# Vehicle Operator Behavior Monitoring System

Final Report - December 10th, 2019

—

Client

—

Andrew Guillemette

Team

—

Andrew Damon
Freya Gaynor
Skand Gupta
Sydney Ehlinger

sddec19-22

# Table of Contents

## List of Figures

## List of Abbreviations & Symbols

| | |
|---|---|
| SDK | Software Development Kit |
| NoSQL | nonSQL or non relational |
| CRUD | Create, Read, Update, Delete |

## List of Definitions

**Angular**: Version 2+ of a TypeScript framework developed by Google used for full-stack web application development[1]. Note the difference from AngularJS.

**AngularJS**: Version 1.x of Angular, rewritten by the same team as Angular.

**CRUD**: An abbreviation defined in the List of Abbreviations & Symbols, used as shorthand for the major functions of data in a database; creating a new record, reading an existing record, updating an existing record, and deleting an existing record.

**End-to-End Testing**: is a technique used to test whether the flow of an application right from start to finish is behaving as expected.

**Firebase**: Google's mobile and web application development platform.

**Firestore**: fast, fully managed, serverless, cloud-native NoSQL document database that simplifies storing, syncing, and querying data for your mobile, web, and IoT apps at global scale.

**Fleet**: A collection of vehicles owned by a company or individual that is used for similar purposes (e.g. public transportation and similar services).

**Google Cloud:** a set of solutions and products, including GCP and G Suite.

**Software Development Kit**: Tooling provided to use external projects as resources for the current project.

**Vehicle Moment**: a single recording of data at any point in time during a driver’s trip.

**Segment**: a collection of Vehicle Moments with details about the driver’s performance.

**Lap**: One loop of a driver’s trip. A lap is broken down into segments.

# 1 | Introduction

## 1.1 | Acknowledgement

**Project Client**: Andrew Guillemette

**Project Adviser**: Dr. Daji Qiao

**Graduate Student Team**: Archit Shashidhar Joshi, Ashraf Shaikh Mohammed, & Shankar Sridhar developed the grading algorithm to score the performance of vehicle operators. The data is gathered via a Raspberry Pi chosen exclusively by the graduate students and by the client.

## 1.2 | Problem and Project Statement

Our project will help make buses safer by monitoring bus drivers for complacent driving that could eventually lead to an accident. Using data collected from the bus, we will look to identify bad trends/behaviors and correct them before an incident occurs. Any company that manages a fleet of vehicles will eventually have to deal with an automotive collision. This inevitably results in many consequences, to the repair/replacement of the damaged vehicle, losing a valuable driver, and usually a spike in insurance premiums. Most collisions are the direct result of negligence on the part of a driver, either from lack of training or complacency.

Our goal is to provide an interface for safety managers, dispatchers, and bus drivers to receive alerts when driving performance deviates from ideal behavior. This system will allow the users to address the deviations *before* they become vehicle collisions by knowing when to providing training refreshers for the drivers or - in the case of repeated problems - terminate the contract of the driver.  Examples of potential warning signs that demonstrate a need for action on the part of the safety managers include: sudden starts/stops, jerky turns, side-swiping mirrors, et cetera. These warning signs and other data collected from the vehicles are analyzed for trends by the graduate students' project **[1.1]** and then presented in a user-friendly interface by our team.

# 2 | Project Design

## 2.1 | Operating Environment

The majority of heavy-lifting for the fleet management software is server-side, as it will collect data from various fleet vehicles and analyze it for trends to form a dataset that can be compared against for irregularities in behavior. Meanwhile, the front-end that uses this dataset is a dynamic web application.

## 2.2 | Intended User(s) and Use(s)

The primary initial user for our software will be the bus drivers from DART and DART's safety managers, although it is our client's intention for this to be available as a product for other companies as well.

## 2.3 | Limitations

- Graduate student's algorithm was hardcoded for one route, so we had to rework scope and were not able to complete all tasks we had planned for last semester.
- Storage server cost is unknown until the proper scale of the data is known.
- As student developers, we will not be able to maintain this product after we have completed it for the client.

## 2.4 | End Product and Deliverables

At the end of Senior Design 492, our client will be presented with:

- A web-based software capable of:
    - Viewing grades that will represent the driver's performance - it indicates to Management if the driver had abnormal behavior that might warrant a warning or possibly additional training.
    - Pulling driver data and trip data from Firestore
    - Searching for specific drivers, filtering them, and sorting them based on name, username, overall grade, or their amount of warnings.
    - Using an interactive map to display information about a driver's lap

A copy of the software documentation to help them in the event the client wishes to continue to grow and develop the software in the future.

# 3 | Implementation Details

## 3.1 | Project Design

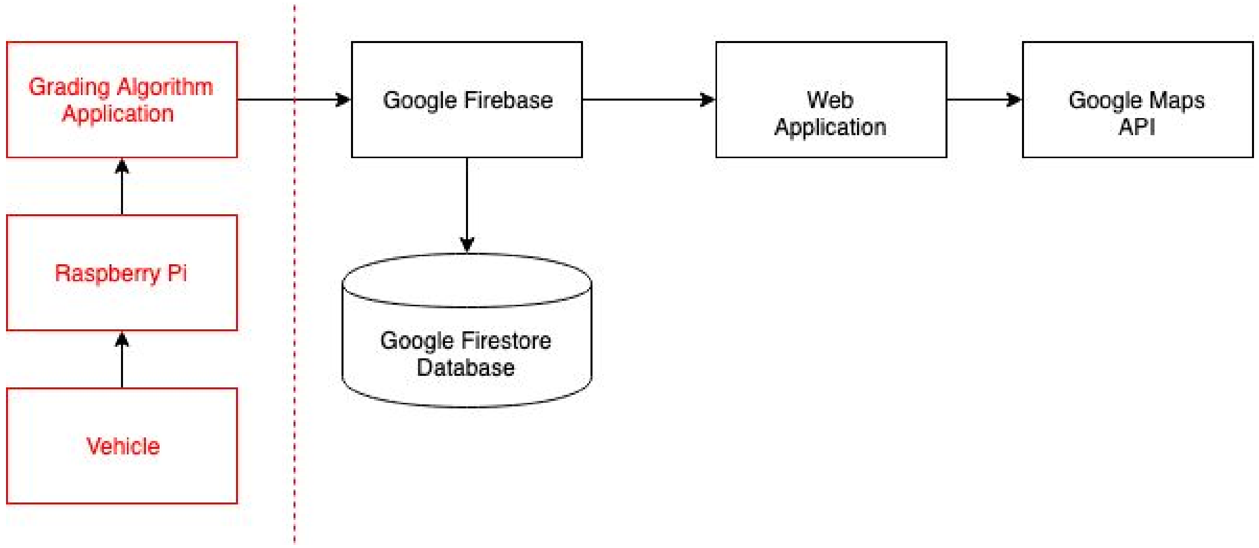


Figure 3.1 Block Diagram

For the design, we decided, based on our client's needs, to use Google Firebase[1] to store the graduate student's data collected from the Raspberry Pi to our Google Cloud Firestore[2] database and web application. The reason why we are storing data with Google Cloud Firestore is because it is a flexible and scalable NoSQL cloud solution that meets our needs. It is used to store and sync data for client- and server-side development. As for our web application, we are using Angular[4] for its reactive framework. For the initial release, the data from the vehicles will be uploaded at the end of the day. The web application retrieves the data in real time, however, so if future changes allow the data to upload with greater frequency it should automatically be handled. In addition, we are using the Google Maps API as most users have some experience with Google Maps in the past. This should allow for a more intuitive and familiar user experience. The Google Maps API also has additional tools such as distance matrices and speed limits that will help aid us in determining the buses' locations with greater accuracy than relying on GPS alone. Additionally, as the API is another Google product, it is highly compatible with both Firebase and Firestore.

On the software side, the JavaScript Angular framework is designed to work in component modules. We have designed individual components for features, and they have been chained together or composed inside of other components to build the application as a whole. Individual components have their own design, functionality, and unit tests.

The web application will make requests to the Google Maps API to render maps and calculate information about the maps.

## 3.2 | Functional Requirements

**As an Admin, I want to be able to...**

- View raw and processed data of routes
- View statistics and reports about Drivers.
- Receive some sort of flag when a driver receives a poor grade for their performance.
- Filter and search the reports
- Access the website without using a VPN or similar networking tools - in other words, it should be accessible to the World Wide Web.

## 3.3 | Constraints Considerations

- Consideration 1: The UI should be intuitive for first-time users. It should not have a steep learning curve for usability.
- Consideration 2: The application will be developed for DART initially, but it is the hope of the client to scale it up to multiple fleet management authorities.

## 3.4 | Technology Considerations

- Consideration 1: Fleet must be able to communicate with graduate student's driver grading application.
- Consideration 2: Server database should store Driver data and generated reports.

## 3.5 | Security Considerations

As is the case with any prototype project, like what we are delivering, security is a topic that is addressed only to a minimal degree. Our intention is to deliver a product that can be secured and extended by our client to meet his needs, ideally with additional security experts and engineers to ensure the capability of the project. We will be providing ample documentation to aid in their efforts, but that is not the focus or goal of our deliverables.

## 3.6 | Safety Considerations

- Consideration 1: Liability if there is an accident should be on the company.
- Consideration 2: Liability for false positives and false negatives in grading should be accounted for in the software license.

## 3.7 | Previous Work & Literature Review

Currently, there are several other products on the market have been created to accomplish similar goals. Multiple insurance companies have different monitoring systems for vehicles to track good driving behavior, such as Progressive Snapshot[4], Allstate

Drivewise[5], Nationwide SmartRide[6], Safeco RightTrack[7], and Travelers IntelliDrive[8]. These products are generally usage-based, meaning you pay your insurance based on how much you drive and how safe your driving is. Most of these insurance companies have an app that allows you to see your driving status, your most recent trips, and how to improve your driving. From there, you can view the specifics of a trip via a map that highlights where you had a hard break, fast acceleration, idle time, time of day you were driving, or even if you used your phone. While all these apps utilize much of the core concepts that we were seeking to implement, most lack all of the features that our client required in one, central product, especially at an enterprise scale.

## 3.8 | Project Tracking Procedures

Biweekly meetings with our client and faculty advisor help strengthen communication and help update everyone on the past week's progress. These meetings were used as a chance to demo our progress, receive feedback, update our timeline, determine changes in scope, and determine our next steps. Additionally, the evaluation criteria for each milestone will allow us to gauge how far along we are in the project and adjust our speed/focus accordingly.

## 3.9 | Design Analysis

We did many weeks of research to come up with a solution that we believe will be effective for our clients' needs. Throughout our research, we analyzed several different tools for their potential. We took into consideration how complex each tool is, its learning curve, its hardware requirements, its resource costs, and its availability for our project use.

**SQL Database vs. Google Firestore**

While a SQL database would have provided us with a tried-and-true approach to data storage, it was not ideal for our solution. SQL mandates certain relationships between various data points in the database, however, the actual design and format of our data was in flux until the final delivery by the graduate students at the beginning of this semester. As such, we had to be able to adapt on-the-fly to changes.

Google Firestore provided an excellent solution to this problem. Firestore is a NoSQL-type database, meaning that it does not require direct relations and the data can easily be parsed by a JavaScript application, like the one we created. In addition, it is owned and operated by the same company as Angular and several other tools we will be using, which ensures compatibility.

**Angular vs. React.js vs Vue.js**

Since we are doing the data visualization portion of the project, we went through a couple of languages to write our program in but landed upon Angular as our best option. In the future, we knew our application would be receiving the graduate students' data in real time, and to accommodate this we knew we needed a reactive language to be able to update the screen with this data as it came in. In addition to Angular, we also looked into React.js and Vue.js which are other reactive programming languages. However, we already have a team member who is comfortable working with Angular who can assist in teaching and advising. As such, Angular was our selection.

**Google Maps API vs OpenStreetMap API**

Our team decided to go with the Google Maps API over the OpenStreetMap API for a couple of reasons. Most people are already familiar with Google Maps and will make the application overall feel more intuitive since they already have experience. There are also more tools with Google Maps that we plan on utilizing. One downside of using Google Maps over Open Street Maps is there is a cost with using the Google Maps API. However, while building the application there was no cost associated with using the Google Maps API.

## 3.10 | Other Resource Requirements

The project required some external resources to help maintain the team’s documentation and store our code, such as GitHub. Our other resource requirements included the data gathered from the graduate student's system, which was used to develop our application.

# 4 | Testing Process and Results

## 4.1 | Testing

Since we are delivering a proof of concept and not a production ready application, testing was not a huge concern of ours or our client's. We did however, implement a testing framework for both unit and end-to-end testing which can be used in the future when we transfer our project to our client. We will be providing documentation to aid their efforts, but this was not a focus of ours.

# 5 | Closing Material

## 5.1 | Conclusion

Our project is designed to give DART (and companies like it) the ability to track the behavior of bus drivers in their fleet. The end goal is to allow DART to identify and address potential risks in the performance of their drivers before they become costly situations like collisions or lawsuits. Each day, the information collected from the busses will be processed and each driver will receive a driving report. Users can filter or search for a bus driver and then view that driver's history of driving reports. If a bus driver receives a subpar driving report, it will be flagged so the safety managers and other appropriate users in DART's employment can look into it. These users can then make the decision on how to address the risk before it becomes a larger problem; likely through retraining or through termination of employment in severe cases. The design of our project will be flexible enough to be useful for multiple companies in the future, including but not limited to DART. With this in mind, we ensured that the use of the application is intuitive, easily scalable, and could be modified to allow for users to view data for a vehicle in real-time.

By using the Angular framework for development, the page can automatically refresh relevant data at the time of its change without requiring user intervention or refreshing the page. This allows the project to be extended in the future so users could view a vehicle's data in real-time, rather than at the end of the business day. The Google Maps API is used to display information regarding a vehicle's route. We selected this over its competitors as it is commonly used and well-known, providing a familiar interface for most users. For storing data, we selected Firebase; primarily due to the compatibility with other Google tools and products we are using for the application. For the same reason, we selected Firestore as a hosting solution for the application. Firebase and Firestore are able to easily communicate data and provide many useful features that can be used for the project.

## 5.2 | References

[1]Firebase - https://firebase.google.com/
[2]Firestore - https://firebase.google.com/docs/firestore/
[3]Angular Wikipedia - https://en.wikipedia.org/wiki/Angular_(web_framework)
[4] Progressive Snapshot - https://www.progressive.com/auto/discounts/snapshot/
[5] Allstate Drivewise - https://www.allstate.com/drive-wise.aspx
[6] Nationwide SmartRide - https://www.nationwide.com/personal/insurance/auto/discounts/smartride/
[7] Safeco RightTrack - https://www.safeco.com/products/righttrack
[8] Travelers IntelliDrive - https://www.travelers.com/car-insurance/programs/Intellidrive

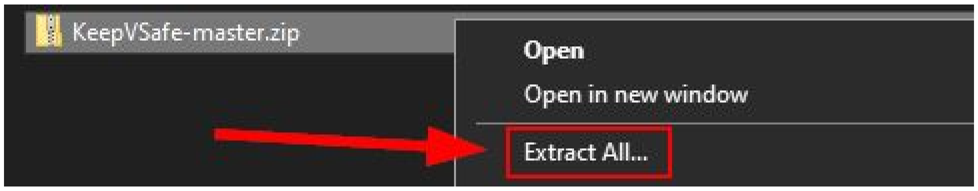
# 6 | Appendix

## 6.1 | Operation Manual

**Step 1.** - Download the project as a zip

**Step 2.** - Make sure you have Node.js installed
Download here: [https://nodejs.org/en/download/](https://nodejs.org/en/download/)

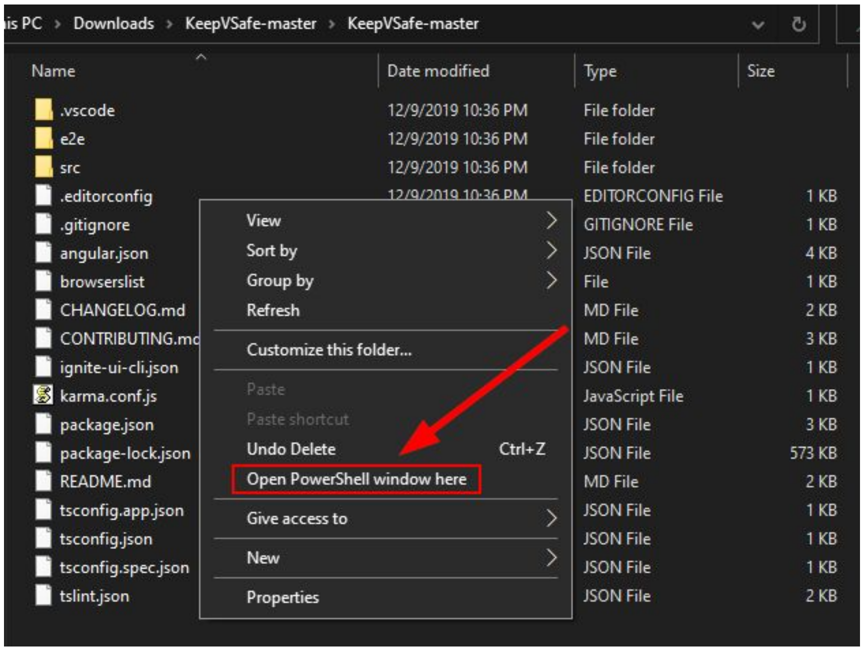
**Step 3.** - Extract the project folder from the zip



**Step 4.** - Go to the main folder and open your operating system's terminal there

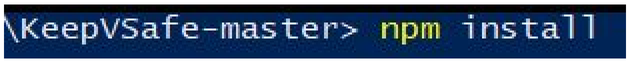
| | |
|---|---|
| Windows terminal: | Powershell or Command Prompt (cmd) |
| Linux terminal: | Terminal |
| Mac: | Terminal |

(For Windows, a shortcut is Ctrl + Right click and then click
"Open Powershell window here")



| Name | Date modified | Type | Size |
|---|---|---|---|
| .vscode | 12/9/2019 10:36 PM | File folder | |
| e2e | 12/9/2019 10:36 PM | File folder | |
| src | 12/9/2019 10:36 PM | File folder | |
| .editorconfig | 12/9/2019 10:36 PM | EDITORCONFIG File | 1 KB |
| .gitignore | | GITIGNORE File | 1 KB |
| angular.json | | JSON File | 4 KB |
| browserslist | | File | 1 KB |
| CHANGELOG.md | | MD File | 2 KB |
| CONTRIBUTING.md | | MD File | 3 KB |
| ignite-ui-cli.json | | JSON File | 1 KB |
| karma.conf.js | | JavaScript File | 1 KB |
| package.json | | JSON File | 3 KB |
| package-lock.json | | JSON File | 573 KB |
| README.md | | MD File | 2 KB |
| tsconfig.app.json | | JSON File | 1 KB |
| tsconfig.json | | JSON File | 1 KB |
| tsconfig.spec.json | | JSON File | 1 KB |
| tslint.json | | JSON File | 2 KB |

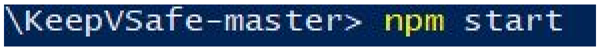
**Step 5.** - Run the command `npm install` to install the project's dependencies



```
\KeepVSafe-master> npm install
```

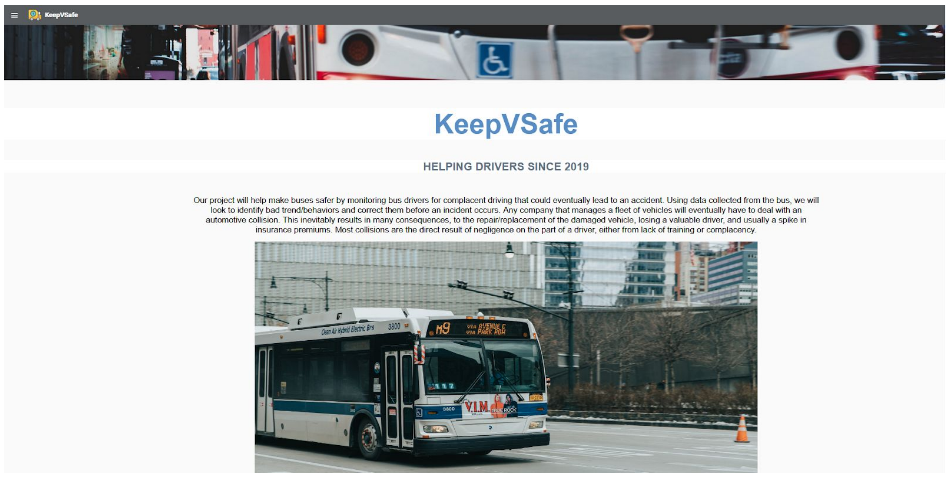
(May take a few minutes)

**Step 6.** - Run the command `npm start` to begin running the application



```
\KeepVSafe-master> npm start
```

**Step 7.** - Open your browser and go to [http://localhost:4200/](http://localhost:4200/)



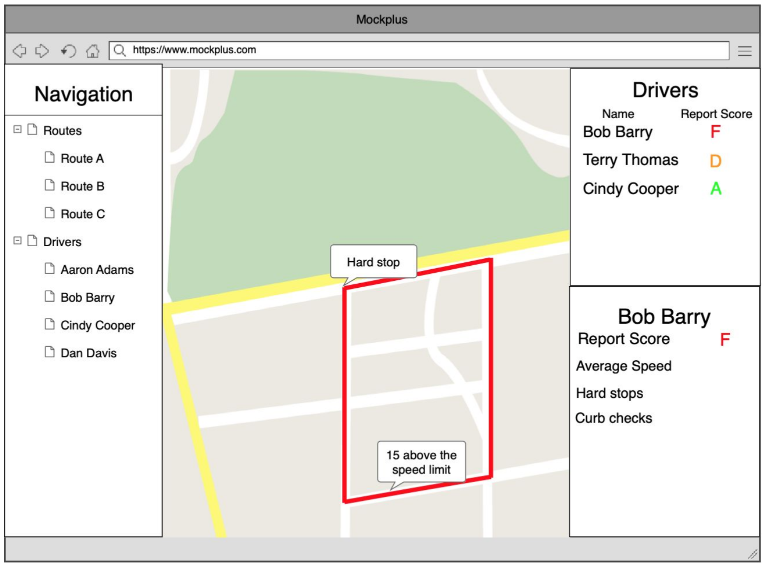
KeepVSafe

HELPING DRIVERS SINCE 2019

Our project will help make buses safer by monitoring bus drivers for complacent driving that could eventually lead to an accident. Using data collected from the bus, we will look to identify bad trend/behaviors and correct them before an incident occurs. Any company that manages a fleet of vehicles will eventually have to deal with an automotive collision. This inevitably results in many consequences, to the repair/replacement of the damaged vehicle, losing a valuable driver, and usually a spike in insurance premiums. Most collisions are the direct result of negligence on the part of a driver, either from lack of training or complacency.

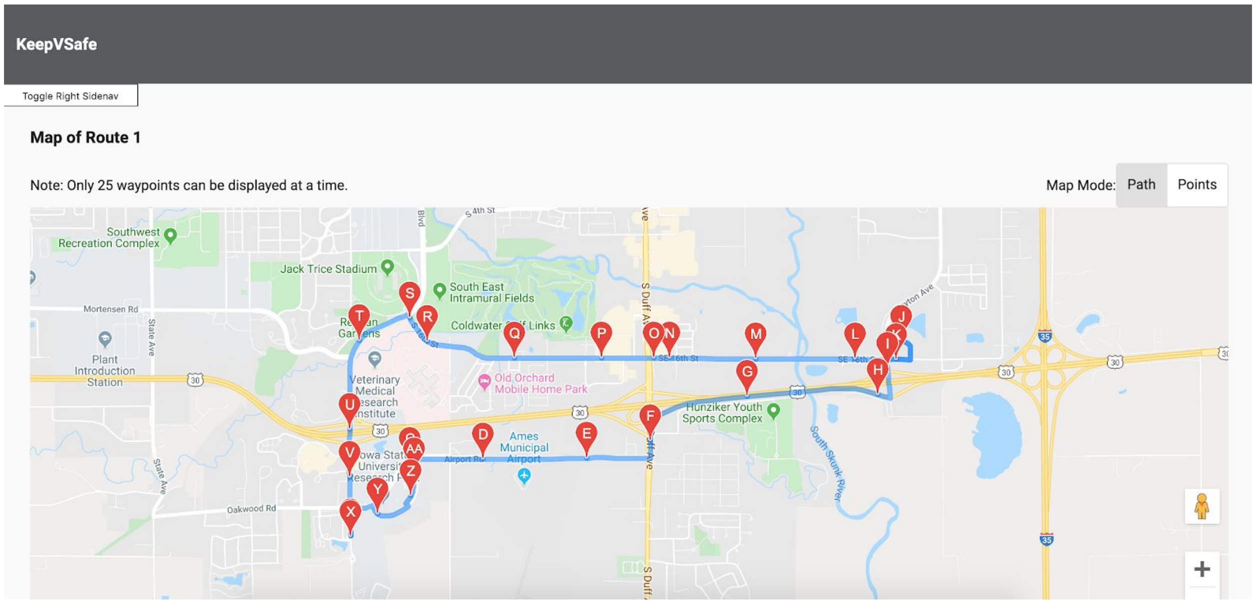
## 6.2 | Alternative/Other Initial Versions of the Design

**Early concept design:**



During Senior Design 491, we created concept art of how we wanted our interface to look. We were under the impression that the graduate students' grading application and data would be easy to integrate. However, this was not the case, and a lot of features that we planned for during last semester we were unable to complete. The grading application was hardcoded for only one route and mainly relied on using distance to map segments. The hardcoded data for the one route meant that we wouldn't be able to use their grading application for new routes. Additionally, because the segments were created using distance, we didn't have an automated way to create them in Google Maps and had to manually add them in.

**First attempt at displaying a lap:**



Map of Route 1

Note: Only 25 waypoints can be displayed at a time.

Early into the semester, we began working on trying to use the Google Maps API to display a lap from the graduate student's data. We hadn't transferred the data over to Firestore yet, so we were pulling it from CSV files. The module that we installed was able to easily create a route from point to point, however, it lacked many features that we needed. We were forced to display the different segments using waypoints, and we couldn't even click on the route to display the closest Vehicle Moment. We realized that we'd have to override much of the module's code to add the functionality that we wanted, so instead we found a more versatile module.

## 6.3 | Other Considerations

Coming into this project, three-fourths of our team had no experience with Angular and had the opportunity to learn a new programming language they hadn't known. In addition to this, we really want to stress how excited we are that we were able to deliver this proof of concept to our client with absolutely no cost.