# Fleet Monitoring System

DESIGN DOCUMENT

sdmay18-19
Lotfi Ben-Othmane
Venecia Alvarez - Point of Contact
Kendall Berner - Project Manager
Matthew Fuhrmann - Report Manager
William Fuhrmann - Test Engineer
Anthony Guss - Technical Lead
Tyler Hartsock - Web Manager
sdmay18-18@iastate.edu
https://sdmay18-18.sd.ece.iastate.edu

Revised: 10-9-2017/1.0

# Table of Contents

# 1 Introduction

## 1.1 Project statement

Our project is to make a fleet monitoring system. We have been provided an Android embedded device that can interface with a vehicle's CAN BUS network. We will develop an Android application for the Android embedded device that queries the vehicle's components and relays the collected data to a server. This server will process and store the data. The server will also provide a website with real-time and historical data on location of all vehicles and internal data from the vehicles such as gas consumption and vehicle diagnostics. The website will also show useful interpretations and statistics of the data, and will allow website users to message the Android devices that are in the vehicles.

## 1.2 Purpose

The purpose of this project is to give fleet managers a better way of tracking information on the vehicles in their fleet. Fleet managers often find it difficult to know the current location of their vehicles or where their vehicles have been, and knowing this information helps them ensure that their vehicles are being used correctly and efficiently. Fleet managers also need better ways of predicting maintenance needs and vehicle operation costs. Data on fuel consumption and vehicle diagnostics can help fleet managers predict the needs of their vehicles. Having statistics on fleet operation will help fleet managers who are trying to improve the efficiency of their fleet.

## 1.3 Goals

We want to create a system that efficiently gathers and transmits information from vehicles, processes and stores that data, and displays it in a meaningful way for fleet managers. Our project has several different components involved. Our main goal is to have all those components working and integrated with each other before we get to our final deadline. In regards to soft skills, our team has set a goal to learn more about project management and agile development. Through this project experience, we will see what the entire software development process and life cycle really look like.

# 2 Deliverables

The webpage will be made for use by the manager of the fleet. It will have a simple and intuitive design made for ease of use. This web page will display a live map tracking the locations of all vehicles in the fleet in real time. A single vehicle can be selected, which will bring up a profile that will display information about that vehicle such as remaining gas, miles driven, driver, etc. The webpage will also have a page that details the stats of the fleet as a whole, so that the manager can see how much gas the entire fleet uses over a month, for instance. Graphs will be provided to make the statistics and information easier to parse.

Our server will be created using Node.js. The server will handle incoming raw data from the embedded devices and process this information into usable information that it will then put into a database. The server will also handle data request from the web page. Both the incoming and

outgoing data will be transmitted through REST API calls. This will allow us to use a single protocol across our entire system. The Node.js server will be hosted on the Google cloud platform, allowing us to scale our resources to the demands of our services.

The Android microcontroller will run an Android app that is used in the vehicles. It will connect to the vehicle's CAN BUS network and query for useful information. It will also acquire GPS data from a separate GPS hardware device. The microcontroller will store the data locally so that it can guarantee that it gets sent to the server. The microcontroller will be able to send data that it collects to the server, and will be able to receive messages from the server and display them on a screen connected to the microcontroller.

# 3 Design

## 3.1 System specifications

Our back-end is required to be written in Node.js.

### 3.1.1 Non-functional

The product shall:

- Be used by vehicles at any time and location
- Utilize Google Cloud services
- Send messages from the manager to the driver(s) in a timely manner
- Only allow managers to view fleet data on the dashboard
- Have the server side code made in Node.js
- Use AngularJS on the client side

### 3.1.2 Functional

The product shall:

- Gather data from a vehicle
- Transmit data from the vehicle to the server
- Process raw data from the vehicle on the server
- Record vehicle data into a database
- Display a map with a location of all vehicles in the fleet
- Display historical data for a certain vehicle (location, gas usage)
- Allow managers to register vehicles that belong to a particular fleet
- Display vehicle information only to users who have that vehicle registered to their fleet

### 3.1.3 Standards

We will use the agile development method. Our coding standards will align with the basic coding standards for each programming language used. We will also use vulnerability testing to ensure that the data can only be accessed by processes or processors with permission. This ensures that we will not be doing anything unethical. Coding standards are important for our project, because the client intends to modify our software in the future to fulfill the needs of future clients. Following coding standards for the programming languages we use will make fixing and modifying the software set much quicker, easier, and cheaper.

## 3.2 PROPOSED DESIGN/METHOD

Our team will develop three different components that will provide the manager with the information that they need on their fleet: an on-board Android microcontroller that will query the CAN BUS network of the vehicle and forward the data it collects to a server, a server used to handle incoming data from the Android device and send necessary data to the website, and a website that will be the interface that shows important information to a manager about their fleet. The server is required to be in Node.js, and will provide APIs for putting data into the database for vehicles and APIs for the front-end to get the information that they need. The front-end will be an AngularJS

website that presents location data on a map and vehicle and fleet statistics using charts and graphs.

The on-board Android device may require a USB GPS receiver in order to send location data to the server. Android location services often use triangulation through Wi-Fi connections to get location data, and if this is precise and reliable enough we will not need the USB GPS receiver. The on-board Android device does have the capability to use Wi-Fi that can be provided by a mobile phone's Wi-Fi hotspot feature. The majority of work for the on-board device will be creating an interface for the CAN BUS network. Because we are only trying to query data, we will mainly work on creating a background process that uses OBD-II PIDs to query data from the network and periodically sends that data to the server.

The server will be designed to be extremely modular. This will be done by separating our implementation of the API into a model, a controller and a route to the API. This will allow each implementation of the API to easily be modified, and will make it easy to add new API calls to the server. We have created an example API that can be used as an example of how to implement a new API call. The server will handle incoming data from the Android device through POST and PUT API calls. The server will send data to the web front end through GET and DELETE API calls.

## 3.3 DESIGN ANALYSIS

We originally planned to use AngularJS on our front-end. This has proven to be a slight challenge, and our team is considering if it is necessary to use this framework at all anymore. We plan on continuing to experiment with AngularJS and trying to integrate it with our existing prototypes for the map and charts for at least another week. Our plan is to pivot towards a different framework if AngularJS seems to be more trouble than it's worth moving forward.

The work for the Android on-board device so far has mostly entailed understanding the hardware and finding out what other hardware we need. We have prototyped all of the functionality not related to the CAN BUS network querying and our ideas seem to work. We do not have the cables and hardware yet to fully test the CAN BUS interfacing hardware, and we are working with our client and the vendor of our microcontroller to get the proper materials needed to begin working on our project. The library that the board works with for CAN BUS interfacing was only provided as a .so file, so we are not entirely sure of what functions it provides us. We do have an example of using the library in Android using 'native' API calls, and those calls seem to be very low-level, so we will likely spend a large amount of time developing the CAN BUS interfacing code from the ground up. We are trying to get the source code of the library, both so that we can understand how the library works, and so that we can eventually have a product that works with more types of processors, as the compiled library will only work with ARM processors.

We originally intended to implement our server using a micro-service architecture using Java and Spring. However, our client made it a requirement that our server be written in Node.js. Because of this, we lost a week of time researching and prototyping the old server. After transitioning into a Node.js server, we were able to prototype our API that will be used by the Android device and the Website. Currently, the API is using a basic model for the data. This model will be changed and improved as we make more progress with the Android device.

# 4 Testing/Development

## 4.1 INTERFACE SPECIFICATIONS

We are using an Android microcontroller to query data off of the CAN-BUS network of a vehicle. We will develop an Android app that will query the CAN-BUS network for useful information and send that data along with location data from the Android device to our google cloud server.

## 4.2 HARDWARE/SOFTWARE

We will use JUnit to unit test for the Android application.  We will also use a CAN-BUS OBD-II simulator that allows the application to query for dummy data and ensure that all the parts are functioning as intended.

In order to test our server, we will be using Postman to verify our API and perform regression testing. Postman allows us to setup automated tests that will run whenever we update the API for the server.

## 4.2 PROCESS

The server was tested using Postman. This allowed us to validate that the API calls were working correctly. We used Postman to setup automatic tests to ensure that the API calls remain correct. This is done by calling a specific API call and validating the response from the server. These tests will grow more complex as our models and data grow as we implement new features.

# 5 Results

On the front end, we have successfully created several prototypes of what we like our final dashboard for fleet managers to look like. It will use the Google Maps API to display a map with the location of all of the vehicles in the fleet. It will also use Chart.js to make multiple different types of charts that will display vehicle/fleet statistics. So far, prototypes of maps and charts have been made that can be modified going forward to meet the specific requirements of our front-end and to use real data from vehicles. One challenge we recently faced was setting up our environment to run our server in order to hit our API endpoints on the front-end. Through collaboration and research, we were finally able to get our application up and running. Another challenge we are facing on the front-end is using AngularJS to its full capabilities. To progress in this area, we plan on doing a lot more research and practicing AngularJS with examples we find online.

On the Android microcontroller, we have so far been unable to query the CAN BUS network. We have figured out that the reason the connection wasn't working is because we have been using the wrong connector. Our client has ordered the correct connector and we should be able to query the CAN BUS from the microcontroller soon. As far as Android programming, we have created a prototype application that gets latitude and longitude and a prototype application that builds a JSON object from dummy data and sends it to the server via POST API. This should give us a good foundation for when we get the CAN BUS data query working. To complete this part of the project, we need to get the microcontroller to get data from the CAN BUS network, combine the prototype applications, and make design decisions as to what types of data the fleet manager will want to see from the vehicle.

On the server, we have successfully prototyped some basic API calls. The first API call allows the Android device to send data containing the date and time, longitude and latitude, and speed. The Android device also sends a unique identification number (UID) with this data which is used to assign the data to a specific vehicle. Using a GET API call, the website can access this data based upon the UID. In order for the data to make sense for a specific manager, we created a manager model. This model contains a login system for managers to login, and shows which vehicles the manager owns. This will allow the website to pull the specific vehicle data for a manager from the server. We have also implemented the ability to delete vehicles, both from the data set and from a manager. This allows old data to be removed if, for example a vehicle has been sold or is no longer in use. The data used for the vehicles is currently minimal containing only location, speed and date. This will expand as we add new features. For example, a feature we want to add is fuel management. To do this, we will have to have data regarding the current levels of fuel. Once we have this data, the server will accept this data and make it available to the website. The website will in turn use this data to provide the manager information about fuel usage.

# 6 Conclusions

The goals for our team are to create a viable fleet monitoring system that provides managers with relevant information on the vehicles in their fleets by collecting data from the vehicles, and to gain more experience with the development process.

We have successfully created prototypes for the map and various charts that we would like to have on our fleet manager dashboard. At this point, the client can make calls to the server API to grab data and display that data on the front-end.

We have created a prototype Android application that can use dummy data and send it to the server in JSON format. This application holds a data class that will be used for data storage on the Android device before it is sent to the server. Our application allows the driver to start a trip, input their employee id, and put logistical data to the server. We also has a prototype application that prints the current location of the device. From here we will need to combine the two prototype applications and get the microcontroller to be able to query the CAN BUS network.

We have successfully created a prototype for our server. This includes basic API calls that allow incoming data from the Android device, and outgoing data to the website. We have created a good base for our server, and it will be easy to add new data as well as new API calls to implement new features going forward.

Our plan is to work primarily on getting the connection to CAN BUS network working so that we can begin to collect real data from cars and pass that to our server. While this is going on, some work will be spend on preparing the server and front-end to handle the data when we begin collecting it, all though much of the prototypes necessary for this have already been completed. This is the best possible plan because it helps us create the framework to provide managers information while focusing on the aspects that will be the most difficult and are causing us the most uncertainty for estimation. Once we finish the data estimation, we should be able to focus on adding features and improving the product.

# 7 References

List any references used in the document. These are an essential part of your review so far.

# 8 Appendices

If you have any large graphs, tables, or similar that does not directly pertain to the problem but helps support it, include that here. This would also be a good area to include hardware/software manuals used. May include CAD files, circuit schematics, layout etc. PCB testing issues etc. Software bugs etc.