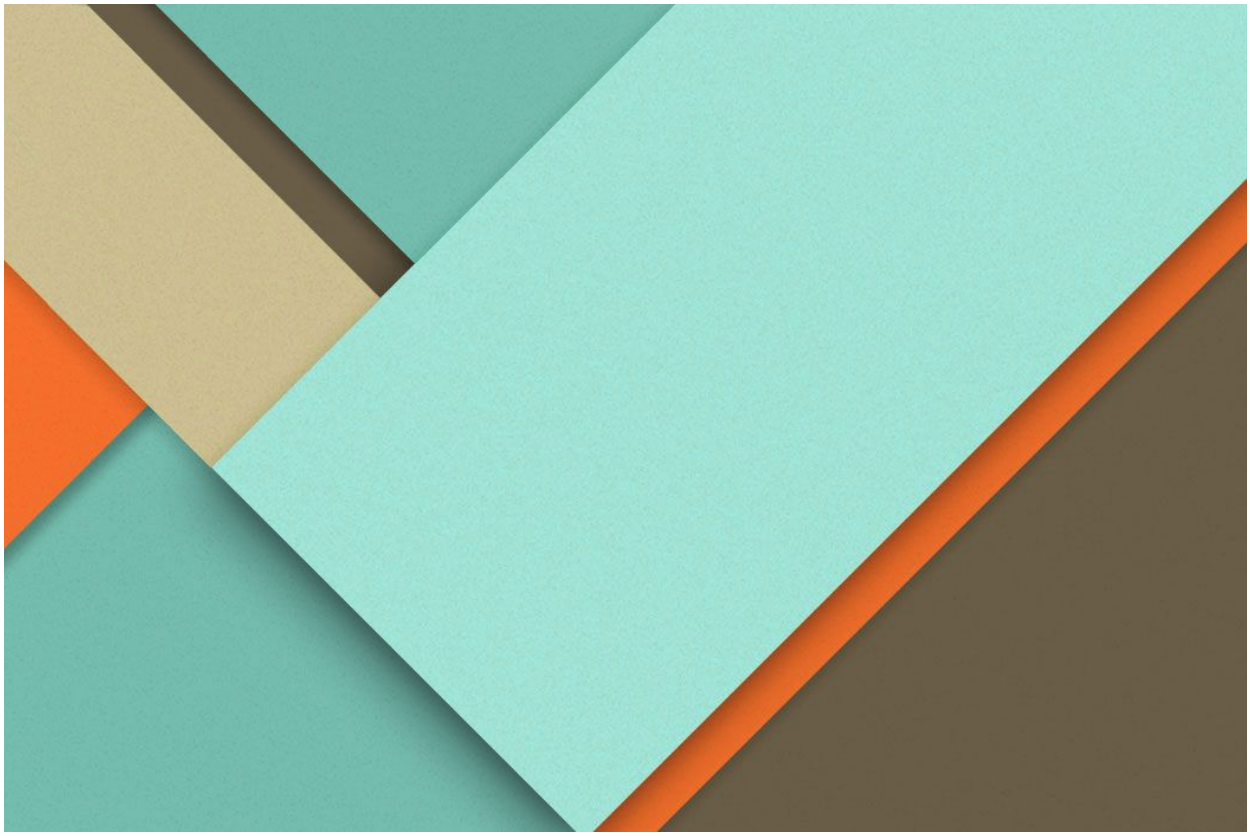

Vehicle Operator Behavior Monitoring System

Project 22

Revision 2 - April 24th, 2019

Client

Andrew Guillemette

Team

Andrew Damon
Freya Gaynor
Skand Gupta
Sydney Ehlinger

Table of Contents

List of Figures

List of Tables

List of Abbreviations & Symbols

AWS Amazon Web Services

DART Des Moines Area Regional Transportation Authority

JS JavaScript

TS TypeScript

List of Definitions

Amazon Web Services: A hosted cloud computing service with a paid-subscription plan, providing persistent access.

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.

Des Moines Area Regional Transportation Authority: A public transportation company providing services for Des Moines and the surrounding area.

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

TypeScript: A syntax subset for JavaScript to allow for static typing of variables. It compiles to standard JavaScript for compatibility.

1 | Introductory Materials

1.1 | Acknowledgement

Project Client: Andrew Guillemette

Project Adviser: Dr. Daji Qiao

Graduate Student Team: Archit Shashidhar Joshi, Ashraf Shaikh Mohammed, & Shankar Sridhar are working on a grading algorithm to score the performance of vehicle operators. Their data is gathered via a hardware interface chosen exclusively by the graduate students and by the client.

1.2 | Problem 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 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.

Our goal is to provide an interface for DART's safety managers, dispatchers, and bus drivers to receive alerts when their driving performance deviates from ideal behavior. This system will allow the users to address the deviations *before* they become vehicle collisions by 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 by the graduate students' project **[1.1]** will be analyzed for trends and then presented in a user-friendly interface by our team.

Figure 1.1 Use Case Diagram

1.3 | Operating Environment

The majority of heavy-lifting for the fleet management software will be 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. The front-end to use this dataset will be a web or mobile application.

1.4 | Intended Users & Intended Uses

The primary initial user for our software will be safety managers, dispatchers, and bus drivers from DART. In the future, the product will also be used by companies outside of DART. Their first use will be to set up a route using integrated mapping software, including the various parameters along the route (any data points that the graduate students require for their algorithm). Next, they will need to be able to assign drivers for a route, which will likely also include times. Finally, they will need to be notified of variations in driver behavior so that they can take action.

1.5 | Assumptions & Limitations

Assumptions

- The data will be available from the fleet for monitoring.

- The team of graduate students will be able to design an algorithm that can alert our service when drivers behavior dangerously.
- Angular will be able to communicate with AWS.
- Angular will be able to meet all of the needs for the web application.

Limitations

- Algorithm must be provided by graduate students for the final product.
- 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.

1.6 | Expected Product & Deliverables

At the end of Senior Design 491, our client will be presented with:

- Documentation of the project including:
    - One or more Use Case Diagrams.
    - One or more Block Diagrams.
    - A Project Plan, outlining the deliverables of the project and the timeline of work.
    - A Design Document, outlining the implementation of the design and what steps we will take to progress from start to finish.

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

- An Angular web application that will allow the user to:
    - Display the grades representing the driver's performance in a readable format - if the driver had abnormal behavior that might warrant a warning or possibly additional training. Some of these abnormal behaviors include hard stops, fast acceleration, and turning too hard.
    - Add routes, select areas to monitor, and receive alerts when a driver receives a bad grade.
    - Search or filter for specific reports based on data points like the average grade of a route, the timespan of a route, the identity of a driver, et cetera.
- A copy of the software documentation to aid future developers in growing the project.

2 | Proposed Approach & Statement of Work

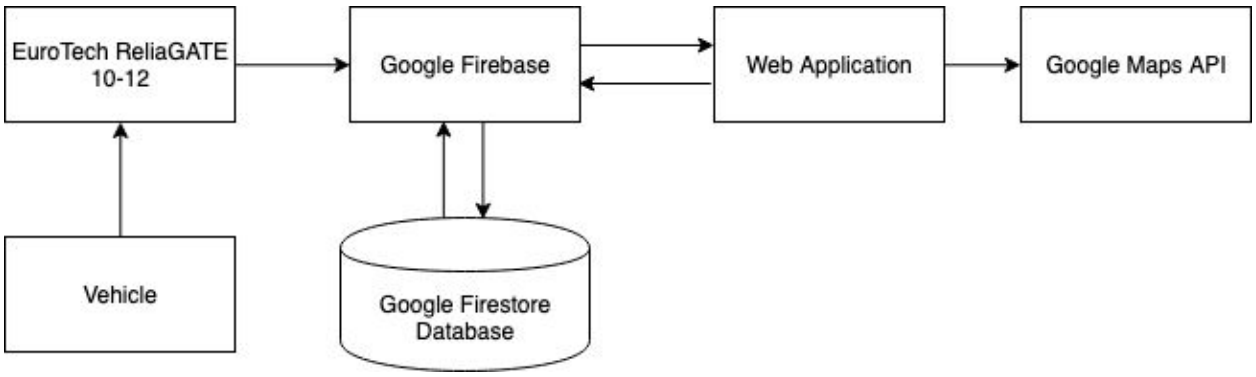

Figure 2.1 Block Diagram

The block diagram above shows that the graduate student's EuroTech ReliaGate 10-12 which gathers the data from the vehicle. At the end of each day the data is uploaded to the server and onto the Google Firestore Database. Our web application - hosted on Google Firebase - then access that data and displays it. In addition retrieving data from Firestore, the web application will make requests from Google Maps API to render maps for the end-user.

2.1 | Functional Requirements

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

- Register and delete Drivers.
- View statistics and reports about Drivers.
- Create a bus route.
- Populate bus routes with grading points such as stop signs, traffic lights, et cetera.
- Add, remove, and modify User accounts.
- Permission what Users can do on the website.
- Receive a notification when a driver receives a poor grade for their performance.

As a User, I want to be able to...

- Generate reports on the data that I am allowed to access.
- Filter the reports I have generated.
- Search for reports I have generated.
- Log in while another User is online.
- Access the website without using a VPN or similar networking tools - in other words, it should be accessible to the World Wide Web.
- Authenticate myself to the website to view the data I am allowed to access.

2.2 | Non-Functional Requirements

- At Least 100 users can be logged in concurrently

- Service should recover/restart after unexpected shutdown within 10 min
- Driver grades should be generated at least 1 min after all of that day's data is received
- Admins should be notified at most 20 min after a driver receives a bad grade

2.3 | 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.

2.4 | Technology Considerations

- Consideration 1: Fleet must be able to communicate with graduate student's driver grading application.
- Consideration 2: Algorithm server must be able to communicate with web application.
- Consideration 3: Web application must be available on mobile and web.
- Consideration 4: Server database should store Driver data and generated reports.

2.5 | Security Considerations

- Consideration 1: User passwords should be securely stored and encrypted.
- Consideration 2: Communication between the site and other resources - including users - should be done over SSL.
- Consideration 3: Least Privilege policy should be applied to user role permissions.
- Consideration 4: User input fields should be sanitized.
- Consideration 5: Cross-site scripting should be accounted for and blocked.

2.6 | 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.

2.7 | Software Standards Compliance

World Wide Web Consortium (W3C)[10] and the Internet Society (ISOC)[11] are the two main organizations that set the standards for software development. W3C was founded by the developer of the internet - Tim Berners-Lee. We chose W3C standard as it is extremely user friendly. It provides a validator tool[12] which takes a url and returns the details of a complete compliance check. We will be using this tool to ensure that our website meets the standards requirement.

2.8 | 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[2], Allstate Drivewise[3], Nationwide SmartRide[4], Safeco RightTrack[5], and Travelers IntelliDrive[6]. 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 are seeking to implement, most lack all of the features that our client requires in one, central product, especially at an enterprise scale.

2.9 | Possible Risks & Risk Management

Our test plan uses the STLC V-Model of testing. The major advantage of this model is that defects are found at a much earlier stage and are therefore easy to isolate. Additionally, it allows for testing to be done in parallel with development.

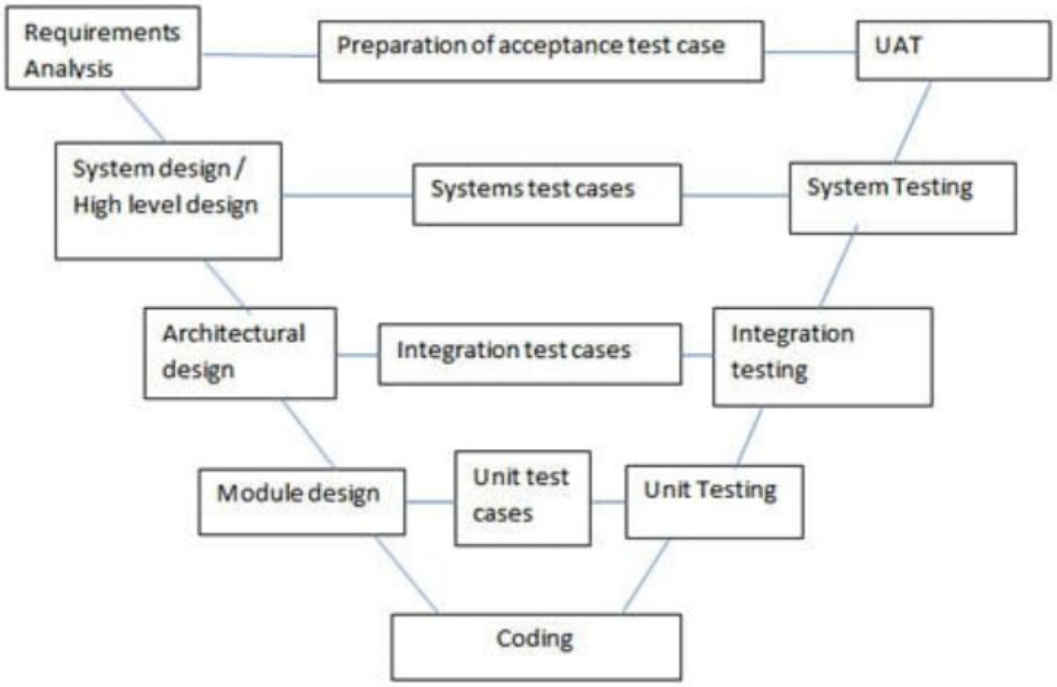

Figure 2.2 STLC V-Model Diagram

For every phase, there is a corresponding testing phase:

1. **Unit Testing (Component Testing):** It is performed on standalone modules to check whether they are developed correctly. This phase has three sub phases:
    a. **Smoke/Sanity Testing:** The goal here is not to check for bugs but to test module functionality. It is quick and non exhaustive.
    b. **White-Box Testing:** These are unit tests that are created by the developer of the module.
    c. **Black-Box Testing:** If a module passes white box testing, a different developer will test the functionality without looking at the module's code.
2. **Integration Testing:** In this stage, the modules are combined and tested as a group. If a connecting module is not complete, a stub is created to simulate data from that missing module.

3. **System Testing:** This type of testing is concerned with the behavior of the system as a whole. Unlike integration testing, which focuses on data transfer amongst modules, system testing checks complete end to end scenarios, the way a user would use the system. Non-functional tests such as Load, Performance, Compatibility, Usability, Security, and Localization will also be performed at this stage.
4. **User Acceptance Testing:** This will be done by the client. The focus of acceptance test is not to find defects, but to check whether the system meets their requirements. Acceptance testing can be done in two ways:
    a. **Alpha Testing:** A small set of employees of the client.
    b. **Beta Testing:** A small set of customers.

The decision to select the type of user acceptance test will be made by the client at a later stage. Exhaustive testing is neither feasible nor possible. The testers will assess the risk and write tests for the most likely cases. Diagrams will be used to show the details of the tests and the test data will be documented in a table format.

For example, the test for checking login functionality will be documented as follows:

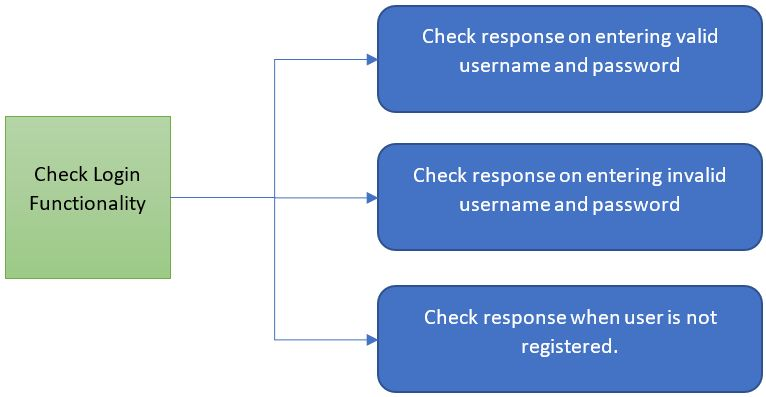

Figure 2.3 Example test

Table 1: Example of test plan

Test Scenario	Test Case	Test Step	Pre-conditions	Test Data	Expected Result	Actual Result	Pass/ Fail
Check login functionality	Check response on entering valid username and password	1. Launch Software 2.Type username 3. Type password 4. Click “login”.	User must be registered	Username: leoMessi Password: Yn\/\/@	Login must be successful	Login was successful	Pass

2.10 | Proposed Milestones & Evaluation Criteria

- **Milestone 1: Requirement Review** - A project plan that has been thoroughly reviewed and accepted by the client.
- **2: Preliminary Design Review** - Production of a high-level system architectural model that accomplishes all of the project requirements.
- **3: Critical Design Review** - Production of detailed designs that fully implement the system architecture.
- **4: Test Plan Review** - Test plans for all project features have been created.
- **5: Test Readiness Review** - The project has been unit tested and is ready to begin integration testing.
- **6: System Test Review Review** - The project passed system testing and is ready for acceptance testing.
- **7: Operational Readiness Review** - The project passed acceptance testing and is ready to be testing in a live environment.
- **8: Product Operational** - All documentation has been updated and completed

2.11 | Project Tracking Procedures

We have laid out our project timeline in Section 3.1. This will allow us to track our progress through both semesters and ensure we hit our deadlines. Weekly team meetings and weekly meetings with our client and faculty advisor help strengthen communication and help update everyone on the past week's progress. These meetings will also be used as a chance to update our timeline if we need to accelerate or slow down the project. 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.

2.12 | Statement of Work

Each task approach is broken down into more manageable tasks to complete the objective.

1. Requirement Review
    a. The objective of requirement review is to gain the necessary knowledge in order to create a project plan detailing the specifications of our project through a project plan outline and completion of the project plan.
    b. Task approach
        i. Outline project plan.
        ii. Complete project plan.
    c. The expected result of requirement review is a finished project plan with complete, correct, approved, and suitable input of our project that we will be able to use as an initial roadmap.
2. Preliminary Design Review

a. The objective of this preliminary design review is to research different architecture models to help us outline our system architecture.
b. Task approach
    i. Outline project system architecture.
    ii. Complete project system architecture.
c. The expected result of the preliminary design review is to have a completed system architecture outline that satisfies all the project requirements through feedback from our client and advisor, so we are well prepared to build a successful architecture for the project.

3. Critical Design Review
    a. The objective of critical design review is now that we have our project plan and architecture plan created, we can create an in-depth design plan to help us establish the specifics of our project.
    b. Task approach
        i. Outline in-depth design plan.
        ii. Complete in-depth design plan.
    c. The expected result of critical design review is to have a completed, in-depth design plan to help us be prepared to build our project.
4. Test Plan Review
    a. The objective of the test plan review is to create and individually test all the project features to ensure they are ready to be used in production.
    b. Task approach
        i. Set up server/servers.
        ii. Create build system.
        iii. Integrate map API.
        iv. Hook up database to website.
        v. Add/remove bus routes.
        vi. Add/remove stop lights, stop signs, etc.
        vii. Integrate graduate students' algorithms.
        viii. Add driver alerts/notifications.
        ix. User registration and login.
        x. Outline test cases document.
        xi. Test cases document complete.
        xii. Review/update project document.
    c. The expected result of this task is to have all of the project features bug free and ready to begin the next stage of testing.
5. Test Readiness Review
    a. The objective of test readiness review is to test multiple different aspects of the project to confirm they are working together the way we intended them to.
    b. Task approach

i. Test server scalability.
ii. Test map API.
iii. Test database resilience.
iv. Test bus routes.
v. Test stop signs, street signs, etc.
vi. Test graduate students' algorithms.
vii. Test driver alerts/notification.
viii. Test user registration and login.
ix. Test for security vulnerabilities.
x. Review/update documentation.

c. The expected result of the test readiness review is to confirm that the different aspects of our project is working together the way they are intended to. It not we will have learned what we have to fix to be able to get it working.

6. System Test Review
   a. The objective of the system test review is to test our project in a live environment to verify that our project works in a live environment setting.
   b. Task approach
      i. Test in live environment
      ii. Review/update project documentation.
   c. The expected result of this task is our project was able to work in a live environment, and if not we will have learned what we have to fix to be able to get it working.
7. Operational Readiness Review
   a. The objective of operational readiness review is to confirm that the project has passed acceptance testing, is suitable for deployment, and is prepared for deployment.
   b. Task approach
      i. Prepare project for deployment.
      ii. Review/update project documentation.
   c. The expected result of this task is to have the project prepared for deployment to the target environment and have the project documentation updated to reflect this.
8. Product Operational
   a. The objective of the product operational task is to complete all documentation detailing the completion of our project.
   b. Task approach
      i. Review/update project documentation.
   c. The expected result of this task is that our documentation is all up-to-date.

3 | Estimated Resources & Project Timeline

3.1 | Project Timeline

Our timeline for this project is divided between the two semesters. During the first semester, we will research the problem and create the necessary documentation. During the second semester, we will implement test, and deploy our project. Table 2 enumerates the tasks for our projet while figure 3.1 shows how long those tasks are expected to take.

Table 2: Gantt Table

	Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Requirement Review**	**24 days?**	**2/14/19 8:00 AM**	**2/21/19 5:00 PM**		
2	Project Plan Outline	5 days?	2/14/19 8:00 AM	2/15/19 10:00 AM		Freya Gaynor;Andrew ...
3	Project Plan Complete	4 days?	2/21/19 8:00 AM	2/21/19 5:00 PM	2	Freya Gaynor;Andrew ...
4	**Preliminary Design Review**	**100 days**	**2/22/19 8:00 AM**	**3/28/19 5:00 PM**	**1**	
5	Outline Project System Archit...	56 days	2/22/19 8:00 AM	3/13/19 5:00 PM	1	Andrew Damon;Freya ...
6	Project System Architecture	44 days	3/14/19 8:00 AM	3/28/19 5:00 PM	5	Andrew Damon;Freya ...
7	**Critical Design Review**	**102.5 d...**	**3/29/19 8:00 AM**	**5/3/19 2:00 PM**	**4**	
8	In-depth Design Plan Outline	37 days	3/29/19 8:00 AM	4/11/19 10:00 AM	4	Andrew Damon;Freya ...
9	In-depth Design Plan Complete	65.5 days	4/11/19 10:00 AM	5/3/19 2:00 PM	8	Andrew Damon;Freya ...
10	**Test Plan Review**	**43 days?**	**8/12/19 10:00 AM**	**8/26/19 5:00 PM**	**7**	
11	Set up server/servers	8 days	8/12/19 10:00 AM	8/14/19 10:00 AM	7	
12	Intergrate map API	12 days	8/14/19 10:00 AM	8/19/19 10:00 AM	11	
13	Hook Database up to website	10 days?	8/14/19 10:00 AM	8/16/19 3:00 PM	11	
14	Add/Remove Bus Routes	15 days	8/14/19 10:00 AM	8/19/19 5:00 PM	11	
15	Add/Remove Stop Lights, Str...	10 days?	8/14/19 10:00 AM	8/16/19 3:00 PM	11	
16	Intergrate Grad Student Alg...	10 days?	8/14/19 10:00 AM	8/16/19 3:00 PM	11	
17	Add Driver Alerts/Notification	15 days	8/14/19 10:00 AM	8/19/19 5:00 PM	11	
18	User Registration and Login	10 days?	8/14/19 10:00 AM	8/16/19 3:00 PM	11	
19	Outline Test Cases Document	10 days?	8/20/19 8:00 AM	8/22/19 1:00 PM	12;13;14;15;16;17;18	
20	Test Cases Document Compl...	10 days?	8/22/19 1:00 PM	8/26/19 5:00 PM	19	
21	Review/Update Project Docu...	10 days?	8/20/19 8:00 AM	8/22/19 1:00 PM	12;13;14;15;16;17;18	
22	**Test Readiness Review**	**10 days?**	**8/27/19 8:00 AM**	**8/29/19 1:00 PM**	**10**	
23	Test server scalibily	10 days?	8/27/19 8:00 AM	8/29/19 1:00 PM		
24	Test map API	10 days?	8/27/19 8:00 AM	8/29/19 1:00 PM		
25	Test Database resiliance an...	10 days?	8/27/19 8:00 AM	8/29/19 1:00 PM		
26	Test Bus Routes	10 days?	8/27/19 8:00 AM	8/29/19 1:00 PM		
27	Test Stop Lights, Street Sign...	10 days?	8/27/19 8:00 AM	8/29/19 1:00 PM		
28	Test Grad Student Algorithms	10 days?	8/27/19 8:00 AM	8/29/19 1:00 PM		

Fleet Management System - page1

	Name	Duration	Start	Finish	Predecessors	Resource Names
29	Test Driver Alerts/Notification	10 days?	8/27/19 8:00 AM	8/29/19 1:00 PM		
30	Test User Registration and L...	10 days?	8/27/19 8:00 AM	8/29/19 1:00 PM		
31	Test for security vunerabilities	10 days?	8/27/19 8:00 AM	8/29/19 1:00 PM		
32	Review/Update Project Docu...	10 days?	8/27/19 8:00 AM	8/29/19 1:00 PM		
33	**System Test Review**	**10 days?**	**8/29/19 1:00 PM**	**9/2/19 5:00 PM**	**22**	
34	Test In Live Environment	10 days?	8/29/19 1:00 PM	9/2/19 5:00 PM		
35	Review/Update Project Docu...	10 days?	8/29/19 1:00 PM	9/2/19 5:00 PM		
36	**Operational Readiness Re...**	**10 days?**	**9/3/19 8:00 AM**	**9/5/19 1:00 PM**	**33**	
37	Review/Update Project Docu...	10 days?	9/3/19 8:00 AM	9/5/19 1:00 PM		
38	**Product Operational**	**10 days?**	**9/5/19 1:00 PM**	**9/9/19 5:00 PM**	**36**	
39	Review/Update Project Docu...	10 days?	9/5/19 1:00 PM	9/9/19 5:00 PM		

Figure 3.1 Gantt Chart

3.2 | Personnel Effort Requirements

Individual Tasks: Set up server/servers, Create build system, integrate map API, hook up database to website, add/remove bus routes, add/remove stop lights, stop signs, etc., integrate graduate students' algorithms, add driver alerts/notifications, user registration and login, outline test cases document, test cases document complete, and review/update project document.

Table 3: Major Tasks

Task	Task Description	Projected Effort Required
Design System Architecture	Designing the project's architecture is an important task. Research must be done on which architecture best fits our timeline, requirements, and budget.	35 hours
Design Individual Project Features	This task will require lot of time to research various ways to implement the project features will still adhering to the overall system architecture.	60 hours
Implement Individual Tasks (located above the table)	This task is when any flaws in our project's design will most prominently appear. A large amount of time should be set aside because unplanned issues will increase the amount of effort required.	120 hours
Test Project Features	Tests must be designed and executed on each feature to ensure it meets the designated requirements. Time must be taken to debug, fix, and test any issues that arise.	45 hours
Test how individual features work together	Tests must be thoroughly designed and executed to ensure the individual features will work together	25 hours

	as intended. Time must also be taken to debug, fix, and retest.	
Test Project in a live environment	The project must be configured out of testing mode and prepped for use in a live environment.	20 hours
Prepare Project for Deployment	The project must be reviewed for any issues that need to be resolved. The hardware and software need to be set up based on the client's needs.	30 hours
Update and finalize all documentation	All project documents must be updated.	15 hours

3.3 | Other Resource Requirements

The project will require some external resources to help maintain the team's documentation and store our code, such as GitLab. In addition to that, we will have to get a server outside of Iowa State as our client does not want it hosted through ISU. Our other resource requirements include the data gathered from the graduate student's system and access to their database, which will be used to develop our application.

3.4 | Financial Requirements

- AWS Server[7]
    - Extremely variable based on needs.
    - An alternative would be Google Cloud's hosting.
- Google Cloud Firestore[8] - $0.28 per gigabyte per month, charges accruing daily
- Google Maps & Routes API[9]
    - See Table 3 on the following page.

Table 4: Google Maps & Routes API Costs

SKU		$200 Monthly Credit Equivalent Free Usage	Monthly Volume Range (Price Per Thousand)		
			0-100k	100,001-500k	500,001+
Maps API	Mobile Native Static Maps	Unlimited loads	$0.00	$0.00	Contact Sales for Volume Discounts
	Mobile Native Dynamic Maps	Unlimited loads	$0.00	$0.00	
	Embed	Unlimited loads	$0.00	$0.00	
	Embed Advanced	Up to 14,000 loads	$14.00	$11.20	
	Static Maps	Up to 100,000 loads	$2.00	$1.60	
	Dynamic Maps	Up to 28,000 loads.	$7.00	$5.60	
	Static Street View	Up to 28,000 panos	$7.00	$5.60	
	Dynamic Street View	Up to 14,000 panos	$14.00	$11.20	
Routes API	Directions	Up to 40,000 calls	$5.00	$4.00	
	Directions Advanced	Up to 20,000 calls	$10.00	$8.00	
	Distance Matrix	Up to 40,000 elements	$5.00	$4.00	
	Distance Matrix Advanced	Up to 20,000 elements	$10.00	$8.00	
	Roads - Route Traveled	Up to 20,000 calls	$10.00	$8.00	
	Roads - Nearest Road	Up to 20,000 calls	$10.00	$8.00	
	Roads - Speed Limits	Up to 2,000 elements	$20.00	$16.00	

4 | Closure Materials

4.1 | Closing Summary

Overall, our goal for this project is to create a web application that allows our client to track the behavior of drivers in their fleet. This application will allow DART to be more proactive rather than reactive with the performance of their drivers. Based on the data they receive from our application, they can view how their employees are performing and make the decision to either retrain the employee with a bad driving report, or let them go before an accident occurs. This will also increase safety on the road for all drivers, not just DART employees.

We will begin by analyzing the format, type, and volume of data that is provided to us by the graduate students' algorithm engine. We will then research the most efficient and cost-effective methods to display that data. Our straightforward workflow and test plan will ensure that our web application is fast, lightweight, and compatible across multiple devices and operating systems.

4.2 | References

[1] Angular Wikipedia - https://en.wikipedia.org/wiki/Angular_(web_framework)
[2] Progressive Snapshot - https://www.progressive.com/auto/discounts/snapshot/
[3] Allstate Drivewise - https://www.allstate.com/drive-wise.aspx
[4] Nationwide SmartRide - https://www.nationwide.com/personal/insurance/auto/discounts/smartride/
[5] Safeco RightTrack - https://www.safeco.com/products/righttrack
[6] Travelers IntelliDrive - https://www.travelers.com/car-insurance/programs/Intellidrive
[7] AWS Server Pricing - https://aws.amazon.com/ec2/pricing/on-demand/
[8] Google Cloud Firestore Pricing - https://firebase.google.com/docs/firestore/pricing
[9] Google Maps & Routes API Pricing https://cloud.google.com/maps-platform/pricing/sheet/
[10] World Wide Web Consortium (W3C) https://www.w3.org/
[11] Internet Society (ISOC) https://www.internetsociety.org/
[12] W3C standards validator https://validator.w3.org/