Fleet Monitoring System

DESIGN DOCUMENT

Team #12
Lofti Ben Othmane
Joe Herrera - Mobile Developer
Marco Yepez - Embedded System and Mobile Developer
Nicolas De la Cruz - Back-end Developer
Lorenzo Chavarria - Web Application Developer
sddec20-12@iastate.edu
http://sddec20-12.sd.ece.iastate.edu/

Executive Summary

Development Standards & Practices Used

- Agile Development Standards
- JavaScript Coding Standards
- Google Java Style Standards
- Oracle PLSQL Coding Guidelines

Summary of Requirements

- Develop Android and iOS mobile applications for Fleet Monitoring System
- Develop a web application for Fleet Monitoring System
- Provide GPS location of each vehicle in a fleet
- Provide data analytics of vehicle data
- Provide client chat
- Provide SMS notifications

Applicable Courses from Iowa State University Curriculum

List all Iowa State University courses whose contents were applicable to your project.

- CPR E 288
- COM S 309
- COM S 363
- COM S 352
- S E 319
- S E 329
- S E 339

New Skills/Knowledge acquired that was not taught in courses

- Server configuration
- Rest API creation
- Hardware equipment knowledge, e.g. resistors

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Introduction

1.1 ACKNOWLEDGMENT

Acknowledgment to team SDMay18-2018, the team with the first version of this project.

1.2 PROBLEM AND PROJECT STATEMENT

With UPS, FedEx, and numerous fleet companies out there. We need something to monitor and manage these fleets. With our fleet monitoring system, a company would be able to use our user-friendly solution. Our solution revolves around web and mobile applications to view GPS locations of each vehicle, SMS notifications, client chat, and data analytics. Our solution would be based on MongoDB for our database and a server running Node.js Rest API for microservices. The fleet would need a PiCan for each vehicle to be able to send data from the vehicle to the server. Through the connectivity of each component, our solution would provide clients a fleet monitoring system.

1.3 OPERATIONAL ENVIRONMENT

The conditions for this product are regular conditions inside the car for the PiCan device. Regular conditions would be restricted to the device being dry, avoiding all liquids.

1.4 REQUIREMENTS

- Develop Android and iOS mobile applications for Fleet Monitoring System
- Develop a web application for Fleet Monitoring System
- Provide GPS location of each vehicle in fleet
- Provide data analytics of vehicle data
- Provide client chat
- Provide SMS notifications

1.5 INTENDED USERS AND USES

- Managers
  - Add users to groups
  - Accept new driver account request
  - Analyze data from vehicles
- Drivers
  - Submit driver account request
  - Indirectly produce usage data while driving

1.6 ASSUMPTIONS AND LIMITATIONS

Assumptions:

-There will be one manager and many groups that will consist of drivers. The manager will be able to control the different groups and see different data analytics.

Limitations:

-The idea of hardware is that it has to be in a small workspace.

-PiCan device can only be tested and used on cars of the year 2008 and newer.

-Limited knowledge of hardware and server.

1.7 EXPECTED END PRODUCT AND DELIVERABLES

Deliverable 1: Web Application - This web page will allow a user (manager or driver) to log in. As a manager, they would be able to view real-time tracking of all the vehicles currently being driven. A chat system will be implemented so that the manager can communicate with drivers.

Deliverable 2: Mobile Application - The mobile application, in practice will be similar to the web application. There are performance bottlenecks with mobile devices as they don't have the computing power of a computer. Therefore, the mobile may contain only a smaller amount of data analytics and may not have the performance to run real-time tracking.

Deliverable 3: PiCan - The PiCan is the hardware that will be installed on the vehicles to extract data and send it to the server. Data that is extracted from the vehicle includes mileage, fuel, speed, etc.

Deliverable 4: Server - The server will be a Node.js Rest API to provide Microservices for PiCan data posting and the client can request and post data to the database.

Deliverable 5: Database - Mongo Database to store user information, registered vehicles, registered drivers, and vehicle data.

The end product would include the web application, mobile application, PiCan, server, and database that all work together to create a fleet monitoring system. The fleet monitoring system includes hardware that must be installed on a vehicle. The hardware will communicate data to the server so that the manager and drivers can use the web/mobile application to view data analytics. The Fleet Monitoring System is set to be completed by December 2020.

2. Specifications and Analysis

2.1 PROPOSED APPROACH

Approach:

- Use MongoDB for scalability, as we need to collect extreme amount of data from vehicles

- Rest API for Microservices in Node.js using Microsoft Azure Web API Design
- Dynamic web page creation using Node.js and Express.js
- Google Java Style Standards

2.2 DESIGN ANALYSIS

In these weeks, we have met with our adviser and client to discuss what the previous team from a past semester completed. We discussed what and how we should implement for a successful project. Our team has been able to produce design prototypes for an Android and website app. We have also been able to successfully replicate what the previous team completed for the raspberry pi device and vehicle. In the end, we want to implement these front-end designs, refractor the embedded system program, and create a successful, sustainable backend. Our strengths will be that we are most familiar with how our product and the software will work. Our weakness is mostly that we lack some knowledge of hardware and server.
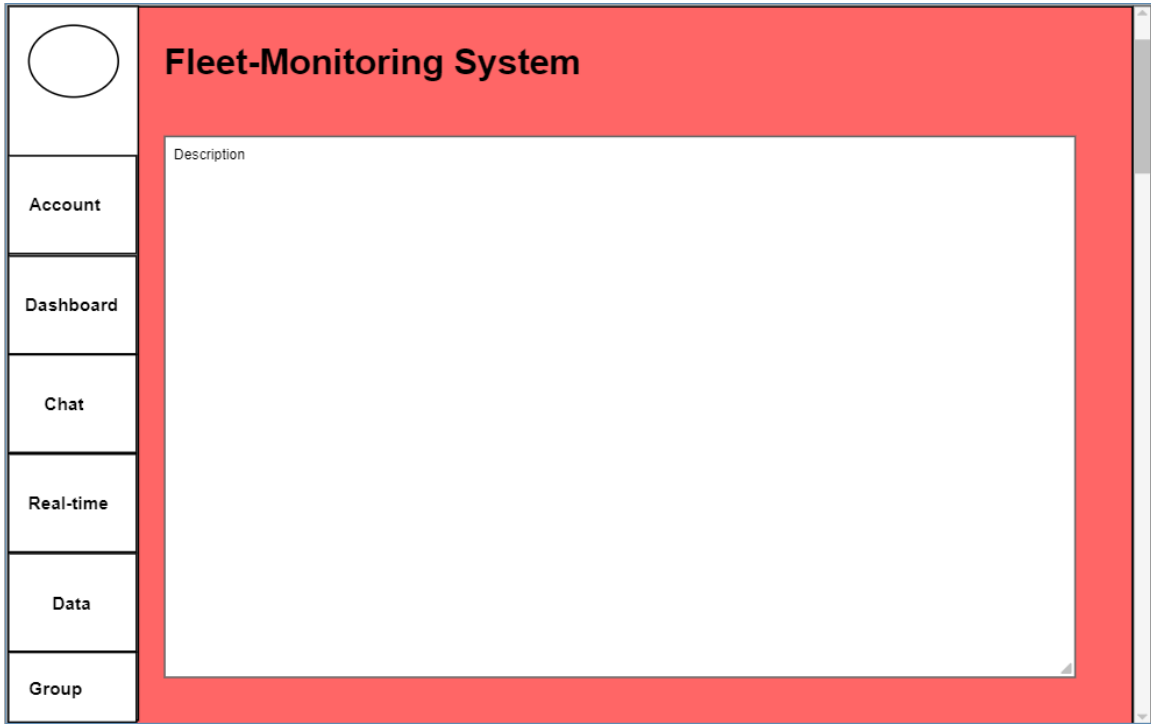
2.3 DEVELOPMENT PROCESS

Our team will follow the Agile software development process, our development sprints will be three-weeks long. We meet with our advisor and team weekly for updates, questions asked, and any other relevant topics for our project.

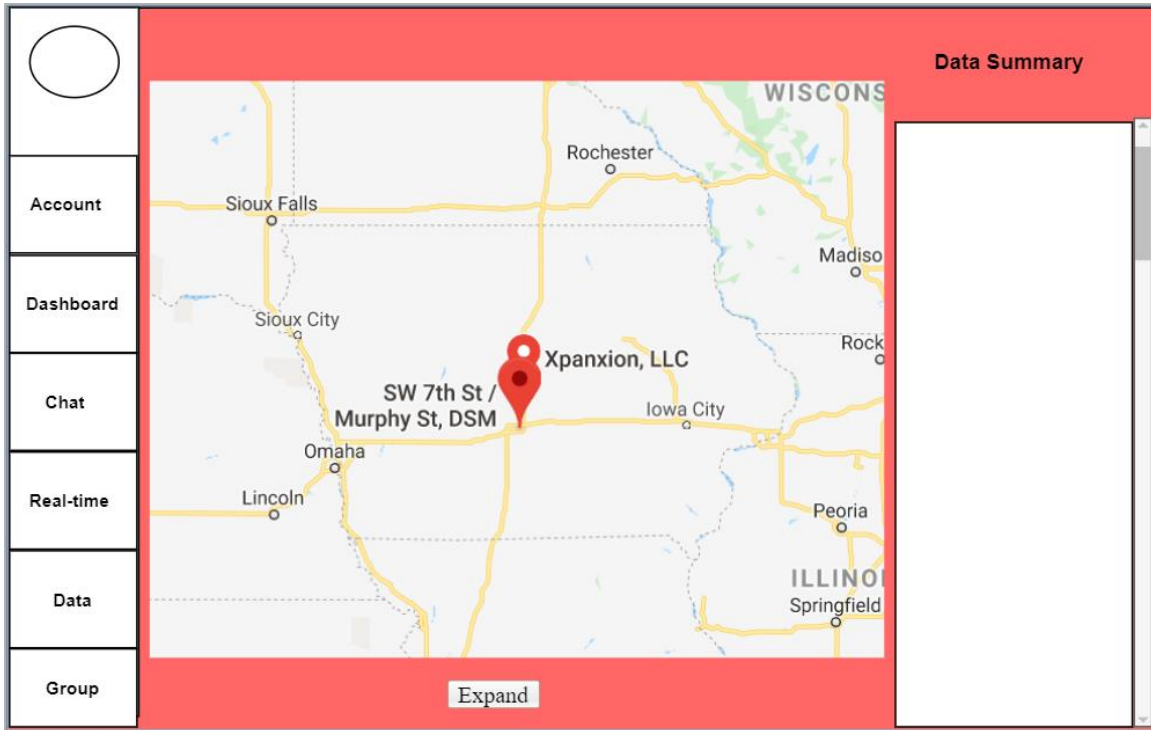## 2.4 CONCEPTUAL SKETCH

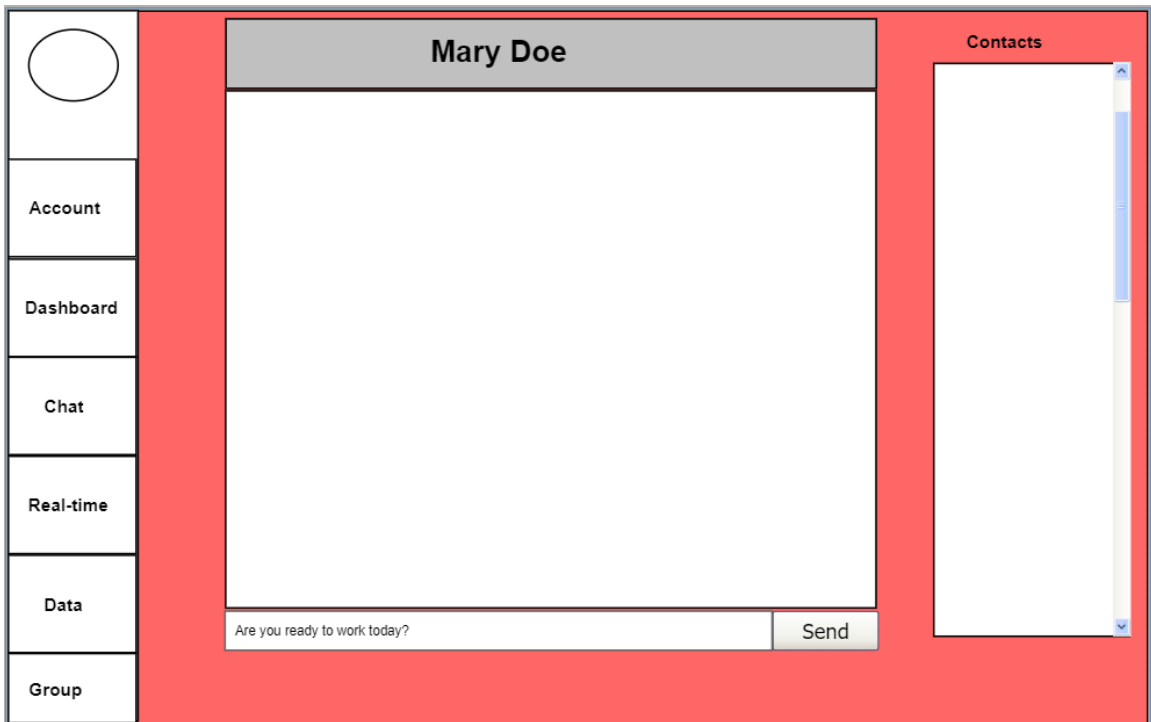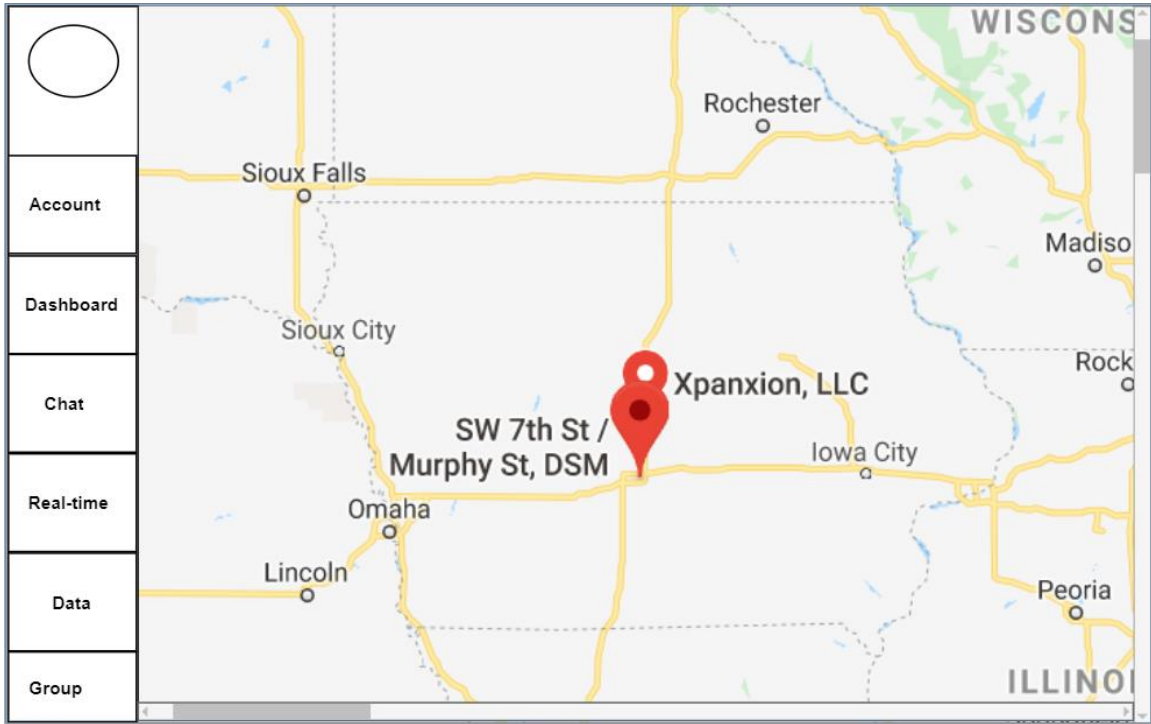**Database Architectural Design:**
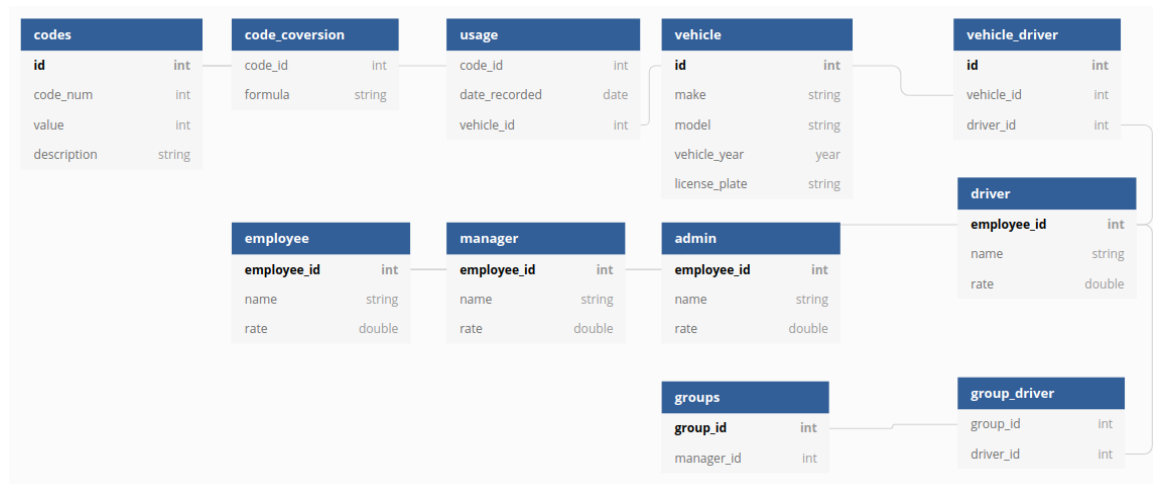
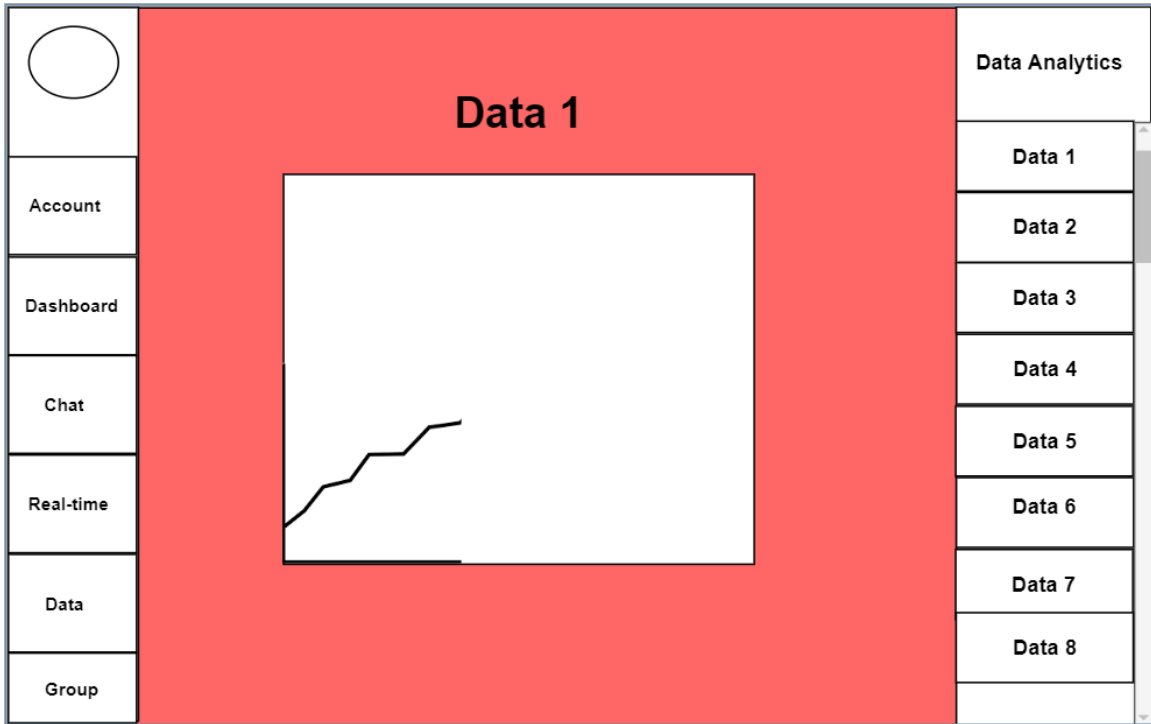**Web Application Design:**

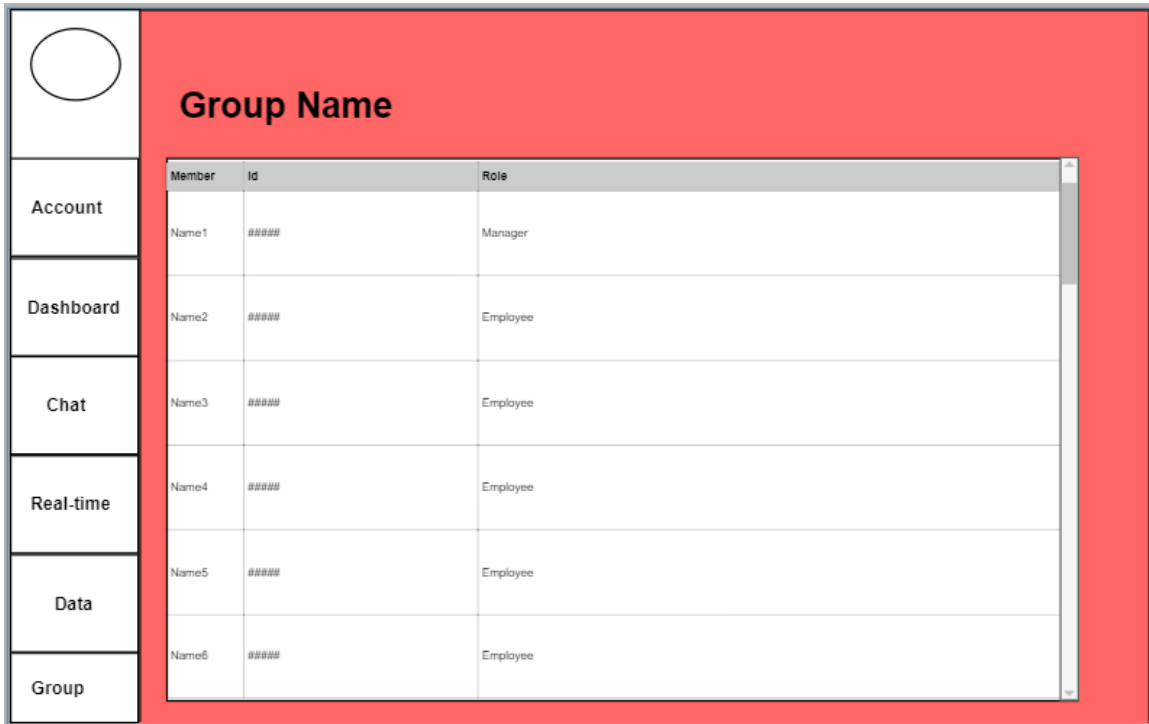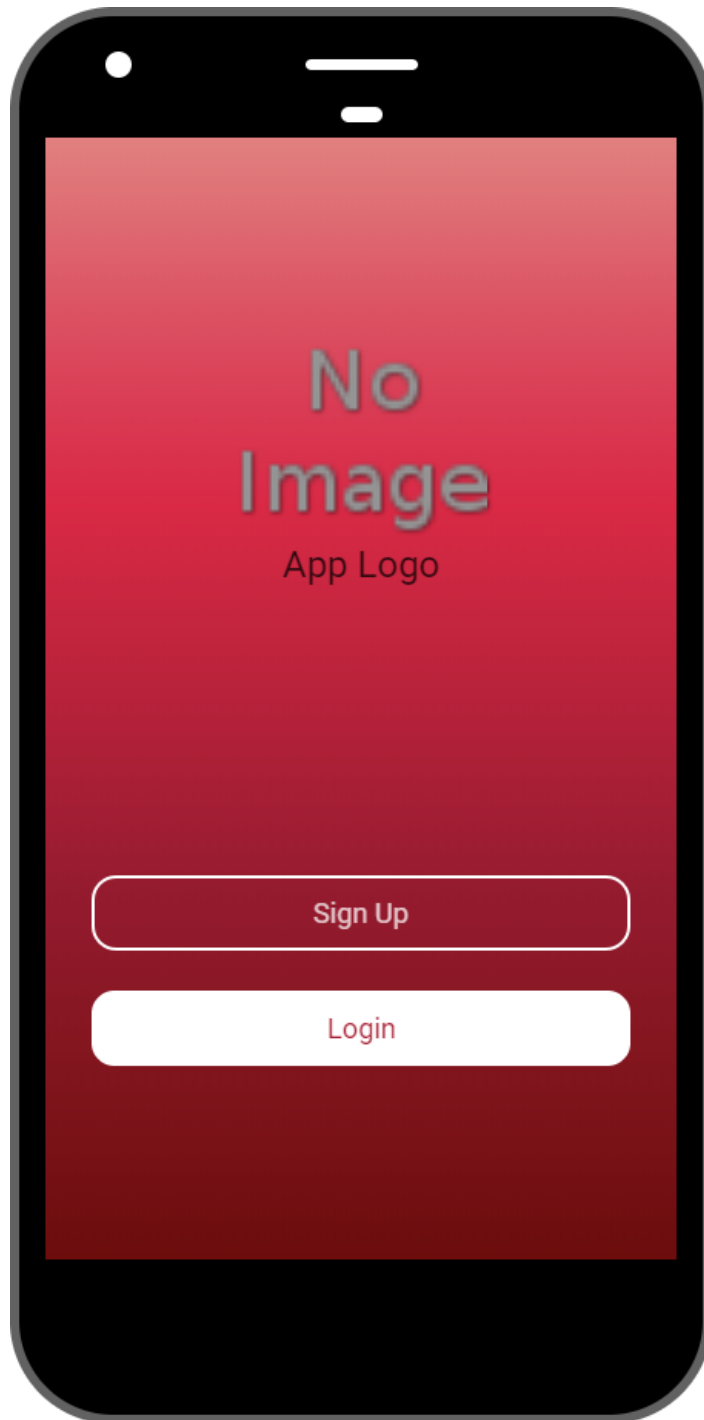Home Page

Account Page

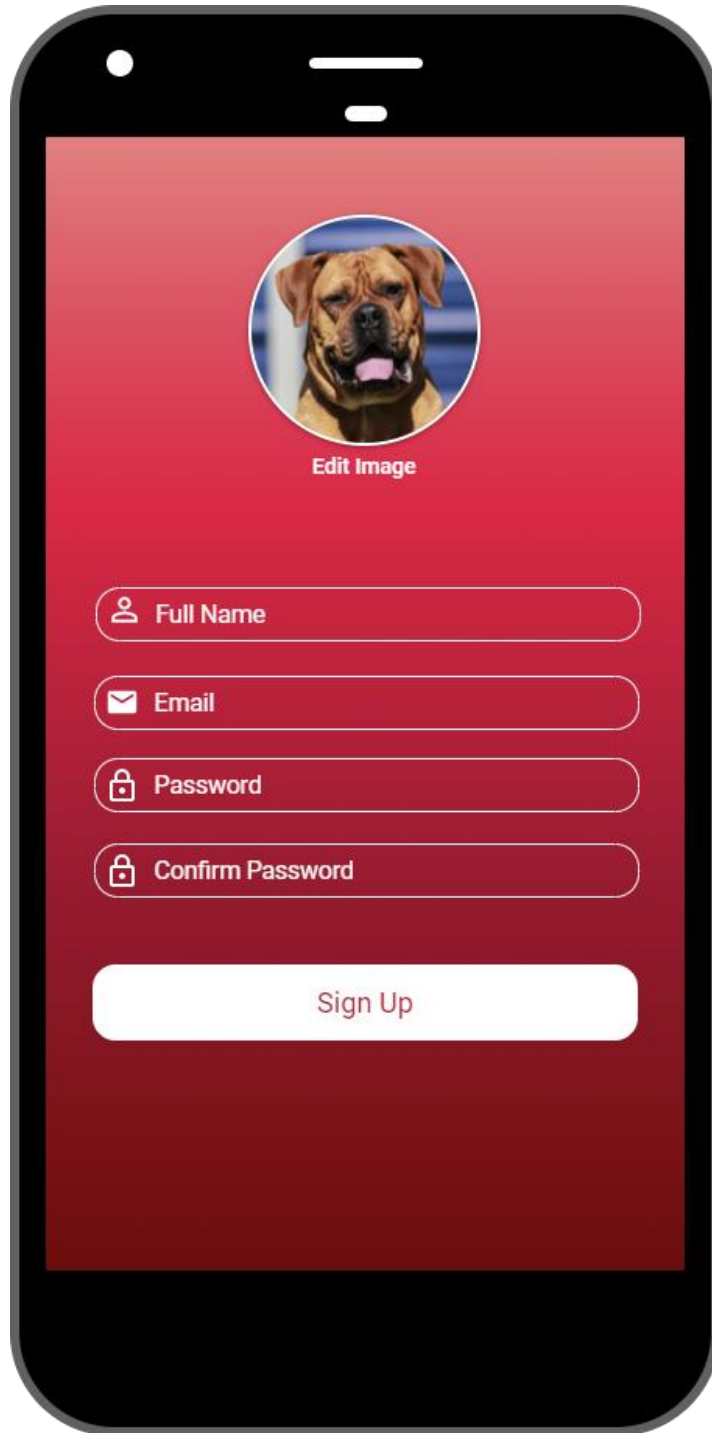Dashboard Page

Chat Page

Real-time Data Page

Data Analytics

Group Page

Mobile Application Design:

No
Image

App Logo

Sign Up

Login

Welcome View

Sign Up View
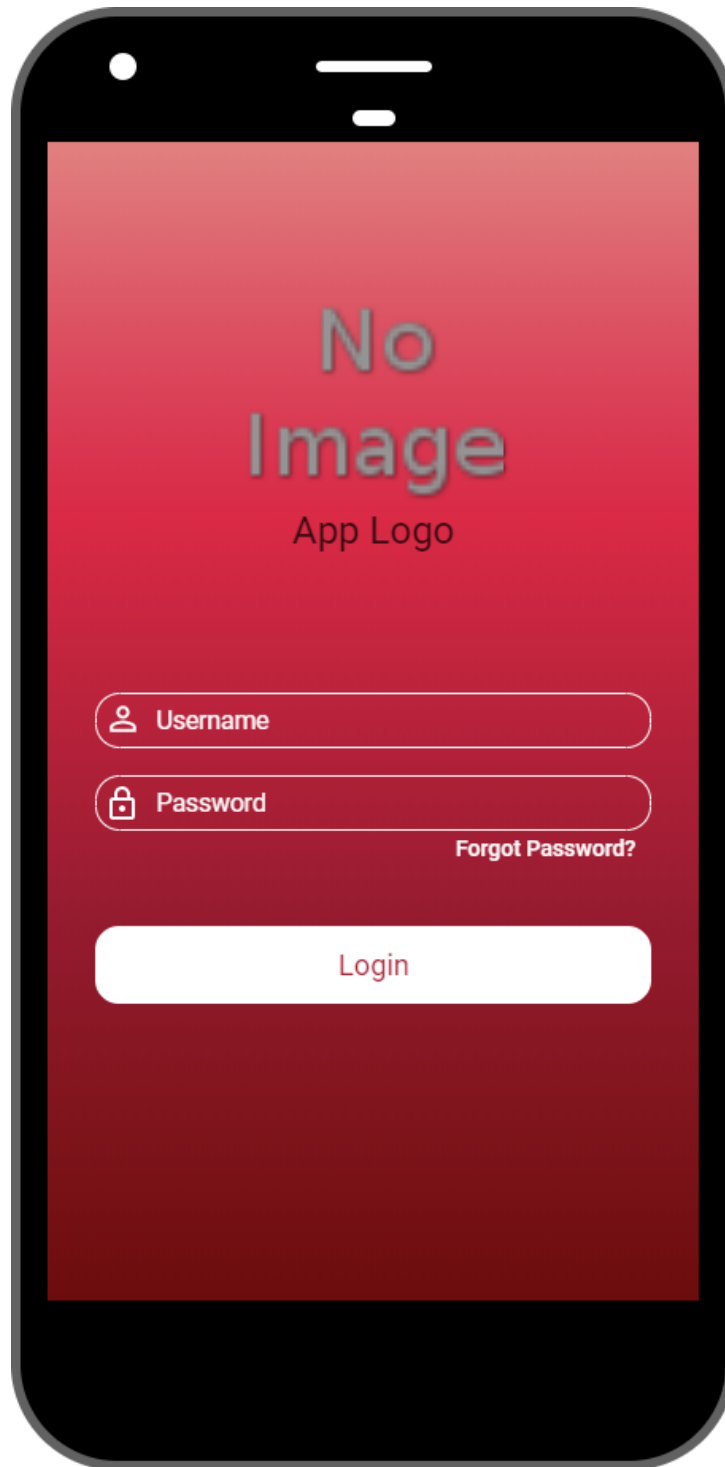
Login View

Dashboard View
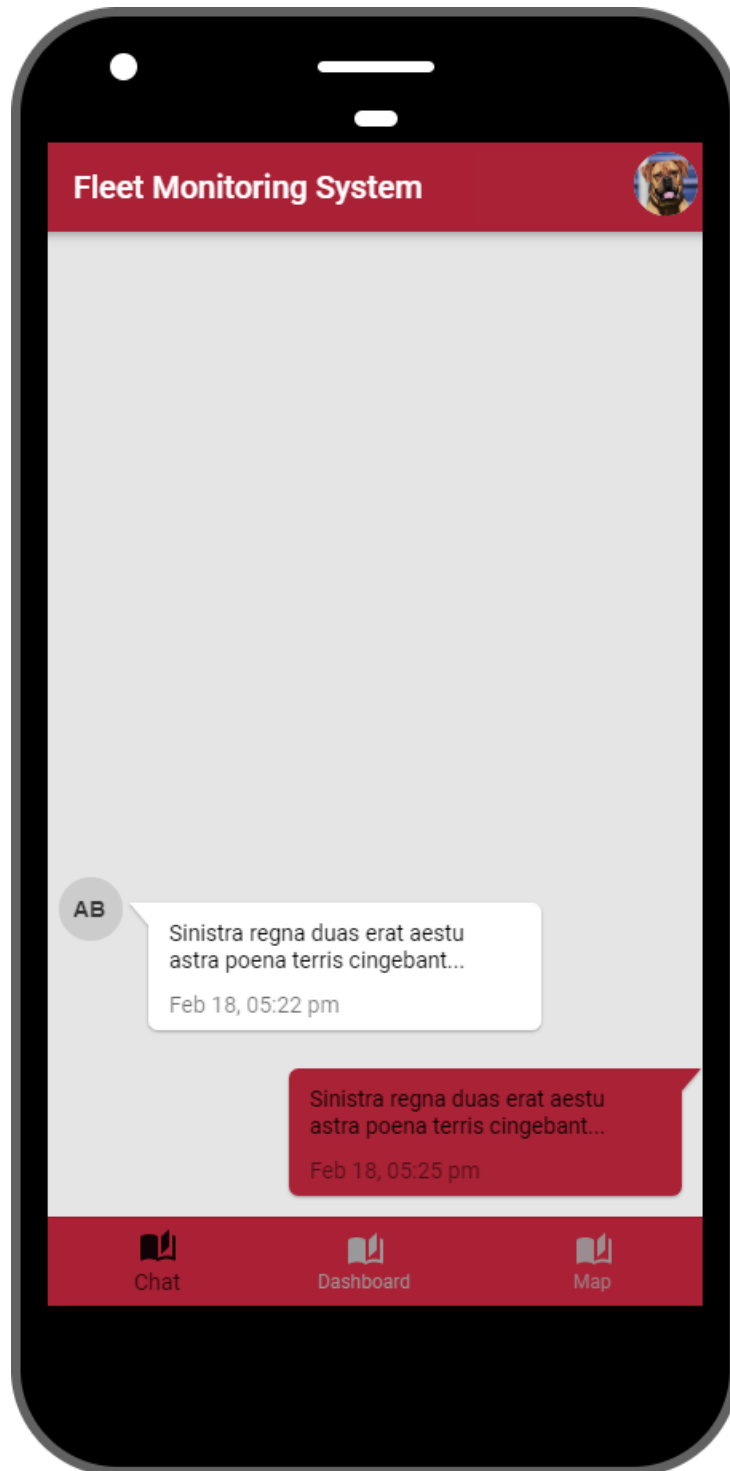
Chat View

Map View
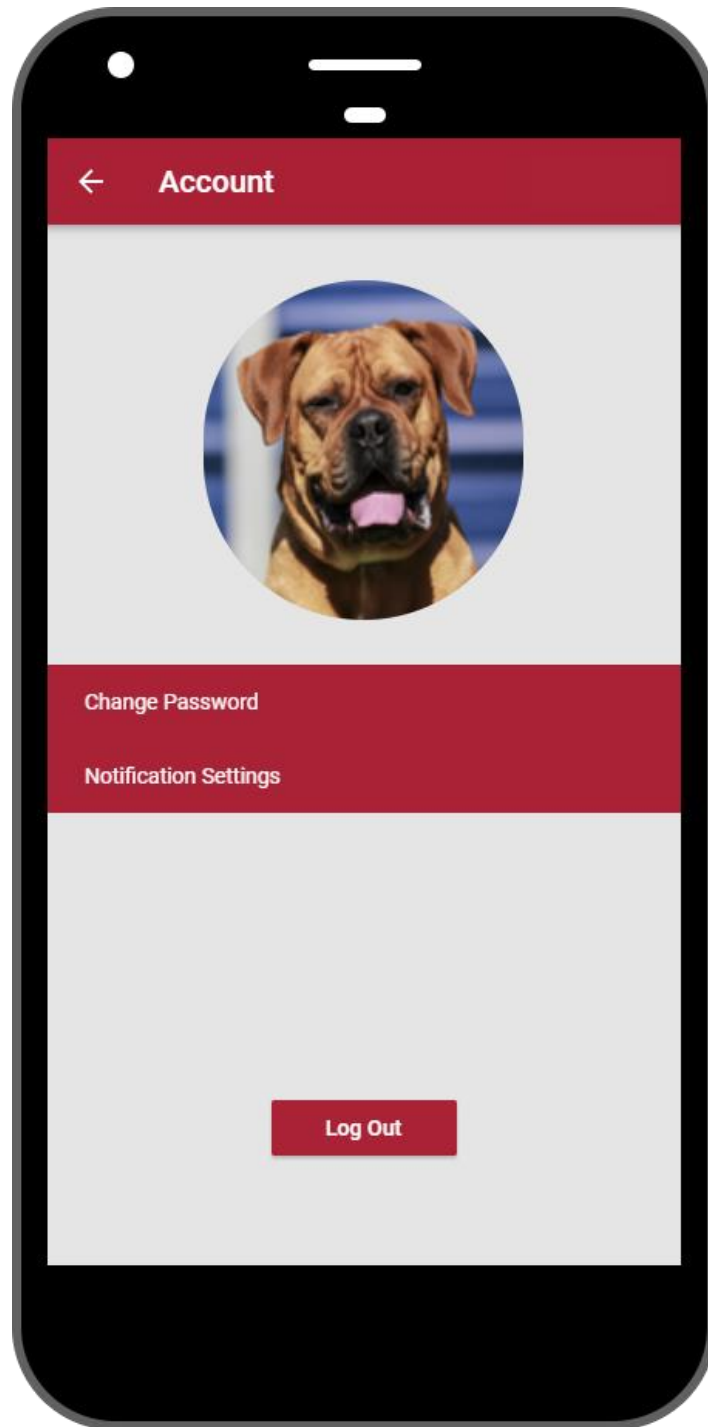
Profile Settings View

Embedded System Class Diagram:

Composition Classes to FleetMonitorApp

CANHandler

+ ID_FIRST_BYTE: int
+ ID_SECOND_BYTE: int
+ PID_REQUEST: int
+ PID_RESPONSE: int

+ __init__(self)
+ startup(self)
+ shutdown(self)
+ read_messages(self)
+ write_requests(self)
+ run(self)

Used to connect start and read nodes. While checking for new data, it'll send data using the API_ENGINE

ConfigStore

+ __Singleton: class
+ default_vehicle: dictionary
+ default_server: dictionary
+ default_mode: string

+ __init__(self)
+ get_pids(self)
+ get_vid(self)
+ get_did(self)
+ get_bitrate(self)
+ get_server_uri(self)
+ set_vehicle(self, vehicle)
+ set_server(self, server)

Class is used to configure vehicle, driver, and pids

GPSHandler

+ api_engine: API_ENGINE

+ __init__(self)
+ startup(self)
+ run(self)

Class is used to retrieve data from the device that the GPS has given it. The class sends latitude and longitude by using the API_Engine

API_Engine

+ __Singleton: class

+ run(self)
+ set_logging(self, log)
+ set_send(self, send)
+ pid_send(self, pid, data)
+ pos_send(self, lat, lon)
+ get_vehicle(self)
+ do_request(self, type, url, body)

Appears to be a class where external devices or program can send and get data from the server

Left-side of diagram (next image is right-side)

FleetMonitorApp

+ __init__(self)
+ startup(self)
+ run_hardware(self)
+ run

Class wraps all the other composition classes together. Here it starts the CAN, GPS, and log

retrieve data from
the GPS has given
ends latitude and
ng the API_Engine
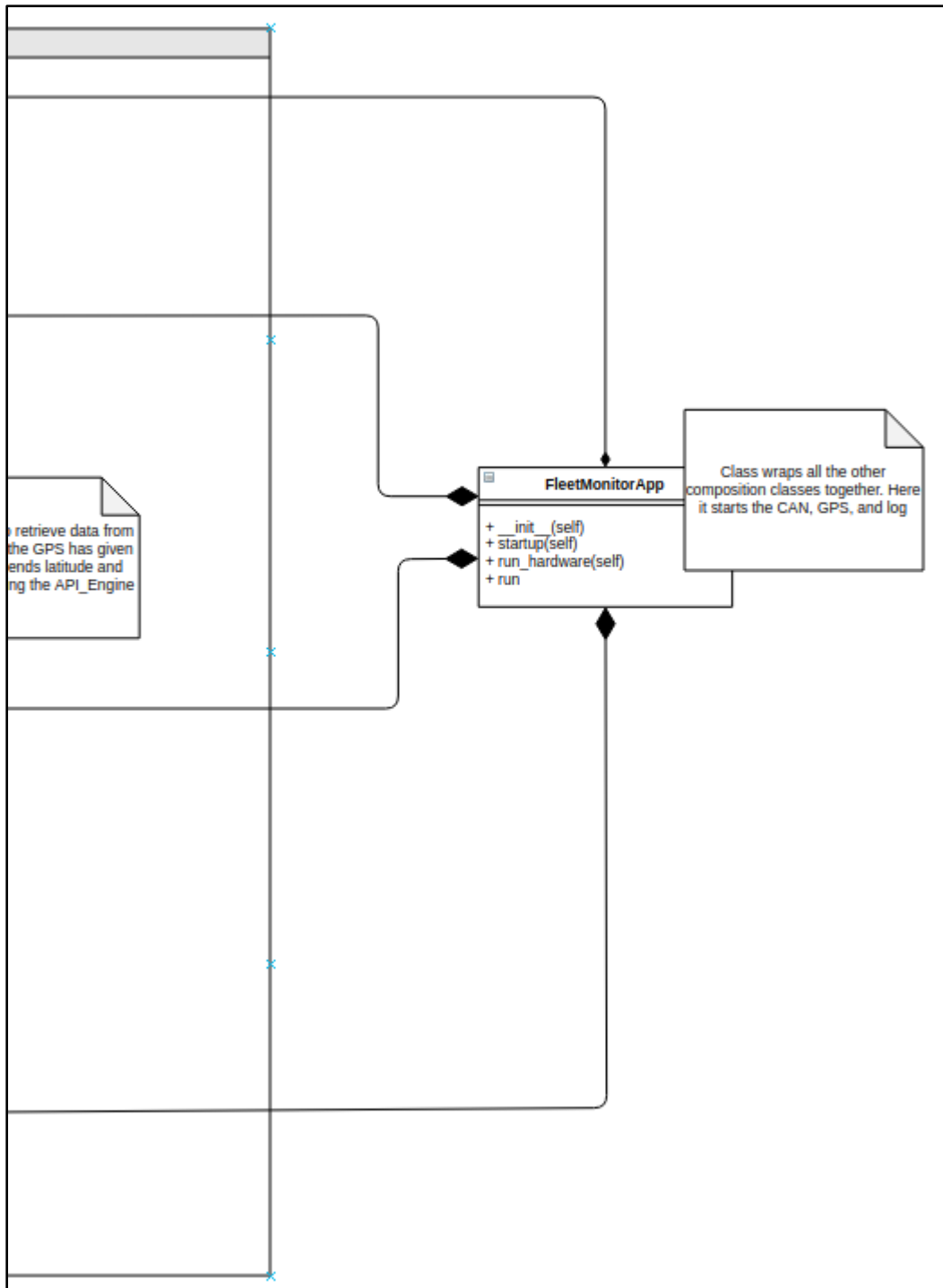
Right-side of diagram (previous image is left-side)

3. Statement of Work

Teletrac Navman

Features:

- Data analysis of fuel use-per trip and averages, idle time-total events and duration, and more
- Communication between drivers and dispatchers in a two-way messaging system
- Allows for monitoring activity within a user-set perimeter
- Alerts and proactive management

Advantages/shortcomings:

- No real-time tracking
- Data analysis

Disadvantages:

- Cost

WebNMS Fleet Management Solution -- Amazon's:

Features:

- Real-time visibility of vehicle location
- Driver behaviour monitoring
- Fuel/cargo management
- OBD data monitoring

Advantages/shortcomings:

- Made by a well known company, Amazon

Disadvantages:

- Meant for large fleets that do delivery
- Cost

Onfleet features:

- SMS notifications
- Real-time tracking
- Driver chat
- Data visualization

Advantages:

- Drivers have the ability communicated to each other

- Data visualization

Disadvantages:

- Cost

Sources:

https://www.teletracnavman.com/

https://aws.amazon.com/marketplace/pp/ZOHO-Corporation-Private-Limited-WebNMS-Fleet-Mana/B07V9V4CTF

https://onfleet.com/

Note: For the server, android app, and webpage we started from scratch. The embedded system inherianted code from the previous team's project. Although, refracting was needed for the embedded system.

3.2 TECHNOLOGY CONSIDERATIONS

Technology includes:

- RaspberryPi
    - Strength: Adaptable to other technology
    - Weakness: High cost, bulky, small computational power, and inconvenient
    - Tradeoff: Inexpensive
- Android Studio
    - Strength: Very friendly community, tons of features, constantly being updates
    - Weakness: Large learning curve, some components may be hard to research about to learn how to implement, only android applications can be created, Java is used
    - Tradeoff: Free to use, open source, android studio can be used on any operating system
- ExpressJS
    - Strength: Large community, matured framework, Fast app development because you can use the same language for frontend and backend.
    - Weakness: No error handling, tedious tasks, Code organization
    - Tradeoff: Free to use, open source
- Digital Ocean
    - Strength: Easy to use, and deploy for starters
    - Weakness: Smaller company runs it, less features, a lot of components have to be implemented on your own.
    - Tradeoff: As a student I get a lot of credit towards server costs
- MongoDB
    - Strength: Easy to use, and scalable
    - Weakness: High data consumption; no default transaction support
    - Tradeoff: Free to use for a single database

3.3 Task Decomposition

Develop a server

- Migrate to AWS
- Develop API
- Implement Microservices
- Implement Socket Broadcasting for real time tracking

Develop an android application

- Login screen
- Sign up screen
- Real-time tracking of drivers
- Chat system
- Data summary

Develop a web application

- Login screen
- Sign up screen
- Real-time tracking of drivers
- Chat system
- Data summary
- Data analytics

Develop the communication between embedded system and vehicle

- Obtain vehicle real-time diagnostic data
- Retrieve location of current vehicle
- Implement user authentication

3.4 Possible Risks And Risk Management

Some concerns that may affect our plan include knowing which frameworks and resources are best for our project. An example of this knowing which resource is best to use on our server and knowing which framework best fits for creating our web application. A possible risk for our project includes the PiCan, no one in our group has a lot of knowledge with embedded systems. So there are some risks with learning about as we are working.

3.5 Project Proposed Milestones and Evaluation Criteria

Most of our milestones include being able to read data with Pican. being able to run our server, and having different parts of both of our UIs working. An example of one of these milestones includes having a use-case for the mobile application and web application. This includes having both of these UIs read information from a server that is received from the PiCan. A lot of milestones will include having components working correctly with each other. To test these we will meet with our groups and run test cases and see if our components will communicate pass data correctly.

3.6 PROJECT TRACKING PROCEDURES

Our group will use Trello to keep track of each task that our members will be doing and we will be using to keep track of tasks that are to be done and tasks that are completed. We will also have weekly meetings to talk about project progress. This involves two meetings, one as a team meeting and with our stakeholder.

3.7 EXPECTED RESULTS AND VALIDATION

The desired outcome is to have a fully functioning Web Application, Mobile Application, PiCan, Server, and Database. These 4 components should be able to communicate with each other and be able to handle data correctly. Our solution will work at High Level if everything will work correctly and the use-cases will be done correctly by our UIs.

4. Project Timeline, Estimated Resources, and Challenges

4.1 PROJECT TIMELINE

4.2 FEASIBILITY ASSESSMENT

Senior Design Group 12's project will be a multi-platform for mobile and web application. The applications will have numerous features like group chat, data visualization, and real time tracking.

The challenges from the project is that a lot of the technologies being used are new to our developers, and have to spend numerous hours on learning the new technologies.

4.3 PERSONNEL EFFORT REQUIREMENTS

Task	People Worked	Time Spent (hours)
Research current fleet monitoring services	Joe, Lorenzo, and Nicolas	3 (total)
Research database to use for project	Nicolas	1
Run inherited code for RaspberryPi on vehicle	Joe and Marco	10 (total)
Refactor RaspberryPi code	Marco	29
Boot up new server on DigitalOcean	Nicolas	2
Set up database	Nicolas	1
Design database architecture	Nicolas	2
Develop vehicle routes: • POST vehicle OBD status • POST vehicle GPS status	Nicolas	8
Develop vehicle routes: • GET vehicle OBD status • GET vehicle GPS status	Nicolas	8
Develop user routes: • POST login • POST sign up	Nicolas	8
Design mobile app user interfaces	Joe	4

Develop mobile app login user interface		Joe	12
Develop mobile app data display user interface		Joe	12
Design web app user interfaces		Lorenzo	5
Develop web app login user interface		Lorenzo	13
Develop web app data display user interface		Lorenzo	13

4.4 OTHER RESOURCE REQUIREMENTS

Materials:

-Hardware: Simulator, HOVA Hologram, GPS MicroSD, Raspberry Pi

-Server: DigitialOcean Web Servers, ISU Web Server

4.5 FINANCIAL REQUIREMENTS

Hardware: $100

TOTAL: $100

5. Testing and Implementation

5.1 INTERFACE SPECIFICATIONS

An Interface that we are using for testing, is Postman. Postman is used to test that the correct data is sent and received using requests from or server.

5.2 HARDWARE AND SOFTWARE

Android:

- Espresso UI Tests
- Espresso testing framework to be used for testing UI activities in the Android application.

Server:

- Requests Framework
- Requests testing framework to be used for testing API.

Web:

- MochaJS and JasmineJS
- MochaJS testing framework to be used for testing the front-end design and JasmineJS is used to test the behaviour of the web application.

5.3 FUNCTIONAL TESTING

Stress testing for the server and unit testing for Android and web application.

5.4 NON-FUNCTIONAL TESTING

Web-Application app is run on localhost and here the UI is tested to make sure that it runs correctly. It is tested here to make sure that it looks correct and that each component does it's job correctly. Android app is tested in a similar way as the web-application. The app can be run on an android phone and here similar tests like the web-application.

5.5 PROCESS

Web-application and the android app were tested each time a component was created. For the server and PiCAN, there were weekly tests to make sure that the data was correctly received and sent out by the server from the PiCAN.

5.6 RESULTS

All tests are successful for all components of the project. We learned that testing has gone well so that we don't need to make any changes so far, but as we keep on developing more tests will be created.

6. Closing Material

6.1 CONCLUSION

Implemented embedded system, server, android application, and a web application.

Goals:

- Implement Raspberry Pi to communicate vehicle data to the server
- Implement Server to store and send the data to the website and mobile application
- Implement an web application to display data to an user. Data displayed include visual data (graphs), gps locations, and a chat system.
- Implement an mobile application to display data to an user. Data displayed include a brief summary of vehicle data, gps locations, and a chat system.

Best plan of action is to continue with implementations of these components to achieve a final product at the end Fall 2020. This is the best plan of action because development has been going smoothly, therefore, time is all the project needs.

6.2 REFERENCES

Standards
Agile Development Standards
JavaScript Coding Standards
Google Java Style Standards
Oracle PLSQL Coding Guidelines

6.3 APPENDICES

Item	Link
OBD II PIDS	OBD-II PIDs Link
Google Maps JavaScript API	JavaScript API Link