Team members:
Waldo van der Merwe; u15018556 (Team leader)
Joan Mwaniki; u16159323
Brian Ndungu; u15322913
Duncan Smallwood; u130027205
Dewald van Hoven; u15030378

Taxi Boss SRS

Supreme Internect

May 2019

Contents

1 Introduction

1.1 Purpose

This Software Requirements Specification (SRS) aims to provide a detailed overview of the software product to be developed and delivered, as well as the parameters and goals of the project.

1.2 Scope

A new system will be developed that will be used on mobile devices. This system will aim at monitoring taxi drivers through their mobile device and report any violations that the taxi commit to a monitoring sub-system. The taxi driver will also receive ratings from the commuters, through a different sub-system that will be installed on the commuter's mobile device.

Commuters will also be able to request rides at predetermined pickup spots which will in turn pop up on the taxi driver's app.

However, the main aim of the system is to monitor and rate the taxi drivers. The purpose of this is to provide a platform to improve the behaviour of taxi drivers, as well as the states of their vehicles. The approach is to empower commuters by giving them a platform on which they can help make the taxi industry safer for everyone.

1.3 Definitions, Acronyms, and Abbreviations

TDA: Taxi Driver app
CA: Commuter app
WM: Web monitor
SSP: Server-side processing
UC: Use Case
R: Functional Requirement

1.4 Overview

This document contains all of the software requirements and specifications, as well as an overview of the software product and its functions. User characteristics, constraints, assumptions, and specific requirements will also be outlined in this document.

2 User Characteristics

There are three types of intended users that will use the taxi boss system; the taxi driver, taxi commuter and the taxi monitor.

The taxi driver will need to have a smart mobile device which will enable him to see who requested rides, his current rating and any driving violations that he has committed. This user does not need any high technical level of understanding to use the app.

The taxi commuter will also need to have a smart mobile device which will enable them to "follow" a certain taxi driver, request a ride and give a rating. This user also does not require any high technical level of understanding to use the app.

The monitor needs access to any device that has access to an internet browser which will enable this user to monitor the drivers that are currently registered with this user. They need to be internet literate and have some understanding on the different traffic violations and the severity of each one of them.

Other intended users includes the software engineers who will ensure that the system runs smoothly, as well as take note and fix any problems that may occur. They will require a high educational level and good technical skills in software engineering in order to interact with the system at the lowest level to ensure that it is functioning as required.

3 System Architecture

3.1 System Type

- The system is an Interactive System. The system is driver by users interacting with the system to perform business transactions. Commuters and drivers both interact with the system through their direct input and GPS information when using the app. The transfer of data to and from the server forms the core functionality of the system.

- The system is a Transformational System. Because of route building that will happen on the server as taxis drive, the app will transform in real-time to give a very responsive and real-time system

3.2 Architectural Style

The system has a Client-Server Architecture. There is a central server that contains all of the database information. This data includes: user data, location data, ratings, driving patterns of drivers, driver complaints and compliments and business level data for monitors. The monitor will access and update data as necessary from the monitor portal. The driver will send location data, pull location data, view ratings, complaints, compliments and do account related tasks from the driver portal. Drivers will also get updates on commuters requesting rides, send notifications to followers and receive notifications of traffic violations. They will also pull route-related data. Commuters will give ratings, request rides, comment on drivers and stop and end rides. These are all of the clients of the server with how they interact with the server.

3.3 Architectural Requirements

3.3.1 Database Server

A scalable real-time database server is required. We chose to use Cloud Firestore for our database. It is a highly scalable real-time database hosting service by Google. It uses no SQL and stores data in Documents in JSON format. The content already being in JSON format makes it very easy to work with when data transactions are done because it mixes well with the programming languages we are using for the client interfaces. Cloud Firestore is free to start using with enough storage and transactions. It is scalable to a very large and load intensive database with payed options.

3.3.2 Server

A server will be hosted on Firebase Hosting. It is a highly scalable server service which start out with a free plan which can be upgraded based on our needs. The monitor interface which will be hosted on the server is what will be displayed in the monitor portal. Logic for building routes will also be done from this server.

3.3.3 Web Framework

The monitor portal will be written using the Angular 7 web framework. It will serve us well in the way that it is constructed which will reduce redundancy when coding.

3.3.4 Native app

The native client app that drivers and commutors will use on their smart phones will be coded in Flutter. It is a very dynamic language which will give us all of the capabilities we need and enable us te create a very responsive UI. It also pairs well with Cloud Firestore because it is also a Google product. Apps written in Flutter are also instantly compatible with both iOS and Android operating phones. This makes it easier to distribute and maintain.

3.4 Architectural Design

Architecture.jpg

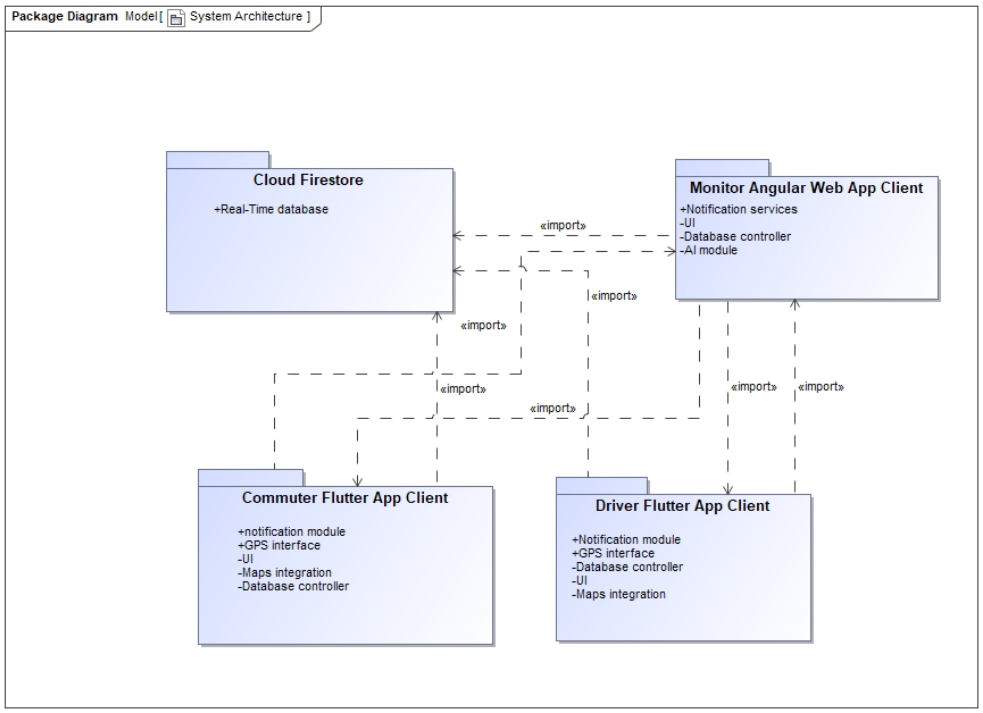

Figure 1: System Architecture

3.5 Deployment Model

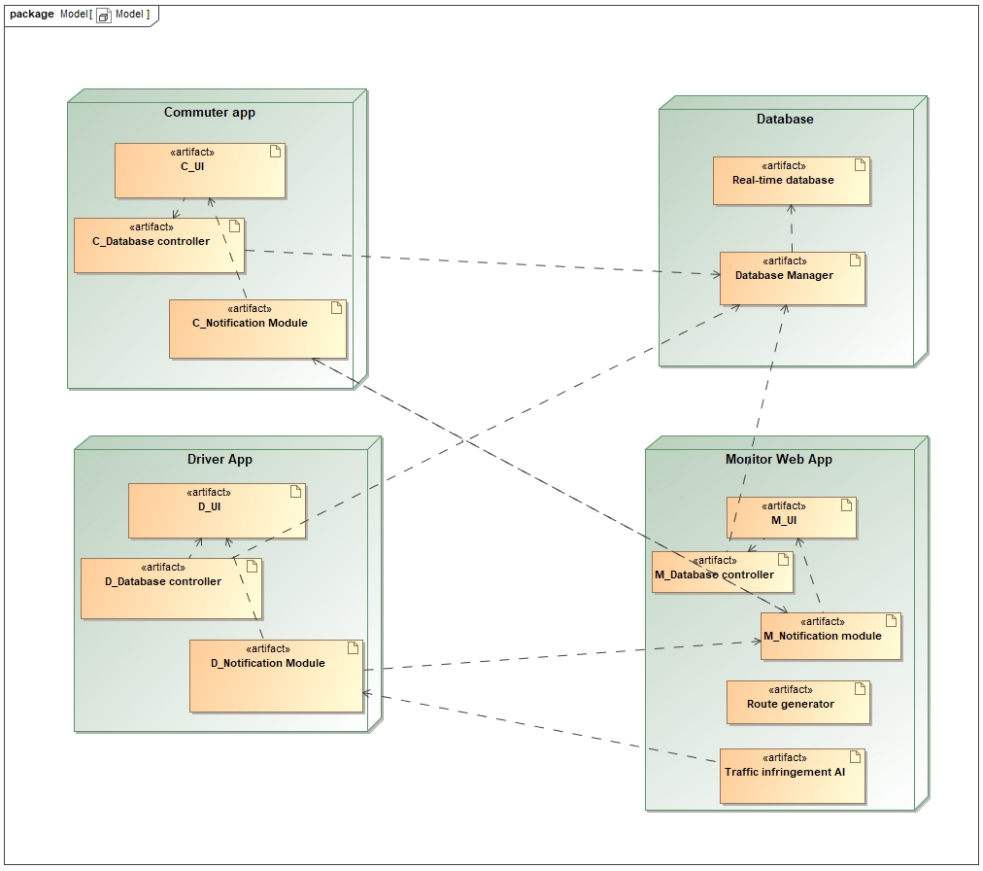

Figure 2: Deployment Diagram

4 Product Perspective

4.1 System Interfaces

The system requires cellphone or internet browser client and there is a central database server.
There will be a commuter and driver interface in the form of a cellphone app in Android and iOS.This will interact with the database. Interaction from the phone to the central database server will be handled with the Flutter UI framework. The database itself is a Cloud Firestore database.
The following data will be communicated with the server from the commuter app:

- Location data for pickup locations
- Ratings for drivers
- Follow drivers and receive updates

The following data will be communicated with the server from the driver app:

- Location data to show where the driver is
- Receive a notification when a pickup is requested at a pickup spot
- Receive a notification of bad behaviour that must be corrected
- View his most used routes
- Driving infringements
- When one of his frequent customers or followers request a ride
- View his driver rating

The following data will be communicated with the monitor browser interface:

- View drivers under this monitor
- View a driver's rating
- See the route of a specific driver

The following data will be communicated with the database server interface:

- Location information of the commuter when requesting a ride
- Location of the taxi

- Driver ratings given by the commuters
- Driver ratings of the taxi
- Route information
- Taxi driving infringements
- The amount of followers a driver has

4.2 User Interfaces

Users will interact with either one of the cellphone interfaces or with the web interface.
There will be a taxi driver app that is available on Android and iOS. There will be a commuter app that is available on Android and iOS. There will be a monitor web app.

4.3 Hardware Interfaces

The following hardware interfaces will be used:

- A smart phone with a mobile network connection and a GPS controller
- A computer
- WiFi connection to the computer
- Mobile networks
- Server side hardware

4.4 Software Interfaces

Software interfaces that will be used:

- Android
- iOS
- Flutter UI framework.
- Firebase's Cloud Firestore database interface
- Google Maps
- Any web browser
- Firebase Hosting

4.5 Communication Interfaces

The application and website will make use of the TCP/IP protocol and asynchronous HTTP requests to communicate with the server. This connection will be encrypted for security. Smart phones will communicate GPS location information with GPS satellites.

4.6 Memory

In terms of the memory requirements of the users' mobile device, it will depend on the memory of the actual device. In terms of the software application itself, it will require somewhere between 60 MB - 90 MB of storage (this is an estimate). The user's mobile device will therefore need to be able to accommodate this requirement.
The only other consideration to make in terms of the memory, is the amount of memory the database will require on the actual server.

5 Constraints

- There are no constraints relating to the type of software to be used to implement any functionality, the decision is left to the design team.
- There is a budget constraint that the project will include no additional costs for software to be used. Exceptions can be made if it is really necessary, but should be avoided.

6 Domain Model

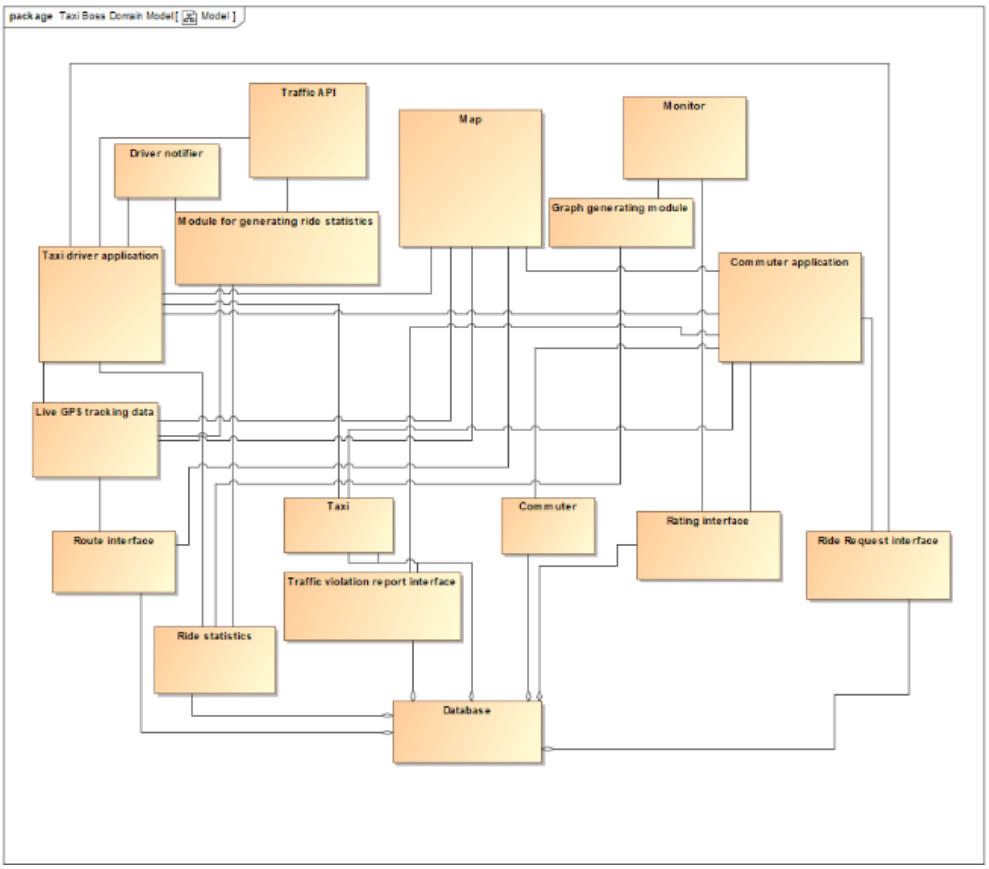

Figure 3: Taxi Boss Domain Model

7 Functional Requirements

Functional requirements are broken up into the various types of interfaces. The number of the functional requirement corresponds to the numbering scheme used in this list.

7.1 Subsystem: Taxi Driver app

1. The app will allow for a specific driver to log in and out of his/her account.
2. The app will allow for a specific driver to register an account. The registration information that will need to be provided is:
   (a) Name and surname of the driver
   (b) Email address of the driver
   (c) Driver's ID number
   (d) Registration number of the taxi the driver drives
   (e) The User ID of the monitor the driver is linked to
3. The app will allow the driver to change the registration number of his taxi.
4. The app will allow the driver to delete his/her account.
5. The allow tapp will he driver to view his current location on the map.
6. The current location of the driver will be communicated to the server.
7. The app will allow the driver to view his most-used route in the form of a heat map.
8. The app will allow the driver to send a broadcast message that will be sent to all of his followers.
9. The app will receive notifications of bad driving.
10. The app will allow the driver to view his current average star rating in terms of:
    (a) Overall rating
    (b) Taxi quality
    (c) Etiquette
    (d) Safety
    (e) Comfort
11. The app will allow the driver to view pickup locations on the map.
12. The app will receive a notification if a commuter has requested a ride at a nearby pickup spot.
13. The app will display a pickup spot differently if a commuter has requested a ride at a certain pickup spot.

7.2 Subsystem: Commuter app

14. The app will allow the commuter to log in and out of his/her account.
15. The app will allow the commuter to register an account with the following information:
    (a) Name and surname of the commuter
    (b) Email address of the commuter
16. The app will allow the commuter to delete his/her account.
17. The app will allow the commuter to view his/her current location on the map.
18. The app will allow the commuter to view pickup spots on the map.
19. The app will allow the commuter to request a ride. The request will be made if the commuter is close enough to a pickup spot. The ride request will be logged at the pickup spot the commuter is close enough to and has requested to be picked up at.
20. The app will allow the commuter to follow a certain taxi. The functionalities included in the following of a taxi are:
    (a) Receive notifications from the specific driver.
    (b) See where a followed driver is on the map.
21. The app will allow the commuter to see different routes of different taxis drawn out on the map.
22. The app will allow the commuter to report any bad or good behaviour of the driver.
23. The commuter will be able to note that he/she is currently in a certain taxi by clicking on the taxi on the map while close enough to the taxi.
24. The app will allow the commuter to give the driver a star rating in terms of:
    (a) Taxi quality
    (b) Safety
    (c) Comfort
25. The app will allow the commuter to see the average star rating of a driver in terms of:
    (a) Overall average rating
    (b) Taxi quality
    (c) Safety
    (d) Comfort

7.3 Subsystem: Web monitor

26. The website will allow a monitor to create an account with the following credentials:
    (a) Name and surname
    (b) Email address
27. The website will allow a monitor to log in and log out of his/her account.
28. The website will allow a monitor to add or remove a taxi driver from his/her account.
29. The website will allow a monitor to view the taxi drivers under his monitoring account.
30. The website will allow a monitor to view the rating of a taxi in terms of:
    (a) Overall average star rating
    (b) Taxi quality
    (c) Safety
    (d) Comfort
31. The website will allow a monitor to see comments that commuters left of a certain taxi driver.
32. The website will allow a monitor to see traffic violations committed by the driver.

7.4 Subsystem: Server-side processing

33. The server will create heat maps of where the taxis drive most frequently and build and store routes of the taxis.
34. If the server picks up that the driver has committed a traffic violation, it will notify the driver of the violation. This will be handled with an AI module.
35. The server will calculate average star ratings of the drivers.

7.5 Use Cases

7.5.1 Request a ride and rate the driver

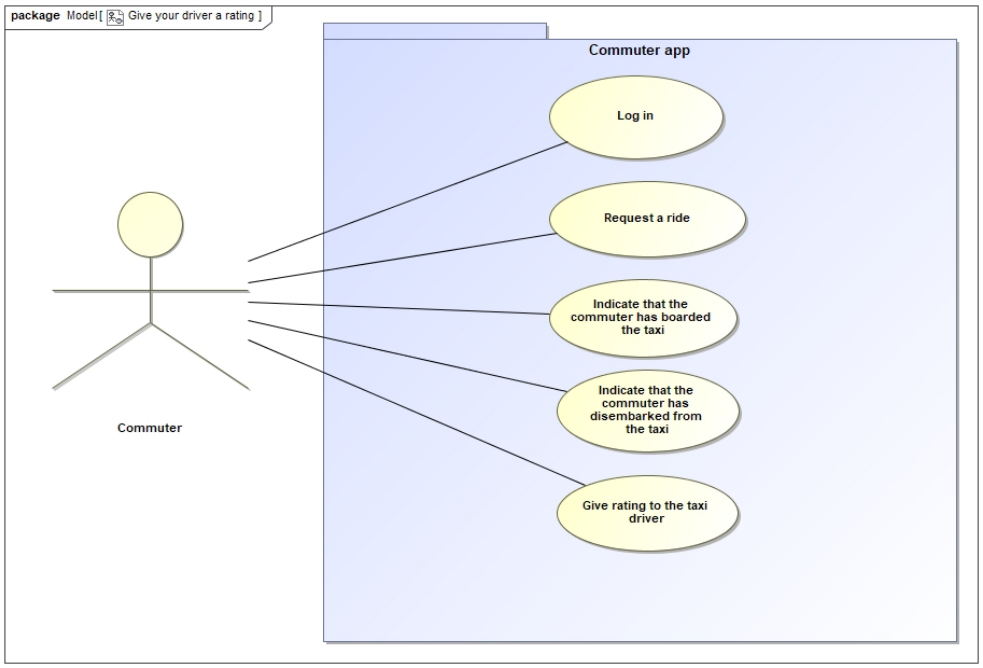

Figure 4: Request a ride and rate the driver

7.5.2 Taxi driver views his rating

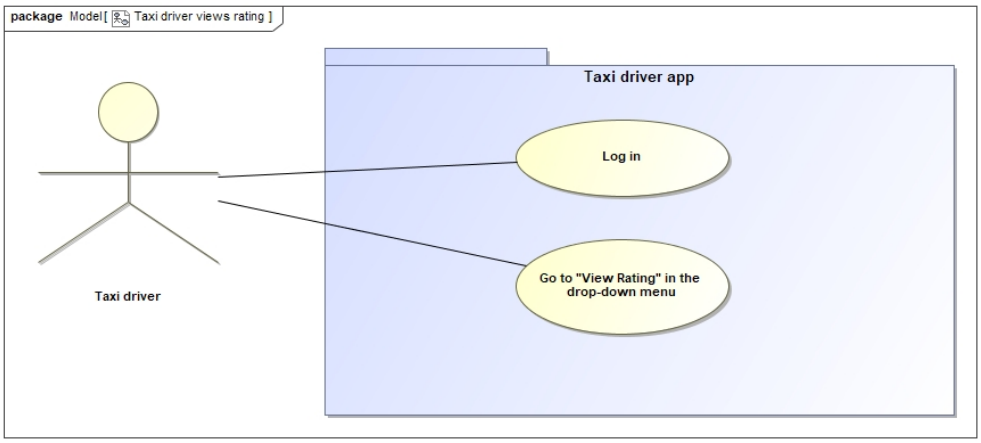

Figure 5: Taxi driver views his rating

7.5.3 Taxi driver views pickup spots

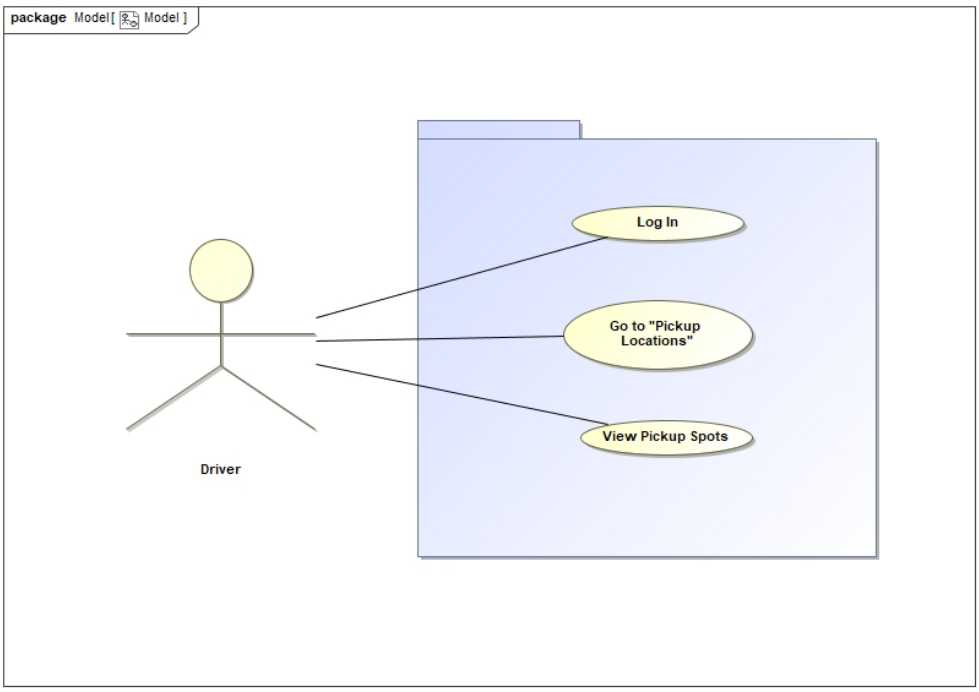

Figure 6: Taxi driver views his rating

8 Quality Requirements

8.0.1 Quality Factors

1. Commuters and drivers should have at least a 3G cellular data connection or WiFi connection of equivalent or higher speed.
2. Location data should be accurate to 20m of the current location of a user. If this is not the case, however, the app will still function satisfactorily.
3. Phone specs for drivers and commuters:
   (a) 2GHz dual core CPU.
   (b) 2GB RAM.
   (c) 3G or higher network card and or a WiFi controller.
   (d) 200MB free storage.
   (e) An internet browser that supports HTML5.
   (f) The Android Nougat or higher Android operating system or iOS 9 or higher for apple smartphones.
4. Laptop/PC specs for the monitor:
   (a) 2Hz dual core CPU.
   (b) 2GB RAM.
   (c) Internet connection of 1MB/s or higher.
   (d) An internet browser that supports HTML5.
5. If the server goes down, users will be notified in under 1 minute.
6. Data on the driver, commuter and monitor interfaces will all be live and updated in real time.

8.0.2 Server Factors

1. Uptime: The Cloud Firestore database provides 99.999% uptime. The monitor website is currently hosted on Firebase Hosting. It provides 99.5% uptime. If maintenance needs to be performed on the server, it should be done between 00:00 midnight and 04:00. This is the time when drivers and commuters are the least active.
2. Speed: The database updates in real time with a latency of 2 seconds on average.
3. Cloud Firestore is a scalable database. As traffic and storage size increases, it will be easy to upscale the server based on payed accounts. Currently it is free.

4. Sensitive data on the server will will be encrypted.

5. Monitors will have special accounts to view statistics of their various drivers.

6. Firestore Hosting is a scalable server service. As processing requirement and number of hits increases, the server can be upscaled easily with payed accounts. Currently it is free.

8.0.3 User Interface Factors

1. The commuter and taxi driver interfaces will be easy to understand so that everyone is capable of using the app to its full advantage.

2. The monitor interface will have a small learning curve to it, but given that the data to be viewed by the monitor is not very complex, it will be easy to learn.

3. All interfaces will have a logo and accompanying styling to give the app a pleasurable and recognizable feel.

8.0.4 Quality Metrics

1. Smart phones using android should ideally have the Android Pie operating system installed.

2. Smart phones using iOS should ideally have the iOS 12.2 operating system installed

3. The smart phones being used should have more than 2GB of Memory and more than 5GB of storage as well as 3G or LTE mobile network data transfer speed capabilities.

4. The Cloud Firestore database should scale according to the amount of transactions being performed and the amount of users using it. Cloud Firestore offers this in that it is free when there is only low-scale data and traffic going through it, but can easily be upgraded to a payed account that will allow all necessary transactions to take place and space be used.

5. Most laptops and desktop PC's that will be used by the monitor will have a web browser installed. The monitor view will be easily accessible.

8.1 Functional Requirement to Use Case Traceability Matrix

	UC1	*UC2*	*UC3*
R1		x	x
R2			
R3			
R4			
R5		x	x
R6		x	x
R7		x	x
R8			
R9			
R10		x	
R11		x	x
R12			x
R13		x	x
R14	x		
R15			
R16			
R17	x		
R18	x		
R19	x		
R20			
R21	x		
R22			
R23	x		
R24	x		
R25	x		
R26			
R27			
R28			
R29			
R30			
R31			
R32			
R33	x		
R34			
R35	x	x	