Haaga-Helia

Android Application for Taxi Management

Usman Ali

Bachelor's Thesis

Business Information Technology

2018

Haaga-Helia

Abstract

Author
Usman Ali

Degree programme
Business Information Technology

Thesis Title	**Number of report pages and appendix pages**
Android Application for Taxi Management	32 + 7

Android applications are indeed a very handy tool for android users. Android users can access to several Applications at the same time from one android device. Applications did not only remove the problem of communication but also solved several issues in daily life such as tracking assets, banking information, media, education, sports and so on. The importance of applications for android devices cannot be ignored. It brings several benefits for users and owner of the app. Moreover, the number of android users is also in millions and growing all the time.

This thesis is also about android application and the goal is to track location of device holder especially in the taxi business. Therefore, we researched on location tracking of android phone devices and to record the location for long term that will also be accessible for future. The purpose was to find out that how is it possible that applications can record location of android users. Moreover, admin and others can access the location of taxi driver and the fuel recharged in car by driver.

Several technology tutorials and courses were taken online to understand that how tracking application works. The study of android application helped to make a multiple small apps to get the idea about android platform. After that, the development was narrowed down to android google maps. Furthermore, studying google maps docs was also making a way to achieve the main goal of the project. All the learning methods were implemented to test the application in multiple android devices.

Finally, we found out a way that connected us to the destination of research project. It was server and database who were sitting behind the application and the android google maps api was involved with them. Google maps api was taking the coordinates and sending to the server that was storing into database. The data was accessible at any time in future so android device was making the get request as well as fetching the data from database. The data was drawing lines by collecting all the coordinates. Therefore, we found the actual path and movements of device holder.

Keywords
Android maps, Location Tracking, Polylines, Retrofit, Java, Google Maps

Table of Contents

1 Introduction

A smart mobile device has become a basic need for human being because it is very easy to connect other humans, services and products. The mobile device has several benefits for its users, device sellers and application creators. The major benefit is to have several useful features known as applications. The applications run in these mobile devices that help to make life easy as well as improve daily life activities. For instance, a mobile device can have almost everything that can benefits daily life like calendar, phone calls, messages, online browsing, video streaming, recording audio, video, camera, emails and so on. Furthermore, additional features can be added to mobile devices to get extra fun and useful information. Similarly, this text provides information about a project that is an application made for location tracking system for android mobile devices.

Android is an operating system primarily used in smartphones and tablets that is currently owned and developed by Google. Google provides applications by a platform called playstore where applications can be downloaded and installed. Number of available applications in google playstore are in millions. Similarly, this text is also about google android application of location tracking that defines current and previous location of user as well as provide the feature of adding, removing, updating and deleting the fuel cost.

Android device can help us to track the previous, current and moving location of an object. For instance, a person who has a mobile device can be traced easily with his phone or a car that has android Auto system. A car can be found in parked or moving situation with the help of location tracking system. The tracking location system helps us to analyze the activities of a person or some object to feel secure about it. For instance, if a car is stopped for a long time so it can be understood as nowhere to go, some issue with the car or driver is just busy around that location.

The application in this project is developed in native language with native development environment such as Java and android studio. Moreover, it is also recommended by developer of android operating system to use the native environment. Currently, there are several other options to develop the android application such as react native, ionic, vue native, hybrid app etc. However, the

selection of language, platform and development environment was based on requirement of this project. This text does not cover the development of hybrid app or any other way of developing application.

The overall objective of this thesis is to describe the processes and stages of development of location tracking application in android platform in addition to providing api knowledge to communicate with server natively. For this sake, previous researches related to this project are taken into consideration and discussed to support the text in second chapter. Moreover, planning and implementation of development process is added in third chapter. Furthermore, logical part and results of the project is mentioned in fourth chapter. Additionally, conclusion took place in chapter five and lastly appendix is explaining the code used in development of application after the references.

1.1 Goal(s) of this Development project

The goal of this project is to get real time information about taxi location and its fuel cost in android mobile devices. This information will help the admin to analyze the business situation. The possibility to achieve this goal in digital world is to provide application that can allow drivers to add fuel cost and track them.

Another goal is to make use of google maps api to make location tracking system in android. Furthermore, the goal is also to use location to track the driver and save location for long term to see in future as well as enable the driver to add the fuel cost.

1.2 Research questions

How a user can be tracked through android mobile device? How tracking data can be retrieved in future? How polylines make the routes in Maps? How CRUD functionality is developed in android platform.

1.3 Scope of thesis

User can see his current location. Moreover, the places a user visited yesterday can also be retrieved as a path in maps. Similarly, the fuel information can be added and can also found earlier fuel entries added in past. The app user can also see other user’s location and fuel entries. Fuel entries can be viewed, updated and deleted at any time.

1.4 Out of scope

Driver will not be able to share his current location without internet connection and without location permission granted. This is because it is not possible for google maps api to serve without internet or permission granted. Similarly, driver would also not see the fuel list or add fuel cost without internet. This is because internet enables application to connect with database. However, internet connection will retrieve the location path but recording location will depend on location access. Furthermore, user interface of application will be available without internet and location access. Lastly, the user interface without access to location and internet will neither provide any service related to fuel object nor to location object.

2 Theoretical Framework

This chapter explains the theory presented in earlier research papers, journals or any of the established sources. The technologies mentioned in this chapter are subtopics related to the technology used in location tracking development project. Additionally, I have discussed researchers' views about the technologies at general level. For instance, technology's key concept, importance, use, criticism and benefits to support the project.

2.1 Hybrid applications

I described the reason of not using hybrid app development in first chapter. Xanthopoulos & Xinogalos described that hybrid applications are mobile apps that are made by web technologies. For example, HTML, CSS, JavaScript, jQuery etc. They mentioned the advantage of Hybrid apps that it gives dual benefit of iOS and Android platform by using cross-platform development technologies. Another good point in hybrid app is that same skills used in developing websites can be used in developing a hybrid application. Additionally, the benefit of developing hybrid apps is that it targets multiple platforms. These apps are same as a native application and the reason is that native environment uses the WebView of device platform. Generally, it gives access to hardware of the device such as GPS and accelerometer. However, if it does not give access then it cannot use the native browser functionality. (Xanthopoulos & Xinogalos 2016)

There are several benefits of hybrid application and one is that it is capable to get the geolocation even in the background. Moreover, several plugins are available that can be utilized and improve the battery life. The location feature works in Phonegap as GPS sensor of a device and geolocation is used to access the location. Plugins are implemented by Cordova and they are installed in the project to get additional features. PhoneGap uses geolocation api that follows the W3C requirements. Moreover, it runs on devices that even don't support the geolocation functionality. Authors also argue that despite of several benefits of hybrid app, it is not real native app because it creates the simulation of real native application by using WebView of device. (Xanthopoulos & Xinogalos 2016) However, google recommend using native environment and native language for android development.

2.2 Google Maps

On the other hand, Google strongly advise to use complete native environment for the development of android application. Google also mentioned in docs that android studio is a recommended tool for development environment specifically making a maps application with android. Google docs provide easy steps to get the map on android screen. However, there are certain requirements such as google maps api key from google account to connect the Maps servers. It is generally used with applications that deal with maps SDK in android. Furthermore, there are sufficient instructions to create API key in google docs and ensure that api key is mentioned in android manifest file. Google maps api key is the prerequisite for developing the maps application and the key should also be enabled. One more optional requirement is that key should be restricted to Android Application for safe use. (Google Maps Android 2018)

2.2.1 Location Permission in Google Maps

Google maps provide two types of permissions for location such as ACCESS_FINE_LOCATION and ACCESS_COARSE_LOCATION whereas hybrid applications usually get location from geo location api with no guarantee of receiving actual location. These permissions provided by google maps also vary in accuracy of location. For instance, Coarse location permission provides location according to the city block. These permissions are added in android manifest file (Developers, Documentation 2018). Xanthopoulos & Xinogalos stated that when device holder moves then location is updated and applications have manifest file that contains location permissions also with other data. Therefore, it is a requirement that application should request permission for receiving updates. (Xanthopoulos & Xinogalos. 2016)

Google also suggests that application should add location permission request in code if the application needs location access. Furthermore, there will also be permission added for location in android manifest file. The application can be tested by connecting a physical mobile device or with Android Emulator. Google also mentioned in docs that multiple devices can be configured through android virtual device manager that can be used with android emulator. (Google Maps Android 2018)

2.2.2 Polylines in Google Maps:

Lines connected with each other on maps is called polylines. This is a helpful feature of android google maps that can be used to draw lines on map. These lines can connect to several locations, creating paths and routes in application. However, the level of complexity is also high depending on required feature and implementation of polylines. PolylineOptions is an object that is required to define the details of polyline. This object is inserted as a parameter in GoogleMap.addPolyline method that is used to draw the lines on map. Google provides the code example as given below.

```
“Polyline polyline1 = googleMap.addPolyline(new PolylineOptions()
      .add(   new LatLng(-35.016, 143.321),
            new LatLng(-34.747, 145.592),
            new LatLng(-32.306, 149.248),
            new LatLng(-32.491, 147.309)));” (Google Maps Android 2018)
```

The above code is a basic example of creating polyline that has four points to connect. However, there is a lot of functionality in polylines that can be added to the application.

2.3 Android and iOS Efforts for Maps

Xanthopoulos & Xinogalos also mentioned that there are other alternatives of Google maps. They further discussed about Apple that is a well-known technology company and a big competitor of android operating system. According to authors, Apple did not continue using google map in 2012 and came up with its own mapping solution with additional features for a better user experience. Apple also worked more on mapping and they acquired the company like iBeacon Bluetooth-based technology for indoor navigation in 2013. Similarly, Google also introduced location API service that is a part of google play services and is used as a prerequisite for most google APIs like google maps, games etc. (Xanthopoulos & Xinogalos 2016)

Authors further elaborated that Google map is not the only option whereas Apple is also providing mapping features. Moreover, the effort is coming from both Google and Apple for an optimal solution for mapping so there is still not perfect solution

yet. Additionally, it can be a discouraging task for the implementation of location-based features as well as a complex process to identify the location of user from a mobile device. There is also the possibility of error in getting the location that can lead to inaccurate location information. For instance, several providers like Wi-Fi and GPS have different qualities and drawbacks in battery usage, speed and accuracy. (Xanthopoulos & Xinogalos 2016)

Moreover, estimation of user's location must be repeated with a different time interval because user also move around. In addition, the location measurement can also be different in accuracy. For instance, the location retrieved from one provider can be more accurate than the others. As the mobile devices have GPS sensors and limitation of these sensors is that they do not work accurately in indoor areas. The reason is that no direct connection with satellite and it destroys the accuracy. Apple was quite active for indoor mapping technologies from several years. They also acquired companies to enhance the indoor mapping service. (Xanthopoulos & Xinogalos 2016)

2.4 Location

We can also find from previous researches that how location technology works in mobile devices. According to Yue, Zhang & Jacobsen, Google play location services are used by clients and these services help in recording location and position of users. Moreover, the services also help in assessing the user movement that whether user is walking, on bicycle or in car (Yue & al. 2013). Xanthaopoulos & Xinogalos also mentioned about location awareness in their research. They pointed out that everyone carries mobile devices with them all the time. Therefore, it is crucial to keep users aware about the world around. For example, people, places, events, festivals and so on. Furthermore, it is possible from geographic information of the mobile phone devices as well as location awareness. The location awareness improves the experience of location feature. There are several sensors available in each mobile phone device and this makes the control on location very easy and feasible. For example, tracking the current location automatically, moving objects and geofencing etc. (Xanthopoulos & Xinogalos 2016) However, Podiyan, Butakov & Zavarsky (2015) argues on giving unauthorize access of mobile devices. Similarly, according to Sarma, Li, Gates, Potharaju & Nita-Rotaru (2012), mobile phone

devices are also in use of financial authentication such as bank accounts, credit cards and so on. Users also access remotely to their business information. Therefore, there is security risk involved in sharing location with the application. (Sarma & al. 2012)

Authors further explained about development of location technology mechanism. Location providers are used in accessing Android location technologies. The access is given to sensors and technology by implementing the information via location providers. There are APIs such as Google play services location API as well as Android framework location API and one of both can be utilized to identify the users' location. The communication must be done between location-oriented applications and one of both APIs to get the location. The fused location provider is used that help in changing the use of location provider dynamically. Google introduced Play services location API and considered as a part of Google Play services that is used for running the most google apis. Google does not encourage to use any other API than Google Play location API. There are several reasons like it has simple interface and high in accuracy in addition to less power consumption. (Xanthopoulos & Xinogalos 2016)

2.5 Location Based Services (LBS)

Authors also have considered location-based services in their research. For instance, several application requests to get the access of device location. These applications provide services based on location. For instance, nearest places like restaurant, sports, gym, temperature, shops and so on. According to Podiyan & al., there are privacy issues that bring the attention of researchers. Number of applications that is based on location is getting huge and mobile device users are unsure about giving the access of location information. There are tools available that can disable some requests in addition to location such as Mockdroid platform. The privacy issue will be resolved but the full functionality and a better experience will not be achieved. This is a must scenario where applications depending on location need access to location to complete their aim. Additionally, this factor totally depends on individual users that whether they want to maintain their privacy or utility of an application. (Podiyan & al. 2015)

Another research also paid attention on location sharing. According to Sarma & al., mobile devices are in huge numbers and not only for personal use but also used for business and several other purposes. In addition, it is also very crucial that users should be aware of the advantages and limitations or threat of using applications of mobile devices. There are two forms of data attached to mobile devices such as private traditional data like phone numbers, contact information, credit card number and so on. Similarly, resources of new types like accurate location, calling on other phones, recording audio, SMS message and so on. These resources require wireless networks with high speed of connection with internet. The trade-off in this shift of computing is a compromise of personal information. However, it seems that user ignores the risk associated with an app downloaded from app store and take the risk on its own. (Sarma & al. 2012)

2.6 Battery Consumption

Battery is the main concern in mobile devices because devices need power to run. Battery also depends on several factors like quality and lifetime of a battery, usage of mobile device, applications running in foreground and background etc. Similarly, some applications take a lot of energy than others. For instance, location-based applications consume high power because applications use several sensors available in mobile device. Discussion about battery consumption can also be found in earlier researches.

2.6.1 HTTP Requests and Battery

There is need of http requests in application to communicate with server. Application behaviour also affects the battery consumption. According to Li & Halfond, energy plays a vital role in mobile apps. All the functionality of an app consumes energy while running. HTTTP requests in an app takes a lot of energy. Authors mentioned that there is need of work for finding the optimum way to consume energy while making HTTP requests in mobile applications. Earlier research of authors revealed that calling http request is very expensive for mobile applications. Http requests in an application takes around 32% of non-idle state energy in average. A major portion for energy efficiency can be unnoticed if developers and researchers ignore the http requests. Authors tested in their earlier study and concluded with energy inefficiency even in making small http requests. (Li & Halfond. 2015)

2.6.2 Location and Battery

According to Android developer guides, battery drains depend on several factors in case of location dependent apps. For instance, accuracy takes the sufficient battery specifically if the accuracy is set on high level. Moreover, measuring the location with a very short interval is also one of the reasons of quick battery drain. For saving battery, location should be computed less frequent. If the latency is not high that also impact the battery. In the case of latency, location should be delivered quickly (Developers, Documentation 2018). On the contrary, Li & Halfond (2015) discussed about battery that it really helps the developer to generate more revenue and get the satisfaction of app users by improving the energy consumption of an app. Moreover, they argued that there are researches available that offered several ways to reduce energy consumption for applications of mobile device but considering http requests for energy consumption was ignored completely. (Li & Halfond. 2015)

2.7 Retrofit

Communication between server and client also needs http request. Moreover, there are several libraries available for http requests to communicate with server in android java and one easy and secure option is Retrofit. According to Pöhlas & Peitek, retrofit is for type-safety in HTTP client request of java and android. Pöhlas explains the problem of http requests that it is inconvenient to work with api especially if there are several scenarios. The major issues are to work with threads like runnable thread or worker thread in addition to Async Tasks. Furthermore, it is painful to deal with implementation of AsyncTasks and threads. It saves time and avoid stress if using retrofit in the application. There are certain requirements in retrofit like annotations are used in java interfaces for each api endpoint and its http request. (Pöhls & Peitek 2017, 1.)

Moreover, retrofit is used to make HTTP requests to server and this process is done with the help of internet. Therefore, internet permission must be mentioned in android manifest file of the android project. After that, define the retrofit dependency in pom.xml in case of Maven. On the other hand, define dependency in build.gradle if having the gradle project. Sync the project so packages can be imported. In addition, GSON is used to convert JSON data in retrofit so we also need to add

manually. GSON is from Google that is used to map Java objects. We need to use GSON library as retrofit does not convert response automatically. Therefore, another dependency is needed to add in build.gradle or pom.xml. Api interface for annotations and endpoints of http requests, service generator for set up of retrofit base URL, standard java class to create get and set methods depending on http requests, activity class to call the actual method available in Api interface. (Pöhls & Peitek 2017, 3-12.)

2.8 Spring Framework:

Spring framework is written in Java and can be used for backend operations as well as several other purposes. According to Saxena, Kaushik & Kaushik, spring is a java application framework that is open source and considered the most popular. Additionally, it follows the industry standard in specifically hibernate and struts in framework as well as put them all together into one package. Authors also mentioned that currently, there are three senses of taste in Java and every sense delivers certain requirements of programming. Furthermore, it is not a bad idea to consider java language for developing enterprise level applications because it is one of the main languages from programming. Additionally, Java did not remain to web browsers but also improved to large scale distributed applications maintained by several servers over the years. Java is also a big business because spring framework as it is considered mostly for application growth structure. (Saxena & al. 2016)

Spring framework is also used by a lot number of designers all over the world for developing reusable code, high performance, testable and effortless applications. Spring is also a very less weight framework. Several unlike structures such as JSF, hibernate, struts etc. are backed by spring and known as a body of the systems. It is java that makes the way for application to execute efficiently and it is possible because it distributes execution of application at multiple levels. Authors also discussed the dependency injection as it is also used in spring framework. There are several benefits of spring framework such as it is less in weight that the whole system can be packed into one jar file with the weight of little more than 1 MB. Moreover, it is aspect oriented, a framework, container and a design pattern also known as dependency injection. (Saxena & al. 2016)

3 Development Plan

This chapter covers the background, planning and implementation of the project. For instance, how I dealt with technology after getting the requirements of this project. What knowledge was needed and how it was planned, how changes were implemented and how the challenges were faced.

3.1 Background

There was not any kind of template used for this project to develop or to put the code on top of some project and extend the development. Therefore, the project started from scratch with no knowledge in development of native android application. However, I had some experience of hybrid applications that I tested on android mobile device in past. Similarly, there was no readymade application or template that could be imported and developed on top of it. Therefore, commissioning party was required to play a vital role to guide about the project because the project started from scratch.

Starting a project from scratch takes time and require skills to accomplish it on time. However, a project with no skills require more effort, resources and hardworking to complete it on time. Therefore, I realized my situation and prepare myself to work hard to complete this project successfully. I was alone in this project with no knowledge of android development, but I was very optimistic and having positive feeling about the completion of project. Similarly, I had to focus on my situation and carefully implement the planning of this project.

3.2 Planning

Planning of this application took several stages such as initial planning after collecting requirements, development stage, testing stage, feedback from commissioning party and improvements. Development stage was to plan about what and how to develop the required feature and where to get the information sources to develop a feature. For instance, reading documentation, articles, watching tutorial video about the feature. Testing stage was to plan about testing the application on different android api levels, emulator and moving the physical device to get the

moving object etc. Moreover, there was not sufficient time to test the code separately through JUnit test or some test framework. Therefore, this project does not include code testing. Lastly, prepare notes to discuss before meeting the commissioning party and implement the suggestions.

The project was discussed with commissioning party to get to know more about requirements of the project. The initial meeting helped us to agree on final output of this project. This also helped us to set goals and timeframe to achieve the goals. The project time was limited enough to achieve the goals. Furthermore, the selection of platform, language and development environment was also discussed in first meeting. The selection was android platform, java language and android studio to develop a complete native android application. After the first meeting, it gave me a good idea to study about technologies that are required. Therefore, I decided to get knowledge to have better understanding of android development. I planned first week to develop several small applications to get the basic idea of application components and activity lifecycle in android.

3.2 Implementation

Project situation was always discussed with commissioning party to ensure that project is on right track and his feedback was helpful throughout the project. Moreover, Initial planning of this project helped me to implement the development method as to break the application in small units and after successful development and testing, merge small applications into actual project. This was also done after getting the requirements that I planned to divide the tasks and I started working on each task separately. I was successful to achieve the goal of current location of mobile devices. Similarly, achievement of small goals was helping me to make progress in the project. For this project, I was going outside very often and moving a lot to test the current location on map. This is how I was making sure that my one location feature is working.

After understanding the needs and requirements of this project, I started narrowing down the development of applications to google maps. After initial meetings with commissioning party, I decided to get familiar with android platform. For this sake, I took several online courses and created several small applications to understand

the development behavior in android. Furthermore, I was developing small applications on each topic like button for toast message, alert box, toolbar, static and custom text, implicit intents, explicit intents, showing the map in application for using them later in project when needed. Moreover, I started to get android specific knowledge from several free courses available in Udemy, YouTube video tutorials as well as reading articles. This method helped me to break my project into small applications and test them separately. For instance, I created an application showing map with static location. Similarly, I created another application showing my current location. If both are fulfilling the requirement of project, then I will merge them by adding the code of another successful application. Moreover, each small application was tested in multiple android mobile devices to understand the behavior of different api levels. Finally, I found this method is working fine for me.

During first meeting, it was also decided to meet commissioning party once in a week to demonstrate the development stage of the project, issues faced and get further guideline to plan the project. Therefore, I was discussing the issues and progress of project to commissioning party once in a week. The meeting was flexible but kind of compulsory even if I am progressing in project but still I was informing the commissioning party about the project. Moreover, the commissioning party was informed if I was implementing any changes in the project.

4 Empirical part

This chapter will discuss about the logical part of the project, results and difficulties faced during the project. The output of this project is a tracking application. Furthermore, the application serves its purpose by fulfilling the responsibility of providing information of driver's movement and Fuel cost. This information will make the taxi owner to feel comfortable about his cars and track them whenever owner wants in case of taxi business. This product can also be used for other purposes such as personal tracking or fuel cost saving. The unique point of this project is that it provides information about complete full stack development of location tracking application.

4.1 Required Technologies

The target of this project was to find out the location of driver and record the location in static and moving situation. Furthermore, drawing paths on the recorded location of user on maps in an application. The server side was done in Eclipse Java Enterprise Edition with spring boot. On the other hand, android studio was utilized for client side. MySQL database was for storing and retrieving the data locally. The server and database were serving in backend to get the data from application and store them into database. After that, retrieve the stored data from database through http request and draw the path by connecting all the points. Therefore, server had to serve the http requests for get and post all the points. The three important tools for this application were required and given below.

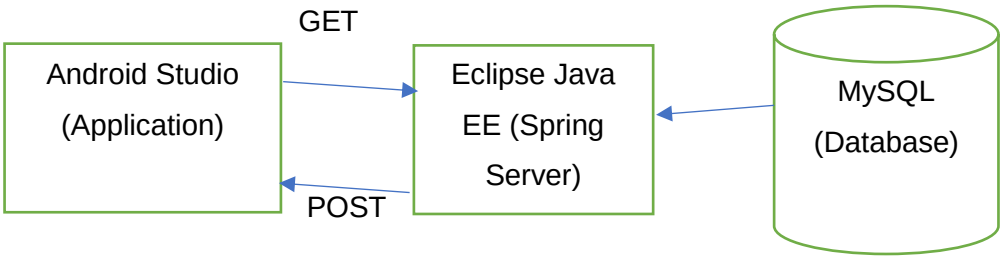


Figure 0-1 Main technologies required for developing application.

4.2 Backend for Project

Server was required for HTTP requests so Spring boot with java was utilized to develop the server as mentioned earlier. Server was handling all CRUD (create, read, update, delete) functionality. Two standard java classes were made for getters and setters. These two classes were Fuel.java and Location.java in server and client side. Controller class of server was also having routes for REST api so JSON data was available to access on localhost for testing. The main task was to deal with location and fuel because situation of taxi can be seen via web browser or in application. Therefore, location and fuel classes were made in addition to all required dependencies as well as other necessary repositories. Controller classes were serving the routes for crud functionality in both classes.

The server was tested, and the goal was successfully achieved by finding the server in running position. However, storing and retrieving the information of these two tables were one of the most important tasks of the project. Therefore, MySQL database was created and configured for storing the object values. It is the virtue of spring boot that dependency injection makes the system automated. Therefore, database was only created manually and table annotation in spring and dependency configuration created the table itself. Database was tested by sending new values and information retrieved was also deleted and updated with the help of POSTMAN http request handler tool.

```
@Entity
public class Fuel {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long fuelId;
    private String userperson;
    private int amount;
    private int litres;
    private String date;

    public Long getFuelId() {
        return fuelId;
    }
    public void setFuelId(Long fuelId) {
        this.fuelId = fuelId;
    }

    public String getUserperson() {
        return userperson;
    }

    public void setUserperson(String userperson) {
        this.userperson = userperson;
    }
    public int getAmount() {
        return amount;
    }
    public void setAmount(int amount) {
        this.amount = amount;
    }
    public int getLitres() {
        return litres;
    }
    public void setLitres(int litres) {
        this.litres = litres;
    }
    public String getDate() {
```

```
@Entity
public class Location {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long locationId;
    private String driverId;
    private double lat;
    private double langi;
    private String date;

    public Location() {
    }
    public Location(String driverId, double lat, double langi, String date) {
        super();
        this.driverId = driverId;
        this.lat = lat;
        this.langi = langi;
        this.date = date;
    }
    public Long getLocationId() {
        return locationId;
    }
    public void setLocationId(Long locationId) {
        this.locationId = locationId;
    }

    public String getDriverId() {
        return driverId;
    }
    public void setDriverId(String driverId) {
        this.driverId = driverId;
    }
    public double getLat() {
        return lat;
    }
    public void setLat(double lat) {
        this.lat = lat;
```

Figures 4-2.1 & 4-2.2 Fuel and Location server classes

4.3 Retrofit in Project

After creating the server, the next focus was user interface as users cannot see in backend. However, users of application only concern with the user interface. The only possible way to connect the application with server and database was http requests. Moreover, there are also several alternatives of making http requests such as Retrofit, Volley and traditional android HttpURLConnection etc. Commissioning party was informed about the selection of http request technology and he suggested to use Retrofit because it has type safety. However, my experience was tough in learning the retrofit.

After learning retrofit, all CRUD operations were tested on mobile devices in separate applications. For instance, I create an application for get request only to display the data. Another application was also developed to test with other api to understand the complications in retrofit mechanism. Same process for post request was repeated but this time was with local servers. This activity confirmed my skills to connect and manipulate the server and database with android device.

4.4 Devices used for project

The project was broken into several small parts and each part was focused very carefully. Several articles, technology docs and tutorials were reviewed about server, maps and database. On the completion of each task in client side or server side, application was tested in multiple actual mobile devices and known bugs were written in to-do list for improvement. For instance, retrofit technology was used in client side to make http requests to server and this request process was not working in earlier android api devices. It took time to figure out after testing on api 5.1.1 that server was fine. However, it is the behavior of android platform that does not allow some functionalities to work under certain api levels.

Two android mobile devices were dedicated for the project and was continuously used for testing the development. 1 android device was 5.1 Lollipop with api level 22 and the other one was Marshmallow 6.0 with api level 23. Two more devices with earlier api level were also used but they were not supported for the project so were resigned from the project because of older api level. Additionally, all the relevant components of android technology were also utilized during development such as intents, broadcast receiver, activities and service etc. In addition, toast and log was mainly used for quick debugging.

4.5 Google Maps Api

After selection and setup of backend and frontend, the most important task was to show the location path on map. Therefore, google maps api was utilized for this task. This is also the most complicated process as one of the previous researches also declared that making a location app with google location api services is a complicated process. Different api levels behave differently so task was continuously tested on api level 22 and api level 23 with every little progress. The nature of request permission in api level 23 requires differently than api level 22. In addition, there are also several ways to get the current location of the device in google maps api service like fusedLocationProviderClient is one way.

The development in client side was done in android studio so all the activities for different tasks with their respective layouts can be seen in the figure given below.

The image is after making the first release on google play store. Every android project usually has prefix “com.example” that does not let application to make a release on play store as google asks to change the prefix so I changed from com.example to com.sometry.

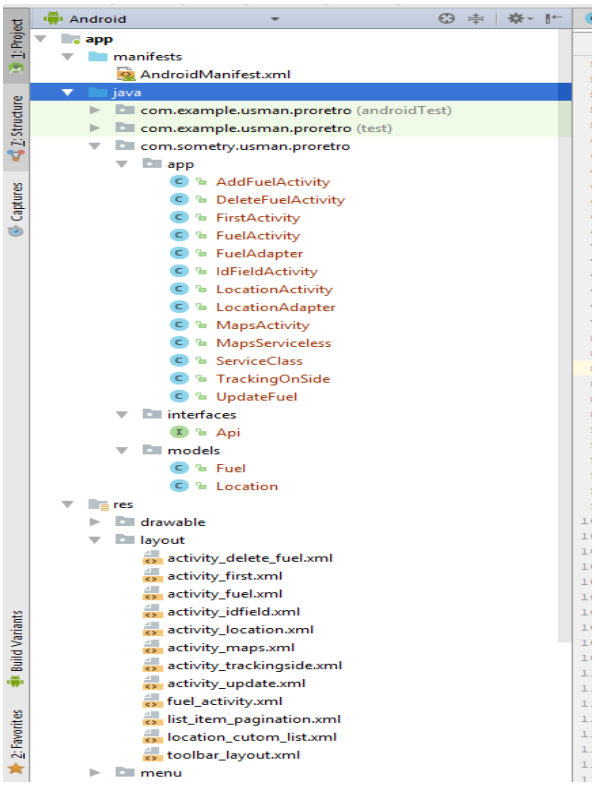


Figure 0-3 Project Structure in Android studio

4.6 Google Maps with http requests

The process of connecting google maps with server from mobile device can be discussed here. Planning of this step was to understand how google maps retrieve coordinates as well as retrofit get and post request. After that, a post request was implemented in client side with every new coordinate received from location updates. Google maps api gets the current coordinates with a short interval so every coordinate received by api was sent to the server and stored into database. These points will also be fetched on client side and can utilized to draw a path. Moreover, drawing path is possible with the help of polylines provided by google maps api services. The commissioning party was asked about the polylines and he suggested that coordinates should make the lines on all points received from database. The result of this task can be found below:

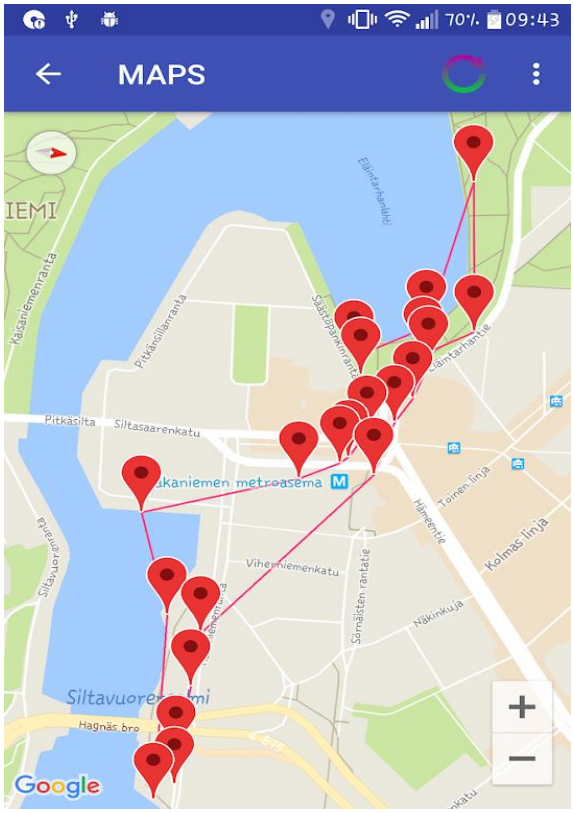


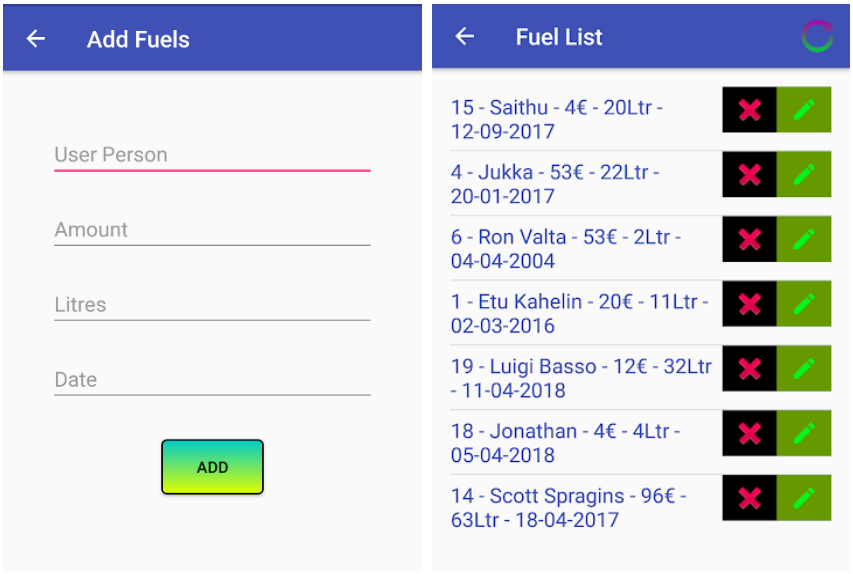
Figure 0-4 Location of user received from server

The above image shows the polylines on google maps where the user was moving. I also found out that location accuracy is also a big challenge in mapping applications. This is because it sometimes records the point close enough to actual location and sometimes the accurate current location of the user. We also found location related issues like accuracy and indoor navigation discussed by researchers in theoretical framework. The location is served in this app with the previous way of getting location. It works fine but google maps has introduced FusedLocationProviderClient method and that is recommended way. Additionally, there is a little difference in traditional way of getting location and FusedLocationProviderClient as simple way does not require to set GoogleApiClient method for location whereas it is required in FusedLocationProviderClient.

4.7 CRUD functionality for Fuel cost

The other side of application was to provide the options of CRUD functionality for fuel cost. Moreover, it was one of the requirements from commissioning party. Therefore, I created four EditText in activity layout in android studio and these four

fields were having standard java class in client and server side. Additionally, id of Fuel table was auto generated that was used in updating and deleting the entries.



Figures 0-5.1 & 4-5.2 Adding Fuels & Fuel list

The above images are representing fuels added by users with amount, quantity and date. The list can also be changed by updating and deleting and this functionality was added by adding 2 buttons appending each item of list. Fuel class was added in server and client-side with same variables and methods. Therefore, the fields of application were sending the values as a post request to the server and fuel class in server was using those values in retrieving and storing into database. In fuel list, these values are retrieved through retrofit get request and displayed by FuelAdapter class attached in appendix.

4.8 Development

We move back to the first step of application to define the maps application more clearly because we need to get access to google maps server as we need map in our application. Google maps have its own way to access the server. Therefore, it requires to have the google maps API key that is a mandatory requirement for working with maps in android mobile devices. Otherwise, it cannot show the maps in mobile device. Furthermore, every api request is recorded and can be seen in dashboard of google developer account. The given below screenshot is from the

dashboard of the project for this application and it shows that 117 times requests were made.

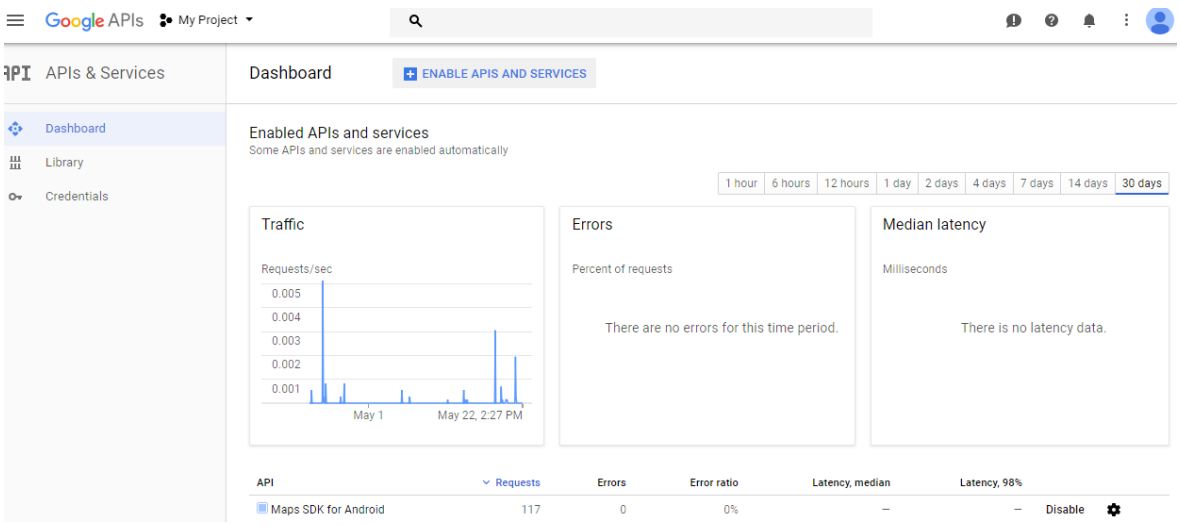
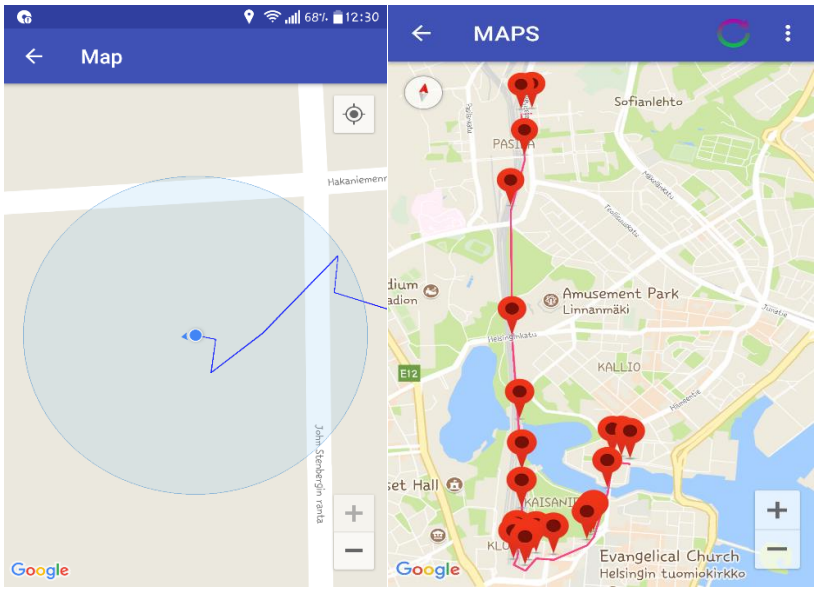


Figure 4-6 Google developer Api dashboard

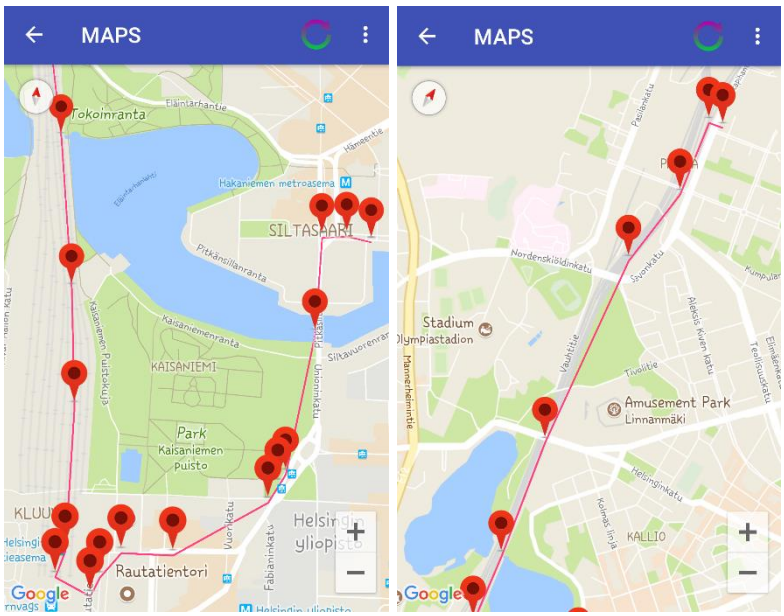
There were bunch of difficulties faced in dealing with android platform as api level specific technology is not easy to understand. Several classes and methods are deprecated in newer api levels in android platform. The way of getting location has also improved in api level 23 and onwards. Therefore, native development in android also requires understanding of platform as well as new ways of implementing technology. It was challenging to face this technology project even from beginning. However, dedication and continuous effort for this project succeed me in each step. Moreover, commissioning party also played a vital role in the whole project as he always motivated and encouraged me.

I understood the challenges of this project as I did not have any background knowledge in android development as well as http requests with android. Therefore, I continued with the approach of slow and steady development for better understanding and implementation. With this approach, I developed an application that takes my current location. After that, I took time to understand polylines and I added with my current location update. About polylines, it is easy to draw static lines but making them dynamic needs more understanding. I tested polylines with my updating location and I found it working so it was making a temporary route by following my location. Multiple screenshots are given below to explain the tracking

routes of this project. Left side of below screenshots is tracking the real time moving location without recording them. On the right side, markers with red color polylines is my route from home to school that is recorded and can be visible in future.



Figures 4-8.1 & 4-8.2 Temporary & Recorded route

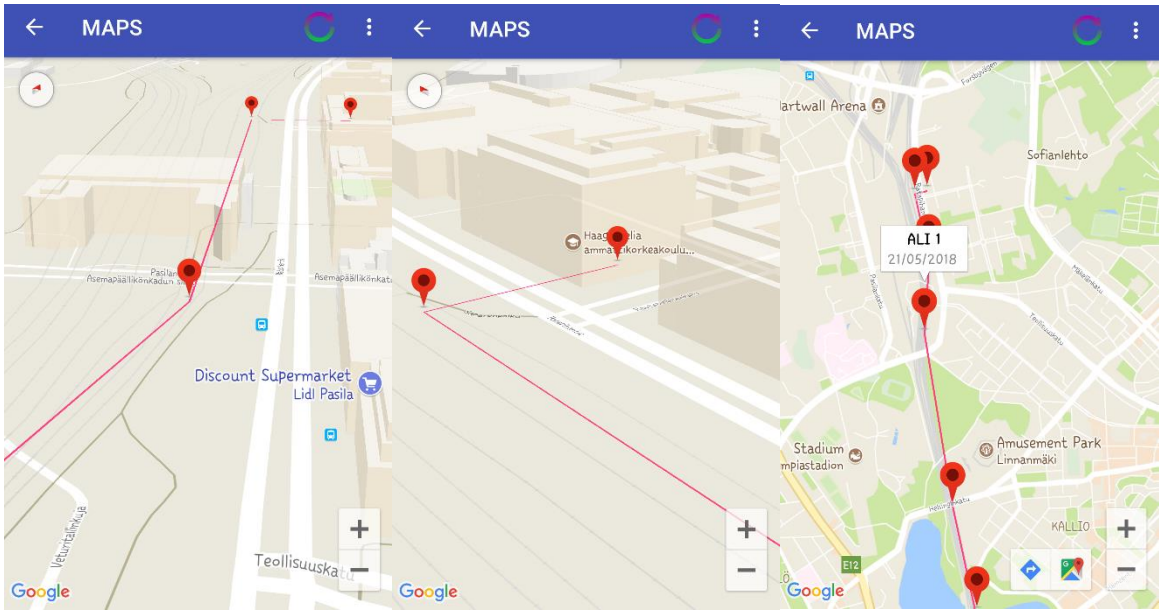


Figures 4-8.1 & 4-8.2 Analysis of Route

Further explanation of recorded location route is here. The above 2 figures show that I took the train at some point and left the train at certain point and turned to the

right. The location is updated after 30 seconds as code is given below with more screenshots.

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000*30, 10,
listener);
```



Figures 4-8.3, 4-8.4 & 4-8.5 showing the destination point and the user's detail information

The above figures show that I went to Haaga-Helia ammattikorkeakoulu.. that is right side of the train station. We can also analyze the information easily from the tracking route. To see who this person is moving, I have added names with hardcoded values in code. This person is Ali and day was actual date in Java as it was 21 May 2018. This location is stored in database and http request is fetching the data and showing on maps. In background, service component is used to send the location to the server. The same location is being retrieved to make path that reaches to the server. The above result is coming from calling these methods given below. The below method is coming when Yesterday item was chosen from ToolBar menu. (Appendix 6)

```
filterPolyLi("1",strDate,locLis,getResources().getColor(R.color.colorAccent),
"ALI");
filterPolyLi("2",strDate,locLis,getResources().getColor(R.color. colorAccent),
"Jukka");
filterPolyLi("",strDate,locLis,getResources().getColor(R.color. colorAccent),
"ALI");
```

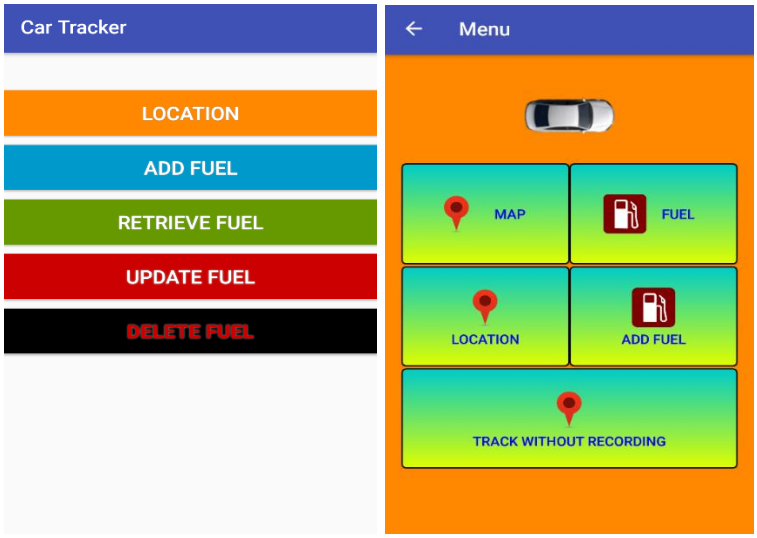
The above code is getting the id of 1 that is hardcoded, date variable that has the current date -1, list of coordinates, color and hardcoded name. I also have added the first activity where several buttons were used for different activities serving different purposes as a menu of this application. For instance, the code after this paragraph explains that each button in layout file directs to another activity by using android intent component. This class is also using broadcast receiver and retrofit library to send the coordinates back to the server. Retrofit takes care of sync and async process in request as call.enqueue is async call. On the other hand, there is call.execute method that is sync call. In the below code, I am using call.enqueue method for async call that is also recommended.

```
public class FirstActivity extends AppCompatActivity {

    private BroadcastReceiver broadcastReceiver;
    Toolbar toolbar;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_first);

Button addFu = (Button) findViewById(R.id.btnAddFuel);
Button fuelS = (Button) findViewById(R.id.fuel);
Button locat = (Button)findViewById(R.id.btLoc);
Button maps = (Button)findViewById(R.id.btMaps);
Button locates = (Button) findViewById(R.id.btnLocate);


toolbar = (Toolbar)findViewById(R.id.refresh);
getSupportActionBar().setTitle("Menu");
getSupportActionBar().setDisplayHomeAsUpEnabled(true);


fuelS.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
startActivity(new Intent(FirstActivity.this, FuelActivity.class));
            }
        });

addFu.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
startActivity(new Intent(FirstActivity.this, AddFuelActivity.class));
            }
        });

locat.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
startActivity(new Intent(FirstActivity.this, LocationActivity.class));
            }
        });

maps.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
startActivity(new Intent(FirstActivity.this, MapsActivity.class));
```

```
            }
        });

locates.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
startActivity(new Intent(FirstActivity.this, MapsServiceless.class));
            }
        });
Intent i = new Intent(getApplicationContext(), ServiceClass.class);
Toast.makeText(this, "Location Service Started", Toast.LENGTH_SHORT).show();
        startService(i);
}

private boolean runtime_permissions() {
    if (Build.VERSION.SDK_INT >= 23 && ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED)
{

  requestPermissions(new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
Manifest.permission.ACCESS_COARSE_LOCATION}, 100);

   return true;
        }
        return false;
    }

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
super.onRequestPermissionsResult(requestCode, permissions, grantResults);
     if (requestCode == 100) {
       if (grantResults[0] == PackageManager.PERMISSION_GRANTED &&
grantResults[1] == PackageManager.PERMISSION_GRANTED) {
   Toast.makeText(this, "Welcome", Toast.LENGTH_SHORT).show();
  } else {
        runtime_permissions();
            }
        }
    }

@Override
protected void onResume() {
        super.onResume();
     if (broadcastReceiver == null) {
       broadcastReceiver = new BroadcastReceiver() {
@Override
public void onReceive(Context context, Intent intent) {

double lat = (double) intent.getExtras().get("lat");
double langi = (double) intent.getExtras().get("lang");

Calendar calendar = Calendar.getInstance();

SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy ");
String strDate = sdf.format(calendar.getTime());
String driveId = (String) getIntent().getExtras().get("idMap");
Location loca = new Location(driveId, lat, langi, strDate);
sendNetworkRequest(loca);
                }
            };
        }
  registerReceiver(broadcastReceiver, new IntentFilter("location_update"));
    }

@Override
protected void onDestroy() {
```

```
super.onDestroy();
  if (broadcastReceiver != null) {
    unregisterReceiver(broadcastReceiver);
}
Toast.makeText(this, "Location Service Stopped", Toast.LENGTH_SHORT).show();
  }

private void sendNetworkRequest(Location location) {

Retrofit retrofit = new Retrofit.Builder().baseUrl("https://fuel-
hero.herokuapp.com/").addConverterFactory(GsonConverterFactory.create())
.build();

Api api = retrofit.create(Api.class);
Call<List<Location>> call = api.addLocations(location);
call.enqueue(new Callback<List<Location>>() {
@Override
public void onResponse(Call<List<Location>> call, Response<List<Location>>
response) {
  Toast.makeText(FirstActivity.this, "sendingLoc in db",
Toast.LENGTH_SHORT).show();
            }
@Override
public void onFailure(Call<List<Location>> call, Throwable t) {
 Toast.makeText(FirstActivity.this, "error :(", Toast.LENGTH_SHORT).show();
            }
        });

    }
```

There is also a screenshot of user interface of this first activity as below. All the buttons lead to other activities developed to achieve the goal of this application. I faced difficulties in designing the interface of this menu as previous version of this application was having just horizontal buttons. After trying several designs, I ended up with the one on right side of below screenshots containing big buttons with custom background of multiple gradient colours. The commissioning party was pleased with the design of this activity and overall functionality of application because he also tested the application on his mobile device.



Figures 4-7.1 & 4-7.2 Previous version & Actual Menu of Application

4.9 Further Work:

The server was deployed to Heroku for remote access after successfully tested on localhost. Similarly, database was shifted to PostgreSQL for smooth working on Heroku. Therefore, there is a lot of improvements required as the server is on Heroku free repository that has several limitations. For instance, it takes time to fetch and post requests as well as memory limitation. Therefore, the server should be hosted with sufficient memory and secure hosting in future. Account creation should be added as well as users cannot access to other members' tracking data without their permission.

Moreover, there can be several interesting features added to the application. For instance, group service can be created so admin can add members to track their location and communication. Furthermore, geo fencing for creating a boundary if the car is going out of limited region then notification will pop up to other member users. In addition, if the location remains the same for an hour then notification will be appeared on screen of admin or owner that will help the taxi owner to know if there is no client or taxi has some problem. There is also the possibility of creating fuel saved by each car separately. For Instance, there should be car id instead of driver name in fuel class. Moreover, each car should have separate fuel list and the kilometers drive by a driver. Above all, these points are possible to implement if there is availability of resources and time. Furthermore, commissioning party also agreed on most of these suggested improvements.

5 Evaluation and Conclusions

This chapter describes the evaluation of results, overall development, difficulties faced, personal learning and satisfaction from the project. The results are positive as they achieved the target as well as application is functional and available in play store. Commissioning party was also pleased with results and overall development. A few goals were not fulfilled as the changes came during the project so they were out of scope after changes. However, application is functioning properly and tested in several devices.

Planning the project correctly and using project management skills is a key to make the project successful most of the times. In my case, project was planned wisely because I divided into several small parts as well as time management was nicely taken into consideration. Therefore, it led to the success of the project as project was ready on time. Additionally, project management skills were also used and improved during the project. In the beginning, I analyzed my actual skills and skills required for this project then I found that I need more time than the available time for this project. Therefore, I worked every day even on weekends with no break.

Overall, I am satisfied because I learned a lot during this application development. I can explain my learning with this fact that I took a week to implement retrofit library in the project earlier because I was unable to understand the simple concept of this library. However, now I take 15 minutes approximately to implement any of the http requests with retrofit. It was similar experience with server development but trying again and again made me confident now. Currently, I am comfortable in implementing any of the technologies that I used during the development of this project.

I started my journey for this project with several small android applications to understand that how applications are developed in android platform. After that, I was adding the correct solution in my project. This small development helped me to learn several other features that I did not even use in this project but they are in my skillset now. I also believe after this project that finding the specific topic or problem is also an art. I improved my skills to search more specific to the problem facing by me in development. Moreover, I improved my skills in creating the server in spring boot,

relationship of tables, databases, Java language, http requests, use of retrofit library, GitHub and gradle based project. Improvements in these skills made me confident in real life to take more android projects and work on them.

Android platform has itself several challenges that I found during android development. The android studio is a marvelous tool for developing android application but the drawback is that it takes a lot of space on disk. Another factor is multiple language can be used in development. For instance, Kotlin is official development language of android now. Although, Java still works but conflict between oracle and Google start closing the doors for development in java for new upcoming mobile devices. Moreover, testing the application on physical device or simulator during development takes a lot of times. Moreover, one drawback of android is that the skills learned for developing android application cannot be used for other platforms like iOS or web. Researchers also have discussed about hybrid apps that can be developed with the web technologies. Moreover, React, Angular and several other frameworks provide platforms to use the same knowledge of web to build the real native mobile apps.

Google maps api also has a major problem that api level of mobile device let the application behaves differently. For example, several older api levels are almost obsolete for development and testing currently. Therefore, app needs to be tested in other api levels on emulator as well as real mobile devices to confirm the quality. There are also several ways of doing the same thing in development and old methods of developing applications are deprecated. The developer must have to be in touch with the docs and new releases in android technology.

References

Saxena, A., Kaushik, N. & Kaushik, N., 2016. Implementing and Analyzing Big Data Techniques withSpring Frame Work in Java& J2EEBased Application, ICTCS '16, Udaipur, India

Yue, Y., Zhang, K. & Jacobsen, H., 2013, Smart Phone Application for Connected Vehicles and Smart Transportation, ACM, Beijing, China

Xanthopoulos, S. & Xinogalos, S., 2016. A Review on Location Based Services for Mobile Games, PCI '16, Patras, Greece

Podiyan, P., Butakov, S., & Zavarsky, P., 2015. POSTER: Study of compliance of Android location APIs with Geopriv, WiSec'15, New York, NY, USA

Sarma, B., Li, N., Gates, C., Potharaju, R., & Nita-Rotaru, C., 2012. Android Permissions: A Perspective Combining Risks and Benefits, Newark, New Jersey, USA.

Li, D. & Halfond, W., 2015. Optimizing Energy of HTTP Requests in Android Applications. DeMobile'15, Bergamo, Italy

Ali, U., 2018. GitHub.AndroidAppCarTracking, 18 March 2018 URL: https://github.com/usmanali598/AndroidAppCarTracking retrieved 29 March 2016

Pöhls, M. & Peitek, N 2017. Retrofit: Love Working with APIs on Android. Leanpub, Victoria, British Columbia, Canada

Google Maps Android, 2018. Project Configuration, https://developers.google.com/maps/documentation/android-sdk/config, Accessed: 21 May 2018

Google Maps Android 2018. Polylines and Polygons to Represent Routes and Areas, https://developers.google.com/maps/documentation/android-sdk/polygon-tutorial, Accessed: 21 May 2018

Android Developers, Documentation 2018. Optimize location for battery https://developer.android.com/guide/topics/location/battery, Accessed: 22 May 2018

Android Developers, Documentation 2018. Get the last known location https://developer.android.com/training/location/retrieve-current, Accessed: 22 May 2018

APPENDICES

Appendix 1- Maps Activity for showing the tracking routes on maps

```
public class MapsActivity extends AppCompatActivity implements OnMapReadyCallback
{

    private GoogleMap mMap;
    Toolbar toolbar;
    private Polyline route = null;
    private PolylineOptions routeOpts = null;
    private boolean drawTrack = true;
    long locationId;
    String driverId, date;
    double lat, lng;

@Override
protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_maps);

SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
mapFragment.getMapAsync(this);

toolbar = (Toolbar)findViewById(R.id.refresh);
getSupportActionBar().setTitle("MAPS");
String respo = getIntent().getStringExtra("respo");
    }

@Override
public void onMapReady(GoogleMap googleMap) {
  mMap = googleMap;
  todayLines();

  mMap.getCameraPosition();
    }

@Override
public boolean onCreateOptionsMenu(Menu menu) {
  MenuInflater inflater = getMenuInflater();
  inflater.inflate(R.menu.menu_overall, menu);
  return true;
    }
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()){
    case R.id.refresh:
    Toast.makeText(this, "Refreshed :)", Toast.LENGTH_SHORT).show();
    finish();
    startActivity(getIntent());
    break;
         case R.id.action_fuel:
         startActivity(new Intent(this, FuelActivity.class));
         break;
    case R.id.action_today:
         finish();
          startActivity(getIntent());
          todayLines();
          break;
    case R.id.action_yesterday:
          yesterdayLines();
          break;
    case android.R.id.home:
          NavUtils.navigateUpFromSameTask(this);
          return true;
```

```
        }
        return true;
    }

public void filterPolyLi(String id, String date, List<Location> locLis, int colr,
String name){

 List<LatLng> latlngs = new ArrayList<>();

 for(int i=0; i<locLis.size(); i++){
        if ((locLis.get(i).getDriverId().equalsIgnoreCase(id)) &&
(locLis.get(i).getDate().equalsIgnoreCase(date)) ) {
latlngs.add(new LatLng(locLis.get(i).getLat(), locLis.get(i).getLangi()));
mMap.addMarker(new MarkerOptions().position(new LatLng(locLis.get(i).getLat(),
locLis.get(i).getLangi())).title(name+" "+id).snippet(""+date)

.icon(BitmapDescriptorFactory.fromResource(R.mipmap.redmark))
    );
                }
            }
  PolylineOptions rectOptions = new PolylineOptions().addAll(latlngs);
  rectOptions.color(colr).width(5).geodesic(true);
  mMap.addPolyline(rectOptions);
        }
public void todayLines(){
Retrofit retrofit = new Retrofit.Builder().baseUrl("https://fuel-
hero.herokuapp.com/").addConverterFactory(GsonConverterFactory.create())
.build();

Api apiService = retrofit.create(Api.class);
Call<List<Location>> call = apiService.getLocations();
call.enqueue(new Callback<List<Location>>() {
@Override
public void onResponse(Call<List<Location>> call, Response<List<Location>>
response) {
    List<Location> locLis = response.body();
    Toast.makeText(MapsActivity.this, ""+locLis.size(),
Toast.LENGTH_SHORT).show();

 if (locLis.size() > 0) {
 LatLng Helsini = new LatLng(locLis.get(1).getLat(), locLis.get(1).getLangi());

mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(Helsini, 15));
  } else {
 LatLng Helsini = new LatLng(60.172503, 24.939974);

mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(Helsini, 15));
                    }
Calendar calendar = Calendar.getInstance();
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy ");
String strDate = sdf.format(calendar.getTime());

//Yesterday Date
calendar.add(Calendar.DATE, -1);
String yesterday = sdf.format(calendar.getTime());

filterPolyLi("1",  strDate, locLis,
getResources().getColor(R.color.colorPrimaryDark), "ALI");
filterPolyLi("2",  strDate, locLis,
getResources().getColor(R.color.colorPrimaryDark), "Jukka");
filterPolyLi("",  strDate,
locLis,getResources().getColor(R.color.colorPrimaryDark), "ALI");
}

@Override
public void onFailure(Call<List<Location>> call, Throwable t) {
Toast.makeText(MapsActivity.this, "Wait for short moment and refresh",
Toast.LENGTH_SHORT).show();
                }
```

```
            });
        }
public void yesterdayLines(){

Retrofit retrofit = new Retrofit.Builder().baseUrl("https://fuel-
hero.herokuapp.com/").addConverterFactory(GsonConverterFactory.create())
.build();

Api apiService = retrofit.create(Api.class);
Call<List<Location>> call = apiService.getLocations();
call.enqueue(new Callback<List<Location>>() {
@Override
public void onResponse(Call<List<Location>> call, Response<List<Location>>
response) {

List<Location> locLis = response.body();
Toast.makeText(MapsActivity.this, ""+locLis.size(), Toast.LENGTH_SHORT).show();

if (locLis.size() > 0) {
   LatLng Helsini = new LatLng(locLis.get(1).getLat(), locLis.get(1).getLangi());

mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(Helsini, 15));
    } else {
LatLng Helsini = new LatLng(60.172503, 24.939974);

mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(Helsini, 15));
    }
Calendar calendar = Calendar.getInstance();
SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy ");
String strDate = sdf.format(calendar.getTime());
//Yesterday Date
calendar.add(Calendar.DATE, -1);
String yesterday = sdf.format(calendar.getTime());
Toast.makeText(MapsActivity.this, "Yesterday Was "+yesterday,
Toast.LENGTH_SHORT).show();

filterPolyLi("1", yesterday,locLis, getResources().getColor(R.color.colorAccent),
"ALI");
filterPolyLi("2", yesterday,locLis, getResources().getColor(R.color.colorAccent),
"Jukka");
filterPolyLi("",  yesterday, locLis,getResources().getColor(R.color.colorAccent),
"ALI");

  }
@Override
public void onFailure(Call<List<Location>> call, Throwable t) {
Toast.makeText(MapsActivity.this, "Wait for short moment and refresh",
Toast.LENGTH_SHORT).show();
                }
            });
        }
}
```

Appendix 2 Api Interface with Http Request methods

```
public interface Api {

    @GET("fuels")
    Call<List<Fuel>> getFuels();

    @GET("locations")
    Call<List<Location>> getLocations();

    @POST("fuels")
    Call<List<Fuel>> addFuels(@Body Fuel fuel);
```

```
    @POST("locations")
    Call<List<Location>> addLocations(@Body Location location);

    @PUT("fuels/{id}")
    Call<Fuel> editFuels(@Path("id")long id, @Body Fuel fuel);

    @DELETE("fuels/{id}")
    Call<Fuel> deleteFuels(@Path("id")long id);
}
```

Appendix 3 Fuel Activity for fetching fuels from server

```
public class FuelActivity extends AppCompatActivity {

Toolbar toolbar;
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.fuel_activity);

Intent in = getIntent();

toolbar = (Toolbar)findViewById(R.id.refresh);
getSupportActionBar().setTitle("Fuel List");

final ListView listView = (ListView) findViewById(R.id.fuel_list);

Retrofit retrofit = new Retrofit.Builder().baseUrl("https://fuel-
hero.herokuapp.com/")
.addConverterFactory(GsonConverterFactory.create()).build();

 Api apiService = retrofit.create(Api.class);

 Call<List<Fuel>> call = apiService.getFuels();

 call.enqueue(new Callback<List<Fuel>>() {
@Override
public void onResponse(Call<List<Fuel>> call, Response<List<Fuel>> response) {
 List<Fuel> fulLis = response.body();
 listView.setAdapter(new FuelAdapter(FuelActivity.this, fulLis));
}
@Override
public void onFailure(Call<List<Fuel>> call, Throwable t) {
Toast.makeText(FuelActivity.this, "error :( "+t.getMessage(),
Toast.LENGTH_SHORT).show();
            }
        });
    }

@Override
public boolean onCreateOptionsMenu(Menu menu) {
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.get_menu, menu);
return true;
    }
@Override
public boolean onOptionsItemSelected(MenuItem item) {

String title = (String)item.getTitle();

switch (item.getItemId()){
case R.id.refresh:
Toast.makeText(this, "Refreshed :)", Toast.LENGTH_SHORT).show();
   finish();
```

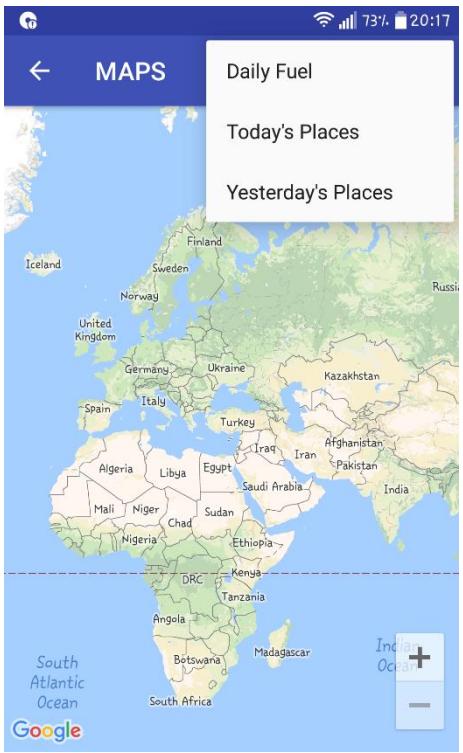
```
    startActivity(getIntent());
    break;
    case android.R.id.home:
    NavUtils.navigateUpFromSameTask(this);
    return true;

      }
         return true;
     }
}
```

Appendix 4 Fuel Adapter for Retrofit request

```
public class FuelAdapter extends ArrayAdapter<Fuel> {

private Context context;
private List<Fuel> values;

public FuelAdapter(@NonNull Context context, List<Fuel> values) {
  super(context, R.layout.list_item_pagination, values);

this.context = context;
this.values = values;
    }

@Override
public View getView(final int position, final View convertView, ViewGroup parent)
{
 View row = convertView;

if (row == null) {
 LayoutInflater inflater =
 (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
 row = inflater.inflate(R.layout.list_item_pagination, parent, false);
   }

final TextView textView = (TextView)
row.findViewById(R.id.list_item_pagination_text);
ImageButton list_butt = (ImageButton) row.findViewById(R.id.editing);
ImageButton delete_butt = (ImageButton) row.findViewById(R.id.deleting);


final Fuel item = values.get(position);

final String message = item.getFuelId() +" - "+item.getUserperson()+" -
"+item.getAmount()+"€"+" - "+item.getLitres()+"Ltr"+" - "+item.getDate();
textView.setText(message);
textView.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
Toast.makeText(context, ""+ message, Toast.LENGTH_SHORT).show();

  }
    });

delete_butt.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
Fuel fuel = new Fuel(item.getFuelId(), item.getUserperson(), item.getAmount(),
item.getLitres(), item.getDate());
Intent intenti = new Intent(getContext(), DeleteFuelActivity.class);
intenti.putExtra("fuelId", item.getFuelId());
context.startActivity(intenti);
     }
 });
```

```
list_butt.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {

Fuel fuel = new Fuel(item.getFuelId(), item.getUserperson(), item.getAmount(),
item.getLitres(), item.getDate());
Toast.makeText(context, ""+fuel.getFuelId()+" / "+fuel.getUserperson()+" /
"+fuel.getAmount()+" / "+fuel.getLitres()+" / "+fuel.getDate(),
Toast.LENGTH_SHORT).show();

Intent intenti = new Intent(getContext(), UpdateFuel.class);
intenti.putExtra("fuelId", item.getFuelId());
intenti.putExtra("userPerson", item.getUserperson());
intenti.putExtra("amount", item.getAmount());
intenti.putExtra("litres", item.getLitres());
intenti.putExtra("date", item.getDate());

context.startActivity(intenti);

  }
     });

 return row;
    }

}
```

Appendix 6 Action Bar Menu for Yesterday location routes



Appendix 7 Server-side Fuel controller class

```java
@RestController
@CrossOrigin(origins = "http://localhost:8000")
@RequestMapping(path = "/fuels")
public class FuelController {

        @Autowired
        private FuelRepository frepository;

        @GetMapping
        public Iterable<Fuel> findAll() {
            return frepository.findAll();
        }

        @GetMapping(path = "/{fuelId}")
        public Fuel find(@PathVariable("fuelId") Long fuelId) {
            return frepository.findOne(fuelId);
        }

        @PostMapping(consumes = "application/json")
        public Fuel create(@RequestBody Fuel fuel) {
            return frepository.save(fuel);
        }

        @DeleteMapping(path = "/{fuelId}")
        public void delete(@PathVariable("fuelId") Long fuelId) {
            frepository.delete(fuelId);
        }

        @PutMapping(path = "/{fuelId}")
        public Fuel update(@PathVariable("fuelId") Long fuela, @RequestBody Fuel fuel) throws BadHttpRequest {
            if (frepository.exists(fuela)) {
                fuel.setFuelId(fuela);
                return frepository.save(fuel);
            } else {
                throw new BadHttpRequest();
            }
        }
```

Appendix 8 Location Api record from server