Vehicle Management System

By

Maj Imran Khan Niazi

Maj Mohsin Riaz

Maj Uzair Nawaz Khan

Submitted to the Faculty of Computing Software Engineering

National University of Sciences and Technology, Islamabad

ABSTRACT

Vehicle Management System

Globalized era and the advancement of technology made human life easier than it was. The more complex the problem is, the more efficient solution has been proposed. If we have to look in the world, we came across some facts that directly or indirectly affect human beings. In the past the systems were manual, but in this era the systems are being automated so the efforts of the humans are being sustained.

Vehicle Management System (VMS) is a system that that is proposed to develop a system that will automate the management aspects of a vehicle and will help keep track for scheduled maintenance activities.

This Software Design document is meant to outline design features of VMS, so as to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other. Following are the main users that would interact with the system directly or indirectly

CERTIFICATE FOR CORRECTNESS AND APPROVAL

Certified that work contained in the thesis – Vehicle Management System(VMS) carried out by Maj Imran Khan Niazi, Maj Mohsin Riaz and Maj Uzair Nawaz under supervision of Lt Col Dr. Adil Masood for partial fulfillment of Degree of Bachelor of Software Engineering is correct and approved.

Approved by

Lt Col Dr. Adil Masood

CSE DEPARTMENT

MCS

DATED___________________

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our families, without whose unflinching support and unstinting cooperation,

a work of this magnitude would not have been possible

ACKNOWLEDGEMENTS

Table of Contents

Introduction

Overview

The application should inform the driver about periodic maintenance points for the vehicle, changing of Engine Oil, Transmission Oil, Air Filter, Fuel filter etc..

Problem Statement

The application of vehicle service and maintenance has a significant role in informing the user that the vehicle is due for service at regular intervals to ensure prime operating condition of the car. Most vehicle users are careless in paying enough attention to the maintenance schedule and obeying servicing due date. The on-time maintenance system can decrease the service time and prevent scar malfunction of the engine other parts. This is why it is important for car owners and users to pay attention to the service maintenance schedule in a timely manner. At a corporate level, insufficient attention to the proper servicing has direct influence on the financial factors of vehicle fleet and logistical operation of their business. Otherwise, it may lead to both unnecessary repair cost and loss of revenue respectively. The problem is that there is no effective indicator or warning maintenance system in most vehicles in terms of a signal to remind drivers that the mileage of their vehicles is approaching time for service.

1.1 Approach

The application should collect driving distance information from the integrated phone sensors and keep logging the driving activity inside an internally integrated database

Scope

The proposed project is designing an iOS vehicle management application for iPhones. The app should make full use of the location services feature of iOS Software Development Kit through iPhone sensors like Global Positioning System Receiver, 3 Axis Gyroscope and Accelerometer. The app should essentially relieve the car drivers from manually entering car management key data and make use of the above-mentioned sensors instead.

The application should collect driving distance information from the integrated iPhone sensors and keep logging the driving activity inside an internally integrated database. The driver should provide initial mileage as an input when the app is downloaded and installed and thereafter, the driver must add refueling checkpoint data manually.

The application should inform the driver about periodic maintenance points for the vehicle, changing of Engine Oil, Transmission Oil, Air Filter, Fuel filter.

1.5 Objectives

During the course of this project, all the aspects of software engineering will be covered i.e Requirement gathering, software design, implementation and testing along with documentation
(SRS, SDS, Test Document, Final Report and User manual). Students are also expected to
develop sound knowledge and technical skills in the following fields:

☐ Programming

☐ Artificial Intelligence

☐ Signal Processing

After these comprehensive documentations and discussion, the project can serve to become a basis for
further development in vehicle management system. Further research can be made to automate the tracking of air pressure in tires, performance of shocks and many other mechanical parts.

Deliverables

Sr.	**Tasks**	**Deliverables**
1	Literature Review	Literature Survey
2	Requirements Gathering	SRS Document
3	Application Design	Design Document (SDS)
4	Implementation	Implementation on computer with a live test to show the accuracy and ability of the project
5	Testing	Evaluation plan and test document
6	Training	Deployment Plan
7	Deployment	Complete application along with necessary documentation

Chapter 2: Literature Review

Recently, some technologies have been changed fromtraditional calendar based maintenance to e-maintenance via interaction through the use of data communication technology, such as the Electronic Product Code (EPC) and RFID. For example, Hiraoka et al. display the method for a sector agent that made a recommendation on part maintenance based on the gathered data from historical data and internet . In addition, Djurdjanovic and Lee used multi sensor machine estimation and forecast to realize predictive vehicle maintenance condition by reducing assessment of reading multiple sensors to examine significant properties of the process or machinery in a networked and tether-free environment. Moreover eethichandra (2000) made multifunctional sensor for measuring viscosity, temperature, capacitance and cleanness of engine oil to make an evident choice on their maintenance state [4]. Mazlan et al. (2008) specified the in-car maintenance system based on receiving satellite signals to compare the mileage data with database for forecasting the service time. This system used GPS toolkit software to measure the speed of the vehicle in Km/h format to take out the distance travel information and inform the user for next service time. Kirchgessner et al. proposed the Wireless ECU Monitoring System (WEMS) used to monitor and manage engine features as engine RPM, oxygen sensor and air flow rate in order to optimize engine functionality. WEMS is able to provide access to ECU data and engine condition to track the maintenance states via a home PC, that is limitation of supported range for this system.

Available applications for vehicle management

1. Car Minder Plus - Car Maintenance and Gas Log (MPG) By Josh Monroe

Car Minder is an application for managing all your car maintenance needs, logging repairs and tracking fuel economy. Keep track of multiple cars, services and repairs. Keep track of oil changes, tire rotations, and other maintenance tasks for all your cars. Never have to wonder again when your car needs its next oil change or the last time you replaced those squeaky windshield wipers. Detect potential problems early by tracking your fuel consumption. Taking care of your car by following recommended maintenance intervals prolongs the life of your car, keeps it running better and increases fuel economy so that you use less gas. Use Car Minder to get the most out of your vehicle. Car Minder's easy and beautiful interface lets you record services, fill-ups and repairs in mere seconds and lookup when a service is due just as fast.

Car Minder's new Gas Log lets you easily track your gas consumption by displaying your average mileage, most recent mileage and a graph of all your fill-ups. In addition, Car Minder displays a nice gauge to help you easily identify how your last fill-up compares to your average.

Car Minder keeps a complete record of your car and allows you to email those records to anyone. Every time you sync with iTunes, Car Minder's database is backed up to your computer and can be restored if you lose or break your phone.

Features available:

- Manage Multiple Services for Each Car

- Repair Log for Each Car

- Gas Log for Each Car

- Lights indicating services that need attention

- Store Notes for Each Car, Service and Service Record

- Miles/Kilometers and Gallon/Liter Options

- Optional Built-In Services to Help You Get Started

- Notification when services are due soon or overdue

- Email Vehicle Service and Repair History

2. aCar

Acar is a straightforward and easy-to-use app that helps you manage all things having to do with your car by means of an appealing user interface. To start, you enter information about your automobile's make, model, year, etc.; if you want, you can track more than one car. Once aCar has the necessary data about your vehicle, it can conduct some really useful calculations for you. For instance, put in your mileage and the price of gas every time you fill up, and aCar will calculate the cost of running your car per day and per mile. It will also tell you how far you are driving each day, predict when your next fill-up should be, calculate your fuel efficiency and chart it all for you.

Record each time you bring your car in for maintenance or repair, include the cost, and you will find out how much your car is costing you. The app will also remind you when your next maintenance check is due. And while a few things have to be entered manually in the beginning, once you enter info such as your mechanic, charge card or gas station, aCar remembers it for you.

There is a $5.99 Pro edition available that adds features such as other languages, manual data restore from an SD card, data import and export, etc. However, I found the free version fit my needs nicely.

FEATURES:

Very clean, modern and user-friendly interface.

Very easy and painless data entry.

Support for logging the specification of your vehicle parts.

Multiple vehicles support.

Powerful searching and filtering.

Geographical location (GPS) support.

Automatic monthly data backups.

Move to SD-Card support (Android 2.2 and later).

Chapter 3: Software Requirement Specification

Introduction

This document details the software requirements specification for the Vehicle Management System. It reflects all the requirements, constraints and design activities of this project. The release number of the software is 1.0. A Vehicle Management System is a 3rd Party iOS application which automates personal vehicle management aspects by keeping a log of distance related aspects of a vehicle and automatically notifying end users regarding their car management issues in time. It will save users a lot of time in efficiently and effectively managing their vehicles without keeping a manual log of everything. The main purpose of developing the Vehicle Management System is to provide those people who do not have very high-end feature packed vehicles and rely on their smart phones for their day-to-day tasks.

Document Conventions

The conventions used to prepare the document is given bellow

1. Font – Times New Roman, size 12
2. Main headings, Bold size 18
3. Sub headings, Bold size 14
4. Sub-sub headings, Bold size 12

Intended Audience and Reading Suggestions

This document is primarily intended for the evaluators and the supervisor of the project. This document will provide guidelines to system developers and testers. Any third party who needs a basic understanding of the system may find this document helpful.

Examiners/Evaluators

The document will provide the FYP evaluators with the scope, requirements and details of the project to be built. It will also be used as basis for the evaluation of the implementation and final project.

Developers

The document will provide guidance to the developers to determine what the requirements are and how they should continue with the project.

Project Supervisor

This document will be used by the project supervisor to check whether all the requirements have been understood and in the end whether the requirements have been implemented properly and completely.

Project Testers

Project testers can use this document as a base for their testing strategy as some bugs are easier to find using a requirements document. It will help in building up test cases for the testing process. This way testing becomes more methodically organized.

Up gradation Engineers

Up gradation engineers can review projects capabilities and more easily understand where their efforts should be targeted to improve or add more features to it. It sets the guidelines for future developments.

End Users

This document can be read by the end users if they wish to know what the project is about and what requirements have been fulfilled in this project.

Product Scope

The proposed project is designing an iOS vehicle management application for iPhones. The app should make full use of the location services feature of iOS Software Development Kit through iPhone sensors like Global Positioning System Receiver, 3 Axis Gyroscope and Accelerometer. The app should essentially relieve the car drivers from manually entering car management key data and make use of the above-mentioned sensors instead.

Overall Description

Product Perspective

The application should collect driving distance information from the integrated iPhone sensors and keep logging the driving activity inside an internally integrated database. The driver should provide initial mileage as an input when the app is downloaded and installed and thereafter, the driver must add refueling checkpoint data manually. The application should inform the driver on requirement his car fuel average from last refueling checkpoint to the current refueling. For the time being, we shall include only this module. Later on, additional modules may be added as well like tracking periodic maintenance points for the vehicle, changing of Engine Oil, Transmission Oil, Air Filter, Fuel filter etc.

Product Functions

This Vehicle Management System will provide following functionalities:

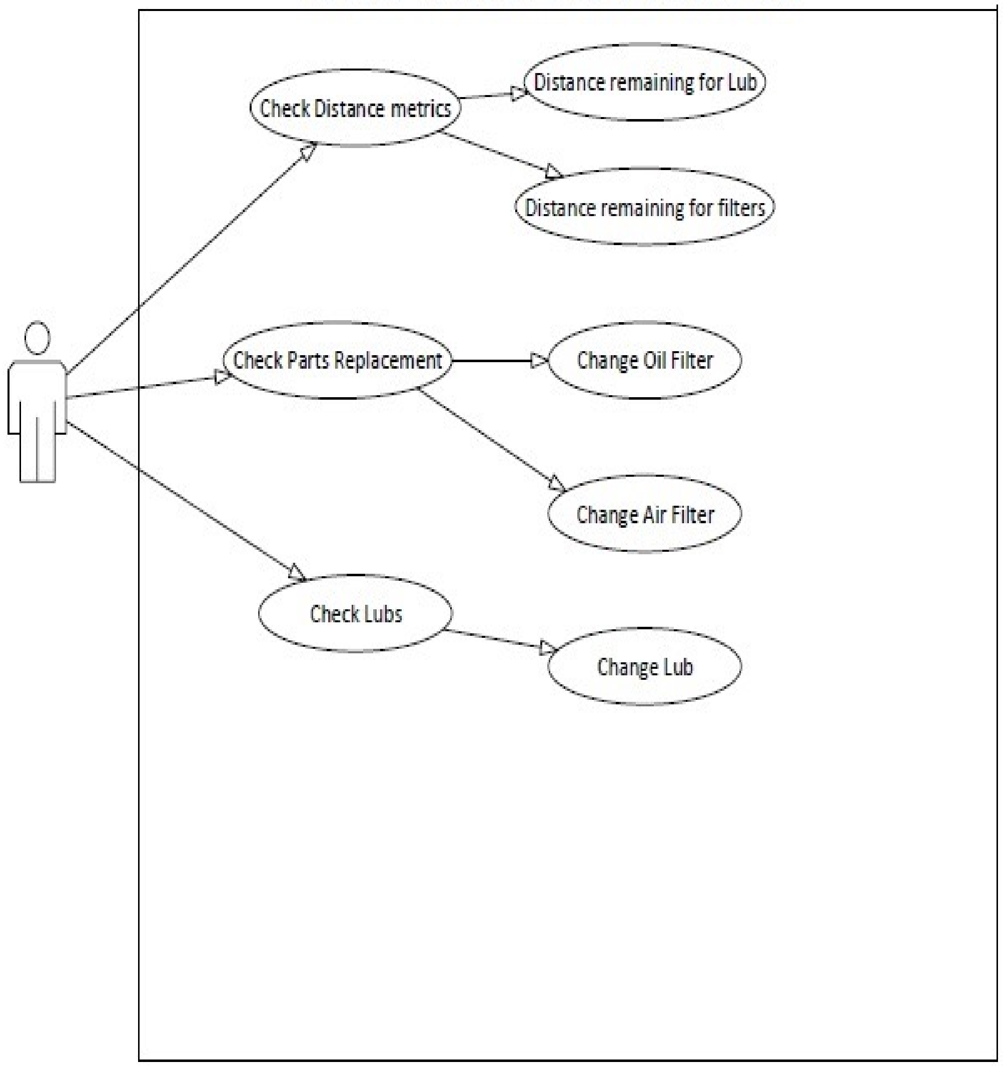
a. Provides a Graphical User Interface to the user for different vehicle management areas.
b. The GUI Main View provides a broader category of maintenance tasks like Oil section, Parts section.

c. The GU I sub-views provide insight into specific maintenance tasks like in Oil Section, it shall have Engine Oil, Brake Oil sub-sections etc.

d. Each section will independently keep a record of related maintenance task.

User Classes and Characteristics

USE CASE DIAGRAM



a. Drivers who own at least one vehicle that does not have OEM fitted vehicle management system in the In-Dash Satellite Navigation / Multimedia / Stereo Unit.

b. Drivers that use iPhones for their daily To-Do tasks.

Following are the user classes and their brief description.

Researchers

Researchers will use this Project as a guide to understand the “Vehicle management system”. They will use this as a base for upgrading and adding new features. They can also use this for developing a new project by using this a reference material.

Tester

Tester will also use this project to check for bug finding. They will also use this to check it is in accordance to the srs document.

Project Evaluator/Supervisor

Project supervisor/Evaluator will also use the product to evaluate. They will use this product to find the accuracy and error in the output.

Operating Environment

The operating environment required for this project is:

a. Mac OS X

b. XCode IDE

c. Swift Language

d. iOS 8 / 9 on an iPhone

Design and Implementation Constraints

Following are the constraints of design and implementation in our project

a. The user has to drive the vehicle along with iPhone.

b. Only one vehicle can be managed at a time.

c. Different profile will have to be activated for each vehicle.

d. The accuracy of data will depend heavily upon GPS sensor and how frequently the user has chosen to give access to the device sensor to his / her location.

User Documentation

An in app tutorial will help the user to get started with the Vehicle Management System application. An FAQ section will also be provided to get the user acquainted with commonly encountered problems.

Assumptions and Dependencies

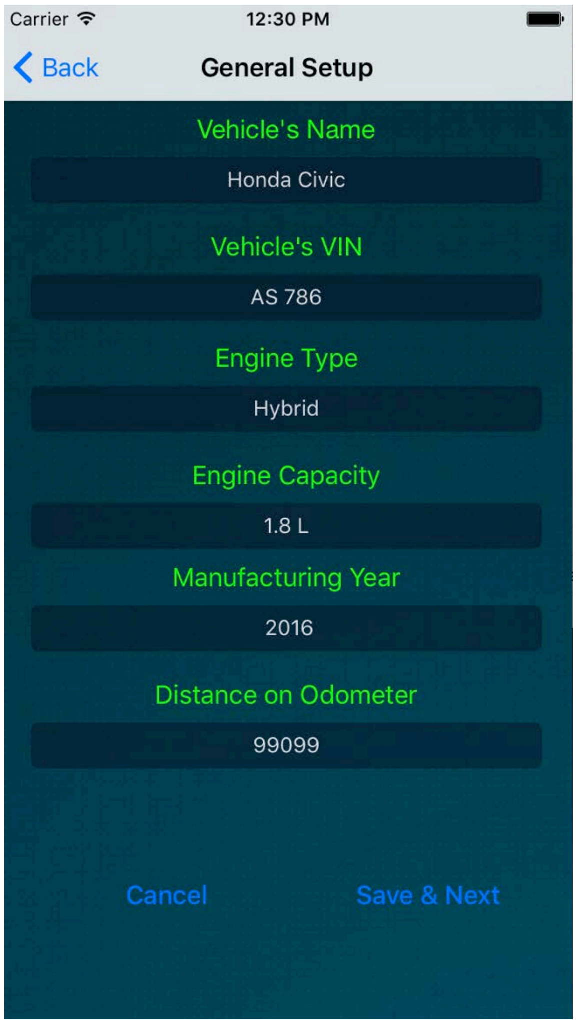
a. The system requires an iPhone with GPS sensor enabled to

use the application.

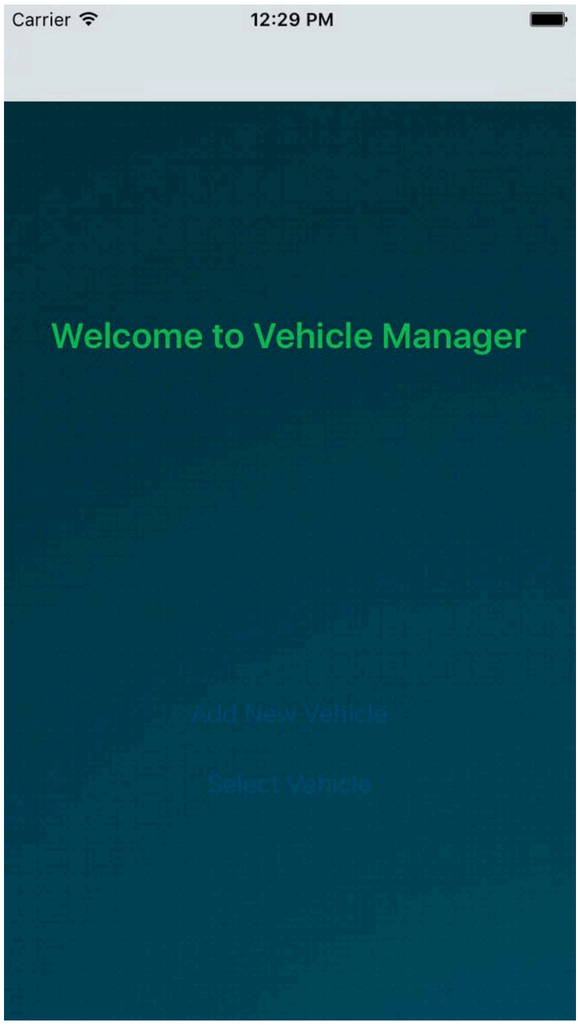
b. The system also requires iOS 8 or 9 installed on the iPhone.

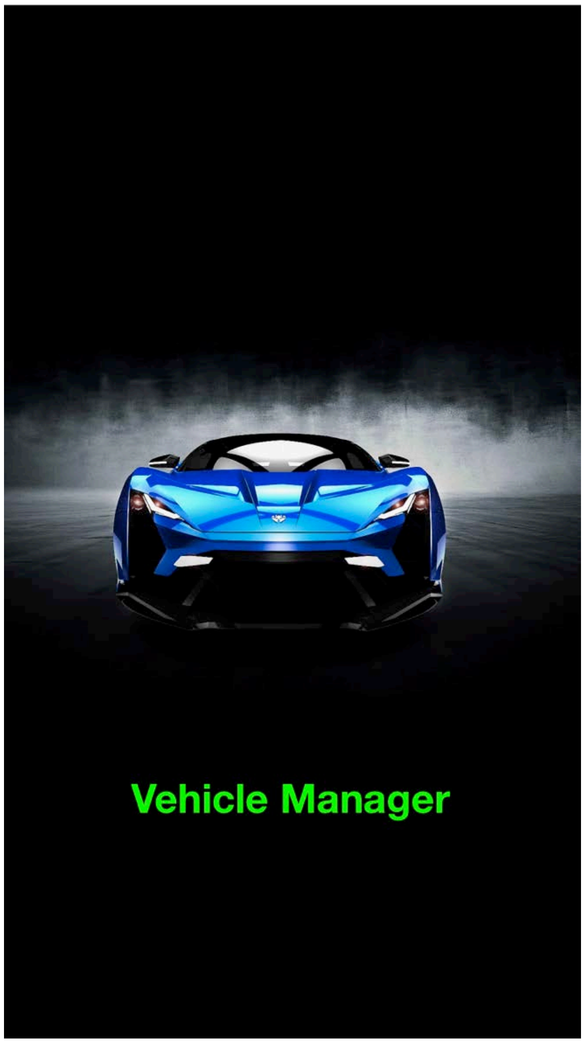
External Interface Requirements

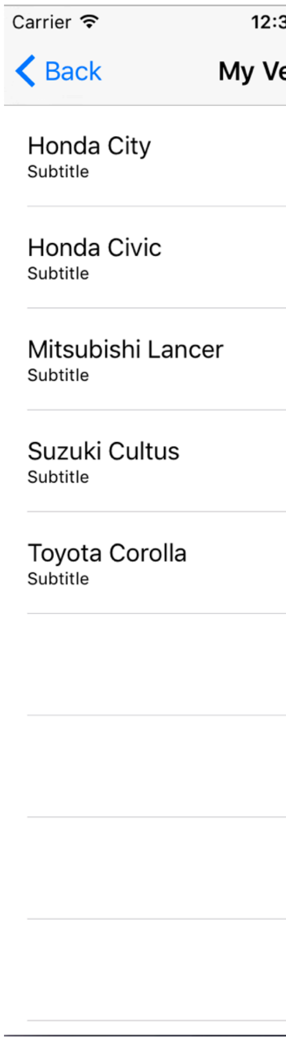
User Interfaces

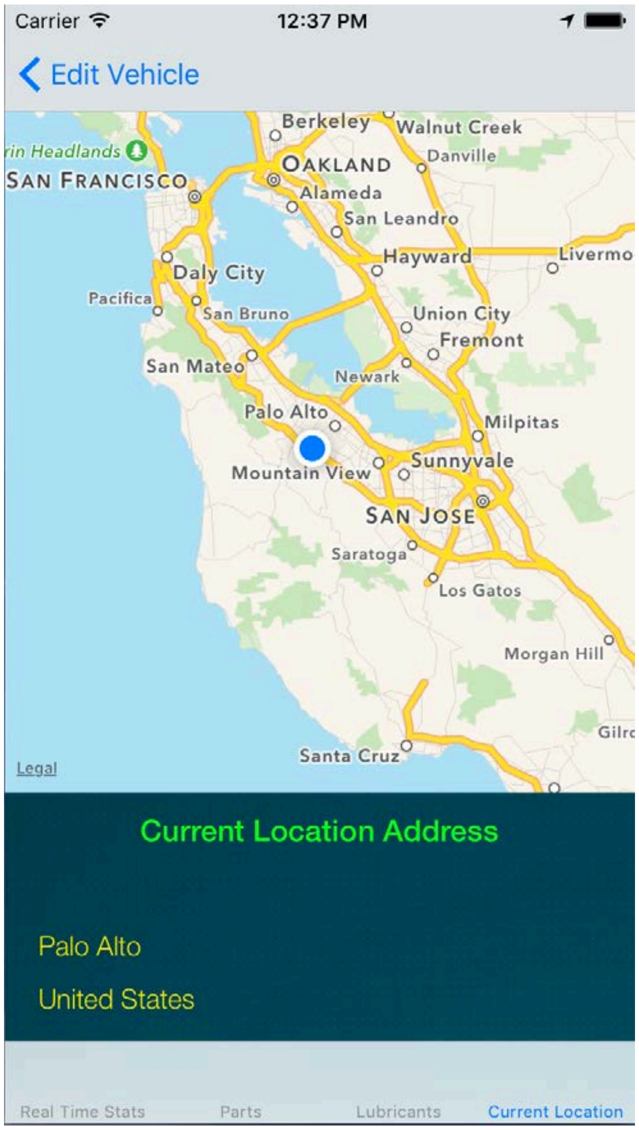


Vehicle's Name
Honda Civic
Vehicle's VIN
AS 786
Engine Type
Hybrid
Engine Capacity
1.8 L
Manufacturing Year
2016
Distance on Odometer
99099


Carrier
12:29 PM
Welcome to Vehicle Manager
Add New Vehicle
Select Vehicle


Vehicle Manager

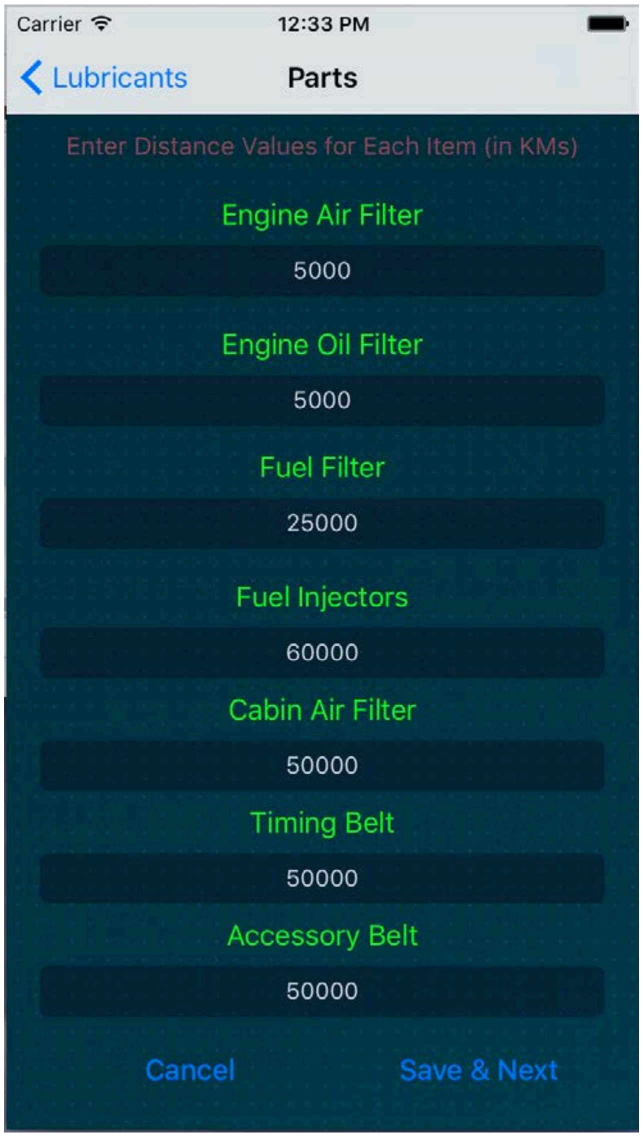

Carrier
12:3
Back
My Ve
Honda City
Subtitle
Honda Civic
Subtitle
Mitsubishi Lancer
Subtitle
Suzuki Cultus
Subtitle
Toyota Corolla
Subtitle



Current Location Address

Palo Alto

United States



Enter Distance Values for Each Item (in KMs)

Engine Air Filter
5000

Engine Oil Filter
5000

Fuel Filter
25000

Fuel Injectors
60000

Cabin Air Filter
50000

Timing Belt
50000

Accessory Belt
50000

Hardware Interfaces

An iPhone is required to run the application.

Software Interfaces

a. The Vehicle Management System app will be developed in XCode IDE using Swift Language.

b. iOS 8 or 9 will be required to run this application on an iPhone.

Communication Interfaces

No communication interfaces are required.

System Features

Layout

The layout of the application is multi-page cross-referenced style with extensive use of Navigation Controller, View Controllers and Modal Segues.

Profile Selector Page

Helps end-user in selecting one of the vehicles he intends to drive. The application will then select the appropriate profile for the vehicle and manage the selected vehicle's data only.

Categories

Once the selected vehicle profile is done loading related data, it will navigate the user to related sections of that vehicle only. The main categories include:

a. Vehicle Information

b. Oil & Lubricants

c. Replacement Parts

d. Routine Maintenance

Sub Categories

Once a broader category is selected by the user, the category selector will navigate the user through further sub categories related.

The sub categories for Oil & Lubricants are:

a. Engine Oil

b. Brake Fluid

c. Power Steering Oil

d. Transmission Oil

The sub categories for Replacement Parts are:

a. Engine Oil Filter

b. Fuel Filter

c. Air Filter

d. Tyres

The sub categories for Routine Maintenance include:

a. CO Emission Test

b. IT Diagnostic Scan

c. General Check Up

d. Engine Tuning

Maintenance Area Information Pages

When the user will press the sub category button, all related information regarding the selected maintenance sub category will display on a new page.

Functional Requirements

REQ-1: For noise cancellation Brain wave will be required which will be stored after Brain Waves reading.

Signal Enhancement:

Description and Priority

This feature will help user to Enhance the recorded brain waves signals spike more understandable .This feature is high priority as well as the more enhanced wave will result in most accurate output.

Stimulus/Response Sequences

1. Once brain waves has passed the noise cancellation step a dialog message will be show for the user "Press continue to Perform Signal Enhancement".
2. User presses continue and the Signal cancellation process will start resulting in wave with Enhanced wave.

Functional Requirements

REQ-1: For signal Enhancement Brain waves with reduced noise will be required which will be stored after Noise Cancellation.

Signal Categorizing and Wave Matching:

Description and Priority

This feature will help user to categorize and match brain waves signals into alpha, beta, gamma etc. This feature is high priority as well if the waves are not categorized correctly it will result in false output.

Stimulus/Response Sequences

1. After signal Enhancement user will press continue to categorize the brain waves and the values of waves will be stored.
2. Waves will also be show in a side window for user with values for easy understanding.

Functional Requirements

REQ-1: For categorizing and matching Enhancement will be required which will be stored after step three.

Output

Description and Priority

This will be the last step. This feature will show the output after the processing of the wave in the features mentioned above. This is also important because it will show what user was thinking.

Stimulus/Response Sequences

1. After categorize and matching of the waves output will be show in a new window.
2. After output is show user will be asked to give feedback/ mark how accurate was the result for future improvements.

Functional Requirements

REQ-1: Categorized and correct matching of brain waves for correct output will be required.

Other Nonfunctional Requirements

Performance Requirements

a. Response Time

The response time of the system would be less than five seconds.

b. Capacity

At a time, user would be able to select only one profile. This project should use the minimum resources.

Safety Requirements

a. The application crash will not result in data loss.

b. Once the vehicle data has been updated, it shall be saved automatically in internal database.

Software Quality Attributes

a. Extensibility and Maintainability

The application has a huge potential to grow in future releases as numerous other functionalities can be added. The application code may also be re used for future development.

b. Portability

Since it is an application developed for mobile operating system iOS, it is inherently portable.

c. Robustness

The limitations of using one profile at a time tenders less opportunity to the user to crash the application. Hence it is robust.

d. Usability

The new user will be able to create new vehicle profile after following a quick in-app tutorial.

Appendix A: Proposal

Brief Description of the Project:

The application should collect driving distance information from the integrated iPhone sensors and keep logging the driving activity inside an internally integrated database. The driver should provide initial mileage as an input when the app is downloaded and installed and thereafter, the driver must add refueling checkpoint data manually.

The application should inform the driver about periodic maintenance points for the vehicle, changing of Engine Oil, Transmission Oil, Air Filter, Fuel filter.

Scope of Work: The proposed project is designing an iOS vehicle management application for iPhones. The app should make full use of the location services feature of iOS Software Development Kit through iPhone sensors like Global Positioning System

Receiver, 3 Axis Gyroscope and Accelerometer. The app should essentially relieve the car drivers from manually entering car management key data and make use of the above-mentioned sensors instead.

Application /End Goal Objective:

the project can serve to become a basis for further development in vehicle management system. Further research can be made to automate the tracking of air pressure in tires, performance of shocks and many other mechanical parts.

Appendix B: Glossary

Car Minder Plus Car Minder is an application for managing all your car maintenance needs, logging repairs and tracking fuel economy. Keep track of multiple cars, services and repairs. Keep track of oil changes, tire rotations, and other maintenance tasks for all your cars.

Chapter 4: Design and Development

INTRODUCTION

The section introduces the Software Design for the Vehicle Management System to its readers. This Software Design Document (SDD) accompanies an architecture diagram with pointers to detailed feature specifications of smaller pieces of the design. The aim of this document is to stable reference, outlining all parts of the software and how they will work..

Purpose of the document:

Globalized era and the advancement of technology made human life easier than it was. The more complex the problem is, the more efficient solution has been proposed. If we have to look in the world, we came across some facts which directly or indirectly affect human beings. In the past the systems were manual, but in this era the systems are being automated so the efforts of the humans are being sustained. Vehicle Management System (VMS) is a system that that is proposed to develop a system that will automate the management aspects of a vehicle and will help keep track for scheduled maintenance activities. This Software Design document is meant to outline design features of VMS, so as to serve as a guide to the developers on one hand and a software validation document for the prospective client on the other. Following are the main users that would interact with the system directly or indirectly:

Scope of the Development Project

The proposed project is designing an iOS vehicle management application for iPhones. The app should make full use of the location services feature of iOS Software Development Kit through iPhone sensors like Global Positioning System Receiver, 3 Axis Gyroscope and Accelerometer. The app should essentially relieve the car drivers from manually entering car management key data and make use of the above-mentioned sensors instead. The application should collect driving distance information from the integrated iPhone sensors and keep logging the driving activity inside an internally integrated database. The driver should provide initial mileage as an input when the app is downloaded and installed and thereafter, the driver must add refueling checkpoint data manually.

The application should inform the driver on requirement his car fuel average from last refueling checkpoint to the current refueling. For the time being, we shall include only this module. Later on, additional modules may be added as well like tracking periodic maintenance points for the vehicle, changing of Engine Oil, Transmission Oil, Air Filter, Fuel filter etc..

Definitions, acronyms, abbreviations

- API: Application Programming Interface
- FAQ: Frequently Asked Questions
- VMS: Vehicle Management System
- GUI: Graphical User Interface
- HTTP: Hypertext Transfer Protocol
- HTTPS: Secure Hypertext Transfer Protocol is a HTTP over SSL (secure socket layer).
- Mac: Macintosh
- Swift: Structured Query Language
- iOS: iPhone Operating System
- FW: Framework

Overview of the document

This document shows the working of our application. It starts off with the system architecture which highlights the modules of the software and represents the system in the form of Block Diagram, component diagram, Use Case Diagram, ER Diagram , Sequence Diagram and general design of the system. Then we move on to discuss the detailed Description of all the components involved. Further we discuss the dependencies of the system and its relationship with other products and the capacity of it to be reused. Then towards the end we shall discuss the Design Tradeoffs and the Pseudocode.

SYSTEM ARCHITECTURE DESCRIPTION

The architectural design or more said as component or class diagram, shows some more technical details about the project, and its low-level architecture of the system. This is totally different to the deployment diagram, which is mentioned later. It shows an insider's perspective of the system by describing the high-level software components that perform the major functions to make the system operational. It overlooks the hardware aspects.

Data Structure Diagram

In this section UML class diagrams are mentioned. Class diagrams are important integral for the description of low level components of the software that include how the different parts of the software work. Classes provide the system different means to perform its functions. Details about the database are also defined in this part.

Deployment Diagram

The deployment diagram describes the very high level architecture of the system, by listing in figures, all of the main parts of the system. The deployment diagram shows how the system looks from the outer world in a non-technical context. It will give the first impression of how the large parts are linked together and what the total parts comprise without going into unnecessary details. This is not about components diagram, because there is a separate components diagram, having all the logical and technical details about the components.

Use Case Realization

Use cases tell how the user shall be able to interact with the system and how the system shall respond in return. It is actually a user’s interaction with the system and the other users. The use cases are used at higher level, often representing missions and goals. They directly relate to the functional requirements of the SRS.

Activity Diagram

Activity diagrams follow a workflow-based approach to describe the overall functioning of the system. They are graphical representation of workflows of stepwise activities and actions with support for choice and iteration. This is basically a good means to see how the steps involved in major tasks inside a system with the flow of control.

Sequence Diagram

Sequence diagrams demonstrate how distinctive objects are involved in the culmination of a functionality of the system. They have a one of a kind arrangement that allows the reader to see many objects and for what extent for the completion of a functionality of a system.

UI Design

A few previews of the different graphical client interfaces are demonstrated in this segment that model the way a client might be communicating with the framework in a Windows Operating System.

Interface Specification

This segment contains all the insights about diverse interfaces i.e. Hardware, software and communication interfaces to be actualized on the system. The design of the interfaces has been exhibited with an agreeable knowledge into working of distinctive segments has been given.

Benefits

The main advantages that could be achieved from this system are

- Advantage to drivers who own at least one vehicle that does not have OEM fitted vehicle management system in the In-Dash Satellite Navigation / Multimedia / Stereo Unit.
- Advantage to drivers that use iPhones for their daily To-Do tasks.
- Advantage to drivers who own at least one vehicle that does not have OEM fitted vehicle management system in the In-Dash Satellite Navigation / Multimedia / Stereo Unit.
- Advantage to drivers who use iPhones for their daily To-Do tasks.

Objectives & Goals

The Objectives and goals that this system can achieve are:

- Developing Swift based application for drivers which will include forms to enter data to sync information.
- Creating a Swift based application for iPhones using and implementing iOS Frameworks.
- Database for the maintenance of the Complete Record of the Vehicles.
- Keep the system easy-to-use for the users.
- Ensure a system free of data loss.

Document Conventions

This section describes the standards followed while writing this document.

Headings:

Headings are prioritized in a numbered fashion, the highest priority heading having a single digit and subsequent headings having more numbers, according to their level. All of the main headings are titled as follows:

All first level headings have single digit number followed by a dot and the name of the section. All bold Times New Roman, size 18, Centered e.g. (2. Heading 2).

All second level sub headings for every sub section have the same number as their respective main heading, followed by one dot and subsequent sub heading number followed by name of the sub section. All bold Times New Roman, size 14, e.g. (2.2 Heading 3).

Further sub headings, i.e. level three and above, follow the same rules as above for numbering and naming, but different for font. All bold Times New Roman, size 12, e.g. (2.3.1.1 heading 4).

Bullets and numbering:

Bullets have been given where there is no need for prioritizing a list. For example the list of use cases, where uses cases may appear in any order.

Numbered lists are normally used for prioritizing purposes. Prioritizing purposes arise when the customer has specified a specific order for the requirements or when a need for prioritizing arises due to business needs. For example, in a use case example, the steps the external may perform are listed in a numbered fashion. Also when specifying the organization of intended audiences, the people in need of reading this document are listed first, and so on.

Figures:

All figures in this document have captions, and are assigned figure numbers. The format of the figure no is “Figure # - Figure Name”. Use case, data flow, state transition, entity relationship and sequence diagrams are based on UML standards.

Reference:

All references to headings in this document are provided where necessary, however where not present, the meaning is self-explanatory. All ambiguous terms have been clarified in the glossary at the end of this document.

Links to web pages:

All links have been provided with underlined font, the title of the web page or eBook is written at the top of the link and the title may be searched using the Google search engine to pinpoint the exact address.

Basic Text:

All other basic text appears in regular, size 12 Times New Roman. Every paragraph explains one type of idea.

Intended Audience and Reading Suggestions

The Intended audience for this document is listed below:

Examiners/Evaluators

The document will provide the FYP evaluators with the scope, requirements and details of the project to be built. It will also be used as basis for the evaluation of the implementation and final project.

Developers

The document will provide guidance to the developers to determine what the requirements are and how they should continue with the project.

Project Supervisor

This document will be used by the project supervisor to check whether all the requirements have been understood and in the end whether the requirements have been implemented properly and completely.

Project Testers

Project testers can use this document as a base for their testing strategy as some bugs are easier to find using a requirements document. It will help in building up test cases for the testing process. This way testing becomes more methodically organized.

Up-Gradation Engineers

Up gradation engineers can review projects capabilities and more easily understand where their efforts should be targeted to improve or add more features to it. It sets the guidelines for future developments.

1.9.6End Users

This document can be read by the end-users if they wish to know what the project is about and what requirements have been fulfilled in this project

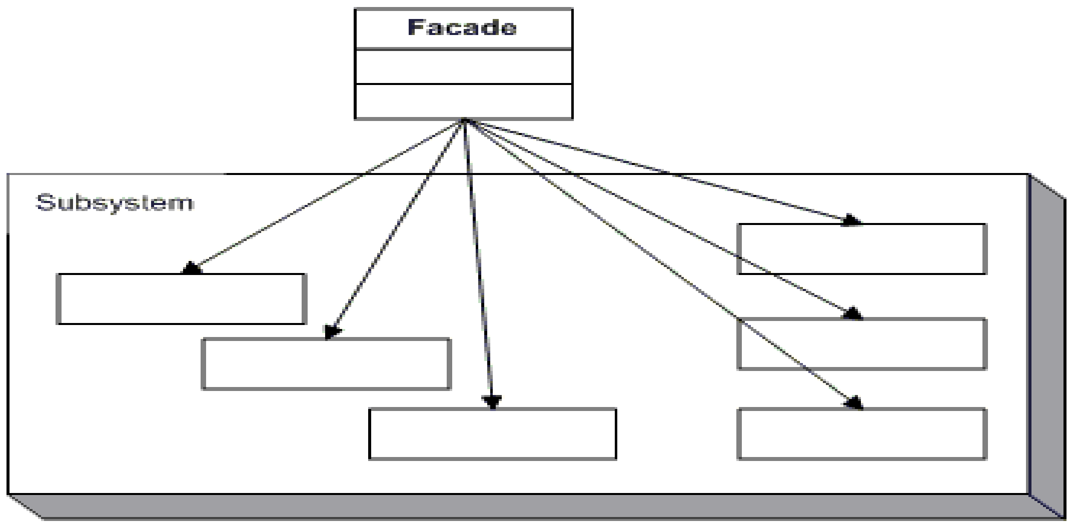
Reading suggestions:

The SDD begins with the title and table of contents. All level 1 and level 2 headings are given in the table of contents, but the lower sub headings are not included. Each main heading is succeeded by a number of sub headings, which are all in bold format. The product overview is given at the start, succeeded by the complete detailed features, including both functional and non-functional requirements. The entire interfaces are also described. The SDD ends with a bunch of appendices, including a glossary.

System Architecture Description

Vehicle Management System is a self-contained project, aiming to automate the maintenance of vehicles by the drivers a hassle-free job. The architecture is based on Façade Pattern, allowing to encompass the working of the components in a single top layer. The pattern gives one entry point for the user for a very easy and intuitive use. The architecture also helps in a modular implementation of the sub-components of the application. The creation of a profile (new vehicle) will be initiated through a Graphical User Interface Screen and the initial information for each vehicle will be entered through same architectural design. The data generated by the driver for a particular vehicle will be stored in an on-device SQLite based database for persistent storage in the events of application fault or crash. Once configured, the application shall measure the driving distance of the vehicle being driven fairly accurately and generate various alerts and warnings for maintenance tasks when due.

Figure 1 Façade Pattern Architecture



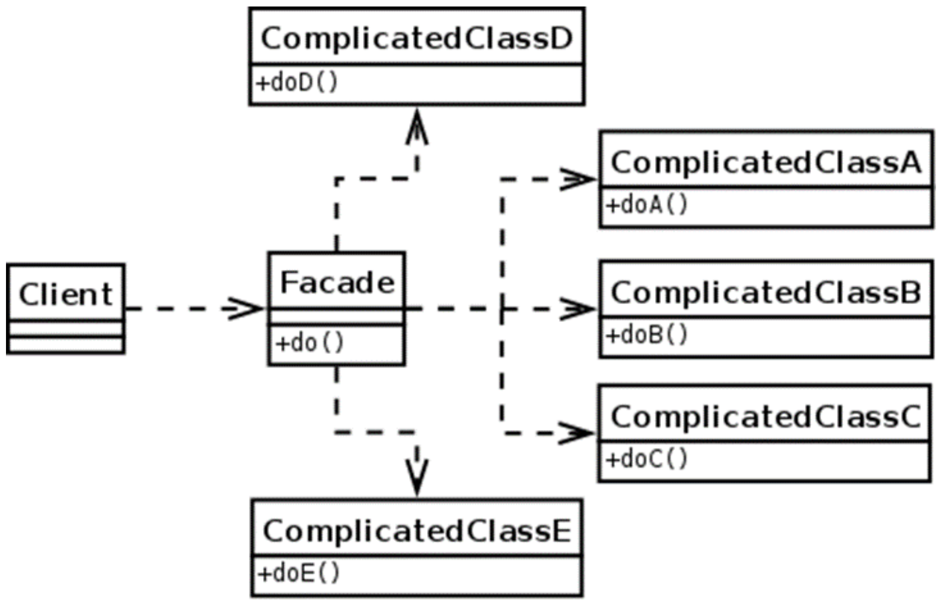


Figure 2 Façade Pattern Architecture

Overview of Modules/Components:

Software Components

Operating Systems

- Apple's Mac OS X (El Capitan)
- Apple's iOS 9

Software Packages

- XCode 7.1.1
- Adobe Photoshop
- Pixelmator
- Sketch
- Appo

1. SQL (Structured Query Language): SQL is a special-purpose programming language designed for managing data held in a relational database management system (RDBMS). In this application, we shall use SQLite which works on the same principles and is used by the Core Data framework of Swift language.

2. XCode is the Integrated Development Environment used for developing software applications for all Apple devices.

3. Adobe Photoshop & Pixelmator are used to visualize and later on implement the Graphical User Interface elements of the application.

4. Sketch is used for application prototyping (wire framing) of application graphical storyboard, which forms basis of application design.

5. Appo is a tool used to generate all picture based resources for the development of application.

Hardware Components

N/A

Product Design Specifications, Assumptions and Dependencies

The team of software engineers is obliged to state the following assumptions while making this SDD:

Hardware and software platforms

The software shall be developed on Intel based Macs and Apple’s OS X Based Operating Systems. The final product has to be deployed on an Apple’s iPhone device operating on iOS 9 operating system.

Product Quality level

The product is intended to serve as a helping tool to facilitate drivers from all walks of life and due to heavy reliance on managing the maintenance aspects of the vehicle, the product should be able to sustain prolonged sessions of usage.

Testing and Debugging

The group shall try its best to remove all possible errors and bugs from the software that are able to be resolved within the limited expertise frame of the group.

User Manuals

The group shall provide the user manual for the VMS with the instruction for the usage of the system. The user manuals will also include graphical guidelines for the operations. It will be provided in both hard and on application. The manual would not be exhaustive, but provide help on the most common features such as installation, basic usage and most frequent errors.

User Interface Issues

Firstly, in order to add a new vehicle, the driver shall follow a separate routine in which he will all the relevant information required by the application to setup the database entities.

Secondly, once a vehicle has been configured, the user will simply go to the newly created profile of the vehicle and can configure alerts, generate reminders etc.

The Screenshots for VMS user interface are as follows:

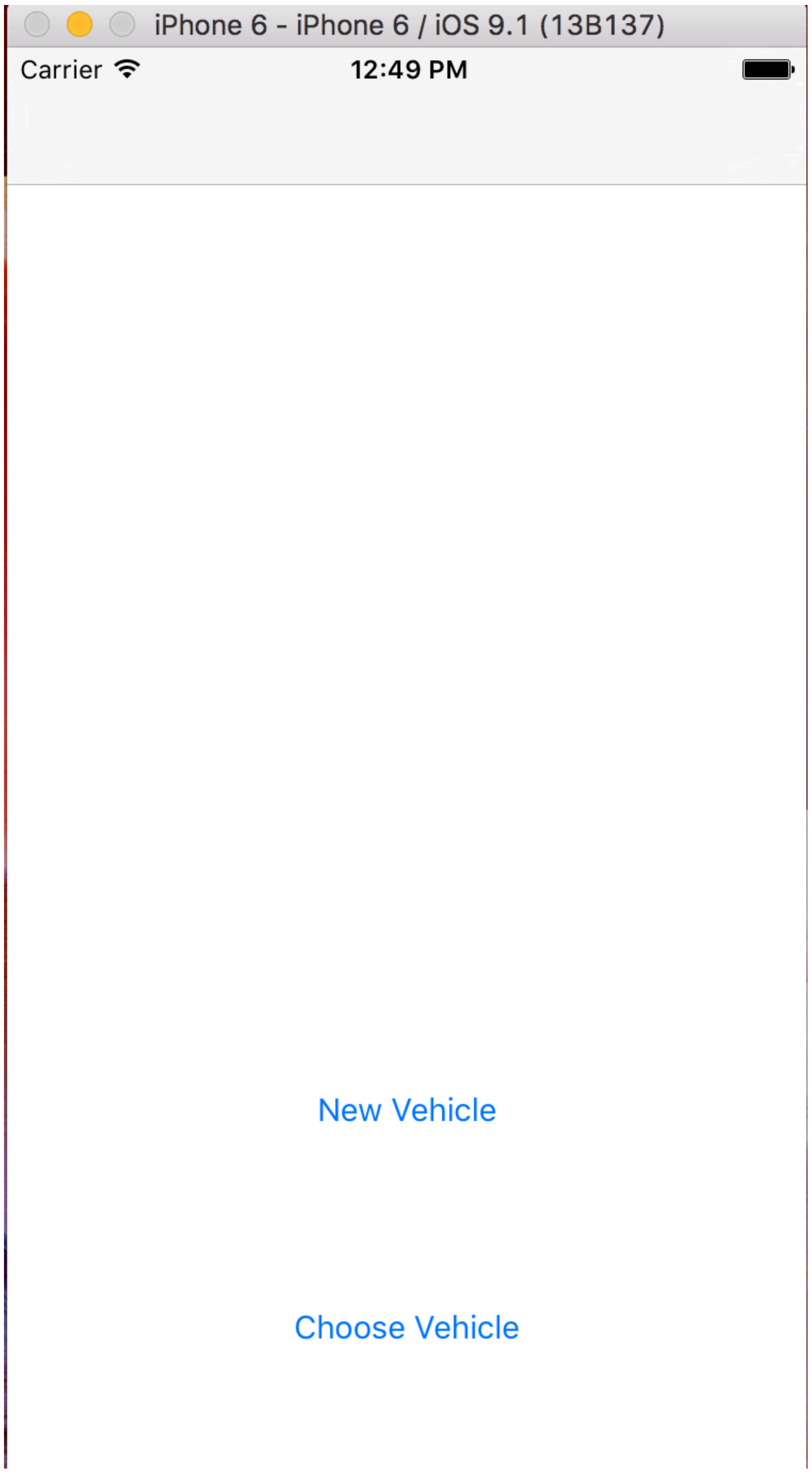
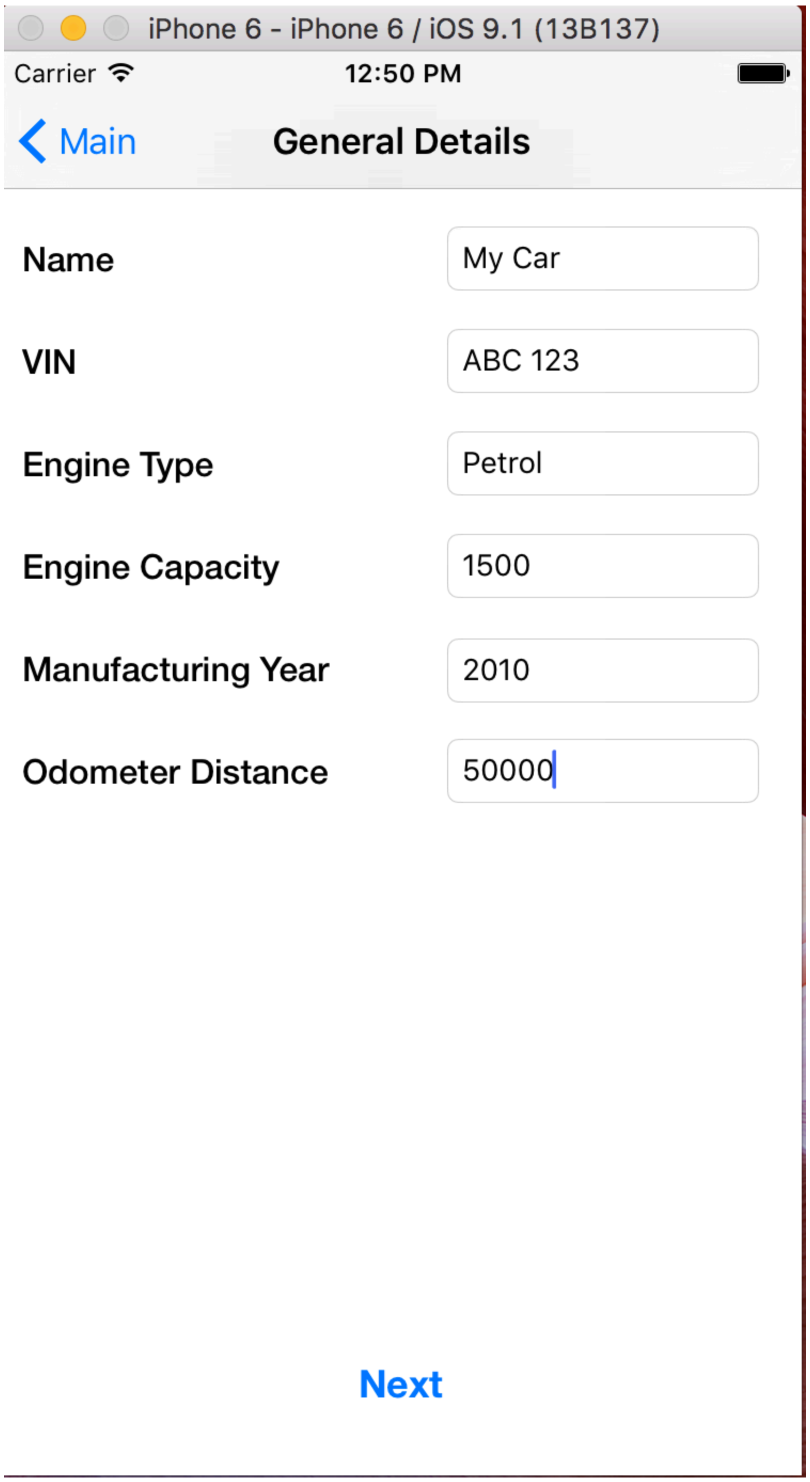
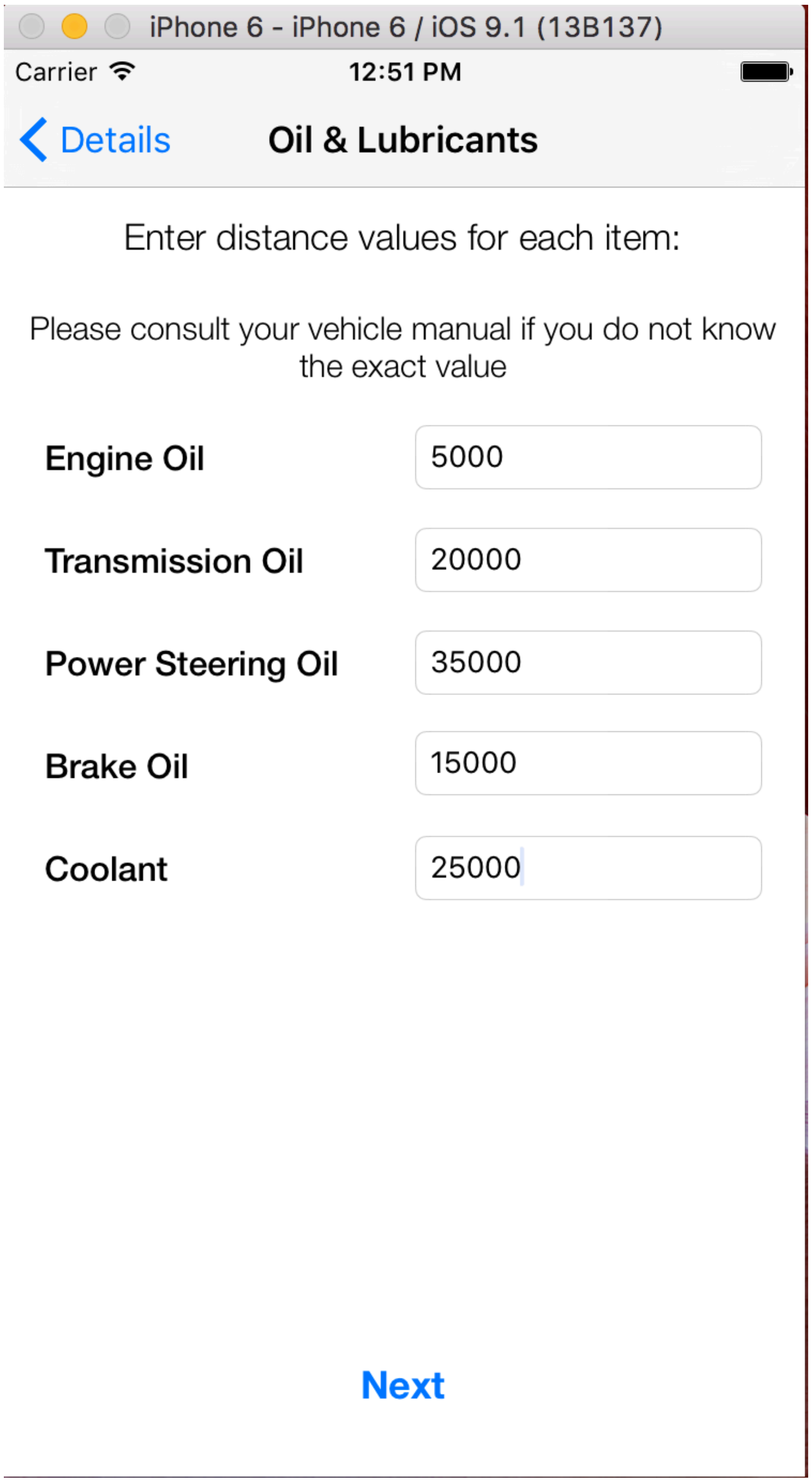


Figure 1 VMSHome Page



Name	My Car
VIN	ABC 123
Engine Type	Petrol
Engine Capacity	1500
Manufacturing Year	2010
Odometer Distance	50000

Figure 2 VMS General Details Entry

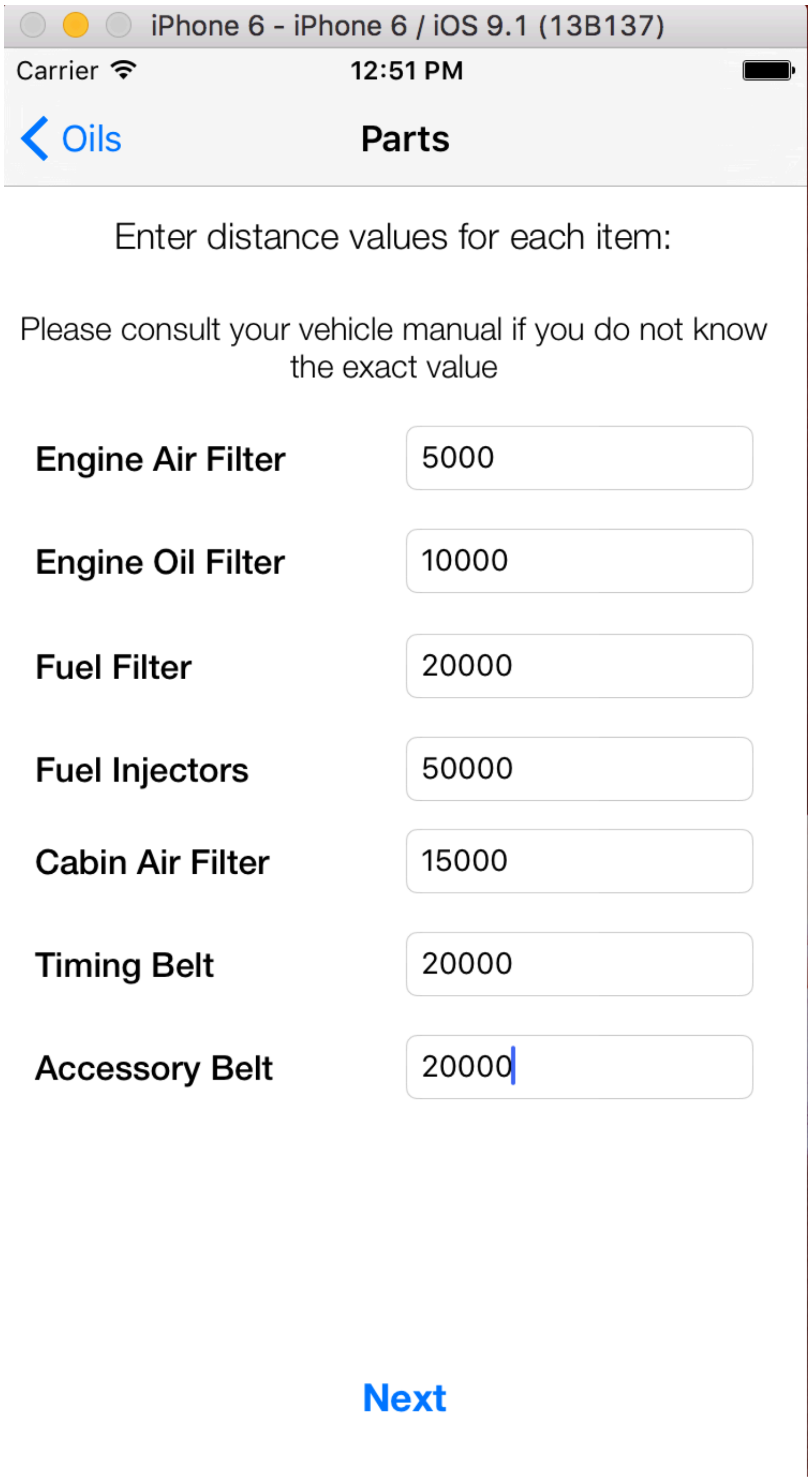


Enter distance values for each item:

Please consult your vehicle manual if you do not know the exact value

Engine Oil	5000
Transmission Oil	20000
Power Steering Oil	35000
Brake Oil	15000
Coolant	25000

Figure 3VMSOil & Lubricants



Enter distance values for each item:

Please consult your vehicle manual if you do not know the exact value

Engine Air Filter	5000
Engine Oil Filter	10000
Fuel Filter	20000
Fuel Injectors	50000
Cabin Air Filter	15000
Timing Belt	20000
Accessory Belt	20000

Figure 4VMS Parts Entry

Detailed description of components

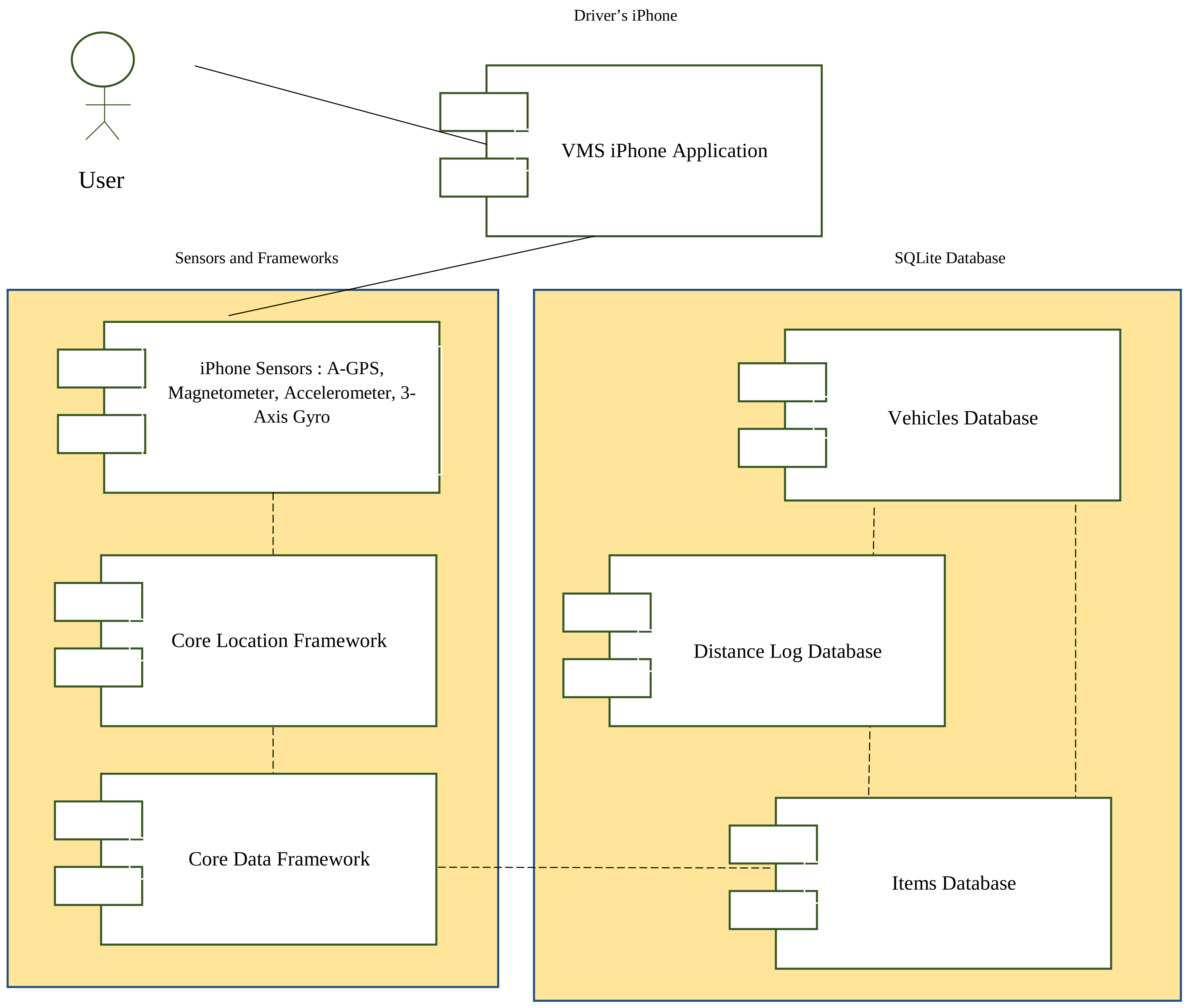


Figure 7 High Level View of the App Architecture and Components

Detailed Description of Components

Following Table will be used to express all the details related to a component. The field which did not belong to a particular component has been marked N/A rather than being left out.

Identification	The unique name for the component and the location of the component in the system.
Type	A module, a subprogram, a data file, a control procedure, a class, etc.
Purpose	Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS, but are implied or adjunct to formally stated SDS requirements.
Subordinates	The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part.
Dependencies	How the component's function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components.
Interfaces	Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here.
Resources	A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services.
Data	For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary.

VMS Graphical User Interface

Identification	GUI
Type	A subprogram.
Purpose	It provides the interface to the user from where he can perform all the required tasks. It is the bridge between the user and the system's integrated sensors.
Subordinates	It consists of forms on which the user enters the information which is to be sent. Similarly it also displays the information which is received by the user.
Dependencies	The system requires GPS Satellite signals in order to work flawlessly. In order to accurately generate the bearing and distance data, the application should be operated in an environment free of heavy electrical appliances.
Interfaces	It is an iPhone application and runs on iOS basedApple devices. For each vehicle required to be managed, it requires the same set of information. The GUI interfaces with Core Location Framework and Core Data Framework embedded in the iOS.
Resources	An Apple iPhone device is required to deploy and run the application.
Data	The major work being done at the application is saving the user data default configuration values to the internal database and thereafter using the same data as well as the distance coming from the integrated sensors to generate maintenance alerts.

Core Location Framework

Identification	Core Location Framework
Type	Framework
Purpose	This is the part of the application which performs the Logical Tasks. The framework modules shall record distance and compare with data inside the database to be able to generate reminders.
Subordinates	It will consist of functions and classes which will ensure the fulfillment of the purpose stated in the above point.
Dependencies	This module depends upon the data from the database which will be acquired through queries whenever required after completing all formalities and the connection with the clients through which this module will receive data which requires processing or any action from initiating till its accomplishment.
Interfaces	An interface screen shall be added to show the driver his current speed, coordinates and other such data in real time
Resources	A-GPS Sensor Accelerometer Sensor Magnetometer Sensor Cellular Antenna
Data	The data used is in different forms depending upon the type of the entity.

Core Data Framework

Identification	Core Data Framework
Type	Framework
Purpose	This is the part of the application manages the database of the app and ensures data persistence which is not otherwise available for iOS app which are terminated once.
Subordinates	It utilizes the embedded property of the iOS by automatically managing the model layer database by inbuilt libraries. The libraries provide various functions for creating and updating object modelsthroughout its life cycle.
Dependencies	This module depends upon the data from the Core Location Framework as well as the data entered by the user.The acquisition will be performed through queries whenever required.
Interfaces	The Core Data Framework would manage the model layer objects of VMS application. It will also be integrated with the application’s controller layer to automatically update the edited entries.
Resources	iOS screen gestures. Core Location Framework. iCloud database.
Data	The data used is in different forms depending upon the type of the entity simple number to complex strings etc.

Code of Components

Data Entry Module:

class GeneralSetupViewController: UIViewController{}

//Starts the vehicle class variables entry and makes it ready to connect clients. It includes defining of the ports aswell.

func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?)
//Prepares the data entered in classes for saving and storage to next class

Requesting user location and distance details (Logic component):

class ViewController: UIViewController, CLLocationManagerDelegate{}

//The delegate class used to instantiate the core Location manager.

func locationManager(manager: CLLocationManager, didUpdateLocations locations: [CLLocation])
// Creation of location manager instance and handling of location updates.

Reuse and relationships to other products

The module used to establish connection between the Core Location framework and the Core Data framework can be reused from some other project but reusing this module also requires medication.
Many of the modules that are being developed in VMS can be reused in the future for other location-based projects.

Design decisions and tradeoffs

The design decision to break the system up into a three-module system was made to promote modularity within separate parts of the system. Modularity provides encapsulation for the important pieces of the system. Using encapsulation, we are able to change important parts of the system without changing the whole system.

Detailed Design

Entity Relation Diagram

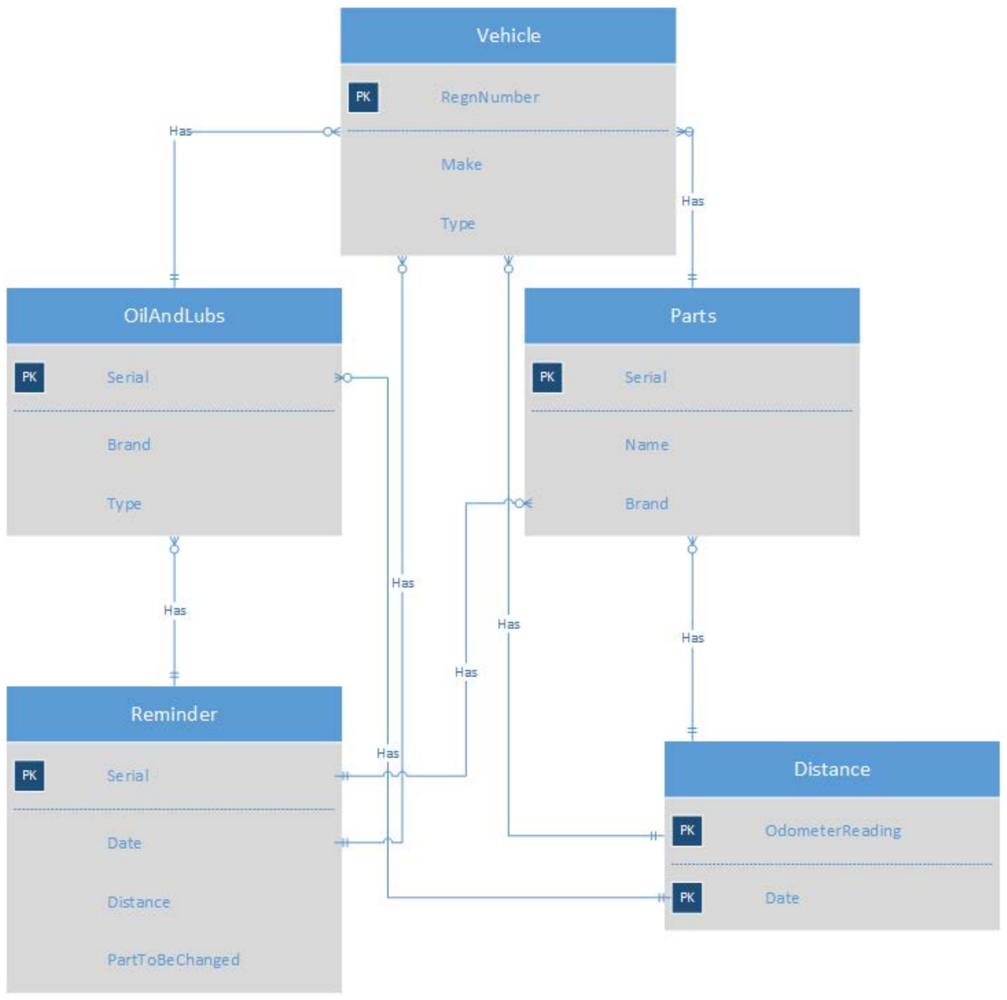


Figure 8 Entity Relation Diagram

Entities & Attributes

Vehicle:

Field	**Type**	**Description**
RegnNumber	Primary Key, varchar(20)	Unique id assigned to each Vehicle.
Make	varchar(20)	The Brand of the vehicle.
Type	varchar(20)	The typeof the vehicle.
Engine	Varchar(25)	Capacity and type of Engine
Model	Varchar(20)	The year of Manufacturing

Table 1 - Entity: Vehicle

Oil and Lubs

Field	Type	Description
Serial	Primary Key, int	Unique id for each Oil entry.
Brand	varchar(20)	The Brand of the Oil.
Type	Varchar(20)	Defines type of Oil

Table 2 -Entity: Oil and Lubs

Parts

Field	Type	Description
Serial	Primary Key, int	Unique id for each letter to identify it
Name	varchar(50)	Subject/title of the letter
Brand	Varchar(100)	Address of the sender

Table 3 – Entity: Parts

Reminders

Field	Type	Description
Serial	Primary Key, int	Unique id for every reminder.
Date	Date	date
Distance	Varchar(15)	Distance to generate reminder
PartsTobeChanged	Varchar(20)	Name of the part to be replaced

Table 4 - Entity: Reminders

Distance

Field	Type	Description
Distance	Primary Key, int	Unique id for each Warning
Date	Primary Key, int	Letter id from which the report is generated

Table 5 - Entity: Distance

Use Case Diagrams

Use Case for VMS Application

All the data will be generated and sent to the database from the clients. The following use case diagram represents the functionality of this subsystem

.

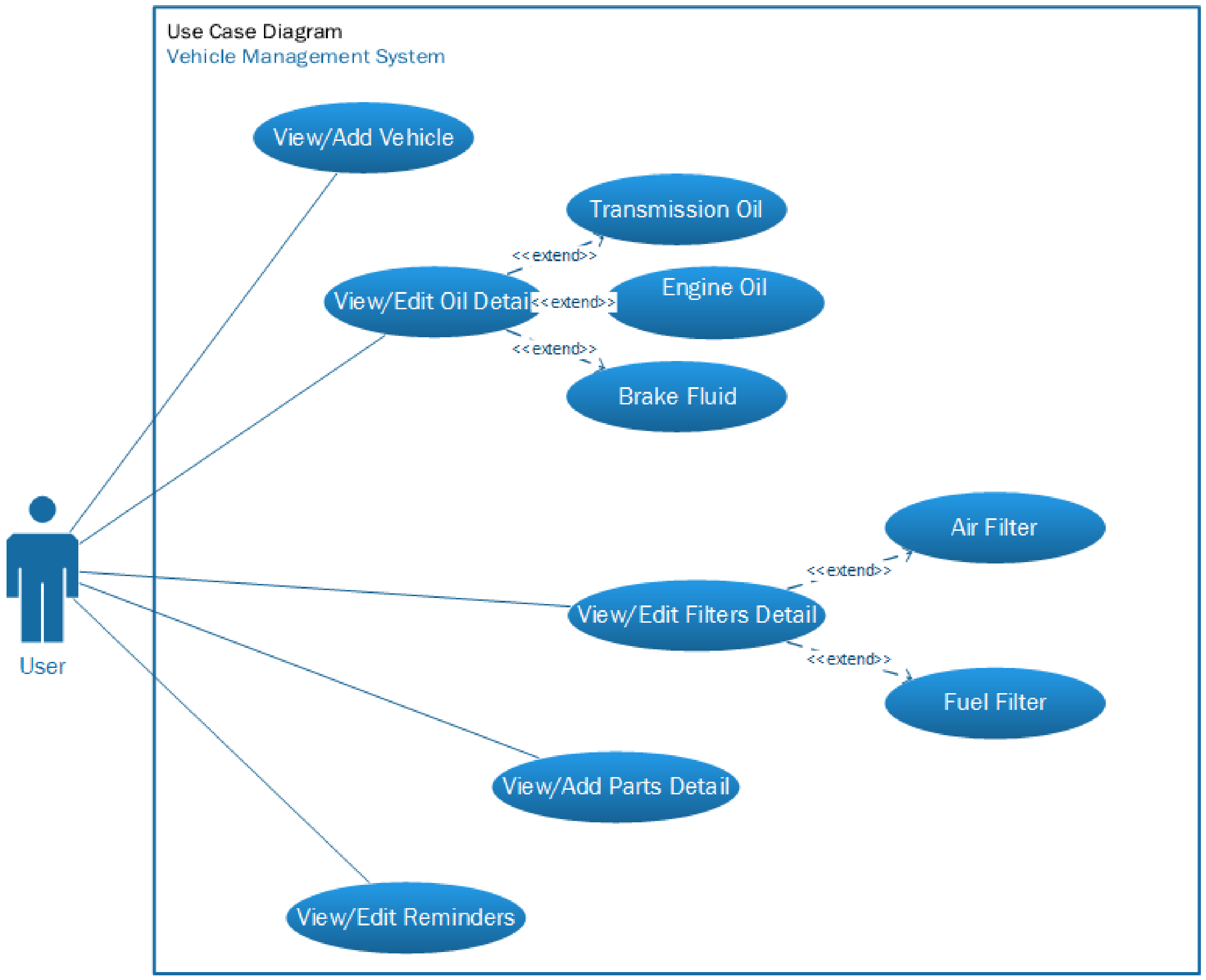


Figure 9 Use Case: VMS

UC-1 View/Add Vehicle

Use Case Description

The application shall allow users to add multiple vehicles for their record keeping. On first launch of the app, the user will have to add vehicles he wants to manage.

Business Justification

The user must add a vehicle in order to use the app. Also viewing a car's details is a primary function of the app.

Use Case Name	View/Add Vehicle
Actor(s)	App User
Pre-Condition	VMS is launched.
Normal Course	• **The vehicle's data is added** • **The vehicle's data summary is shown**
Post-Condition	User saves the added vehicle's data. User can view further parts for detail.
Alternate Course	• **Vehicle is not added because maximum number of vehicles have been reached**
Post-Condition	The user will be able to select already added vehicles.
Assumptions	The iOS version supports the application.
Priority	High
Frequency	Low
Risk	Low

Table 6 - Use Case: View/Add Vehicle

UC-2 View/Edit Oil Details

Use Case Description

The VMS application shall allow its users to view and manually change the details of their oil changes.

Business Justification

Oil and Lubricants are the major part of a vehicle log maintenance.

Use Case Name	View/Edit Oil Details
Actor(s)	App User
Pre-Condition	User has selected a vehicle.
Normal Course	• **The previous Oil change date and the next due distance/date.** • **The user can change the next due distance/date.**
Post-Condition	The user can check other parts details.
Alternate Course	• User has not added Oil details.
Post-Condition	The app will ask for the addition of the detail.
Assumptions	The Oil details has been added.
Priority	High
Frequency	Medium
Risk	Low

Table 7 - Use Case: View/Edit Oil Details

UC-3 View/Edit Parts Detail

Use Case Description

The application allows the user to view or edit various parts detail.

Use Case Name	View/Edit Parts Detail
Actor(s)	App User
Pre-Condition	A vehicle has been added/selected.
Normal Course	• The details about the selected part is shown • The user edits any part's details.
Post-Condition	User can view other parts/features of the vehicle.
Alternate Course	The User has not added any parts details.
Post-Condition	The app allows the user to add a part's details.
Assumptions	User has added a vehicle/part detail.
Priority	Medium
Frequency	Medium
Risk	Low

Table 8 - Use Case: View/Edit Parts Detail

UC-4 View/Edit Reminders

Use Case Description

The main feature of the app is a hassle free vehicle maintenance by not only saving vehicle logs but also generating reminders for maintenance.

Use Case Name	View/Edit Reminder
Actor(s)	App User
Pre-Condition	The vehicle's details have been added.
Normal Course	The user can see/edit reminders for vehicle maintenance.
Post-Condition	The user can access other parts of the app.
Alternate Course	No reminder has been added yet.
Post-Condition	The user is allowed to add a reminder.
Assumptions	A vehicle's data has been added.
Priority	High
Frequency	High
Risk	Low

Table 9 - Use Case: View/Edit Reminders

1.1 Sequence Diagrams

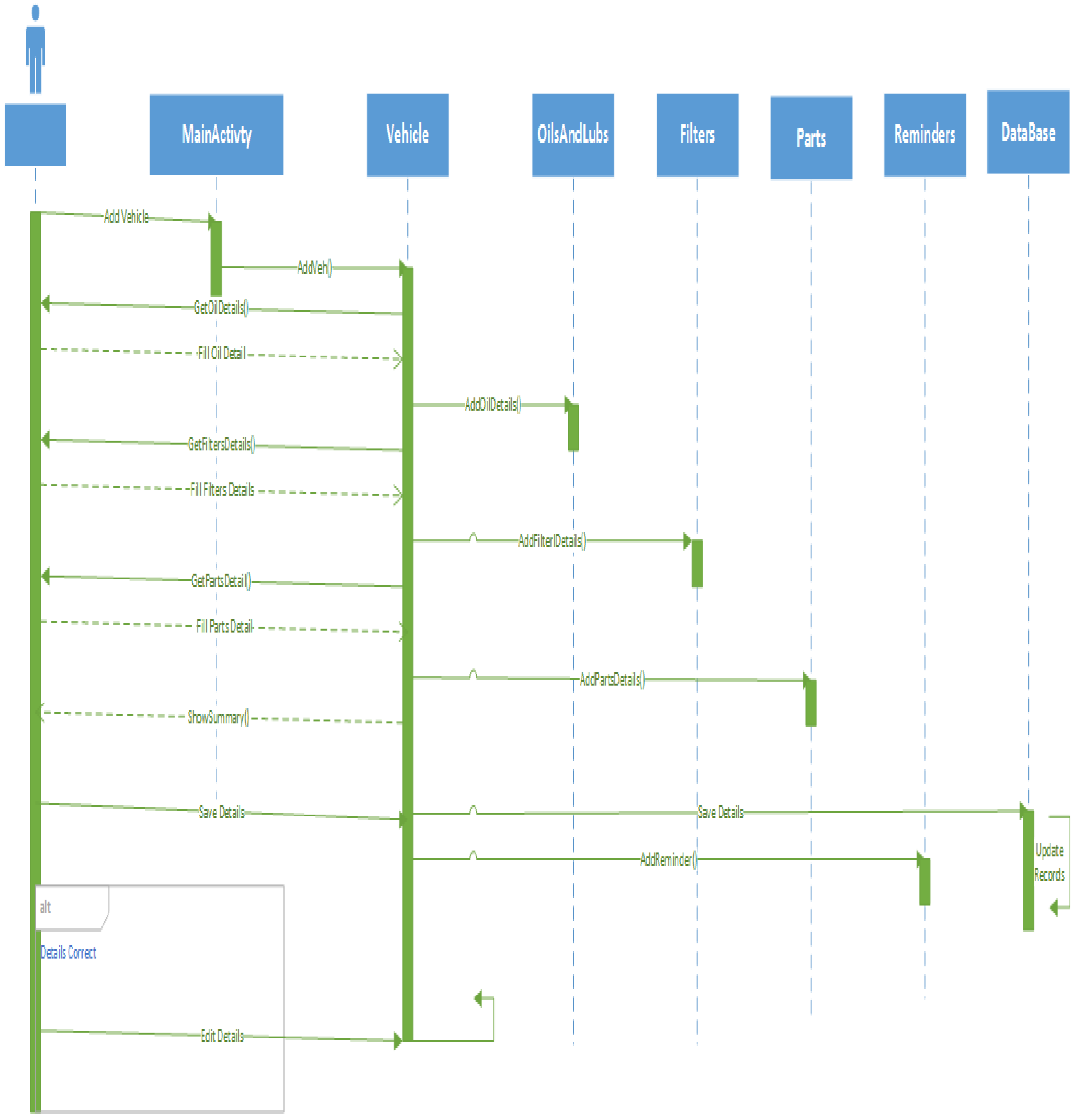


Figure 11 Sequence Diagram: Creating Vehicle Profile

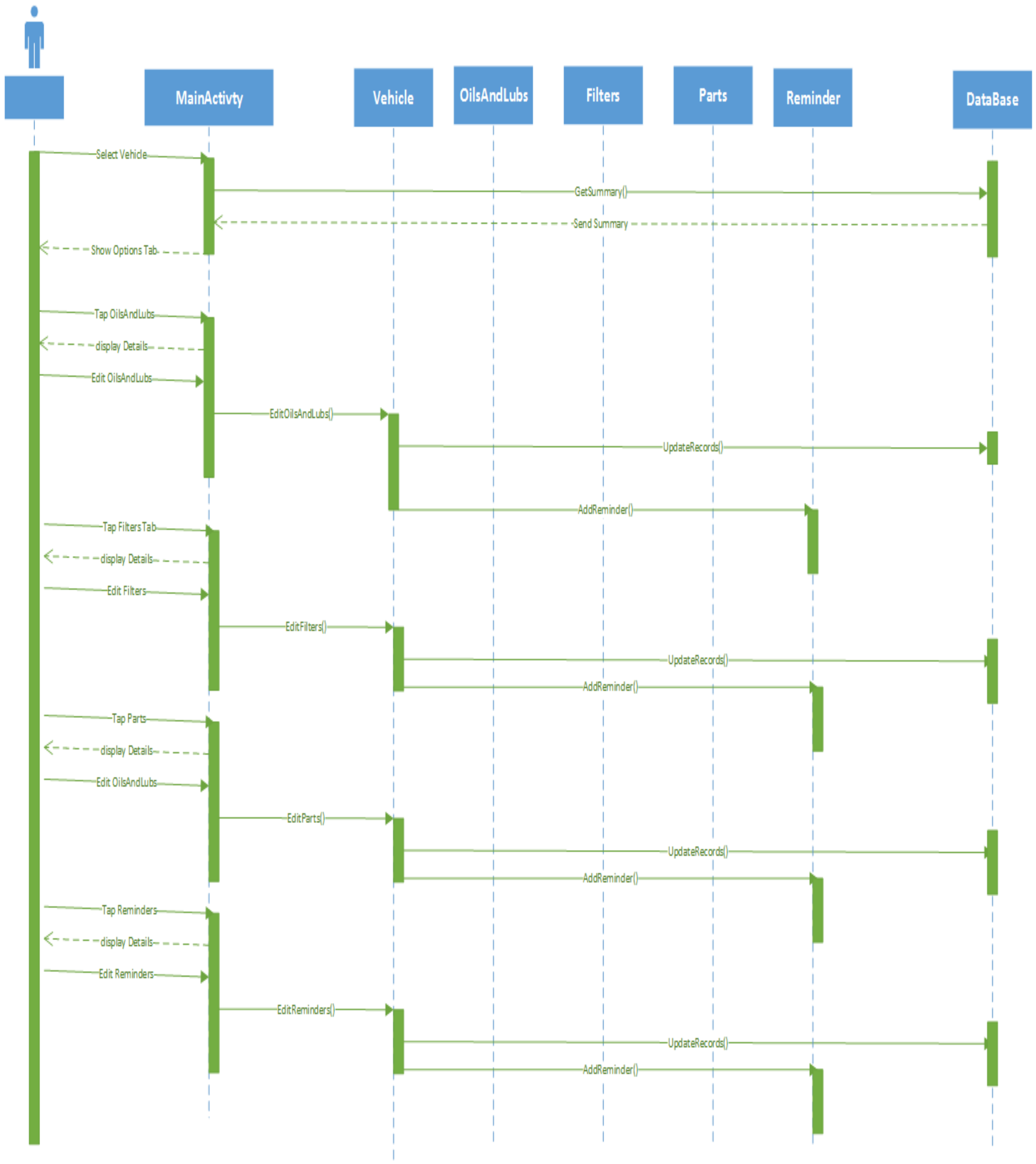


Figure 12 Sequence Diagram: View a Vehicle's Log

1.2 Activity Diagram:

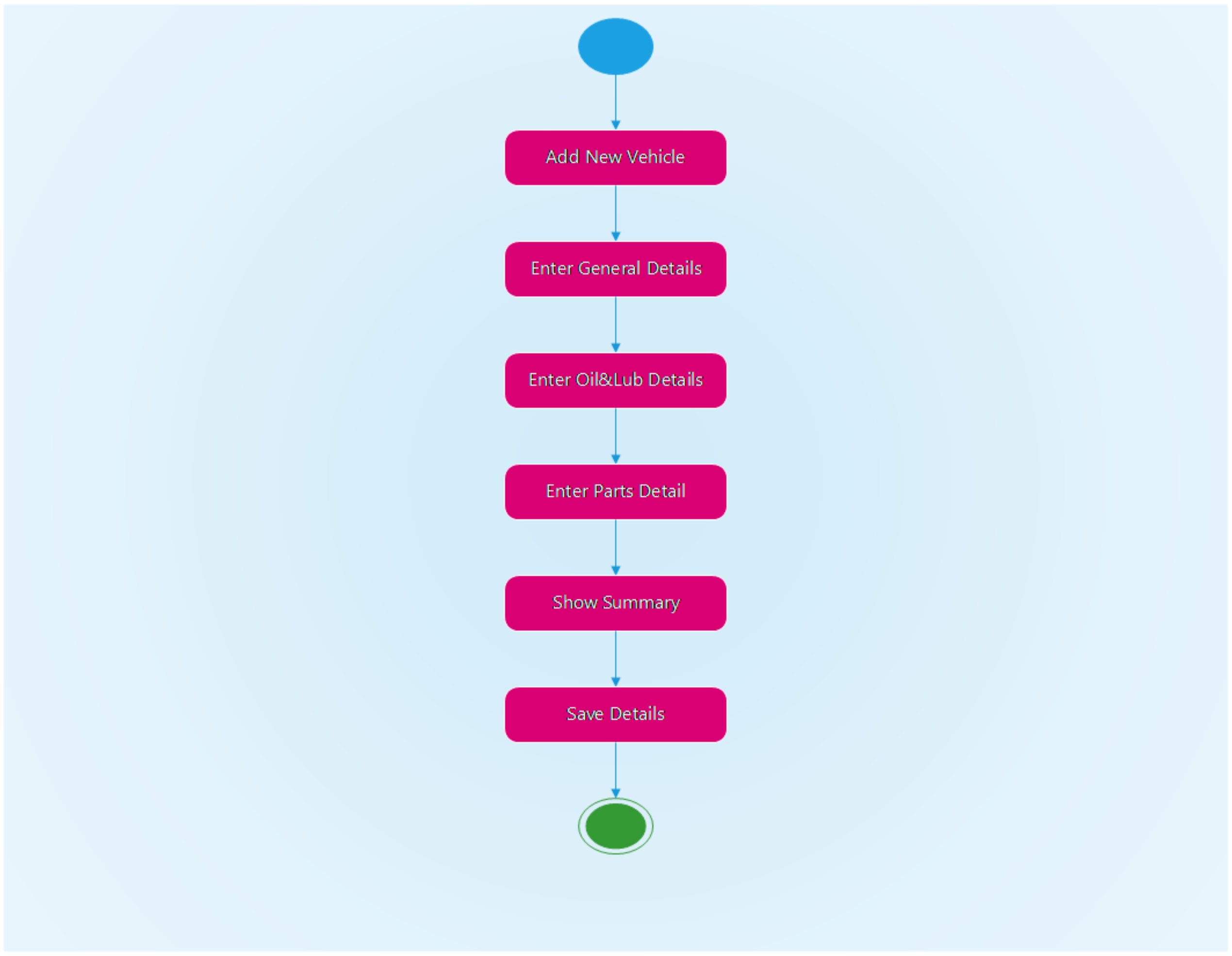


Figure 13Activity Diagram: Create Vehicle Profile

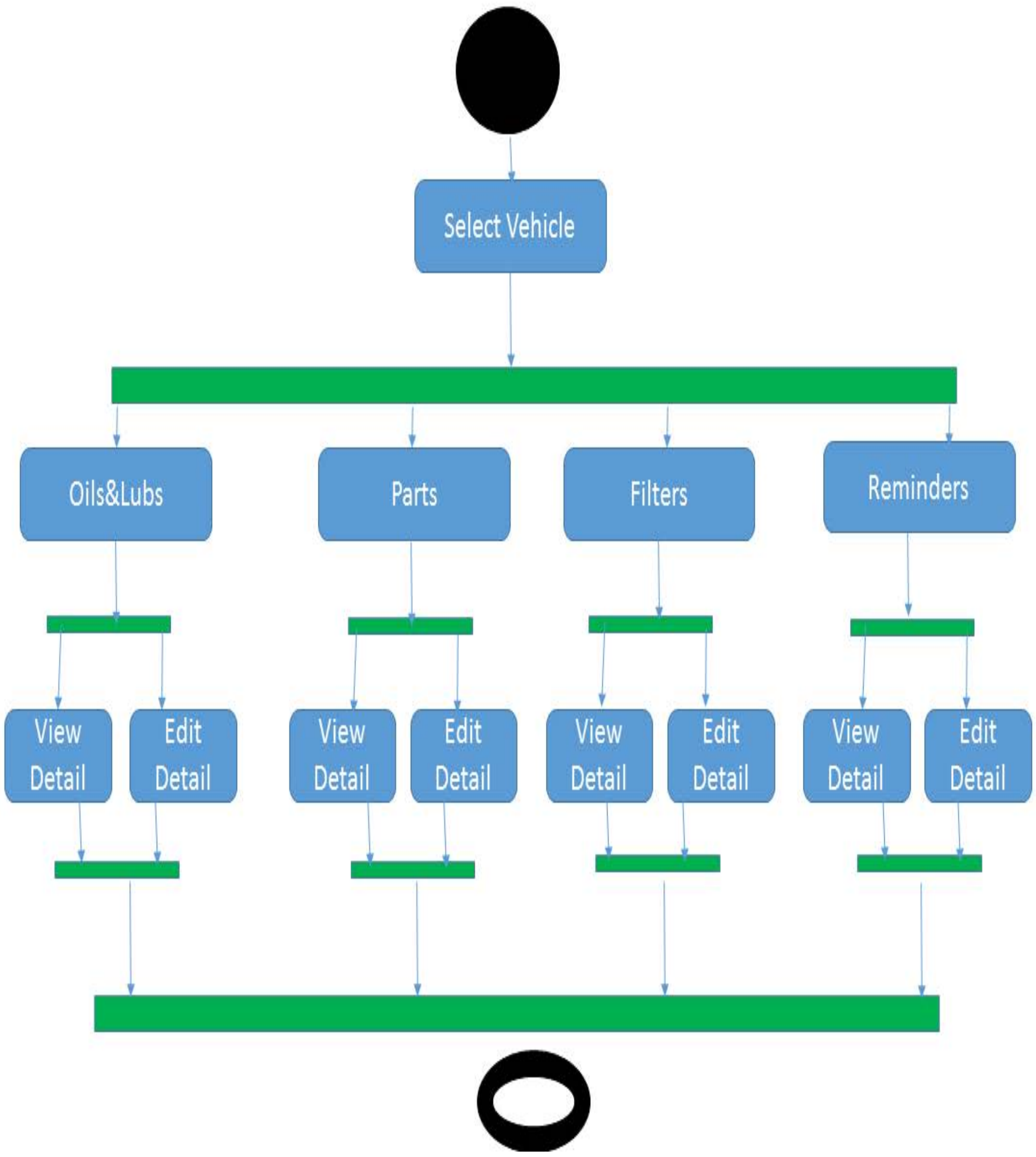


Figure 14Activity Diagram:Viewing a Vehicle's Log

1.3 Class Diagram

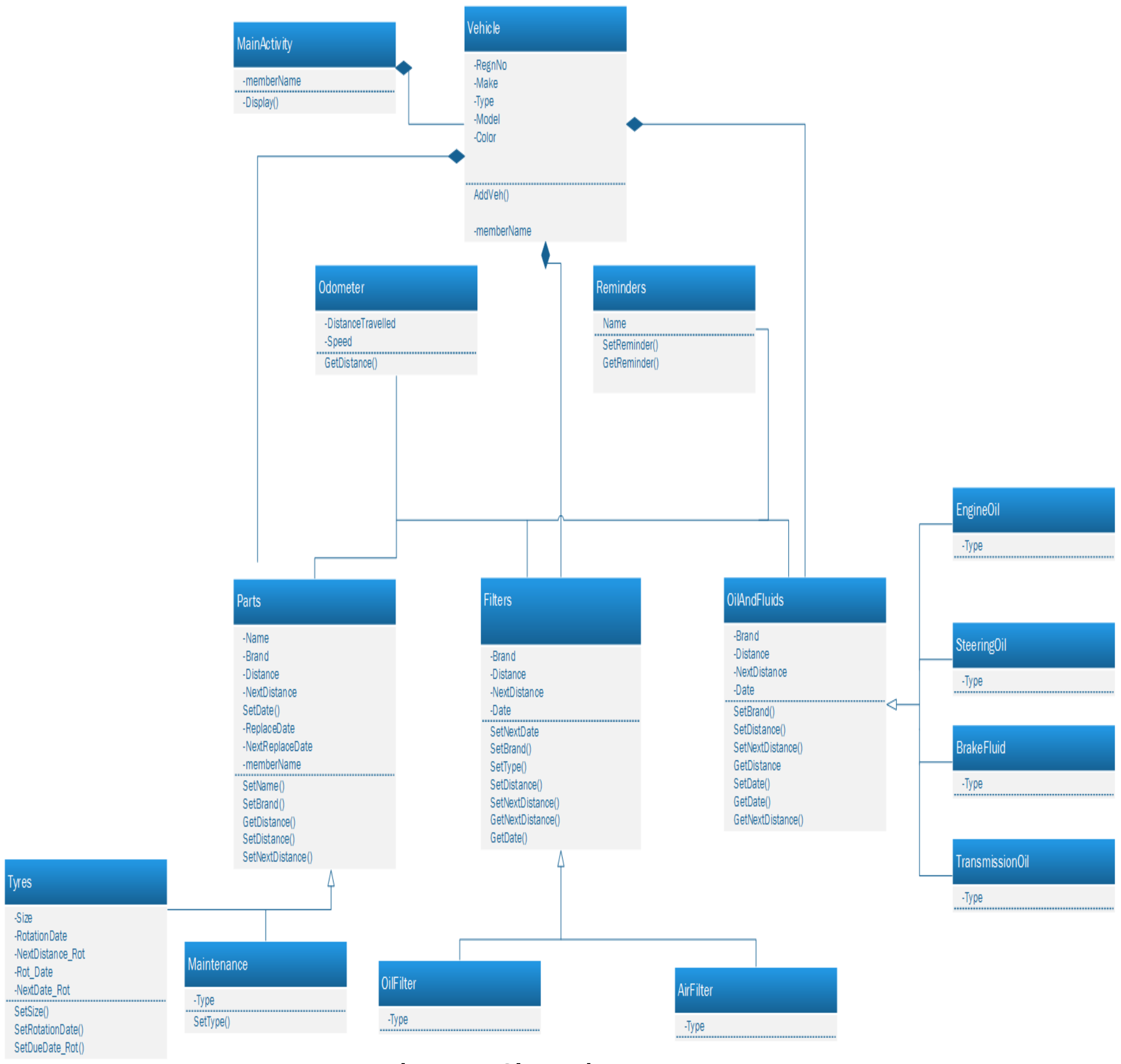


Figure 15 Class Diagram

The following table briefly describes the roles & responsibilities of these classes:

Class Name	Description
MainActivity	This class is responsible for managing, adding, deleting new clients in the network. It is also responsible for monitoring the overall working of the network.
Vehicle	This is the Façade class of the app. It encompasses all the other classes which represents various parts and features of the vehicle. It provides an integration to the various small classes.
OilAndLubs	All the information and the data about the various kinds of Oil changes and the distance will be maintained by the objects of this class.This is a parent class with various child classes denoting various kinds of Oils and Lubricants.
Parts	This parent class will generalize various kinds of parts of the vehicle which need to be maintained for the complete record keeping of the vehicle.
Reminders	The reminder class will keep track of the reminders associated with each part or maintenance carried out on the vehicle and will generate the reminders on completion of the required distance.
Odometer	The job of this class is to keep record of the Odometer readings. Maintaining a record of distance as well as speed while travelling.

WORK BREAKDOWN STRUCTURE

The application should collect driving distance information from the integrated iPhone sensors and keep logging the driving activity inside an internally integrated database. The driver
should provide initial mileage as an input when the app is downloaded and installed and thereafter, the driver must add refueling checkpoint data manually.
The application should inform the driver on requirement his car fuel average from last refueling checkpoint to the current refueling. For the time being, we shall include only this module. Later on, additional modules may be added as well like tracking periodic maintenance
points for the vehicle, changing of Engine Oil, Transmission Oil, Air Filter, Fuel filter etc.
OUTLINE VIEW

1. Vehicle Management System
1.1. Initiation
1.1.1. Evaluation & Recommendations by Group Members
1.1.2. Develop Project Scope
1.1.3. Deliverable: Submit Project Scope
1.1.4. Project Supervisor Reviews Project Charter
1.1.5. Project Scope Signed/Approved
1.2. Planning
1.2.1. Create Preliminary Scope Statement
1.2.2. Determine Project Resources
1.2.3. Project Team Kickoff Meeting
1.2.4. Develop Project Software Requirement Specifications
1.2.5. Submit SRS document
1.2.6. Milestone: SRS Approval
1.3. Execution
1.3.1. Project Kickoff Meeting
1.3.2. Verify & Validate User Requirements
1.3.3. Design System
1.3.4. Procure a Mac System & iPhone (Hardware) / Xcode IDE (Software)

1.3.5. Install on iOS device

1.3.6. Testing Phase

1.3.7. Install completed application (dev version)

1.3.8. User Training

1.3.9. Publish app and demonstrate

1.4. Control

1.4.1. Project Management

1.4.2. Project Status Meetings

1.4.3. Risk Management

1.4.4. Update Project Management Plan

1.5. Closeout

1.5.1. Audit Procurement

1.5.2. Document Lessons Learned

1.5.3. Update Files/Records

1.5.4. Gain Formal Acceptance

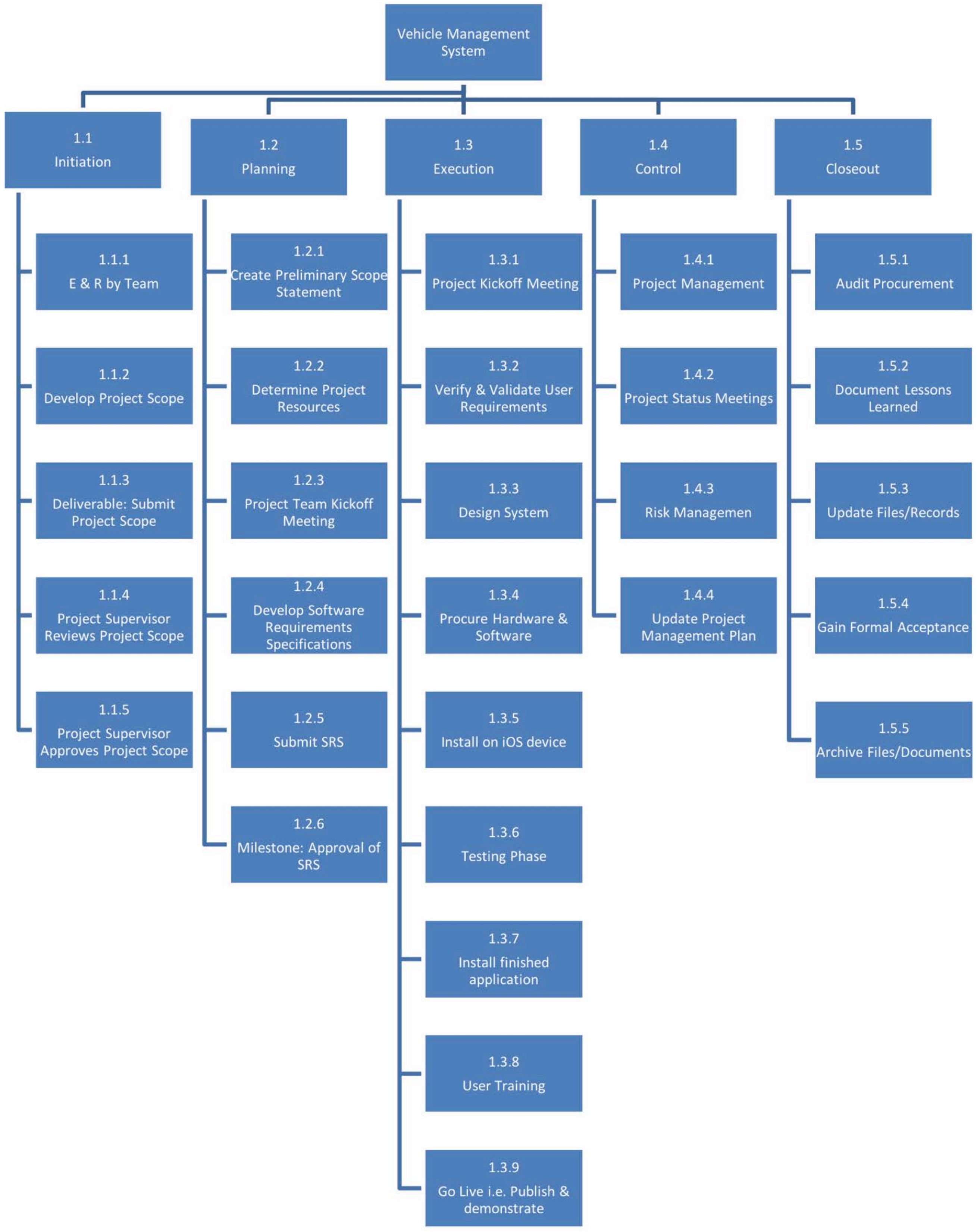
1.5.5. Archive Files/Documents

Tabular View

Level 1	Level 2	Level 3
1 Vehicle Management System	1.1 Initiation	1.1.1 Evaluation & Recommendations by Group Members 1.1.2 Develop Project Scope 1.1.3 Deliverable: Submit Project Scope 1.1 Project Supervisor Reviews Project Scope 1.1.5 Project Scope Signed/Approved
	1.2 Planning	1.2.1 Create Preliminary Scope Statement 1.2.2 De Resources 1.2.3 Meeting 1.2.4 Deve Requirements Specifications 1.2.5 Submit SRS 1.2.6 Milestone: Approval of SRS
	1.3 Execution	1.3.1 Project Kickoff Meeting Validate User Requirements 1 System 1.3. iPhone (Hardware) / Xcode IDE (Software) 1.3.5 Install o device 1.3 finished application 1.3.8 User Training 1.3.9 G demonstrate
	1.4 Control	1.4.1 Project Management 1.4 Status Meetings 1.4.3 Risk Management 1.4.4 Up Management Plan

	1.5 Closeout	1.5.1 Audit Procurement 1. Lessons Learned 1.5.3 Update Files/Records 1.5.4 Gain Acceptance 1.5.5 Archive Files/Documents

Tree View


Vehicle Management System
1.1 Initiation
1.2 Planning
1.3 Execution
1.4 Control
1.5 Closeout
1.1.1 E & R by Team
1.1.2 Develop Project Scope
1.1.3 Deliverable: Submit Project Scope
1.1.4 Project Supervisor Reviews Project Scope
1.1.5 Project Supervisor Approves Project Scope
1.2.1 Create Preliminary Scope Statement
1.2.2 Determine Project Resources
1.2.3 Project Team Kickoff Meeting
1.2.4 Develop Software Requirements Specifications
1.2.5 Submit SRS
1.2.6 Milestone: Approval of SRS
1.3.1 Project Kickoff Meeting
1.3.2 Verify & Validate User Requirements
1.3.3 Design System
1.3.4 Procure Hardware & Software
1.3.5 Install on iOS device
1.3.6 Testing Phase
1.3.7 Install finished application
1.3.8 User Training
1.3.9 Go Live i.e. Publish & demonstrate
1.4.1 Project Management
1.4.2 Project Status Meetings
1.4.3 Risk Managemen
1.4.4 Update Project Management Plan
1.5.1 Audit Procurement
1.5.2 Document Lessons Learned
1.5.3 Update Files/Records
1.5.4 Gain Formal Acceptance
1.5.5 Archive Files/Documents

Pseudo code For components

For Application UI

```
App Launched Successfully

begin

          Show Welcome_Message

          Show Main Screen

          If No New Vehicle Found

            Begin

             Add New Vehicle to database

               Calibrating the user's data

              end

           else

          begin

Go to Existing Vehicle Database

Select Vehicle

          Show Vehicle Stats Tab Bar Controller

                  end
```

```
end

else

        print 'App Crashed on Launch
```

4.3.1 ForLocation Sensor

```
begin

        Connect_Sensor

        Gather_Data

        Send Data to Application

end

else

        Throw error ~ No satellite data available
```

For Distance Logging

```
if Vehicle is mobile

begin

        Get values of distance from GPS Sensor

        Pass the values to distance calculation algorithm
```

```
    Store the fresh values into the database

    Update labels in Vehicle Stats Screens (Maintenance and Oils & Lubricants)



end

else

    print 'No Database found'
```

For Multiple Vehicles Management

```
begin

    Check the Vehicle in Database

    if vehicle exists

        retrieve data

    if vehicle does not exist

        then create new vehicle and add it to database

            gather data for vehiclecalibration

             if user selects edit option then ask for user input

             if user selects delete option then delete vehicle from database
```

```
end
```

Chapter 5: Project Analysis and Evaluation

Testing

Introduction

To ensure quality of the product, testing is conducted. Accuracy of functions performed by Summ-The-Search has to be tested and maintained to improve quality of software. Software testing techniques and results obtained are discussed in the coming sections.

Testing Levels

Separate modules are developed to provide different functionalities of Summ-The-Search. All of these modules are tested at different levels in their development and post-integration process. Different levels at which Summ-The-Search has been tested, and results obtained are described in this section.

Unit Testing

Unit testing involves the testing of each module at the completion and at times, during the very course of development of the module. It is a testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Summ-The-Search app has also been passed through unit testing process, wherein the various units have been tested in accordance with the anticipated output of each unit.

Integration Testing

Summ-The-Search's different modules which were developed and tested independently were also tested during integration to ensure system stability. Integration testing helped in ensuring that different modules when combined give complete functionality and nothing is missed or some functionality doesn't give error when integrated with other modules. Integration testing gave us more than 90% results ensuring that most modules were integrated with others as well as compatible. This shows that errors were minimized during integration testing.

System Testing

System testing was performed at the end of development and integration of Complete system was tested using sample data. Almost all of the test cases were successful ensuring that most of errors and bugs in the system were removed and system was stable enough to perform optimally.

Summary

Testing not only maintains the software quality but also improves overall usability of the project. At different stages of development suitable testing techniques were used to ensure product works accurately and efficiently. All errors detected during testing were removed and the test cases were prepared and made part of this document for the future compliance.

Test Case Name	Core Data Handler
Test Case No	1
Description	Data Storage & Persistence
Testing Technique Used	White Box
Preconditions	Atleast a single profile data was stored by user
Input Values	5 String & 9 Int 32 Bit
Valid Inputs	a. Non- Nil values for Strings b. Non- Zero values for Int 32 Bit
Steps	Specified Values input by user

Expected Output	*Core Data entity attributes to be accepted in valid range by the data model*
Actual Output	*The data model in Core data accepted the valid range of inputs by user for all attributes of entity*
Status	*Working*

Test Case Name	*Assisted GPS Testing*
Test Case No	*2*
Description	*Measurement of Distance data*
Testing Technique Used	*Black Box*
Preconditions	*Medium to High GPS signal reception*
Input Values	*Distance Covered in Kilometers*
Valid Inputs	*a. Non- Stationary distance data* *b. Speed greater than 15 KM/H*

Steps	*User cannot interact. Core Location Framework to handle distance data*
Expected Output	*Core Data entity attributes to be updated at each stamp of distance and the distance coverage should be updated correctly in data model*
Actual Output	*The data model correctly updated the distance factor in all attributes*
Status	*Working*

Test Case Name	*Profile Creation and Removal*
Test Case No	*3*
Description	*User's ability to create and remove cars*
Testing Technique Used	*White Box*
Preconditions	*Correctly entered profile data for each vehicle which is needed to be managed*
Input Values	*Provision of data of all valid values*

	for entity attributes
Valid Inputs	*a. Non- Nil values for Strings* *b. Non- Zero values for Int 32 Bit*
Steps	*Specified Values input by user and then displaying them in Table View UI*
Expected Output	*Object Oriented design shall enable the user to completely remove vehicle profile from both the UI View as well as Core Data Model*
Actual Output	*The data model was correctly updated when the user created or removed a vehicle profile*
Status	*Working*

Chapter 6: Future work

The project developed could then be used as a basis for further work in the field of medical science to help paralyzed patients communicate through brainwaves. If one could detect from a number of test cases the accurate letter which the person is thinking with the help of Neural networks then who knows one day due to technological advancement one could figure out the whole thinking process of the human mind and that could revolutionize the medical fields.

Chapter 7: Conclusion

Vehicle Management System (VMS) is a system that is proposed to develop a system that will automate the management aspects of a vehicle and will help keep track for scheduled maintenance activities.

Our project is only a humble venture to satisfy the needs of vehicle management. Several user-friendly codes have also been adopted. This package shall prove to be a powerful package in satisfying all the requirements of vehicle management. Last but not the least, it is not the hard work that paves the way to success but ALMIGHTY.