Taxi Master: The Smart Solution For Taxi Service Industry

D. Atapattu
Department of Computer Science and Engineering, University of Moratuwa, Sri Lanka.
dulaj.atapattu.13@cse.mrt.ac.lk

***Abstract* — Smartphone solutions are rapidly being applied to many industries to make the day to day life of the human beings easy. Travel and transportation is a service which is useful for any person. Taxi service industry plays a major role in modern transportation services. Even though the taxi service industry has such a position, this industry has not been treated with a proper solution using the latest technology to make the entire industry more smart and efficient. The proposed system, Taxi Master, addresses many underlying issues in taxi service industry and provide solutions to those problems. Taxi Master is a mobile and web based system using GPS, Google maps and many more latest technologies.**

Keywords — Mobile, GPS, Google API, Transportation

I. INTRODUCTION

Taxi services play a major role in transportation in Sri Lanka. Taxi service providers receive hundreds of orders per day. But these providers face a lot of problems and have not been able to take the maximum profit from the resources they have due to the lack of a proper system. Major problems in existing systems are inability to track driver's real-time location and the availability, higher overhead in calling and messaging to drivers, manual allocation of drivers, inability of the drivers to pickup calls while driving, lack of real time information for both the customers and the drivers about the locations of each other. Further, many complaints have filed against taxi drivers for answering phone while driving.

Taxi Master is a mobile and web based system which provides solutions to almost all of the above problems. Both customers and drivers are provided with two separate mobile applications and the taxi service center is provided with a web application. Maximizing the profit of taxi service companies by proper utilization of resources in an efficient manner and providing a satisfactory service to customers are the main purposes of the system. In addition to that, in a world where GSM calls and SMS's are being outdated, this application is easy to use with latest technologies.

The paper is structured as follows. Section II discusses existing systems in the relevant industry. Section III gives an overview of the functionality of the system. Section IV and Section V describes the functional requirements and architecture of the system respectively. Section VI describes the technologies used and the implementation techniques. Section VII discusses the testing strategies used and Section VIII concludes the paper mentioning about future enhancements.

II. LITERATURE REVIEW

PickMe [1] and Uber [2] and are the already existing applications in the same domain. Uber is an international company and PickMe is a Sri Lankan fast growing company. Both of these applications have a similar business objective. Their target is not, providing a system for taxi service companies. Their goal is to build a central taxi system by registering all individual taxis and taxi services within the country with their system. But most of the leading taxi service companies didn't like to join either with Uber or PickMe. All leading taxi companies have the fear that these systems will disrupt the base of their business. Only some small companies and individual vehicle owners and drivers have joined with PickMe, because they don't have that fear which the leading companies have [1].

Uber has different types of cars and price levels. The pricing strategy of Uber is somewhat complex as they charge for the time as well as distance of the hire. In addition to that a base payment should also be paid. Actually the rate per kilo meter is low as 30 Rupees [2]. But the problem is ultimate fare of the hire become too high due to the time based charging of Uber with higher traffic in Colombo. Therefore, Uber has not become very famous within Sri Lanka yet. PickMe is much friendly to Sri Lankan society compared to uber. But it also has drawbacks. One thing is PickMe highly depends on GPS which is not available in some areas of Sri Lanka. Even the drivers complain about this problem.

Compared to Uber and PickMe my solution Taxi Maser has a completely different business plan and an objective. Taxi Master is intended to be used as a central system of a Taxi Service company. It doesn't centralize all the taxi service companies. Meanwhile it's not just only for a single taxi service company; it can be deployed in several taxi service companies with a minimal configuration change. Add on services can also be provided based on the subscription.

Taxi Master supports real time driver location tracking. Taxi Master can handle the complete process of an order from the placement to completion of the order. No single phone call is needed except for exceptional cases. But in PickMe after placing the order rest of the process is handled through phone calls (In most cases driver calls the customer and asks where to come). Drivers can use the dedicated distance fare meter along with Taxi Master. It's not compulsory to do distance calculation using the driver's mobile application. This is useful for a country like Sri

Lanka where GPS is not available in all over the island. A Driver rating system and a customer loyalty point award system has been integrated to the Taxi Master. Customers can use these points as money in next orders. This will increase the loyalty of a customer with a specific taxi provider. Driver rating system helps to improve the quality of the services provided.

III. OVERVIEW OF THE SYSTEM AND PROCESS FOLLOWED

Taxi drivers are provided with an android application. This application can be used to set his current status and send location updates to the central server. Customer can also download an android application from Google Play to access the services offered by the taxi provider.

When a customer wants to place an order for a taxi, he/she can open the app and check nearby drivers or drivers near to a specific location entered by the user. If a driver is available customer can select the driver and place an order. Then that particular driver will get a real time notification about the order.

Driver can either accept or reject the new order. The response of the driver is informed to the customer with a push notification. If driver accepts the order, customer can track the driver's real time location and the driver can see navigation instructions to the pickup location.

In a case where customer calls and place an order, an officer at the taxi service center can manually place the order using the web system. In this scenario customer cannot track the driver as customer doesn't interact with the system directly. Taxi Master can also be used to record and view the order history through the web system.

IV. SYSTEM MODELS

A. System Requirements

The driver can set his current state and then the location update settings are automatically set according to the state. The driver receives new order notifications through the app and he can either accept or reject the order. If driver accepts the order, it will be added to his ongoing orders list. Driver can obtain navigation details to the customer's pick up location when going for the hire.

Customers can enter pick up location and destination and check the available drivers at the moment. If a driver is available, a new order can be placed using the application. Customer can track the real-time location of the driver while driver is coming to pick him/her. At the end of the hire customers can rate the drivers and give a comment. In addition to that customers receive some loyalty points as a percentage of the hire fare. Customers can spend earned loyalty points in future hires. Registered customers of the system receive push notifications about offers and discounts when available.

There is an admin panel for monitoring purposes and resource management activities. Orders received on phone calls can be placed using the admin panel. Admin panel is a web application.

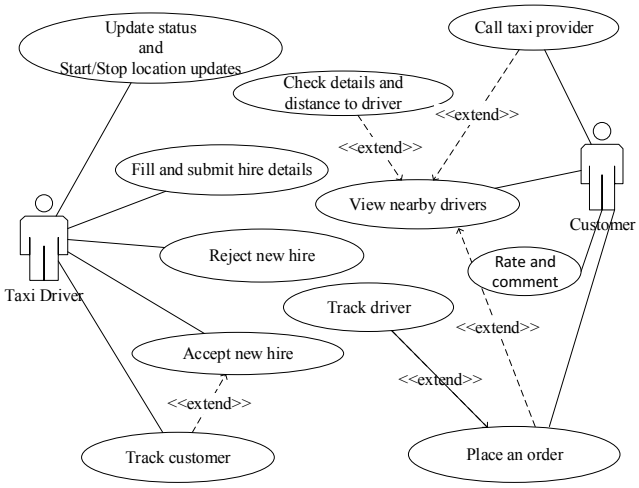

Fig. 10. Use cases related to taxi driver and customer

Figure 1, represents the functional requirements of the taxi drivers and the customers of the system. Customers and taxi drivers completely depend on the system for orders. Therefore, reliability is a major non-functional requirement of the system. Higher customer request load should be expected in peak hours and that load should be manageable by the system without losing the performance. System should be secured from unauthorized access. At least a GPS accuracy of 100 meters is expected form the system.

V. SYSTEM DESIGN

A. Architecture of the system

The main architecture of the system is model-view-controller (MVC) architecture. The web application and the server side implementation exactly follows the MVC architecture and two android applications follow model-view-presenter (MVP) architecture which is an extension of the MVC architecture.

In web and server implementation all the requests are passed to the Laravel routes layer which is an extra security layer added by the Laravel framework [3]. Routes layer direct these requests to controller layer after authorizing the request. Model layer is responsible for all the data access logics. View layer return the well formatted output [3].

In MVP architecture the presenter acts as the intermediate layer between model and view layers. View layer handles all user interface related activities and model layer handles all network and other logics. Presenter is responsible for calling necessary model layer functions related to user interactions and returning the response. Due to this layered architecture the user interface or the logic of the application can be changed with a minimal impact to other layers [4].

B. Logical View

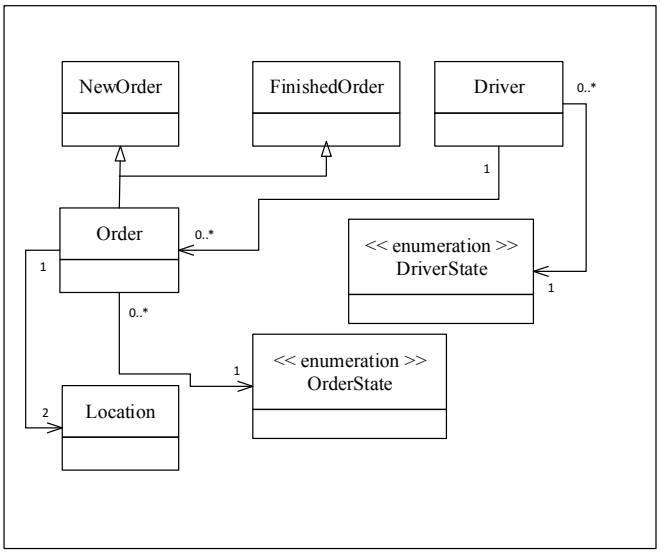

Fig. 11. Class diagram of driver app

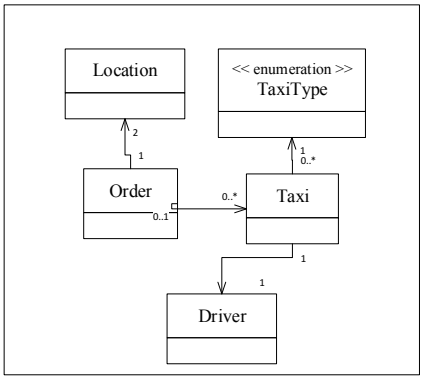

Fig. 12. Class diagram of customer app

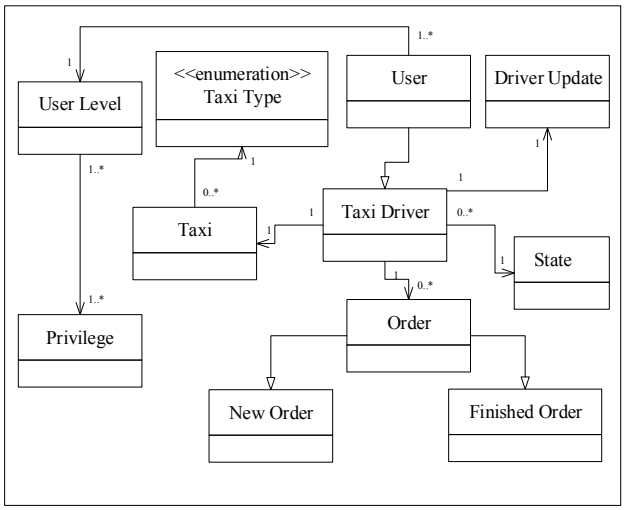

Fig. 13. Class diagram of web app

Figure 2, Figure 3 and Figure 4 represent the class diagram of the Taxi Master classified into sub applications. Each "Driver" has a "Driver State" and that state is used to determine the availability of that driver for an order. Customers place "New Orders" on drivers and the orders also have different "Order States" according to the progress of the order. "Location" represents the origin and destination locations of an "Order". Drivers periodically send "Driver Updates" to the central server informing their current "Location" and the "State". Each "User" of the system has a "User Level" and each user "User Level" is associated with a set of "Privileges".

C. Process View

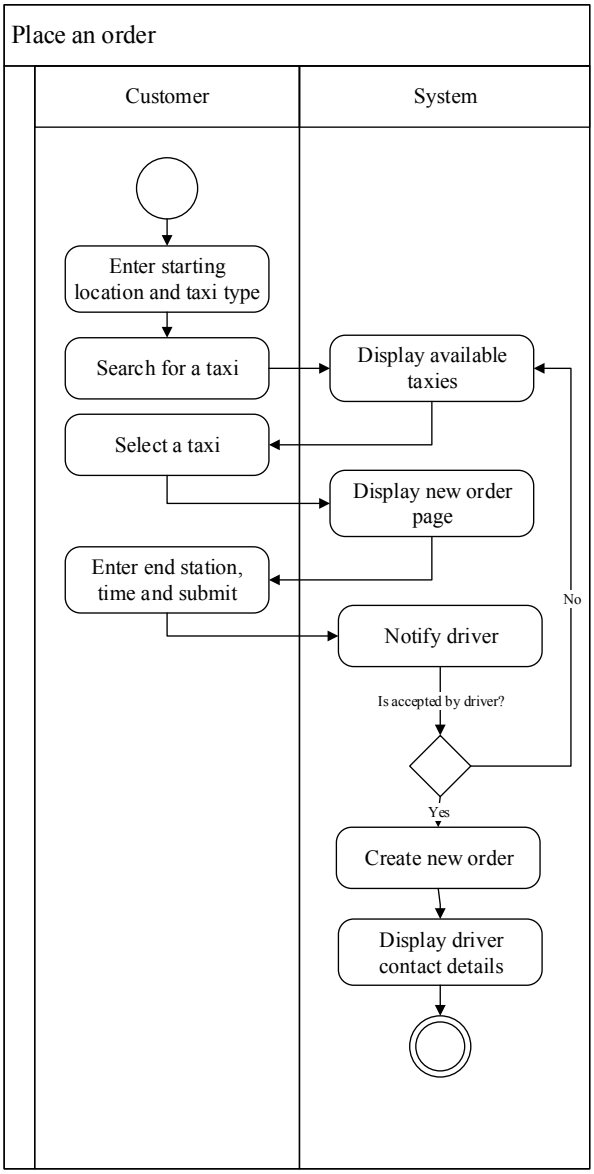

Fig. 14. Activity diagram for place an order use case

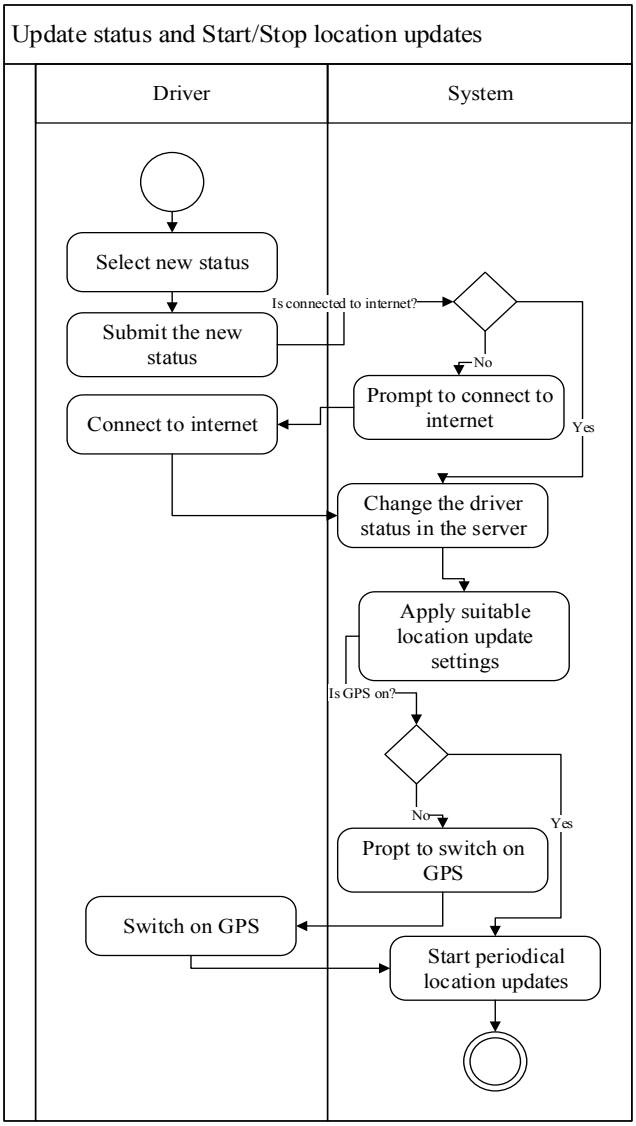

Fig. 15. Activity diagram for update status and Start/Stop location updates use case

Driver and the customer are the main user roles of the system. Setting correct state and sending his own location updates to the server is the main use case and the responsibility of the driver. This will help customer to pick up the drivers suitable for their hires.

Main use case of the customer is placing a new order. Customer can search for nearby drivers and place a new order after filling out necessary details.

D. Database design

User table stores all the personal details and credentials of system administrators, taxi drivers and taxi operators. There is a child taxi driver table to store the details only related to taxi drivers. All users are related with a specific user level and each user level is related with a set of privileges.

The new orders placed by the customers are temporarily stored in new orders table. Successfully completed orders are moved to the finished orders tables for order history management purposes. New orders table is cleared automatically once a month to remove unwanted data from the table.

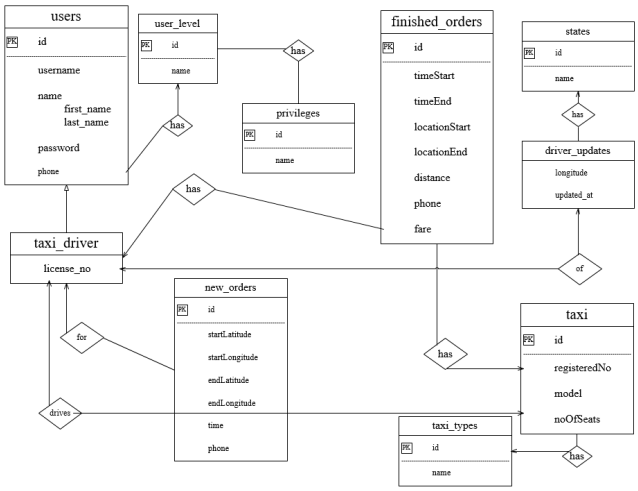

Fig. 16. Entity Relationship diagram of Taxi Master

State table and taxi type tables are static tables and hold all possible states of the drivers and all taxi types respectively. Detail about all the taxis the company have are stored in taxis tables. Each taxi has a taxi type and the taxis are mapped with taxi drivers. Driver updates table contains the latest state and the location of each taxi driver.

VI. SYSTEM IMPLEMENTATION

A. Implementation Procedure

Android studio [5] was used for android application development and PhpStorm [6] was used for web and server side development. Both of the IDE's are IntelliJ based and provide a very interactive and sophisticated development environment.

PHP was used with Laravel framework for server side scripting. Therefore, the database was implemented using Laravel migrations which provide easy version management for the database. The data access layer is completely managed by the Laravel eloquent ORM and raw MySQL queries were not used. This will make the system migratable from MySQL to PostgreSQL without any modification. Access management and authorization is controlled by Laravel middleware and the auth façade. Laravel middleware was also used to handle cross site reference forgery attacks using CDRF tokens. Laravel blades were used for web interface designing which made the process easier and efficient by applying concepts of reuse.

Both of the android applications are native and have been written in java. Material design concepts were used for user interfaces which will make the applications more interactive and self-understandable. GSON library was use to serialize

and de-serialize java objects to JSON when communicating through the network. Google fused location service was used to fetch GPS coordinates instead of the android location provider. Fused location services have been optimized to provide the location with better accuracy with lower power consumption.

Several Google APIs were used for location based data and other calculations. To select drivers for orders distance between the driver the customer's pick up location has to be calculated. This was done by using Google distance matrix API. For better accuracy and interactivity locations are suggested when user types for a pick up locations or a destination. These suggestions are provided by Google Places API. When a driver is going to pick up a customer, driver can enable navigation to driver's pick up location within the application (without switching to Google maps). These navigation services are provided by Google navigation API. Real-time notifications are needed for both customer and driver applications. All notifications are sent as push notifications using Google cloud messaging (GCM) and OneSignal API.

As the initial development step, the database was developed. Then the system was developed in use case wise. To complete most of the main use cases of the system, functionality of all three applications (driver application, customer applications and server) are needed. Therefor all three applications were developed in parallel. Since both of the android application are using GPS, real android devices were used to test the applications from the beginning of the development stage. Server was hosted locally with the help of XAMPP server and has been forwarded to a public IP using the tunneling tool ngrok [7]. The public URL provided by ngrok was used to access the server from android applications.

B. *Algorithm*

```
getAvailableTaxis(request)
      responseList = []
      origin ← request.origin
      taxiType ← request.taxiType
      availableTaxiList = Database.read("select all from
                        taxi_drivers where taxiType=taxiType
                        ordered by ((latitude - origin.latitude
                        + longitude - origin.longitude)/2)
                        asc limit 10")
      distancesList = getDistances(availableTaxiList)
      i ← 0
      while i < distanceList.length
            if(distanceList.status == "OK")
                  responseList
                        .add([availableTaxiLust[i], distanceList[i])
            i++

      return responseList

getDistances()
      url = GOOGLE_DISTANCE_MATRIX_API_URL
      for each taxi in availableTaxiList
            url = url + taxi.latitude + "," + taxi.longitude
      return (get data from google api);
```

Fig. 17. Pseudo code of the algorithm used to filter the available taxis for an order

After customer enters the pickup location available list of taxis should be displayed. In the real world scenario there is a lot of drivers in the database. Therefore, we cannot just return all the available drivers. Fig. 8. Represents the algorithm used for this filtering.

First the average difference between the driver's coordinates and coordinates of the customer's pickup location are calculated. Then 10 taxi drivers having the least coordinate difference are selected form the database. Then real distances between those selected drivers and customer's pick up location are calculated using Google distance matrix API. If API returns a response with status flag "OK" then that taxi and the distance information returned form Google are added to the response. Finally, the response list is returned back to the customer's application.

C. *Main Interfaces*

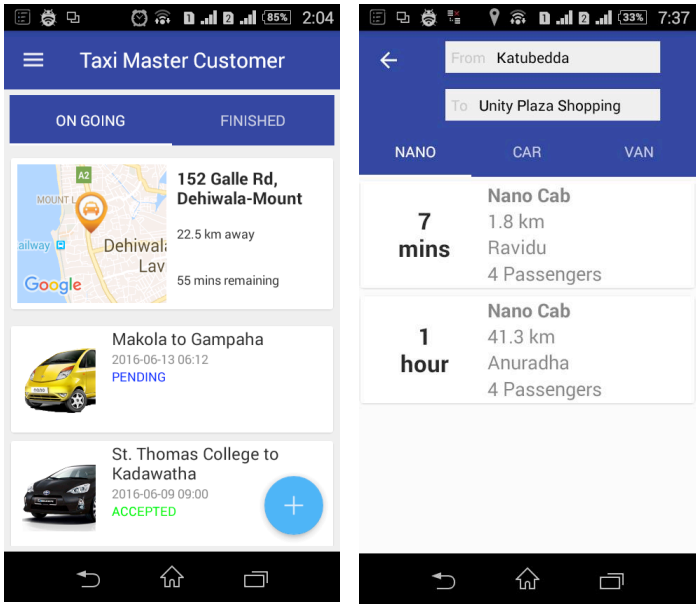

Fig. 18. Customer application main user interfaces

Figure 9. Left user interface is used to search for available taxis. Available list of taxis will be displayed. Right interface is used to place a new order on a selected taxi.

Figure 10. Left interface is used to set the state of the driver and to start/stop sending location updates. Right interface can be used to get navigation details from driver's current location to customer's pick up location (Prior to opening this interface driver has to accept and look for any bookings)

Figure 11. Dashboard is the main user interface of the web application. This is available to both administrator account and taxi operator accounts. Using the dashboard all the taxis of the company can be monitored. Taxis can be filtered using the either the state of the taxi (available, going for hire, in hire, not in service) or the type of the taxi (nano, car, van).

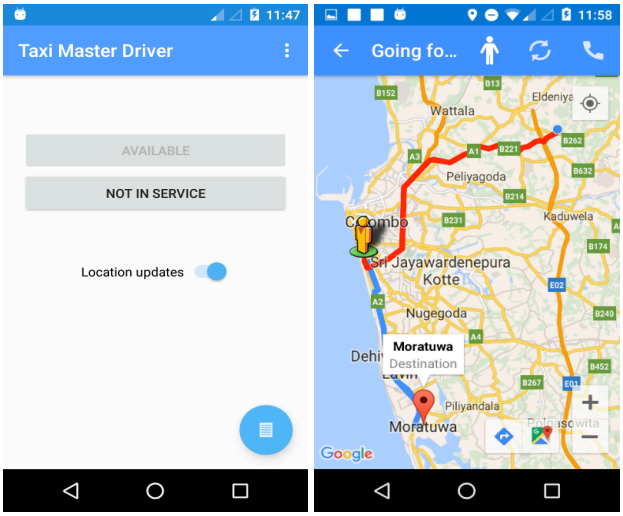

Fig. 19. Driver application main user interfaces

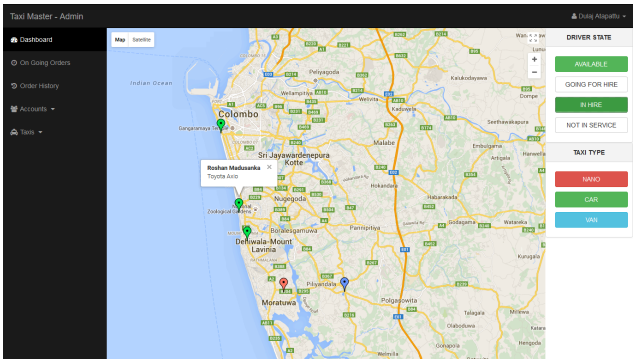

Fig. 20. Web application - dashboard

VII. SYSTEM TESTING AND ANALYSIS

While developing the system, unit testing was used to test all major functions. PHPUnit was used for testing the web application which comes out of the box with Laravel framework [5]. All routes of the web application were requested using test methods and verified that they are working as expected. Authentication and authorization were also tested using both authorized and unauthorized requests. Since the data access layer is managed by Laravel framework, it doesn't need to be tested [8].

Component testing were also done using PHPUnit. Laravel testing supports creating mock objects and mock request to test controllers and other data returning functions. Back end validations when entering new data to the table were tested by trying to input invalid data tests.

Selenium IDE [9] was used for user interface and activity flow testing. All the activity flows were recorded using Selenium IDE for both user types, admin and taxi operators. All the links, buttons filters and all other components where user interacts were tested and verified using user interface testing. All data entering user interfaces were tested for front end validations.

JUnit with espresso framework [10] was used for android unit testing. Both of the apps use android specific and hardware required features such as internet connectivity and GPS. Therefore, all the unit tests were designed using android instrumentation testing not local unit testing. Connectivity to the server and all network related methods were tested by both reading and writing data to the remote database. Location listener was also tested and verified for the required accuracy and the delay.

Android test recorder supplied by Droid Test Lab were used to record test cases. As the automatic assertions are not supported by android test recorder, the test records generated by android test recorder were edited to include assertions for verification.

VIII. CONCLUSION AND FUTURE WORK

Taxi Master is a complete taxi tracking and order management system which can be used at any large or medium scale taxi service company. Real time driver locations, semi-automated driver-customer allocation and real time driver tracking are the highlighting features of the system. With the taxi master, drivers don't need to answer phone calls while driving and go here and there without knowing the exact location of the driver. Also customers can track the real time location of the driver using the android application. Taxi Master is the smartest solution to taxi industry with smart phone technology.

In addition to the existing features "My favorites" feature is a useful future implementation to customer's application which will help customers to save and reuse similar types of routes and orders. And merge rides feature is another big feature which is to be implemented. By merging multiple rides of the same route during the same time interval, taxi companies will be able to provide lower rates to customers. Finally, the web application can be extended to place orders online.

REFERENCES

[1] *Pickme and the taxi revolustion*. (2016, 10 12). Retrieved from ReadMe.lk: http://www.readme.lk/pickme-taxi-revolution/

[2] *uberGO, the most affordable ride in Colombo*. (14, 10 2016). Retrieved from Uber News, Events, Partnerships, Product Updates and More: https://newsroom.uber.com/sri-lanka/ubergo-the-most-affordable-ride-in-colombo/

[3] Architecture of Laravel Applications. (2016, 04 10). Retrieved from Laravel Book: http://laravelbook.com/laravel-architecture/

[4] MVP for Android. (2016, 04 11). Retrieved from antonioleiva: http://antonioleiva.com/mvp-android/

[5] Android Studio, [online] Available at https://developer.android.com/studio/index.html

[6] PhpStom, [online] Available at https://www.jetbrains.com/phpstorm/

[7] ngrok, [online] Available at https://ngrok.com/

[8] Testing - Laravel - The PHP Framework for Web Artisans. (2016, 06 16). Retrieved from Laravel - The PHP Framework for Web Artisans: https://laravel.com/docs/5.1/testing

[9] Selenium IDE, [online] Available at http://www.seleniumhq.org/

[10] Testing UI for a Single App, [online] Available at https://developer.android.com/training/testing/ui-testing/espresso-testing.html