD).

Next application design consideration was how to get the desired data from the vehicle. The design of the ELM327 software operates by servicing a single command. This gives our software design a performance drawback due to the fact that we cannot get all of the vehicle parameters simultaneously. To make our application seem "real time", we need to obtain data as efficiently as possible. To do this, we created the data flow algorithm discussed below:

1.) Send first command string

2.) Wait until ">" is received, (which indicates that the ELM327 is ready)

3.) Send next command string

4.) Go back to step 2 and repeat

From the algorithm, constant data flow between the Android application and the ELM327 has been established. The next thing to consider is the method of parsing each message to obtain the desired data. Parsing of data proved to be one of the big challenges faced in this project as the data was not in a consistent format. Data was received with extra spaces, newlines, and null characters. Messages were also often split up randomly when read. To solve our parsing problem, as discussed in the Technical Challenges section, the use of Java String functions and regular expressions was used. More information about the parsing solution will be discussed later.

Data and Analysis

Two specific values that the CarDoc utilizes are the vehicle speed and the fuel consumption. In the North American automotive industry, the units for these are Miles per Hour (MPH) and Miles per Gallon (MPG), respectively. For the first, the ECU responds in Kilometers per Hour, which is easily converted to MPH

$$MPH = VSS * 0.621371 \longleftarrow \quad \textbf{5.1}$$

Estimating MPG is more involved. Efficient fuel consumption in an internal combustion engine requires a carefully adjusted amount of air. For gasoline fuel, the stoichiometric air-fuel mixture is approximately 14.7:1. This number also depends on the fuel used. Mixture less than a 14.7:1 ratio is considered to be a "rich" mixture, above that ratio is a "lean" mixture. The ECU spends almost 100% of its time maintaining that ratio at 14.7, which it can do quite accurately using a closed loop feedback system involving the O2 sensors. Based on these facts it is obvious that product is not going to look for the amount of fuel, but for the amount of air that is going through the intake system of the vehicle. Hence, the formula for fuel

consumption (Equation 2) involves the Mass Air Flow (MAF) sensor reading and the MPGs are calculated as:

$$MPG = \frac{14.7 \times 6.17 \times 454 \times VSS \times 0.621371}{3600 \times MAF} \longleftarrow \quad \textbf{5.2}$$

Stoichiometric air-fuel ratio - 14.7 grams of air to 1 gram of gasoline Density of gasoline - 6.17 pounds per gallon Vehicle speed in kilometers per hour - VSS Mass air flow rate in grams per second - MAF Conversion - 454 grams per pound Conversion - 0.621371 miles per hour/kilometers per hour Conversion - 3600 seconds per hour Having the two instantaneous readings the software interface is able to give the drivers feedback on their driving habits.

The reference data representation from the OBD adapter is modeled with a line equation, parabola and an exponential decay functions. It provides the relationship between speed and fuel consumption for three different ranges. The instantaneous vehicle speed is provided in increments of 1 MPH which will not alter the accuracy of the feedback.

From 5MPH to 40MPH

$$MPG = 0.521215 \times MPH + 8.457627 \quad \longleftarrow \quad \textbf{5.3}$$

From 40MPH to 69MPH

$$MPG = -0.0261039 \times MPH^2 + 2.791558 \times MPH - 41.181818 \quad \longleftarrow \quad \textbf{5.4}$$

From 69MPH to 150MPH

$$MPG = 61.870034e^{-0.011544 \times MPH} \quad \longleftarrow \quad \textbf{5.5}$$

5.2 Android Application

Getting Started with Bluetooth pairing

The Bluetooth Chat essentially allows devices to create a Bluetooth socket to communicate between itself and another Bluetooth enabled device. Once the Bluetooth communication link was achieved, the application can send PIDs to and receive data strings from the Kiwi device.

Before we are able to use the Bluetooth Chat sample code, a few modifications had to be made in order for it to run on our tablet. The first modification dealt with the AndroidManifest.xml file.

The next modification dealt with the Universally Unique Identifier (UUID) which is used by Bluetooth pairing to assign each connected device an ID. For our purposes of using Bluetooth pairing to connect to SPP devices, the UUID needed to be changed to a specific UUID used by all SPP devices which is shown in Figure 5.2

```
public ConnectThread(BluetoothDevice device, boolean secure) {
    mmDevice = device;
    BluetoothSocket tmp = null;
    mSocketType = secure ? "Secure" : "Insecure";
    final UUID SPP_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
```

Figure 5.1 A Specified UUID

5.3 Android Frontend

When opening up the application, the user must first connect to the OBD Bluetooth adapter by pressing on the button at the top right and selecting the device from the list .Once the application has successfully connected, the user can start the gathering of parameter data by pressing on the toggle Button. The second button is to clear check engine codes (if any are present in the vehicle).

To make it easier for the user to read the current data of the vehicle, Android text Views were also inserted into the gauges to provide a numerical representation. This allows the user to quickly identify the value of each current parameter. During the designing stage in the front end of the application, it was decided that the application would only have one screen to display the entire GUI. This was because implementing multiple screens would require the user to take his/her hands off the steering wheel, which could be potentially dangerous.

In terms of developing the front end of the application, there were two key elements to implement the GUI, creating the layout and creating the animations for the layout. In the OBD-II Android Monitor, the application GUI used Photoshop to create the background and gauges for the GUI.

The interface is kept simple. Four separate views (states) are designed.

1) Info View
2) Settings View
3) Real-Time View
4) Historic View

The android application was developed with the intention of crafting a clean and elegant user interface that was minimally distracting to the operator of a vehicle. Because this application was developed with the intention of being used while a vehicle was in motion, this conscious design decision forced the team Hello World to iterate through multiple GUI mock-ups.

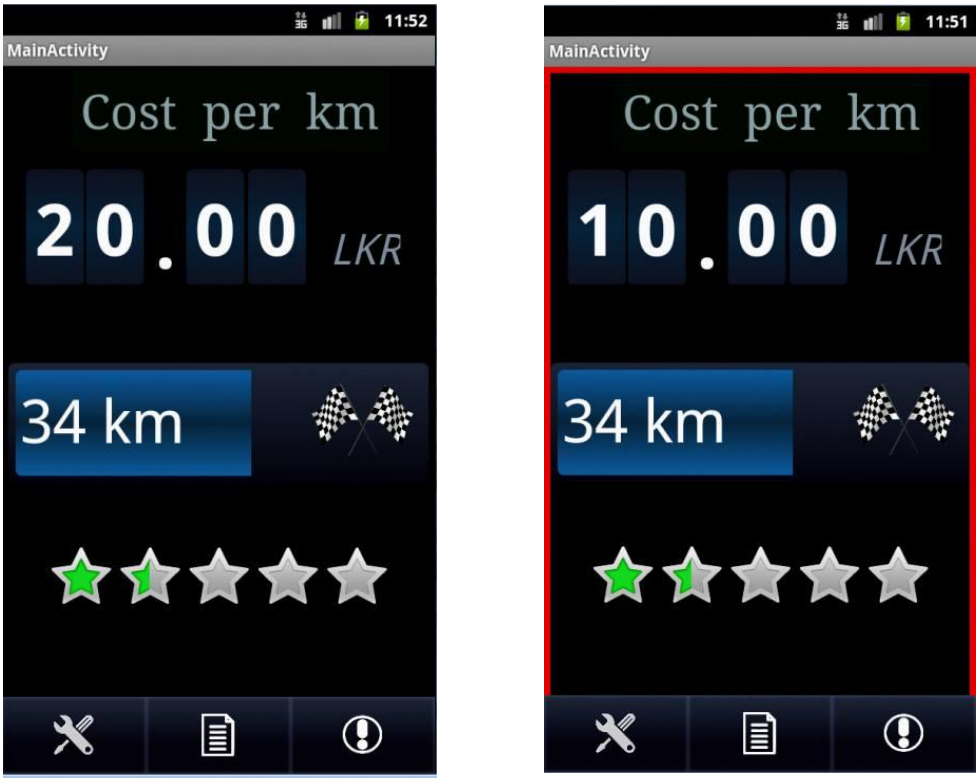


Figure 5.2 UI with ECU Error And UI Without Error

The above images shows how's the driver been alerted with a blinking red bordered interface when there is an ECU error and an interface without an ECU error. By navigating through the interface it will lead to the specific ECU error that has been retrieved by the application and to a crowd based solution.

5.4 Web Interface Testing

Used several Web interfaces in the user side and server side, when it comes to data analysis user need to have a simple interface to understand the usages of his vehicle and other details figure 5.4 shows on of the interface used in the Web site for data analysis part of the site

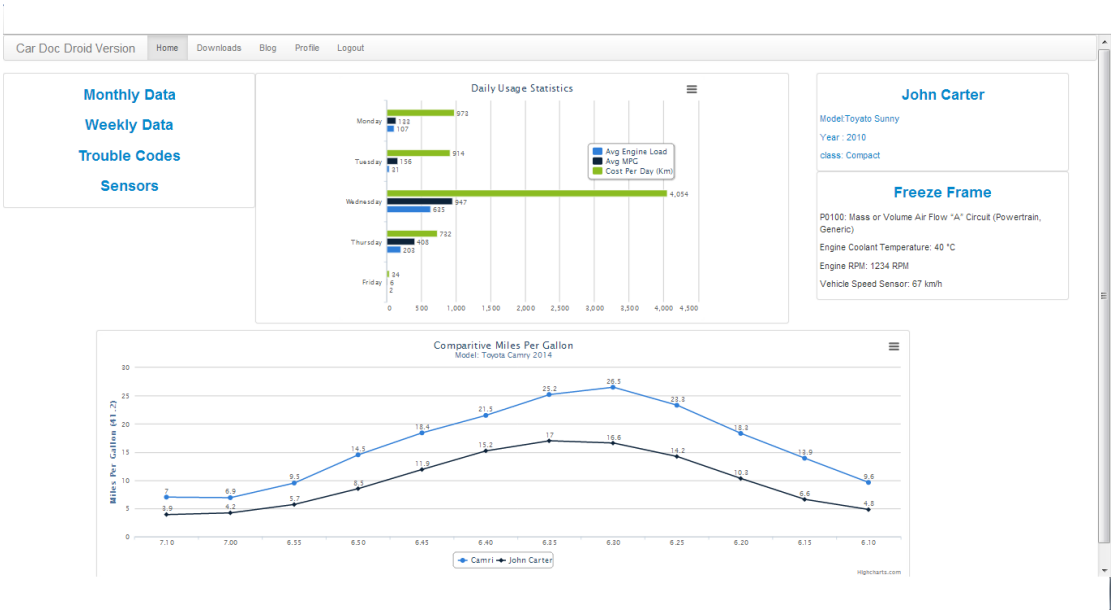


Figure 5.3 Snapshot Of The User Interface Data Analysis Page

Functionality and System Tests

The test cases in this category deal with testing CarDoc's functionality and ensuring the all the requirements work correctly. This involves testing every area of functionality that the end user can perform.

5.4.1 User Authentication and Session Tracking

The goal of testing the user authentication process is to verify security and ensure that authentication cannot be bypassed. Along with testing the user authentication is testing session tracking; for a user who had logged into the system, his authentication must remain valid throughout his interaction with the system; while at the same time, if the user has logged out, he should not be able to access system again through any method without logging in again.

For testing the user authentication, the login system will be tested against a set of valid and invalid user and password pairs. The expected behaviour is that only the authorized user would be able to login, and the invalid input pairs would be rejected. Other extreme cases would be tested, such as blank inputs.

To validate the input system would be resistant to SQL injection, partial SQL commands will be used as user and password pairs. SQL commands and statements will also be tested in all other user-accessible web forms to ensure there are no avenues for injection of malicious code or to compromise the project database.

For testing the session tracking, the session retention would be tested. Once a user has logged in, the user should stay logged in during a page refresh. Thus an authenticated user would not be kicked backed to the login page if he/she refreshed. For user who have logged out of the system, he/she must be kept out of the system. In addition, he/she must not be able to see secure information available before the log out. The main focus for the test would be on the browser history and page caching (e.g. Whether or not the page before the log out could be brought up again).

5.4.2 User Input Handling

Form validation checkers would be attached to the text input field; inputs for the field would only be accepted if it passed the validation. Thus all text inputs processed by the system would be regulated.

To verify the performance, text fields will be tested with a range of inputs. Malicious SQL code statements will be tested in user text entry fields, as well as incorrectly formatted values. The system will be tested to ensure that an appropriate error message is thrown and operation is resumed when unexpected and/or malicious inputs are attempted.

Table5.1 Test cases

No.	Description	Test Procedure	Expected Output
1.0	Register in to the system	Enter already existing password	Message to the user to change the username
1.1	Register in to the system	Null password	Display error message saying that please “enter valid password”
1.2	Register in to the system	Null vehicle Information	Error msg saying that enter information on vehicle
1.3	Register in to the system	Enter correct details	Create a new user profile
2.0	Log in to the user account	Null username Null password	Display error message
2.1	Log in to the user account	Not matching username and password	Display error message
2.2	Log in to the user account	Enter password and username correctly	Log in to the profile successfully

5.5 3G Connectivity Experiment Design

The OBD Bluetooth Adapter will gather information and transmit that data over a 3G or EDGE network to a server. To test this functionality, we will have to ensure the module can acquire, connect to, and transmit data over the 3G and Edge network.

Additionally, the application has to log data for later transmission when neither network is available. To prove this, the device will be loaded with files to transmit and be forced to lose connectivity from either network.

In the first test the application will be connected to the 3G or EDGE network .Then the application will start transmitting the OBD Diagnostics Codes. If the data is received by the server, the application has passed the test.

The third test will begin by loading the 3G connected device with a substantial log to transmit. During transmission, the device will be disconnected from the network and, after a significant delay, reconnected to it. If the server receives all of the preloaded logs, the application is successful.

5.6 System Initialization and Multi-Protocol Tests

Cardoc is designed to work with multiple vehicles and multiple OBD-II protocols. Tests categorised under this description are designed to ensure that Cardoc works with all protocols it can. However, availability of testing resources here was a problem so only a few cars could be tested.

5.7 Duration Tests

Tests in this category dealt with running Cardoc for extended periods of time in an effort to crash it. It is not unusual to see programs crash over time. For example, a memory leak or buffer overflow would become evident in these circumstances.

5.8 Issues with Getting Data from the ELM327

A big challenge in this project was to parse the incoming data sent by the ELM327. It created a challenge for us due to the inconsistency of format in the data. Examples of inconsistencies would be random added spaces, newlines, null characters, or split messages. The typical operation of the ELM327 is to send a command byte followed by a parameter ID (PID) such as "01 00" followed by a carriage return ('\r') byte.

For example, if we were to send the message to obtain intake temperature, we would send ("01 0F" + '\r') to the ELM327. And get a typical response back of:

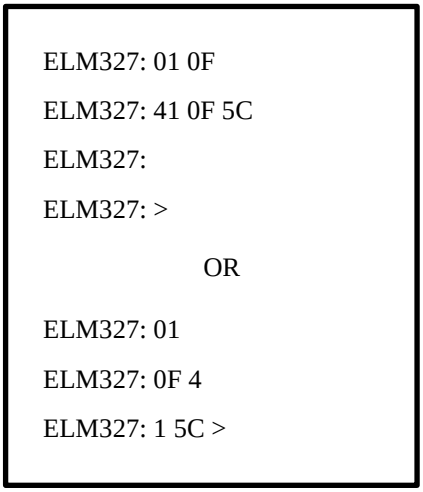


Figure 5.5 Example in ELM 327

From the sample output above, the desired data that we need is the "5C" for this example. As you can see, the "5C" doesn't appear in a predictable manner which makes this a challenge to obtain the data we need. Note that these are only two samples of the possible message format.

To solve this problem, the use of Regular Expressions and built in Java String functions were used. The Java String functions that were utilized were trim(), split(), and matches(). The String.trim() allowed for us to remove any extra padded spaces in front and back of a string message received from the ELM327. String.split() allowed for us to split up the received string message into message bytes using space (" ") as a delimiter. It was desired to split up the received message since the data that follows the PID, is the data we want to retrieve. String.matches() functions by using Regular Expressions to check with selected data. If the data matches the corresponding regular expression then it is parsed, otherwise it is just simply ignored.

The regular expression pattern used to match for MPH, intake temperature, engine load, and coolant temperature was "\s*[0-9A-Fa-f]{2} [0-9A-Fa-f]{2}\s*\r?\n?".

A drawback of using String.matches() and regular expressions is that it is a "hit or miss" scenario. This is due to the fact that the message received will either match or not match with the given regular expression. To address this, all message possibilities must be addressed and accounted for with regular expressions. This is essential to the performance of our application because the more matches obtained will give the user a real time response.

5.9 Strengths

The applications performance exceeded expectations. As the test results show, the application provides high to medium accuracy for all measurements tested. Also, the simple interface, convenience of the smartphone, and relatively low cost of the OBD-II reader makes this application a viable solution to monitor and alert drivers of unsafe and inefficient driving.

5.10 Weakness

While the application shows a considerable amount of potential, there are weaknesses. The identified weaknesses include:

1) Every reading available through the OBD-II standard is not available on every vehicle. This is not because the manufacturer is not compliant with the OBD-II standard, but due to the differences in vehicles. For example, some vehicles are not MAF equipped. The fuel consumption formula used in this application relies on this reading. Time and logic will need to be invested in determining which readings are available for a specific vehicle and new formulas may need to be developed.
2) Testing for this application was very limited and basic.
3) Usefulness was not tested. The question remains, “Can this application make one a safer or more efficient driver?”

Chapter 6

6.0 Conclusions & Further Work

6.1 Conclusions

1. Utilizes a previously developed OBDII adapter in order to wirelessly communicate vehicle information with a mobile device;
2. Allows an incentivized user community to view, upload and critique a database of descriptions and resolutions;
3. Provides access to numerous vehicle performance metrics and maintenance information including
    a. Instantaneous/Average miles per gallon
    b. Instantaneous/Average miles per hour
    c. Instantaneous/Average revolutions per minute
    d. Accurate percentage-based fuel levels
4. Stores personal vehicle information on the remote database including
    a. VIN
    b. Vehicle Year
    c. Vehicle Manufacturer
    d. Vehicle Model
5. Passively connects and reconnects to the vehicle, This implies that the application can be used by the vehicle operator with minimal distraction.
6. Creates a new and exciting platform for marketers and social networks in an environment where users were previously inaccessible.

6.2 Further Work

Given the high levels of access created by this platform, future iterations of this project can explore the varying degrees of vehicle control available via the OBD Bluetooth Adapter. The vehicle‘s OBDII systems are not strictly limited to the performance metrics and DTCs explored in this project. While different amounts of information will be more challenging to access given the closed nature of these systems, the level of vehicle control is nearly unbounded.

Additionally, the DTCs and PIDs explored in this project were typically generic. In order to build the most outstanding application, future implementations should be built to incorporate the more proprietary elements of OBDII systems that are manufacturer specific.

References

[1] Autoblog Green. (2010, August 4). Stud: One-Third of Drivers Don‘t Know What This Dashboard Light Means. Retrieved November 11, 2010 from, http://green.autoblog.com/2010/08/04/study-one-third-of-drivers-dont-know-what-thisdashboard-light/

[2] CBS News. (2010, March 3). Stop Overpaying for Everyday Items! Retrieved October 12,2010 from, http://www.cbsnews.com/stories/2010/03/03/earlyshow/living/money/main6262627.shtml

[3] International Standardization Organization. (2007, March 22). Road Vehicles – Diagnostics on Controller Area Networks (CAN). Retrieved October 12, 2010 from, http://www.iso.org/iso/catalogue_detail.htm?csnumber=33619/

[4] ScanTool.net, LLC, Retrieved October 12, 2010 from, http://www.scantool.net

[5] Rahul Mangharam. Autoplug: Common Automotive Interfaces for Plug-n-Play Servies, from, http://www.autoplug.org/docs/AutoPlug.pdf

[6] Chris Shunk. (2010, November 3). Audi Launches Car Monitor App for iPhone. Retrieved November 3, 2010 from, http://www.autoblog.com/2010/11/03/audi-launches-car-monitor-appfor- iphone/

[7] Jeff Glucker. (2010, November 2). GM and OnStar‘s Mobile Apps Now Available on iPhone and Android. Retrieved November 2, 2010 from, http://www.autoblog.com/2010/11/02/gm-and-onstars-mobile-apps-now-available-on-iphoneand-android

[8] Doug Newcomb. (2010, November 5). Tech Scene At The 2010 SEMA Show. Retrieved November 5, 2010 from, http://www.insideline.com/features/tech-scene-at-the-2010-semashow.html

[9] Chris Shunk. (2010, November 3). Audi Launches Car Monitor App for iPhone. Retrieved November 3, 2010 from, http://www.autoblog.com/2010/11/03/audi-launches-car-monitor-appfor-iphone/

[10] ScanTool.net, LLC, Retrieved October 27, 2010 from http://www.dtcsearch.com

[11] OBD-Codes.com, Retrieved October 27,2010 from, http://www.obd-codes.com

[12] All Data, LLC, Retrieved October 27, 2010 from, http://www.alldataiy.com

[13] YourAutoAdvisor.com, Retrieved October 27, 2010 from, http://www.yourautoadvisor.com

[14] Microchip Technology Inc., Retrieved December 16, 2010 from, http://www.microchip.com/en_US/family/pic32/

[15] Flower, Martin. (2006, July 18). GUI Architectures. Retrieved December 10, 2010 from, http://www.martinfowler.com/eaaDev/uiArchs.html

[16] Microchip Technology Inc., Retrieved December 16, 2010 from, http://www.microchip.com/en_US/family/pic32/

Appendixes A

Individual's Contribution to the Project

N.Selvanathan [114122R]

Each individual in a team have their own strengths and weaknesses. When learning teams are properly structured and everyone is contributing 100 percent, it can be an effective method of developing skills and sharpening existing ones. Within this paper, I will summarize my individual contribution to the team project and evaluate the effectiveness of my contribution to the success of the team project. Additionally, I will identify the areas where I believe additional training and more opportunities to practice would be helpful.

As the group leader, I thoroughly studied all the parts of the system such as the database, web application, analytical engine and web interface. I helped and advised the other members to study their respective research areas, and make their contributions to the project.

The software development process in this project is based on waterfall methodology which required finalising all the contents of the project at the preliminary stage itself. The team was headed to the designing process on daily basis in the early months of the development cycle. That includes the finalising of use cases, story board, interfaces and the flow. The team was headed to searching into the vehicle related information that would be necessary for the user. The designing of the webpage and the android application was initiated. The creation of algorithm to be presented to the user in a non-technical and informative manner was discussed, argued and finalised.

In the initial phases of the project I've learned GWT to implement the website; GWT is an open source set of tools that allows creating and maintaining complex JavaScript front-end applications in Java. With the requirements change we choose JSP for the web part. This helped me to learn to create dynamic web pages with java as the backend.

Our JSP pages were observers of our business model. The model would communicate the changes to the view this way (observer pattern) and the view would also call the model when appropriate. When we I saw it fit or requirements demanded the possibility to change the way the view called the model, I introduced a controller as a strategy for the view to communicate and interact with the model. In this manner I implemented the MVC architecture to the CarDoc site.

In the CarDocs web we needed to show off the historic analysed data in a dynamic way. So I learned jquery ajax so it can be called inside the js to retrieve the dynamic data, and during

this project I learned more about Non RDBMS, by implementing the MongoDb queries it helped me to understand the difference between RDBMS and the mongoDB.

Data analysis part is one of the hardest phases we faced when it comes for the analysing part because we have to obtain the automobile domain knowledge to analyse the retrieved data.

And along working with my team I was exposed to several techniques and I learned about the REST API, JSON parsing, and professional GUI modelling.

N.D.Shanthidewa [114127 L]

Once we got this project from hSenid Mobiles we arrange a group meeting among our group we discuss about how we achieve our goal .We divide whole work load with each other so I got Android App development part. It is the core part of our project. Because we retrieve data from OBD II port to Android App via Bluetooth with help of OBD II Bluetooth dongle .Then we have to sore it in local database. After that data need to send to our web server via HTTP post. Basically I have to learn few core features of Android Development .They are how to retrieve data from OBD dongle, Bluetooth communication, how to create local database and handle insert and retrieve part, how to work with HTTP request and post methods ,Async tasks and UI threads and custom interface designing in android.

First I decide to draw ER diagram of SQLite database with help of our group members and normalize it and also I draw some sketches about interfaces design .after that I decide look overview of Android App's classes .I used separate class for database and database queries .Because we have complex queries and calculation in our project.

I have some knowledge about MySQL ,but in in mobile App they use SQLite .I t's pretty much similar but develop procedure is different .Mr.Shalinda help me about clarifying doubt and give important basic concepts about SQLite database development. SQLite doesn't use user interface like PHP my admin. database also create by using separate class .There is no margin for error because if one syntax missing sometimes not indicate any compile even we use eclipse IDE .It will give exception error which is very hard to determine which part of code that error occurs .

For interface designing part we decide to go beyond the normal Android interfaces and we use custom design for input fields, buttons and text. So I have to learn how to customize design by using drawble XML files .sometime this task very tough for me but my group members always help me to overcome these situation. After interface designing .I added functionalities to these interfaces .Some interfaces are change instantly and some task need to perform as background tasks .So I have to learn work flow of UI threads it can do these kind of things. After that I have to learn how data has been sent to web server via HTTP. It need to happen as background task without disturb interfaces .I use Async task for that thing, because after Android API level 10 internet connection need to perform by using background tasks .I use async tasks because it can pass arrays as parameters to HTTP post .And also it can easily

handle loading bars and other waiting indications staff. So we use JSON format to send data to web server, because it's very much structural and secure way to send data to web server.

After All these things I tried to learn Bluetooth communication and communicated with OBD dongle which we use retrieve data to smart phone. It's very tough because It sends some ASCII values and retrieve ASCII values too. It need to encode to original data .Mr. Harshana help me lot for this job. This is very tough because there are no simulators and other methods for test these communication If it's there they are very expensive .So we have to test these things by using real vehicle.

All above things that I mentioned I don't done only by myself. My group members, our supervisor and Mr. Jestan Nirojan from hSenid mobiles always help me to complete this successfully.

D.M.R.P.Jayawardana [114060 A]

In this second year industry based project we team Hello world choose hSenid mobile solutions company as our partner. In the software development system we were basically in to waterfall method. This was not only a software based project because we had to work with OBD device which is completely based on hardware concepts like Bluetooth, vehicle error findings etc… team members were worked as a team in most occasions. I contributed in several parts of the project.

My main contribution to the project was developing the web site for the users. Firstly company told us to find best strategy to develop the site. They recommended us to do it using GWT (Google web tool kit) but we found that it is not supportive to add tables to the site. Our main purpose of the site is showing data by using graphs to the user of the vehicle so that they can understand the situations very clearly rather than using more complex formulas and other stuffs. Therefore we had to go to a new technology with that the facility to insert graphs and also it had to be dynamic one because it needs to update with the time. So we had to find the best technology to implement the site. At the end I had to create the web site using HTML according to the MVC (Model view control) architecture. We used Mongo BD for the database controlling part so we had to connect the web site and Mobile app in to the Mongo DB server, to that we used JASON flat form. In the android application we used SQL lite database we convert them to the Mongo DB with the help of JSON tool. Basically I involved

in developing the web site; my colleagues of the group helped me a lot in every occasion. And I also helped them with their parts.

Except the web part I help my colleagues to their tasks like database designs, android app and also we visited hSenid mobile solutions several times it was great experience to all of us. In those meeting I was the one who work as the communicator between them and group. At the end of the project I prepared the reports with my friends.
It was a great experience to work with very helpfully team. Got lot of experiences trough the developing process of this process. I was not aware of technologies like android, maven, JASON, MVC web architectures etc… but after the project I was able to get lot of knowledge about them by doing them with my friends. It was a great experience to work with such a team. At the end I would like to thank our supervisors and hSenid mobile for giving us big helping hand

S.Prasanna [114108 F]

The design of the architecture and the finalising process along with the use case was completed. The wireframe design for the website was finalised along with the team mates. The technical side of using a vehicle and the necessary details to create useful information to the user was identified. The initial search about the vehicle and its internal functions was studied. Then the search to figure out the functionality of the
OBD scanner was made. The way the device is connected inside the vehicle with the
ECU and the duty of it was thoroughly examined and shared with the team mates.

The types and use of the current OBD II standards were identified and the details that could be extracted from the OBD scanner was identified. The suitable OBD scanner device that will used to complete this project was searched and identified which should be cheap and compatible to our requirements to make it feasible to make it an appealing product and give a good marketable advantage. The data that should be extracted from the vehicles obd port and the format it will retrieve was analysed in order to make the database creation process easier. The equations that are used to make the necessary display parameters in the GUI were derived and some equations which were extracted from other pre used scenario were examined and comprehended. This was ensured to give the suitable output that is required for this project.

Modified the calculations to give a percentage based outcomes which would intern make the representation and further calculations such as the driver rating could be derived easily. The time period and the mechanism that the data will be retained in the database was discussed and finalised to be as collecting, accumulating the data, and flushing the raw data to allow room for new data. The decision to make the primary key to be the email of the user was made which would be transferred along all the communication process with the server and GUIs. The GUI's were designed in a way in the vehicle that the phone would not distract the driver to lose concentration by viewing the details. It was designed to give the message graphically as well as numerically.
The fleet management system was thought over and the GUI for that was designed along with colleagues to be simple and fully functional to meet administrative needs and a greater detail of real time fleet data. The methods in which the fleet will be controlled by the manager in adding and removing new drivers safely using a security key was designed. The details especially vital for fleet that could be extracted were identified such as vehicle load.

The study in GWT was made which uses java as front and back end. The search revealed that it emits java scripts for the java codes that is entered which made it hard even to integrate an image in the software.
The communication processes to check the possibility of getting to work with the reputed company such as Toyota were made.

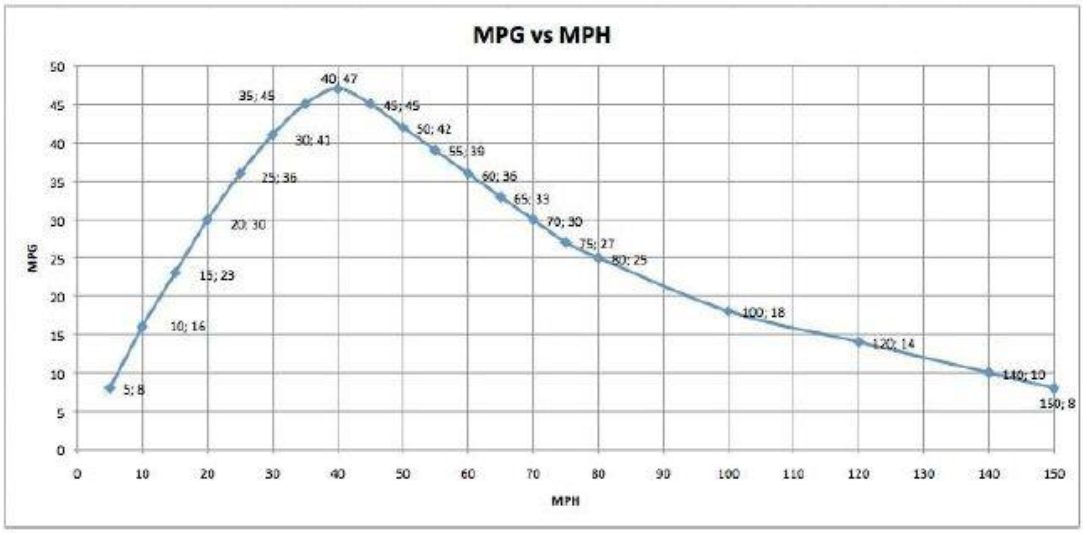
H.M.T.N.Herath [114168L]

When I join to our industry based project, I have to study hard for the project because we have to use more incompatible technologies. We endeavoured to learn GWT(Google web toolkit) but it was uncomfortable work for us because it is a new technology ,less number of resources for learning that technology. All of the team members tried for that we could learn it up to some extent. Other than that I got knowledge about MVC architecture and bootstrap which used for our website, JSON format it is the format that interchange data objects from mobile phone to Mongodb, lite SQL for mobile database and Mongodb for back end and Android. Other than that I got some knowledge about Maven, tomcat, OBD port of a vehicles and OBD II Bluetooth adapter.

We gathered user requirements from our client company. I prepared some use cases for the system and discussed about the architecture of system with team members and Created UML diagrams. We decided to use MVC for high level architecture of website development. Model, view and controller are working independently in this architecture.

I mainly worked on database part of the project. Lite SQL for mobile database, lite sql is a relational database and light weight. First I prepared ER diagram and relational schema. Database analysis, design and implementation were done. Calculated data sent as JSON object via REST API to the mongo DB. We determined that mongo dB is better to use for back end database by the reason of easiness to handle database with big data analysis. It is not a relational database. Created Cardoc database and collections for storing the email addresses, passwords of users and basic details about their vehicle such as vehicle model, brand etc. in their registration. Also created collection to keep updates from mobile phone. According to those updates website shows graphs of that details. Matching E-mail address and password in login done using JSP and servlet. It gives success and error massages.

Worked on user interface designing and Documentation of some parts of project reports and SRS was done. The selection of color schemes for interfaces that would suit both genders was analysed and finalised with the team mates.



Data Analysis Graph

(a) Correlation of MPG with MPH

Source: DRIVER'S EFFICIENCY ANALYZER by Electrical and Computer Engineering of Cornell University

Use case diagram 1

Use case ID	**Car Doc 1.0**
Use case name	**Downloading the application**
Actors	• Application user
Description	• Specifying the and downloading a relevant application
Trigger	• When the user access Google play store
Pre-conditions	• User should be equipped with a smart phone • Internet connection
Post condition	• User downloads the appropriate application
Normal flow	i. Access the Google play store ii. Choose the appropriate application from personal, fleet management and advanced versions. iii. Select the appropriate version among enterprise or normal version. iv. Login by entering E-mail address and name v. Downloading the rest of the application vi. Install the application to the smart phone
Alternative flows	
Exception	• If the E-mail address is invalid then give an error message to the user and let him re-enter the E-mail address
Frequency of use	• Once
Assumptions	• User has the ability to read and comprehend English • User has the ability and knowhow to install an application.
Notes and issues	Problems in the internet connection may prevail

Use case diagram 2

Use case ID	**Car Doc 1.1**
Use case name	**Bluetooth pairing**
Actors	• ODB device • User's smart phone
Description	• Connecting OBD device with user's smart phone
Trigger	• When user plugs in the OBD device to the vehicle
Pre-conditions	• User needs to have an OBD device • Application user needs to have a smart phone
Post condition	• User receives the conformation of paring of two devices
Normal flow	i. User plug in the OBD device to the vehicle ii. Identification of Bluetooth connection between smart phone and OBD device iii. Pairing the application with OBD device
Alternative flows	
Exception	• If the OBD device connected with other device connection conformation will not be received
Frequency of use	• Daily
Assumptions	• User has the ability to read and comprehend English • User has the ability pair devices via Bluetooth

Use case diagram 3

Use case ID	**Car Doc 1.4**
Use case name	**System diagnostic**
Actors	• Application user • Technician
Description	• Displaying the diagnostic details of the vehicle which occupied by the OBD
Trigger	• When the user request a diagnostic details or when there is an error detected by the ECU
Pre-conditions	• User need to have a smart phone • Application user need to have a car which is equipped with OBD port where the OBD scanner is mounted
Post condition	• User receives the diagnostic details in the web and Android platform • User will be notified the error the vehicle
Normal flow	i. Sending request for diagnostic details by user or detecting an error by OBD device ii. Sending error code to the phone via Bluetooth connection iii. Executing the error code iv. Displaying the diagnostic details
Alternative flows	
Exception	• If the application malfunctions or pairing gets interrupted, system should be restarted
Frequency of use	Daily
Assumptions	• User has the ability to read and comprehend English • User has the ability to operate the Android devices • ODB device readings are accurate.
Notes and issues	• Cannot identify the hardware issues of the OBD device

Use case diagram 4

Usecase ID	**Car Doc 1.3**
Use case name	Fleet Management
Actors	• Administrators • Fleet managers • User
Description	• Keeping constant updated track of the total fleet • Manages the vehicles of the fleet
Trigger	• When the user start the car
Pre-conditions	• Fleet manager should be linked with the smart phone • Strong GSM connection between the vehicle and the fleet manager
Post condition	• Every action of the vehicle can be identified by the fleet manager • Fleet manager will poses the ability to the determine the current state of the vehicle
Normal flow	vii. User plug in the OBD device to the vehicle viii. System start the function automatically when vehicle is started ix. System automatically connect with the server /fleet x. When then journey is completed total report about the journey should be submitted to the fleet xi. Fleet management database becomes updated
Alternative flows	
Exception	• If the application cannot connect to the fleet manager, it should try to find the a connection continuously
Frequency of use	• Always when the vehicle is on

Use case diagram 5

Use case ID	**Car Doc 1.3**
Use case name	**Fleet Management**
Actors	• Fleet managers
Description	• Live monitoring of the position and status of vehicles under the fleet • Manages the vehicles of the fleet • Makes vital decisions with additional precise details
Trigger	• When the user start the car and gets added to the fleet
Pre-conditions	• Fleet manager should be linked with internet and the application to run the fleet management system • Priority and authority to login, add and view the details of the fleet • Good internet connection to retrieve constant updates from the server of many vehicles that belong to the fleet
Post condition	• Vehicles position and status with better technical depth, could be monitored live. • Some vital decisions and even assumptions could be made with precise technical updates
Normal flow	i. User plug in the OBD device to the vehicle ii. User pairs with device and logs in form his end and sends verification code along with his username. iii. Fleet managers searches the driver by username and adds by the verification code xii. Server would sent a message to the driver that the enrolment process to the fleet is complete .
Alternative flows	
Exception	• If the application cannot connect to the fleet manager, it should try to find the a connection continuously
Frequency of use	• Always when the vehicle is on

hSenid Mobile
Enabling the Mobile World

Team Hello World Meeting Minutes

[24/01/2013]

Meeting was called to order at 3.30pm at the hSenid Mobile's discussion room.

Attendees Present

- Mr.Harsha Sanjeewa[CTO hSenid Mobile Solutions (Pvt) Ltd]
- Mr.Balathasan Sayanthan [COO at hSenid Mobile]
- Mr.Ruwan Abeykoon [Senior Architect at hSenid Mobile Solutions (Pvt) Ltd]

Group Members Present

- N.Selvanathan
- N.D.Shanthidewa
- D.M.R.P.Jayawardana
- S.Prasanna
- H.M.T.N.Herath

Topics of Discussion

- The first meeting was held with hSenid mobile CTO Mr.Harsha Sanjeewa, Software Architect Mr. Balathasan Sayanthan and Software Engineer Mr.Ruwan Abeykoon.
- The project objective and the final outcome which was expected from us was briefed by them.
- A little insight to the market trends was also discussed.
- The technical obstacles that are to be expected was briefed and discussed with Mr.Ruwan Abeykoon.

- A person in hSenid who will be in-charge of the project was told to be assigned for the team to work the way through.
- The phases that are to be completed were explained and we were asked to prepare the steps to achieve this by starting with the design.
- The company advised us to use the program Xmind to put out the idea in where the workings and interconnections could be clearly identified.
- The request to make a Facebook group in order to share the work, doubts and progress also was made by the company.

Actions to be taken

- To start develop the Software requirements specification (SRS) so it can help to define the format and content of their specific software requirements and developing an SRS model provides a specification process that is unambiguous and delivers a complete specification document for the project.
- To create a social media group so that the company and the team hello world can share the documents and comments regarding the work that have been done.
- To Brainstorm and create a mind map and publish the documents in the social media group.

Meeting adjourned at 4.30 pm

Team Hello World Meeting Minutes

[31/01/2013]

Meeting was called to order at 3.30pm at the hSenid Mobile's discussion room.

Attendees Present

- Mr.Harsha Sanjeewa [CTO hSenid Mobile Solutions (Pvt) Ltd]
- Mr. Balathasan Sayanthan [COO at hSenid Mobile]
- Mr.Ruwan Abeykoon [Senior Architect at hSenid Mobile Solutions (Pvt) Ltd]
- Mr.Jestan Nirojan [Software Engineer at hSenid Mobile Solutions (Pvt) Ltd]

Group Members Present

- N.Selvanathan
- N.D.Shanthidewa
- D.M.R.P.Jayawardana
- S.Prasanna
- H.M.T.N.Herath

Topics of Discussion

- The use of programming language to code the project was discussed. The advantage and the non-complex nature of Clojure above Java during the progress were explained.
- The language of preference to continue to work with was chosen as java.
- The architect advised over the use case diagrams and the advice to have a story board so we would be clear in what should be the next step at every instance.
- The first sample GUI's for the web was shared with the company and they extended their views. The need of a image in the home page that would be attractive as well as

serving the purpose of giving the meaning and purpose of the site was raised and pointed out to work on.

- The phases of the projects were discussed in details. The individual user should be able to add as well as the fleet manager. A map to show the geographical location of the person would be ideal.
- A use of the an additional GPS device was argued with on its pros and cons. Then the decision was reached to have separate GPS that could be paired with the mobile phone for special purposes like using in tight geographical area such as a harbor.
- The idea of using the Facebook user login credentials to the application was rejected and decided otherwise.

Actions to be taken

- To refer the documents which are posted in the social media group and come up with a professional and user friendly User interface for the CarDoc's mobile front end and Website.

Meeting adjourned at 5.30 pm

Team Hello World Meeting Minutes

[14/02/2013]

Meeting was called to order at 4.00pm at the hSenid Mobile's discussion room.

Attendees Present

- Mr.Harsha Sanjeewa[CTO hSenid Mobile Solutions (Pvt) Ltd]
- Mr.Ruwan Abeykoon[Senior Architect at hSenid Mobile Solutions (Pvt) Ltd]
- Mr. Jestan Nirojan [Software Engineer at hSenid Mobile Solutions (Pvt) Ltd]

Group Members Present

- N.Selvanathan
- N.D.Shanthidewa
- D.M.R.P.Jayawardana
- S.Prasanna
- H.M.T.N.Herath

Topics of Discussion

- A GIT repository was created in the company for us to upload the code and a VPN was also created which enables a person to connect to the company server one at a time.
- State diagrams and the story board and comments ideas were exchanged. The architect indicated that the story board is better reference for the team.
- The way that electronic devices communicate (signaling and timing) was briefed to the team.
- A briefing on the objects(XML and JSON) that are used to pass data through various protocols such as SOAP and REST took place.
- The concept of marshaling and de-marshaling was described.

- A detailed session on working with GIT repository was done by the software engineer.
- The need to by the OBD dongle was raised.
- The company asked the team to find the suitable dongle that would be compatible with our work.
- The discussion over the reworked of GUI received the comment from the company as it looked more like a game than an application for a daily user. It required the more professional look at the glance.

Actions to be taken

- To do a research on the OBD Bluetooth adapters those are available in the current market and select a OBD adapter that will suite the project requirements.

Meeting adjourned at 6.00 pm

hSenid Mobile
Enabling the Mobile World

Team Hello World Meeting Minutes

[28/02/2013]

Meeting was called to order at 4.00pm at the hSenid Mobile's discussion room.

Attendees Present

- Mr.Harsha Sanjeewa[CTO hSenid Mobile Solutions (Pvt) Ltd]
- Mr.Ruwan Abeykoon [Senior Architect at hSenid Mobile Solutions (Pvt) Ltd]
- Mr. Jestan Nirojan [Software Engineer at hSenid Mobile Solutions (Pvt) Ltd]

Group Members Present

- N.Selvanathan
- N.D.Shanthidewa
- D.M.R.P.Jayawardana
- S.Prasanna
- H.M.T.N.Herath

Topics of Discussion

- The methods to test the program once completed was discussed.
- A suggestion and discussion was also made on making our own emulator with the necessary IC the ELM 327 which takes the signal from the OBD which comes out of CAN protocol and translates into serial data that could be relayed through Bluetooth connection to any device.
- The cost analysis and the non-availability of the IC locally restricted us from making our own emulator.
- The readily available emulators were very expensive for project of this nature.

- An introductory session on MongoDB was done for the team by the company in order to continue working by integrating it to our project.
- The Google Web Toolkit was introduced which meant that the back end and the front and could be worked in Java itself.
- Its complex nature was highlighted and the company provided us some links to go through.
- The sites interface was designed using a content manager system with attractive graphs to suite our display requirements in a very graphical manner.
- The need of attracting the viewer and providing the a better picture of the site could be achieved by the graphical nature of the display.
- The details which would be displayed in the site will be the same as the display parameters in the phone but with additional ability to view it in different time constraints over the past the application has been used.
- The idea to run the MongoDB on the free hosting site Heroku was given to us and the further discussion of how to set it up by pushing the code through GIT would deploy and run it was explained.
- The fact that the OBD dongle that we required to pull the data out of the vehicle via Bluetooth was ordered and was being processed to hand it over to the team.

Meeting adjourned at 6.00 pm

hSenid Mobile
Enabling the Mobile World

Team Hello World Meeting Minutes

[30/09/2013]

Meeting was called to order at 3.30pm at the hSenid Mobile's discussion room.

Attendees Present

- Mr. Jestan Nirojan [Software Engineer at hSenid Mobile Solutions (Pvt) Ltd]

Group Members Present

- N.Selvanathan
- N.D.Shanthidewa
- D.M.R.P.Jayawardana
- S.Prasanna
- H.M.T.N.Herath

Topics of Discussion

- The dongle was received.
- The ways of testing the inter connectivity of the parts of the whole project was discussed.
- An effort make a local network to run a sample testing by passing the data from the phone via http post was made by setting up a router during the visit.
- Then the idea was pitched by the company to use an ordinary telecom router to setup a local network that would enable the phone to interconnect with a PC that will used as the server during the demonstration.
- The method that we are about to use for the demonstration was discussed. The way the application would be provided to the users was also briefly discussed.

Meeting adjourned at 5.30 pm

Team Hello World Meeting Minutes with Faculty Supervisors.

[18/01/2013]

- The introduction of the team members and the company details were shared
- The discussions about the interview process of hSenid were briefed to the supervisor.
- The general idea of where to start the project and what is expected from us as a development team which was discussed with hSenid at the interview was shared.
- The initial meeting date was fixed with the company through the supervisor.
- The manner in the team should respond and behave was discussed
- The framework of communication was established.

[25/01/2013]

- The initial project idea that was pitched by the company was discussed in detail.
- The unique hardware portion that the team should tackle was explained to the supervisor as it was mentioned by the company.
- The roles of the members and the duration was discussed
- The project objectives were identified
- The technologies that should be looked into were briefed by the supervisor.

[01/02/2013]

- The discussion about the programming language to be adopted for the project was decided as java
- Though the advantages of using Clojure over Java were briefed by the company it was decided to continue with java.
- The story board required by the Architect of the company was discussed with the supervisor
- The project phases were analyzed in depth
- The discussion on the workshop arranged by our supervisor in the Faculty was made.
- The login credential which was suggested by the Software Architect of the company, which was to have a separated username and password specific to the application rather than using Facebook ID was discussed.

[15/02/2013]

- The corrections on the state diagrams that were produced at the company was reworked
- The methods of the data communication by passing XML objects was discussed
- The protocol of the OBD port was briefed to us.
- The way the CAN bus functions, was discussed overall.
- The assistance of Bluetooth pairing codes were received from the supervisor
- The request for the need of a vehicle to test the system was made.
- The use of GIT repository, which was taught us by the company in during the visit, was discussed.
- Discussion over some of the sample GUI of the mobile application indicated that it was less professional and more towards a game GUI.

[01/03/2013]

- The suggestion to visit Toyota for assistance was made by the supervisor
- The discussions of the algorithms and calculations were briefed
- The accuracy of the GPS system in mobile phones were questioned and analyzed.
- The use of non SQL database MongoDB which was taught at the company was discussed.
- The thought to make our own OBD hardware based emulator's feasibility and time constrain was discussed.
- The complexity of using GWT to design the front end was discussed. Our lack of know how that new technology was highlighted to the supervisor.
- The less weighted nature in the Server side programme was highlighted.

[04/05/2013]

- The discussions over the interim presentations were made.
- Supervisor specified to highlight the architectural structure of the programme during the presentation.
- An architectural analysis was made to clarify the reason for choosing Spring MVC over the other architectures.
- The data that would be extracted from the OBD port of a vehicle was explained.
- The thought to use an external GPS device for the Fleet management system without using mobile phone was discussed for various geographical scenarios that would arise in real world fleet management systems. Eg : harbor : fleet would need very accurate devices.
- The duration of presentation and the contents of the demonstration was discussed.

[27/09/2013]

- The discussion over the website GUI made from a content management system made
- clear the sites cover page should be very attractive and should posses the capability of
- providing the picture of the application at a glance.

- The fact that the display parameters would be the same as what it is in the mobile application in a very attractive and self-explanatory manner was shared.

- The advice to run MongoDB in the Heroku site which was the solution given to our team by the company to solve the issue of the necessity of a MongoDB server was expressed.

- The method or the way of presentation which was discussed with the company was also re captured.

[02/10/2013]

- The discussion mainly focused on the methods of the connectivity of the devices in the project.

- Supervisor suggested to make the application connect with another phone or similar type of a functional device like the OBD dongle via Bluetooth to make sure the connectivity is flawless.

- The previous check would ensure the Bluetooth paring would work regardless of the type of device so that the communication protocols could be checked.

[20/10/2013]

- The current work in progress and the level of completion were discussed.

- The main obstacle of getting a real vehicle to check the developed systems functionality was highlighted.

- The attempts made to reach Toyota Lanka Private Limited for the above mentioned purposes was briefed.

- Supervisor requested the report prior to submission.