

**VIETNAM NATIONAL UNIVERSITY
HO CHI MINH CITY
INTERNATIONAL UNIVERSITY**



**SCHOOL OF
COMPUTER SCIENCE AND ENGINEERING**

THESIS REPORT

MOBILE APPLICATION FOR TAXI SERVICE

Advisor: Dr. Nguyen Van Sinh

By

**Trinh Ngoc The Anh
April, 2017**

MOBILE APPLICATION FOR TAXI SERVICE

APPROVED BY: ADVISOR

Dr. Nguyen Van Sinh

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deep and sincere gratitude to my advisor Dr. Nguyen Van Sinh for the continuous support of my study and research, for his patience, motivation, enthusiasm, and immense knowledge in all academic years. I appreciate all his contributions of time, ideas, and valuable instructions to make my experience productive and encouraging.

I would like to acknowledge EdunetJSC for give me opportunity to implement Android and learn knowledge concern to my project.

I would like to specially thank my family for their love and encouragement throughout my undergraduate study. My thankfulness would also send to my friends who have been studying in the university for their sharing to overcome all obstructs in the study-time.

Besides, I would like to thank everyone who supported and helped me to complete research work directly or indirectly.

Last but not least, I would like to acknowledge the academic and technical supports of the International University, Vietnam National University, Ho Chi Minh City. I also thank the Department of Computer Science and Engineering for the support and assistance during my thesis study.

ABSTRACT

Convenience of smartphones can be seen in lately years, beside with taking a call, they can help people to do a lot of things as role as music player, camera, and a map. Development of smartphone promoted the release of many mobile applications serve people in almost of services in life include taxi service. With increasing travel demand of people, taxi service become more important when people want to move more easily without using their own vehicles, so what they need is something that can control their ride as many as possible. To manipulate that convenience of smartphone, in this thesis, I would like to research and build a mobile application that can support users in transportation. Therefore, I will combine map function of smartphone and taxi service to implement a mobile app called “Mobile Application for Taxi Service” (referred to MATS) which can help passengers and drivers manage their taxi-booking and tasks directly on their smartphones.

Table of Contents

ACKNOWLEDGEMENTS	i
ABSTRACT.....	ii
Chapter 1 . Introduction	1
1.1 Overview	1
1.2 Problem Statement.....	1
1.3 Scope and Objectives.....	1
1.4 Solution.....	2
1.5 Structure of Thesis	3
Chapter 2 . Literature Review	5
2.1 Mobile Application.....	5
2.2 Android Platform.....	6
2.3 Google Maps.....	7
2.4 Google Maps API.....	7
2.4.1 Google Maps API for Android	7
2.4.2 Google Maps Web Service APIs.....	7
2.5 Existing Taxi Service Mobile Applications.....	9
2.5.1 Uber.....	9
2.5.2 Grab	10
2.6 MySQL	11
Chapter 3 . Methodology.....	12
3.1 Android Application	12
3.2 System Architecture	12
3.3 Use Case Diagram	13
3.4 Activity Diagram	17
3.5 Database Structure	18
3.6 JDBC	20
3.7 Google Direction API.....	21
3.8 Finding out the Nearest Taxi	21
3.9 Taxi Animation.....	21
3.10 Class Diagram	23
3.10.1 Passenger App Class Diagram	23
3.10.2 Driver App Class Diagram	29
3.11 State Diagram.....	33
3.11.1 Load map and identify user position process:	33
3.11.2 Display all taxis around process:	34
.....	34
3.11.3 Choose destination and display estimate values process:	34
3.11.4 Call a taxi process:	36
3.11.5 MATS Driver process.....	37
Chapter 4 . Implementation.....	39
4.1 Login and Registration Activity	39
4.2 Main Activity.....	40
4.3 Driver Application.....	47

Chapter 5	. Conclusion and Future Work.....	52
5.1	Conclusion	52
5.2	Future Work.....	52
References:	53

Table of Figures

Figure 1 - Thesis Structure.....	3
Figure 2 - Market Share of Mobile Applications [4]	6
Figure 3 - A Part from Response JSON [10]	8
Figure 4 - MATS System Architecture	12
Figure 5 - API Key to using Google Maps Service	13
Figure 6 - Use Case Diagram of MATS	13
Figure 7 - Activity Diagram of the Project	17
Figure 8 - Entity-Relationship Diagram	18
Figure 9 - Database Structure	19
Figure 10 - Sequence Diagram for Calling a Taxi Services	22
Figure 11 - Passenger, TaxiDriver and Trip Classes	23
Figure 12 - DataConnection, DriverDataAccess, PassengerDataAccess and TripAccess Classes	24
Figure 13 - MapsActivity Class Diagram	25
Figure 14 - SignupActivity Class Diagram.....	27
Figure 15 – LoginActivity Class Diagram.....	27
Figure 16 - History Activity Class Diagram	28
Figure 17 - HistoryDetailActivity Class Diagram	28
Figure 18 - SettingActivity Class Diagram.....	28
Figure 19 – Driver App Passenger and TaxiDriver Classes	29
Figure 20 – Driver App DataConnection and DriverDataAccess Classes.....	29
Figure 21 – Driver App MapsActivity Class Diagram	30
Figure 22 - Driver App SignupActivity Class Diagram	31
Figure 23 - Driver App Login Activity.....	31
Figure 24 - Driver App Setting Activity Class Diagram	32
Figure 25 - Load Map and Identify User Position Process Diagram	33
Figure 26 - Display All Nearby Taxis Process Diagram	34
Figure 27 - Choose Destination and Display Estimate Values Process Diagram.....	34
Figure 28 - An Example of JSON Data Returned by Google [17]	35
Figure 29 - Call a Taxi Process Diagram.....	36
Figure 30 - MATS Driver Process Diagram.....	37
Figure 31 - Registration Activity	39
Figure 32 - Login Activity	39
Figure 33 - Edit Account.....	40
Figure 34 - Phone Location Opening Request	40
Figure 35 - Main Layout	41
Figure 36 - Choose the Destination	41
Figure 37 - List of Locations	42
Figure 38 - Ride Estimation.....	42
Figure 39 - Ride Pick-up Place	43
Figure 40 - Uber Fare Calculation	43
Figure 41 - Calling Taxi Activity.....	44
Figure 42 - Taxi Driver Information.....	44
Figure 43 - Cancel a Trip	45
Figure 44 - Start Trip Notification	45
Figure 45 - Taxi Driver Rating When the Trip Finish	46
Figure 46 - Trip History	46
Figure 47 - Trip History Detail	47
Figure 48 - Notification to Open GPS on MATS Driver.....	47

Figure 49 - Login for Driver App	48
Figure 50 - Registration for Driver App	48
Figure 51 - Driver's Position on Main Layout of MATS Driver	49
Figure 52 - Edit Account for Driver App.....	49
Figure 53 - New Taxi Call Notification for Driver.....	50
Figure 54 - Passenger's Position on MATS Driver.....	50
Figure 55 - Passenger Detail Information on MATS Driver	51
Figure 56 - Driver Can Make a Phone Call to Passenger	51

List of Tables

Table 1 - Three Basic Types of Mobile Applications [2]	5
Table 2 - Use Case 1: Register.....	14
Table 3 - Use Case 2: Login	14
Table 4 - Use Case 3: Edit Account.....	14
Table 5 - Use Case 4: Call Taxi	15
Table 6 - Use Case 5: Cancel Call	15
Table 7 - Use Case 6: View trip History.....	16
Table 8 - Use Case 7: Receive Taxi Call	16
Table 9 - Use Case 8: Call Passenger	16

Chapter 1 . Introduction

1.1 Overview

In the evening in 2008, Travis Kalanick and Garrett Camp had problem to call a taxi. So they imagined a simple idea and a new first taxi service using through smartphones named Uber was born rely on idea tap a button, get a ride [1]. By using taxi service through mobile application, everyone including passengers and drivers who join in this service can have more convenience, passengers can control there ride and drivers can employ by themselves.

With the development of smartphone, this taxi service application have more opportunities to become popular. The application is built relying on Google APIs, with accuracy of Google satellites, users feel more comfortable when they use this app. Therefore, mobile application for taxi service is a technical method to give people assurance when they use taxi service, to support drivers has a flexible new way to earn money and to help the city develop local economies, improve transportation, and make streets safer.

1.2 Problem Statement

When using a taxi service, there are main two situations that could make passengers afraid. The first situation is paid estimation. With a normal taxi service, passenger only know about price of their ride after they arriving the destination, they do not know how much money they have to spend when they hail a taxi, and sometimes the price could be so high. The second situation is coming from dishonest drivers. Especially, drivers from unreliable taxi service, instead of choosing the shortest way, to get a higher price for a trip, these drivers will choose passengers come from another place and take them on a very long road because passengers can not control their ride.

1.3 Scope and Objectives

This thesis concentrate on booking taxi for passengers, help passengers to control whole their trip from pick-up place to destination, view all taxis around pick-up position, view all information by using smartphones based on an Android application.

The objective of the thesis is creating an application that can help passenger to call taxi as convenient as possible. The application must be easy to use, every functions must show clearly for passenger and the interface must help user know which function is showing on screen and where they can open a function they want in a very simple way they can visualize.

1.4 Solution

In order to solve two above problems concern to taxis, passengers should use taxi service based on mobile application. By using Google Maps API, the application can calculate distance between starting point and end point, price and show them for passengers. The Google Direction API could show the nearest way for passengers to know which way is the best choice. Besides that, the application also has more other functions such as time estimation, show taxi drivers' information and mange passenger's account. With this taxi service app, passengers can be confident because they totally control their trip. With some reasons as mentioned, the purpose of this thesis is trying to rebuild a Mobile Application for Taxi Service (referred to MATS) with almost necessary features supporting to taxi passengers.

1.5 Structure of Thesis

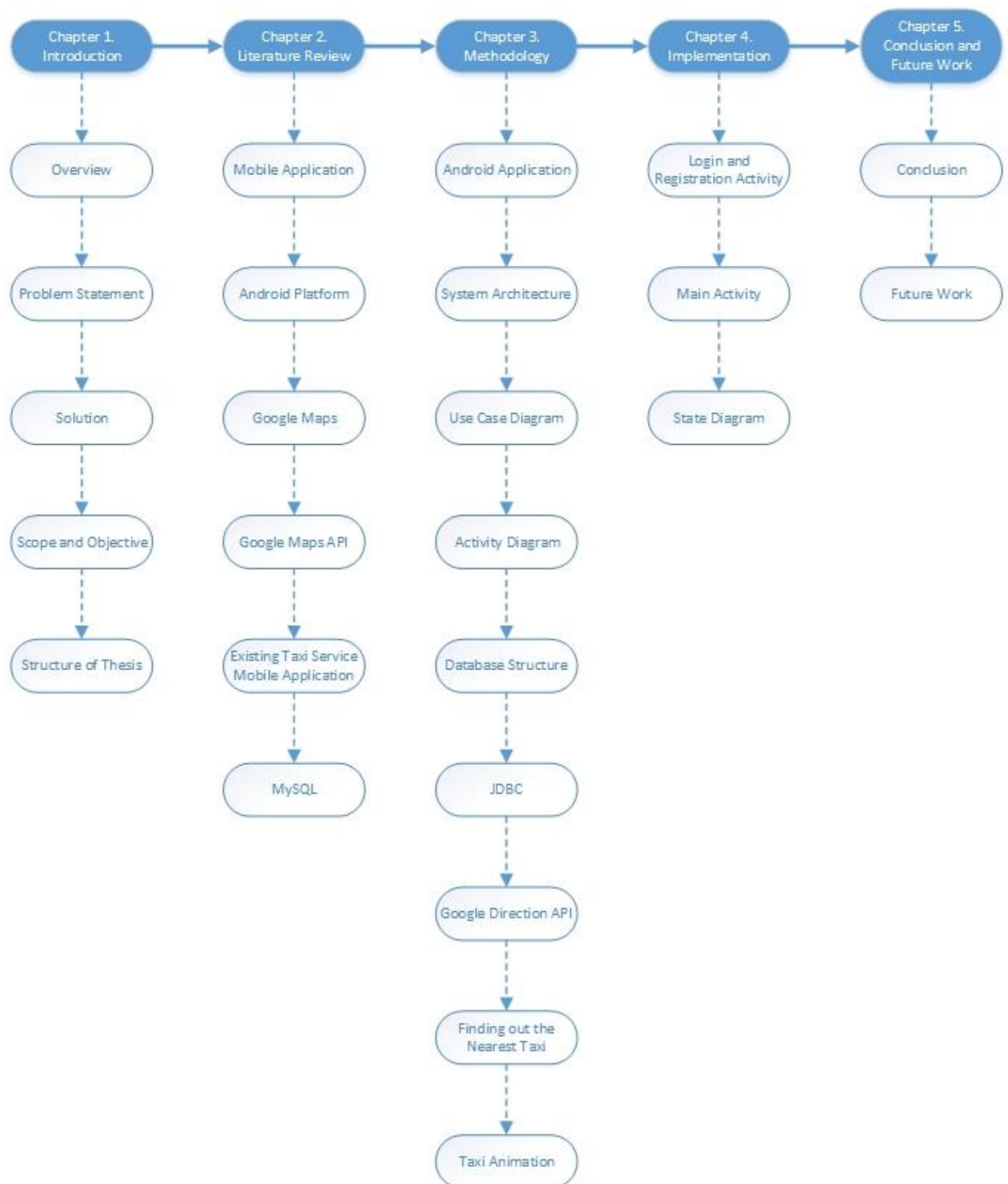


Figure 1 - Thesis Structure

Chapter 1 introduce about my thesis, where the idea come from, why we need an application and which problem the application can solve in our life.

Chapter 2 mention some methods, technique used in the application. Definition and introduction about the techniques and methods such as Mobile Application, Google Maps and Database Server which are used to build application in the thesis.

Chapter 3 include diagrams of the app such as Use Case Diagram, Activity Diagram and also include the way to apply methods and techniques mentioned in chapter 2.

Chapter 4 show process of the app, include pictures of mobile screen when the running functions of the app on smartphone and state diagrams to show the running code process in the app.

Chapter 5 conclude the thesis, mention some experiences and what I have learned though the thesis and the work I should do in the future to improve the application more effectively.

Chapter 2 . Literature Review

In this second chapter, I would like to review some concepts which is related to process that will be used to build the taxi service application.

2.1 Mobile Application

Mobile applications or mobile apps are applications developed for handheld devices, such as mobile phones, smartphones, Personal Digital Assistants (PDAs) and so on. Mobile apps are already exist on devices or can be added by downloading from app stores or the Internet by users. Android, iOS, Windows Phone and BlackBerry are the most popular mobile platforms which support mobile apps today.

The following chart breaks down the three basic types of mobile apps:

Table 1 - Three Basic Types of Mobile Applications [2]

	What It Is	Advantages	Disadvantages
Native Apps	Built specifically to the needs of the various operating systems such as Apple's iOS or Android	Speed, performance and user interface are optimized Works without Internet connection	Must build a specific app for each operating system Takes more time to develop and deploy Higher development costs
Web-Based Apps	Websites built using HTML that are designed specifically for smaller screens	No need to distribute using iTunes or Google Play Works on any device with a browser, but experience varies Lower deployment costs	Slower performance Internet connection is required
Hybrid Apps	Native app shell with feeds from the website	Caches content, so it works offline to a degree Downloadable from app stores	Doesn't run as smoothly as native apps Offline performance can be inconsistent

		Easier to deploy cross platform than native apps Lower cost than native app	Built to specific operating system
--	--	--	------------------------------------

In this project, the application is built as a native app on Android Platform.

2.2 Android Platform

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets [3]. Android provides variety libraries and framework which can provide tools to developers to build new and necessary apps for mobile devices using Java programming language.

According to International Data Corporation, in 2015, Android totally had the highest number of smartphones transported worldwide, with 78% market share in the first three months. In December 2014, according to statista.com, the total number of Android devices used in the north was 1.6 billion. This is a staggering amount, and a very large market potential of users [4].

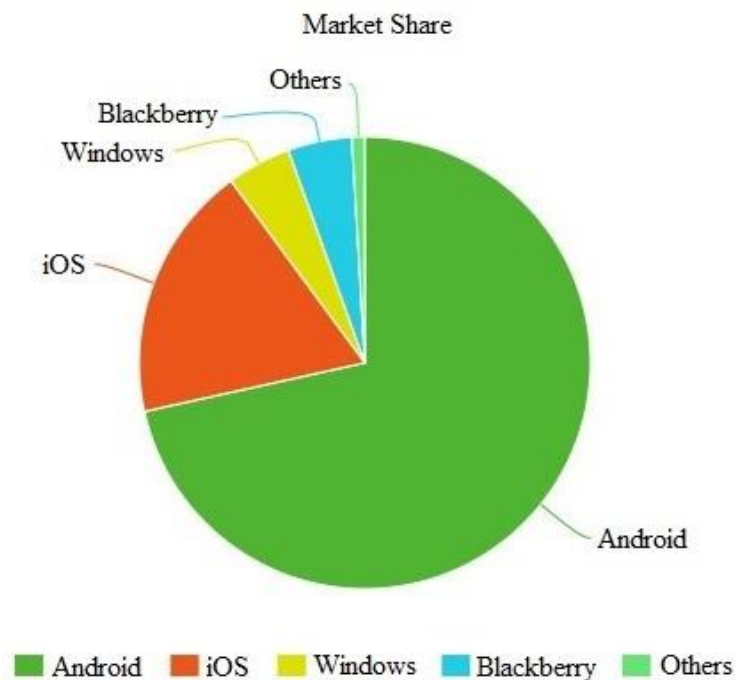


Figure 2 - Market Share of Mobile Applications [4]

With the popular of Android and support form Android SDK, Android is a great idea for building a mobile application.

2.3 Google Maps

About Google Maps, Google Maps has activated for 10 years, Google is still a leading brand in the digital map field. According to Nielsen, only on smartphones, the Google Maps mobile application has an average of 79 million users a month. Google Maps also has an official desktop application and millions of third-party applications that use Google Maps APIs.

Google Maps APIs has a lot of features including an Embed API, Maps Image APIs, Places API, Web Services API and Google Maps API for Work. On programmable Web of Google, there is a list of these APIs.

API documentation as well as code samples, libraries, SDKs and other digital mapping tools is provided very detailed by Google. There is also an API picker that developers can use to find the right mapping API for their projects. Therefore, in this project, I would use Google Maps API for the most effective support in Android app development [5].

2.4 Google Maps API

The Google Maps API allow for the embedding of Google Maps onto web pages of outside developers, using a simple JavaScript interface or a Flash interface. It is designed to work on both mobile devices as well as traditional desktop browser applications.

2.4.1 Google Maps API for Android

There are two service from Google Maps API for Android [6]:

First one is Google Maps Android API: Add maps to Android app. Integrate base maps, 3D buildings, indoor floor plans, Street View and Satellite imagery, custom markers.

The other one is Google Place API for Android: Implement device place detection, autocomplete and add information about millions of locations to app.

2.4.2 Google Maps Web Service APIs

There are eight web services [7]:

Google Maps Geocoding API: Convert between addresses and geographic coordinates.

Google Places API Web Service: Implement autocomplete and add up-to-date information about millions of locations to your site or app.

Google Maps Elevation API: Elevation data for any point in the world.

Google Maps Directions API: Calculate directions among multiple locations.

Google Maps Roads API: Enable snap-to-road functionality to accurately trace GPS breadcrumbs.

Google Maps Time Zone API: Provide time zone data for anywhere in the world.

Google Maps Geolocation API: Find a location based on information from cell towers and WiFi nodes.

Google Maps Distance Matrix API: Estimate travel time and distance for multiple destinations.

These web services use HTTP requests to specific URLs, passing URL parameters as arguments to the services. Generally, these services return data in the HTTP request as either JSON or XML for parsing and/or processing by the application. A typical web service request with a link follow form of Google [8].

To use all APIs of Google, we must have a Google API Key, this Key could be create by access Google Developers Console Website [9]. After we got an API Key, we can start to get database from Google API.

For example, using Directions API to request json data from International University in Thu Duc District and International University in District 1 using request URL of Google and a JSON data will be responded.

Response JSON:

```
"copyrights" : "Map data ©2016 Google",
"legs" : [
  {
    "distance" : {
      "text" : "18.7 km",
      "value" : 18663
    },
    "duration" : {
      "text" : "33 mins",
      "value" : 2002
    },
    "end_address" : "7 Lý Tự Trọng, Bến Nghé, Quận 1, Hồ Chí Minh, Vietnam",
    "end_location" : {
      "lat" : 10.7826157,
      "lng" : 106.7053003
    },
    "start_address" : "Đông Hòa, Di An, Bình Duong, Vietnam",
    "start_location" : {
      "lat" : 10.8775025,
      "lng" : 106.8013837
    },
  },
]
```

Figure 3 - A Part from Response JSON [10]

These APIs support a lot of necessary database for developers to exploit almost data form Google Maps.

2.5 Existing Taxi Service Mobile Applications

To orient the building application process, analyzing and learning from other existing app is necessary.

2.5.1 Uber

Uber is a mobile app, which allows consumers with smartphones to submit a trip request, which the software program then automatically sends to the Uber driver nearest to the consumer, alerting the driver to the location of the customer. The methods built Uber app are Google Maps for map provider, MySQL for managing database and Braintree for payment system [11].

There are 8 features of Uber [12]:

First, registration. For using the app, users may register their accounts and credit cards from their mobile app or they can login with Facebook account or Google account.

Next one is tracking. Passengers have an ability observe drivers' location, and to set a location of pickup by showing the destination on the map or through Google Places.

The other feature is selecting taxi type. Users should be ready to choose which type of car they would like to book with details of the ride like price per kilometer and minimum fare for all type of the car.

With cost calculator feature, a mechanism that would determine the fair rate by calculating a distance between pickup place and destination.

Feature called live tracking. After the taxi is booked, the customer may update the cab live from their app from the initial stage like starting and ending of the trip or arrival of the cab.

For automatic payment feature, with all cashless payments, when the tour is completed, payment is withdrawn automatically through payment infrastructure's API. Later the invoice is sent to the smartphone and to a registered email address. Additionally, clients could able to add various credit cards to the system.

With ratings and feedback feature, passengers should be ready to write a review of their experience. This practice is great for improving service delivery to users.

And the final feature is booking history. Users can know which trip they arrived in the past by seeing trip history list.

2.5.2 Grab

Founded after Uber but no less competitive, Grab or formerly known as GrabTaxi is a Southeast Asia focused taxi-booking app founded by Anthony Tan and Tan Hooi Ling. The idea for GrabTaxi first started when Anthony Tan was a student at Harvard Business School and a classmate pulled him aside to gripe about how hard it was to hail a cab in Malaysia. Tan drew up a business plan for an Uber-like service that won backing from angel investors to start GrabTaxi in 2012, a mobile application that assigns available cabs nearby to commuters using mapping and location-sharing technology [13].

Like Uber, Grab use Google Maps for their map provider.

And Grab Application also has 8 main features:

The first feature is registration. Users need accounts to use the app, they can register by phone number, email or they can login with Facebook account or Google account.

Next feature is tracking. Passengers can follow drivers' location, and users can choose pick-up place or destination by Google Places.

The other one is selecting vehicle. Users can choose GrabCar, GrabShare, GrabBike, GrabTaxi, etc. with details of the ride like price per km as well as price per minute & minimum fare for all type of vehicle.

With cost will be calculated a distance between points of pickup & destination by a price calculator.

With live tracking feature. After the taxi is booked, the customer may update the taxi live from their app from the start point to end point of the trip or arrival of the taxi.

For automatic payment feature, when the trip is completed, payment infrastructure's API will get a pay form users automatically. After that the invoice is sent to the smartphone and to a registered email address. Additionally, users are able to add various credit cards to the system to do payment.

With ratings and feedback feature, passengers could rate point for the drivers. This practice is great for improving service delivery to users.

And the final feature is booking history. Mobile app for taxi service may provide passengers tracking booking history.

Almost features between Uber and Grab do not have too much differences. The competition between them is about the convenience of application interface and fare. Therefore, referring in two these famous taxi service app, the taxi service mobile app in this

thesis will have necessary features and interface which is enough simple for users to use this app.

2.6 MySQL

In the project, information of passengers and drivers such as names, positions, phone number, etc. are saved by MySQL database server. The MySQL database server is the most popular open source database around. In MySQL, the "SQL" part represents for "structured query language" which is a standard interactive and programming language for getting information from and updating a database. This "language" allows users to make queries about the information in database - data selection, insertion, updating, and locating. A database is a collection of information that is organized to allow for easy retrieval [14]. In taxi service app, MySQL is used to collect data from passengers, drivers, save their information, locations and connect database between mobile devices.

Chapter 3 . Methodology

This chapter introduce all of methods that I used to build this mobile application that include the way to request the Google API, connect to database, and method to make taxi moving.

3.1 Android Application

For building the mobile application, I have used Android Studio. Android Studio is the official Integrated Development Environment (IDE) for Android app development. Android Studio includes everything programmers need to build an app, including a code editor, code analysis tools, emulators and more [15]. Java is the official language for Android development. Large parts of Android are written in Java and its APIs are designed to be called primarily from Java. That said, it is possible to develop C and C++ apps using the Android Native Development Kit (NDK), however it isn't something that Google promotes. So in this project, Java is the programming language I almost used [16].

Beside Java, XML is also required when we use Android Studio. Java goes for the basic functions and controls of the app and XML is all about the user interface, which actually isn't used very often, as Google provides palettes in Android Studio where programming can design interface by drag and drop and XML is used to edit the interface when “drag and drop” can not satisfy programmers.

3.2 System Architecture

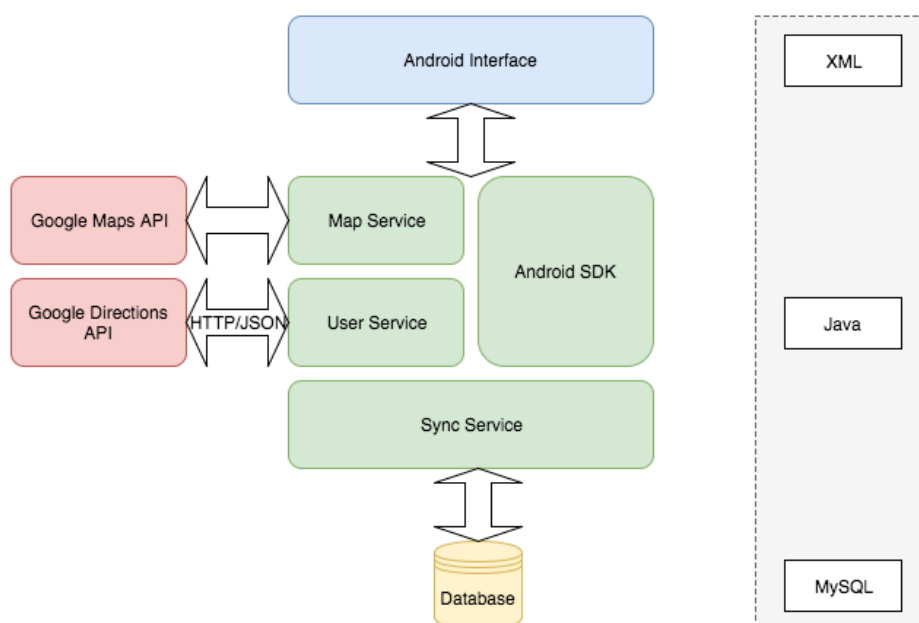


Figure 4 - MATS System Architecture

First of all, an important thing need to implement other functions is display a map. Map will be load by Google Maps API using Android SDK in Android Studio. A Maps Project with framework to implement Map Activity should be created by Android Studio. To use Map Service from Google, we have to register an API Key and turn API enable on <https://console.developers.google.com/>, after the registered key is added to source code, the app will have ability to connect and send request to Google Maps API and Google Map will be shown on the application.

```
<string name="google_maps_key">AIzaSyBTZRgx—PzXbY7qSgTV1Rf8hxBDKPZ3Ec</string>
```

Figure 5 - API Key to using Google Maps Service

After map is loaded, to find the shortest path, we can request to Google Direction API. Android SDK include libraries to support programmers develop on Android platform with some necessary functions to connect to database, interact with Google Map and other services.

3.3 Use Case Diagram

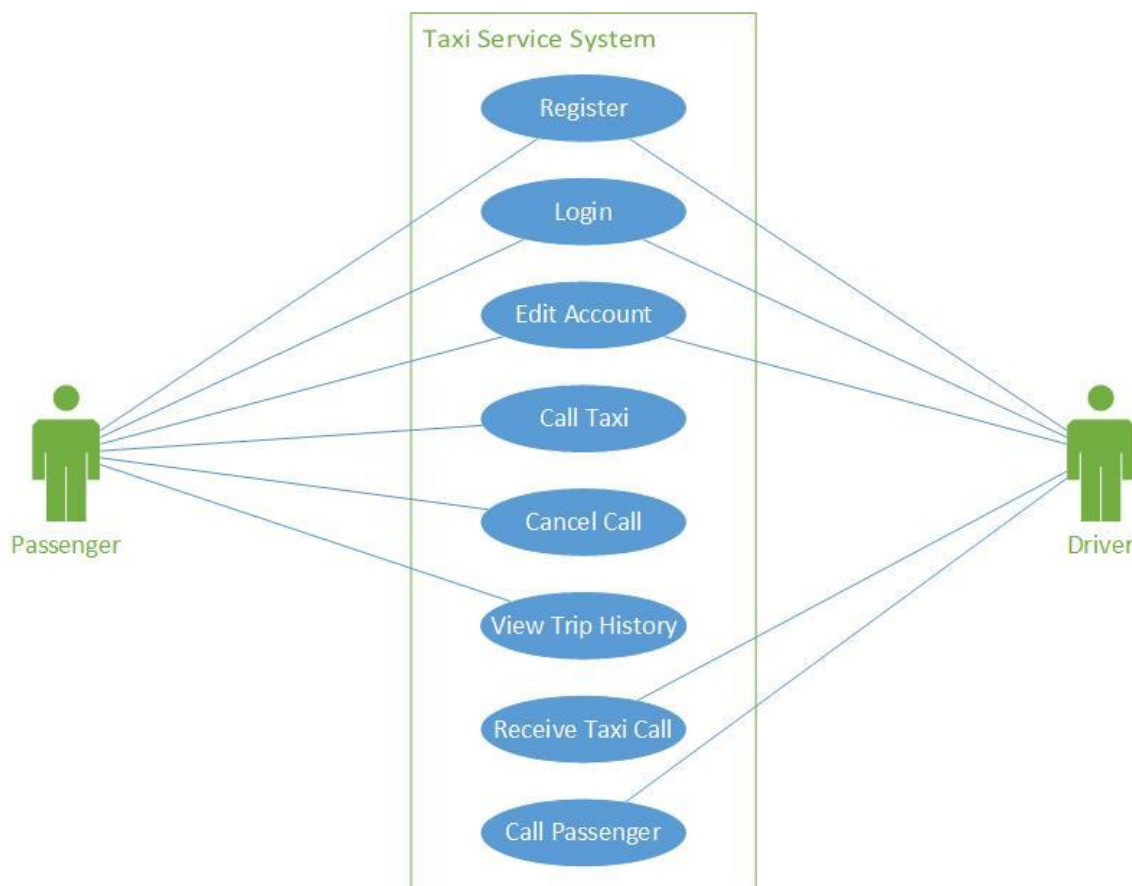


Figure 6 - Use Case Diagram of MATS

Description of use cases:

Table 2 - Use Case 1: Register

Name	UC-1: Register
Summary	Register to become a user of the app
Rationale	Each user has different information, an account is very necessary for the system to manage user.
Users	Passenger, Driver
Pre-conditions	Users want to use the app but do not have any account.
Basic Course of Events	1. Open the app for the first time. 2. Then click 'No account yet? Create one' line. 3. The register successfully confirmation appears.
Alternative Paths	In step 3, if guests leave any information blank, enter wrong format or enter existed email in database, the error will be indicated. Back to step 2.
Post-conditions	User register an account successfully.

Table 3 - Use Case 2: Login

Name	UC-2: Login
Summary	Log-in is required to use the app
Rationale	Users have to log in to help system manage necessary information and support passenger.
Users	Passenger, Driver
Pre-conditions	Users who already have an account.
Basic Course of Events	1. Open the app for the first time. 2. In log in layout, passenger input email and password, then click 'Login' button 3. Login successfully to map layout
Alternative Paths	In step 3, if users leave any information blank or input wrong email or password, the error will be indicated by a message. Back to step 2. Users can log out of the current account by click on 'Logout' button in menu bar on the left of screen.
Post-conditions	Users can access map layout.

Table 4 - Use Case 3: Edit Account

Name	UC-3: Edit Account
Summary	Change registered information of passengers
Rationale	To help users can view and change their account's information.
Users	Passenger, Driver
Pre-conditions	Users who already logged in.
Basic Course of Events	1. Open menu bar on the left of screen and click on 'Settings'. 2. Change information. 3. Click on 'Save' button.

Alternative Paths	If users leave any information blank or input wrong email or password, the error will be indicated by a message. Back to step 2.
Post-conditions	Users have new information.

Table 5 - Use Case 4: Call Taxi

Name	UC-4: Call Taxi
Summary	Call the nearest taxi to have a trip.
Rationale	Main function of this application. That can help user call taxi more easily.
Users	Passenger
Pre-conditions	Passenger who already logged in.
Basic Course of Events	<ol style="list-style-type: none"> 1. (Optional) Choose pick-up place of passenger and destination passenger want to come. 2. Click on 'Call Taxi' button. When passenger do step 1, the nearest taxi driver will come to pick-up place to pick passenger up when passenger set pick-up place or taxi driver will come to current place of passenger as they see their position on map when passenger do not set. When passenger set the destination, they can see estimation of the trip's information and when taxi driver come to them, the app will show notification for passenger to know the trip is started and the taxi will move. 3. Rating after taxi come to the destination by click on rating dialog (when passenger set destination).
Alternative Paths	In step 2, if passengers do not set the destination, rating dialog will show and they can rate taxi drivers after their trip finished.
Post-conditions	Passenger successfully call a taxi.

Table 6 - Use Case 5: Cancel Call

Name	UC-5: Cancel Call
Summary	Cancel a taxi call when passengers change idea.
Rationale	When passengers do not need to call taxi anymore, they can cancel the call before taxi driver come to them.
Users	Passenger
Pre-conditions	Passenger click on 'Call Taxi' button to call a taxi.
Basic Course of Events	<ol style="list-style-type: none"> 1. Click on 'Cancel' button. 2. Choose 'Yes' when the app show question dialog 3. Taxi will stop.
Alternative Paths	In step 2, if passenger choose 'No' the dialog will disappear and the taxi will continue moving.
Post-conditions	Users can cancel taxi call and the taxi stop moving.

Table 7 - Use Case 6: View trip History

Name	UC-6: View Trip History
Summary	Show list of trips and detail of each trip which passenger called.
Rationale	Help passenger can see information of each trip and manage their trips in the past.
Users	Passenger
Pre-conditions	Passenger who made a taxi call before.
Basic Course of Events	1. Open menu bar on the left of screen and click on 'History'. 2. Click on a trip to show trip detail.
Alternative Paths	In step 2, if passenger do not made any taxi called, they cannot see any trip on screen.
Post-conditions	Passenger can view detail of the trip.

Table 8 - Use Case 7: Receive Taxi Call

Name	UC-7: Receive Taxi Call
Summary	Receive a notification when passenger call taxi.
Rationale	To notify taxi driver there is a customer.
Users	Driver
Pre-conditions	Driver who already logged in.
Basic Course of Events	1. Click on "OK" button when driver receive a notification to see passenger's location. 2. Slide from the bottom of device screen to view passenger's information.
Alternative Paths	If passenger cancel trip after driver click "OK", the driver will have a notification.
Post-conditions	Driver have information and location of passenger.

Table 9 - Use Case 8: Call Passenger

Name	UC-8: Call Passenger
Summary	Call Passenger to confirm trip.
Rationale	To help driver contact with passenger.
Users	Driver
Pre-conditions	Driver have a notification about a taxi call from passenger.
Basic Course of Events	1. Slide from the bottom of device screen to view passenger's information. 2. Click on "CALL PASSENGER" button.
Alternative Paths	If passenger cancel trip after driver click "OK", the driver can not see "CALL PASSENGER" button and have to wait for next taxi call.
Post-conditions	Driver can have a phone call with passenger.

3.4 Activity Diagram

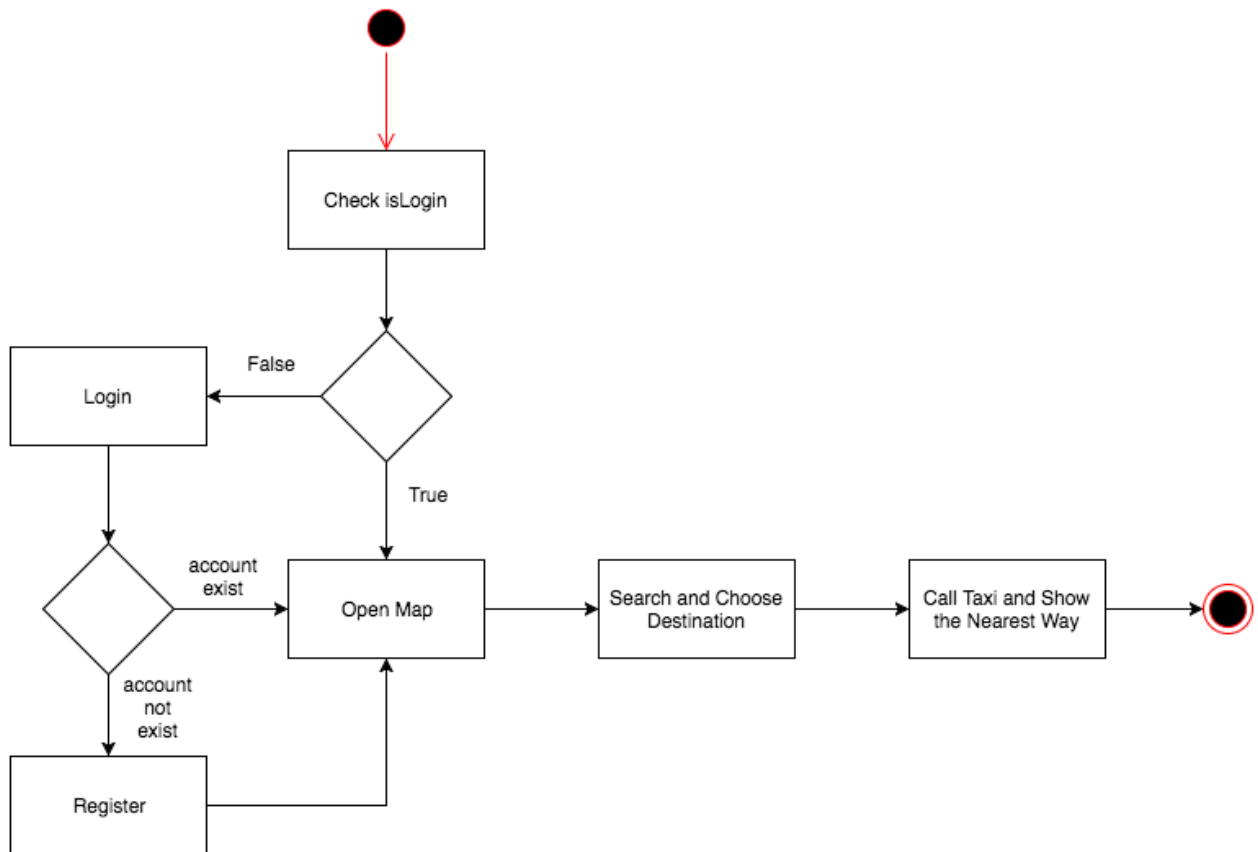


Figure 7 - Activity Diagram of the Project

This is the activity diagram of my application project. When users open the application on their smartphones, system will check are users already login or not. If they have already login, the system will response directly the Google Map or they have to login with their registered account if user would like to use this app. After map layout are presented, users can use functions of the app such as search place, estimate time, fair, distance of their trip and they can call taxi if they want.

3.5 Database Structure

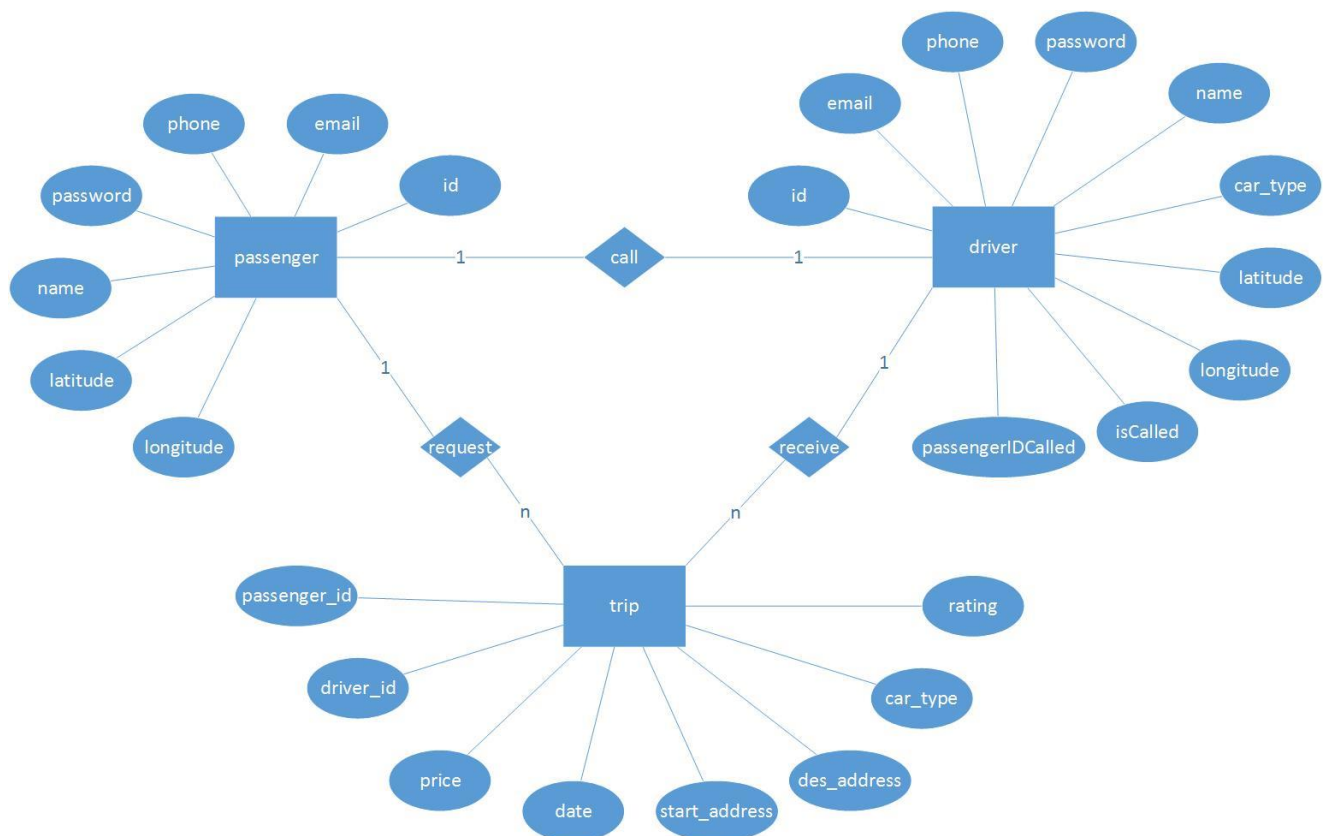


Figure 8 - Entity-Relationship Diagram

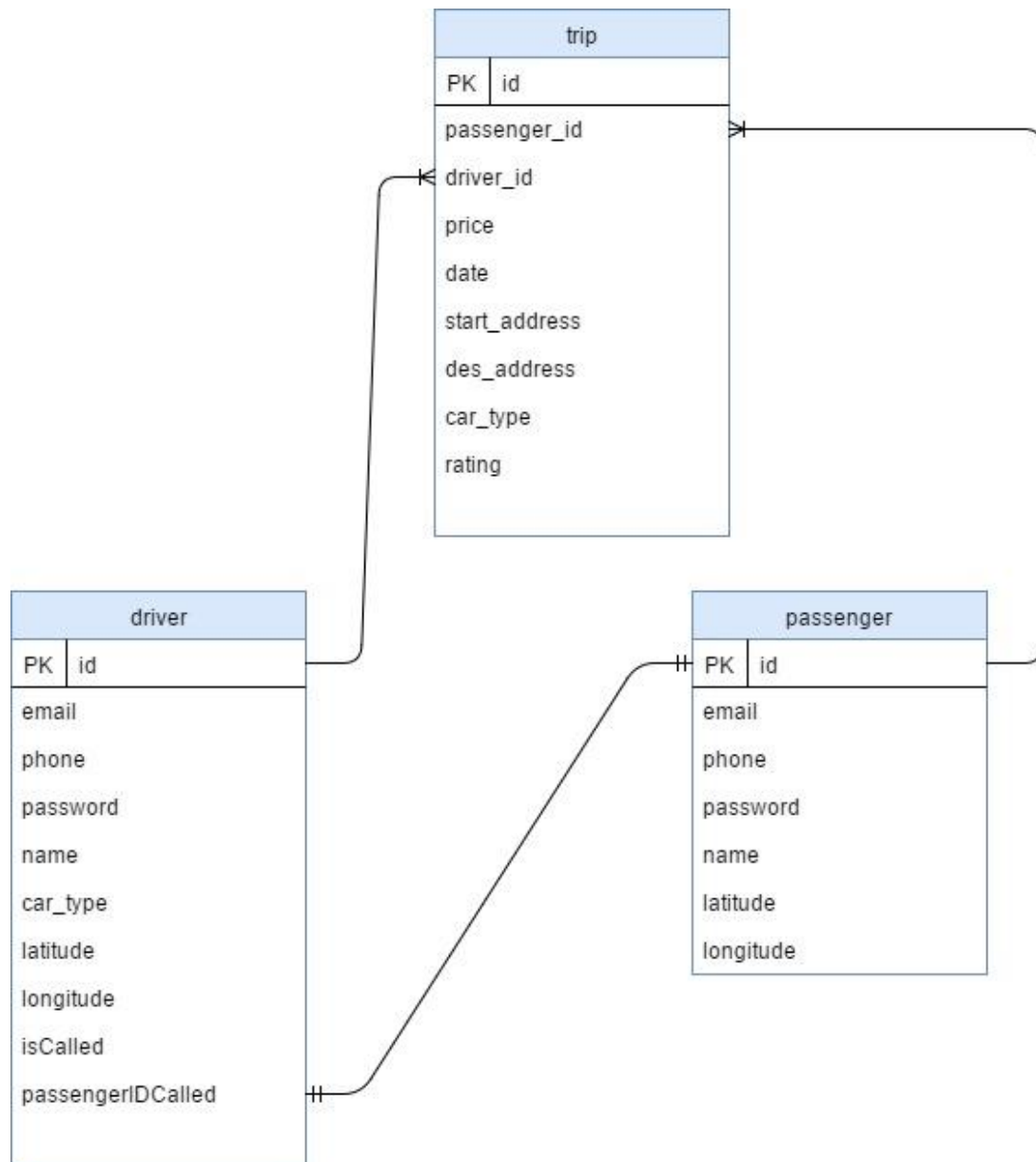


Figure 9 - Database Structure

Database of the app have three tables, one is “**passenger**” save all information of passengers who call taxi, one is “**driver**” which save all information of drivers who drive taxi and “**trip**” table save all information about the trip when passenger call taxi. The relationship between “**driver**” and “**passenger**” is one-to-one, at the moment passenger click on “CALL TAXI” button, when the app choose a driver after passenger call taxi, “passengerIDCalled” attribute will save the ID of the passenger who call this driver, one driver is only called by one passenger and a passenger just can call one taxi driver at the same time. Between “**driver**” and “**trip**” is one-to-many relationship, one driver can receive many trip from many passenger and a trip only received by one driver. The relationship between “**passenger**” and “**trip**” is also one-to-many, a passenger can request many taxi to have many trip and a trip is

only requested by one passenger. Two attributes “latitude” and “longitude” in “driver” and “passenger” table is contain data to know current position of users, these field will be updated frequently by the app. “email”, “phone”, “password”, “name” attributes are personal information of passenger and driver. Attribute “car_type” in tables “driver” and “trip” is save the type of car which driver use. “isCalled” attribute in “driver” table is save value 1 if this driver is called and 0 if the driver is still waiting. “start_address” in “trip” table save the value of address where taxi pick passenger up and “des_address” save the value of destination passenger chose before start the trip. “rating” attribute in “trip” table save the value of quality of a trip which was evaluated by passenger.

3.6 JDBC

To connect with MySQL Server, I have used JDBC. JDBC is an application program interface (API) uses Java standard classes and interfaces which allows the programmer to connect and interact with databases. To establish a connection between the Java application and the database, JDBC follows certain steps:

Step 1: Loading the driver: The driver provides a connection to the database.

Step 2: Creating the connection: Once the driver is loaded, the next step is to create a connection. The connection object uses a URL in the specified format, which includes the machine name, port number and database name. It communicates with the database object.

Step 3: Executing SQL statements: Requires an object for building the SQL statement.

Step 4: Returning the resultset: Retrieves and manipulates the database queries.

Records can be accessed from the first row to the last row of the database.

And then we need a class to run query to get data of a specific object.

We can use “SELECT” query for login function and “INSERT” query for register function.

Notice that, JDBC can not run on same thread with main thread, so if we want to connect to MySQL Server on Android, we have to create a new thread and execute it.

Connecting to Database is also an important task of this project, we need it to implement other functions. This project need database to manage user’s accounts, user’s information, identify user’s position and also driver’s position.

3.7 Google Direction API

To find the shortest way between two places on Map, we can use Google Direction API. Google Direction API is a web service of Google Map APIs, this API will response data on JSON or XML. Therefore, to use the responded data, we need to build a class using HTTP Connection to request data by an URL.

In my project, I have chosen the returned data is JSON, so the data result we receive from Google Web Service is a JSON data which will have a structure like this:

```
{... "route" : [{... "legs" : [{... "steps" : [{... "polyline" : {"points" : "{...}, ...}, ...],},],},],}
```

And we need parse that data to use it in Java.

An array to contain latitude and longitude of all points which place on the shortest way between two locations must be declared. The data of points to get these latitudes and longitudes is what we parsed from the JSON data. Data of this array can be used to estimate moving duration, distance of path, draw the shortest way for users, find the nearest taxi around users and is an assemblage of positions that we can use for taxi to move on map. And the path we have from Google Directions API is exact the shortest path for taxi can move because the coordinates will not be collected and Google Directions will move to the other way to collect coordinates if there is a one-way street on the path to destination.

3.8 Finding out the Nearest Taxi

Android SDK support a function named “distanceTo()” to calculate the distance between two locations on map, but the problem is the result it return is rely on the distance between two points on map based on straight line, and what we need is the distance of the shortest path between that two points, therefore, we need another way to calculate the exact distance.

To measure distance by path, the method I used is using response data from Google Direction API to find the expected distance. For each taxi near the passenger, I send a request to get JSON data by Google Direction API and then it will return the information include distances from each taxi to the passenger in that JSON. Compare all distances between passenger and taxis, the app can find out the result which is the nearest taxi to the passenger through the distance of path on Google Map.

3.9 Taxi Animation

After find out the nearest taxi, we will have the estimate time, estimate distance and assemblage of points on the path from that taxi to the passenger. By the assemblage of points and using thread in Android, we can make the taxi moving by using the thread to continuity

create marker with latitude and longitude of each point on map and this will create the animation for the taxi and make the taxi moving.

This animation only can be applied for demo when the taxi just moving on the shortest path chosen by the app and only can be used for demo app. But in the real app, we have to collect the real taxi driver's position because taxi driver can move on many different path instead of the shortest path chosen by app due to some special situations such as traffic jam, festival, etc., and the customer also would like to know exact information, thus, to come to the passenger position as soon as possible, taxi driver have to move on another path and make the app give incorrect information to the passenger.

Therefore, to obtain the most accurate information of taxi driver position, version of the app for taxi driver have to upload data frequently to the database, and we will use that data to make taxi moving. Method to create animation for taxi is same as the animation method I mentioned above, but instead of using assemblage of points requested once from taxi to the passenger, we have to request frequently to get assemblage of points from the position at two seconds before and current position of taxi driver, that means the app will connect to database and get the current location of taxi driver after 2 seconds because of a large amount of taxis, we should not let database server always open, and then we use the animation on the path with collected data until driver come to the passenger's location.

Sequence diagram when passenger call a taxi:

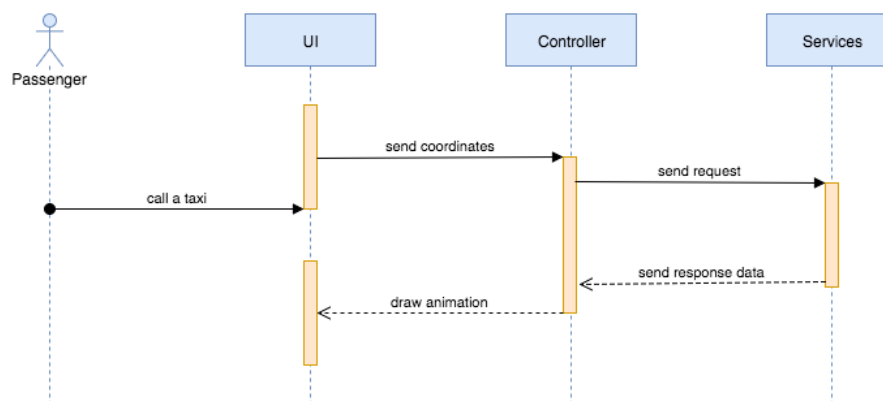


Figure 10 - Sequence Diagram for Calling a Taxi Services

In diagram, when passenger click on screen to call a taxi, the app will send all coordinates concern to passenger's trip and create a request through database to let the driver know. Then the app will get all information of driver and send response data back to the app of passenger and show on screen.

3.10 Class Diagram

3.10.1 Passenger App Class Diagram

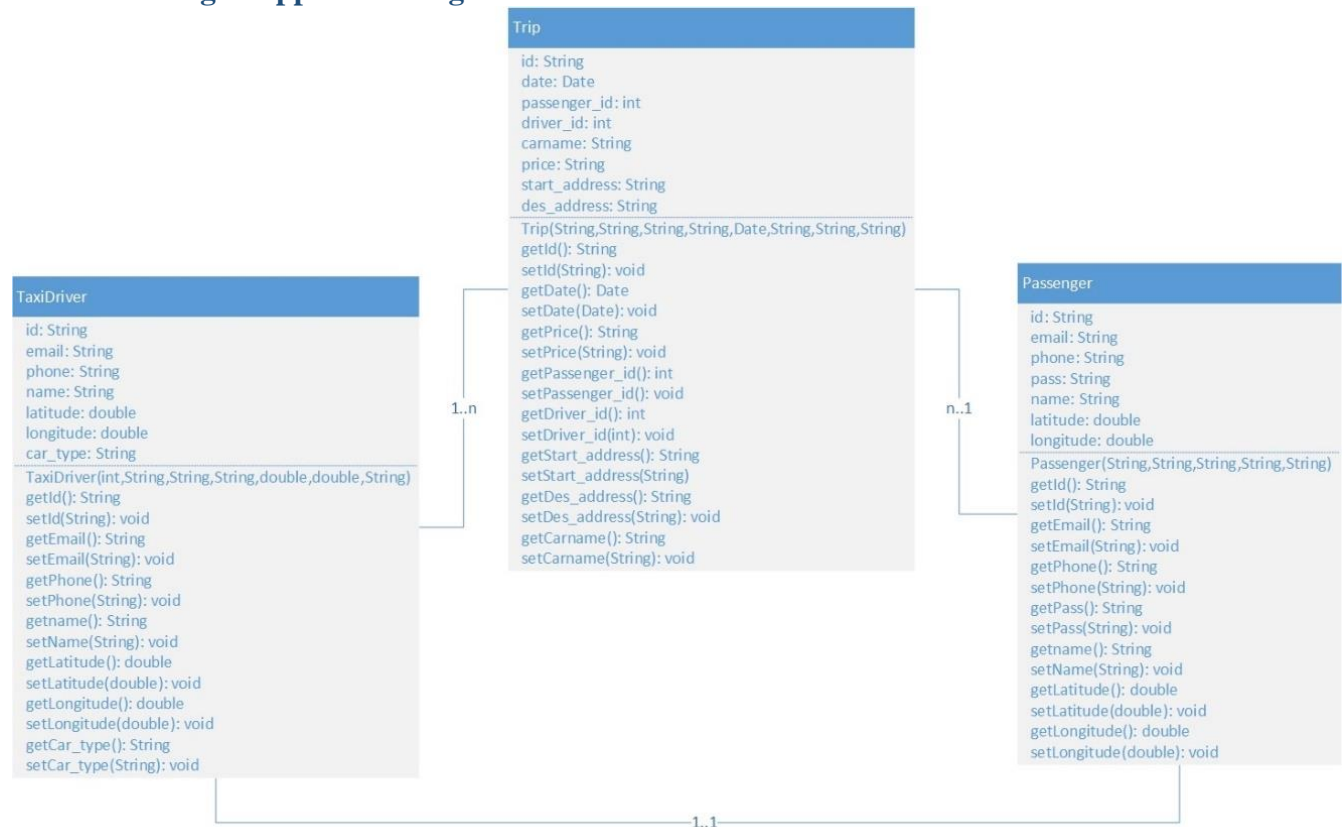


Figure 11 - Passenger, TaxiDriver and Trip Classes

On the diagram above, there are three associations. Between Taxi Driver and Trip, a Taxi Driver can receive many Trips, and a Trip can be arrived by only one driver. Between Passenger and Trip, a Passenger can request many Trips, and a Trip is only requested by one Passenger. Between Taxi Driver and Passenger, this association only appear at the moment Passenger book a trip, so a Passenger just can call one Taxi Driver, and a Taxi Driver is only called by one Passenger. Attributes of them are personal information of Passengers, Taxi Drivers and information of the Trips between these two classes.

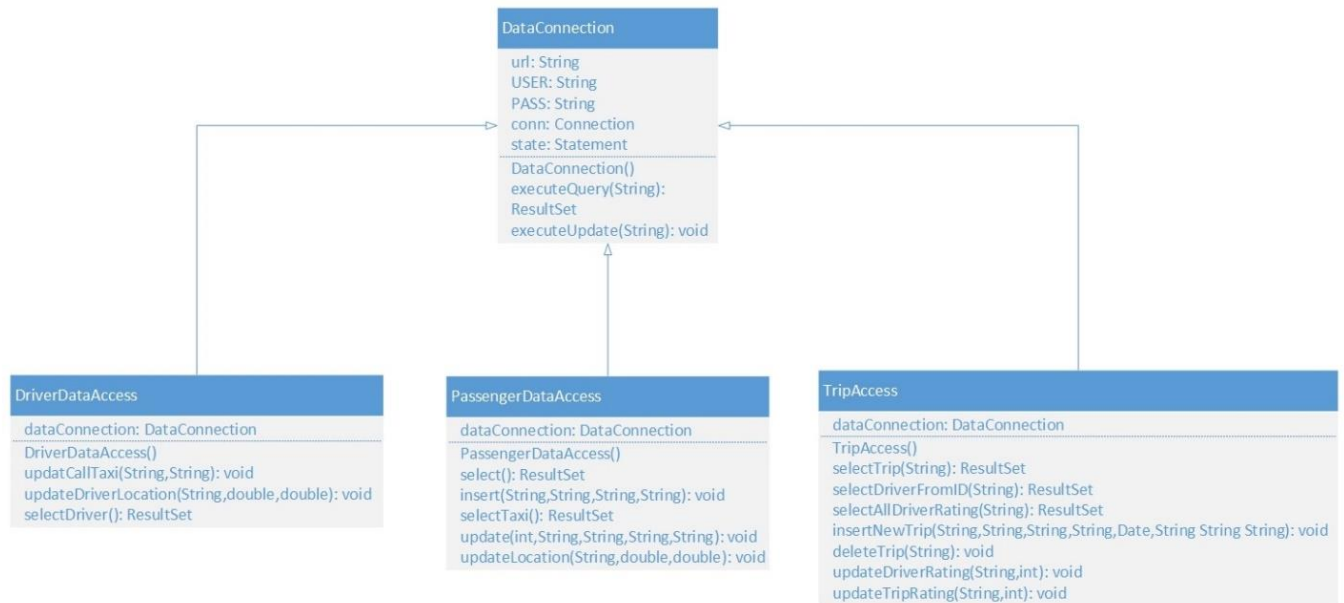


Figure 12 - DataConnection, DriverDataAccess, PassengerDataAccess and TripAccess Classes

DataConnection is class include JDBC to use some functions for connection to Database Server. url is attribute declare Server IP, USER is username and PASS is password of MySQL Server. Three classes DriverDataAccess, PassengerDataAccess and TripDataAccess inherit from DataConnection to request data from Database Server. DriverDataAccess contains queries to get data of drivers, PassengerDataAccess contains queries to get data of passengers and TripAccess contains queries to get data of trips.



Figure 13 - MapsActivity Class Diagram

MapsActivity Class contains main function to run main activity after users log in such as showing map, booking taxi, etc. All other classes except ParserTask, HttpConnection, PathJSONParser Classes, all of them have composition relation with MapsActivity Class, they are a part of MapsActivity Class and need MapActivity Class to run.

In Android applications, there's always at least one main thread that executes the most important tasks in an app: handling user interaction, drawing pixels on the screen, and launching activities. This thread is called the UI Thread, or Main Thread. However, that doesn't mean that all the huge processes must be executed in the main thread. There's a

workaround for this that can improve app's performance and give users a much better experience: background threads. These threads are designed to handle any potentially long tasks that may hang the application. That's why in MapsActivity, some inner classes will be created to use as a background threads. An example function run in background thread is used to connect to database.

SelectTaxiTask Class is a class contain function to get all taxis' information around passenger. UpdateDriverPosition Class will update position of all taxi around passenger and show on device screen. UpdateTrip Class will save all information of trip after passenger complete a taxi-trip. UpdateRating Class will update stars describe for score that passenger rate for trip arrived by a driver. GetDriverRating will get all score of a driver, sum attribute is summary of all score and size is total of trip that driver arrive, sum and size attributes are used to calculate average score of a driver. Distance Class is used to get values of distance and duration between two points on map. UpdateDriverCalled Class is used to save value to let driver know there will be a trip.

HttpConnection Class is used to connect to Google API to get information about path coordination, time, distance, etc. between two points on map and save into JSON data. PathJSONParser Class execute progress to translate JSON data return from HttpConnection Class into value which can use in Java. ReadTask Class is class use to execute HttpConnection Class and ParserTask Class has aggregation relation with ReadTask, ParserTask Class is used to execute PathJSONParser Class after ReadTask Class finish runtime. Three attributes in ReadTask Class will be used in ParserTask Class to identify function when ReadTask is executed. If key is equal to 1 that means function to make taxi moving between two points is called or a path between these two points will be drew if key equal to any other value. When key is equal to 1, if keyMove attribute is equal to 1, taxi will move to passenger or taxi will move to destination which passenger want if keyMove is equal to any other value. isToDes attribute is used to check taxi is at the destination or still moving.

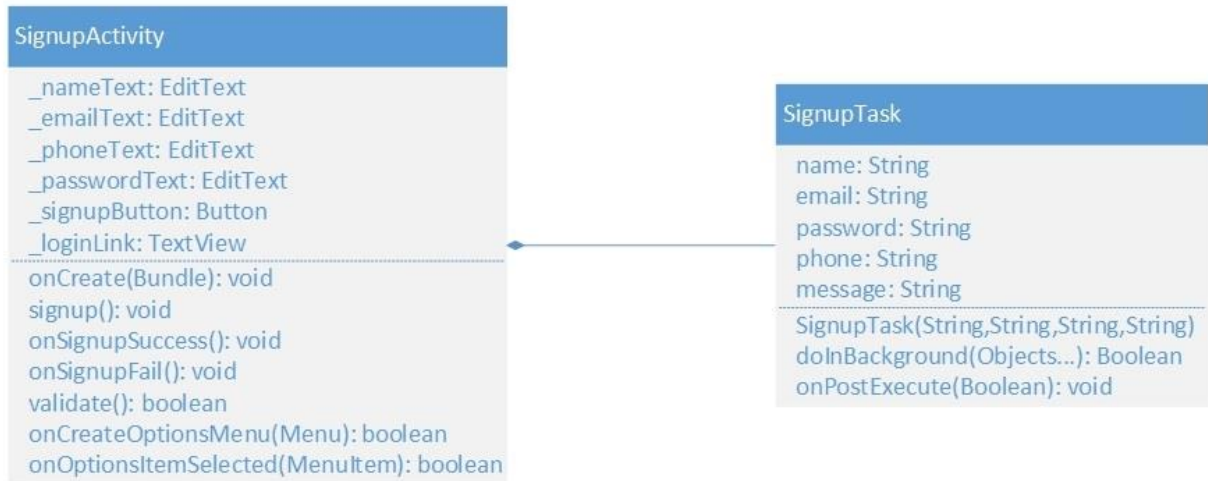


Figure 14 - SignupActivity Class Diagram

SignupActivity Class is used to execute registration function for passengers to create an account. `_nameText`, `_emailText`, `_phoneText`, `_passwordText`, attributes are used to get values of name, email, phone, password from passenger, and these values will be used in SignupTask to insert to database. message attribute in SignupTask is message passenger receive after a sign up activity.

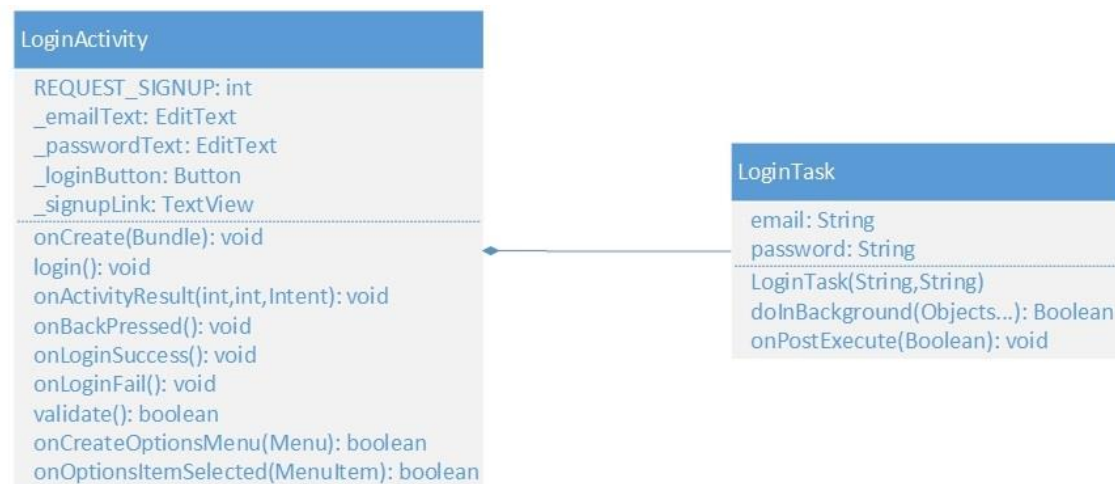


Figure 15 – LoginActivity Class Diagram

LoginActivity Class control sign in activity. `_emailText` and `_passwordText` attribute are values of email and password got from passenger. `REQUEST_SIGNUP` is a key to let the function know when passenger want to change to sign up function. Email and password receive from passenger will be used in LoginTask to connect to database and check account input of passenger.



Figure 16 - History Activity Class Diagram

History Class control showing history activity. LoadTrip Class will get data from database server through HistoryAdapter to show on ListView history_list of History Class.

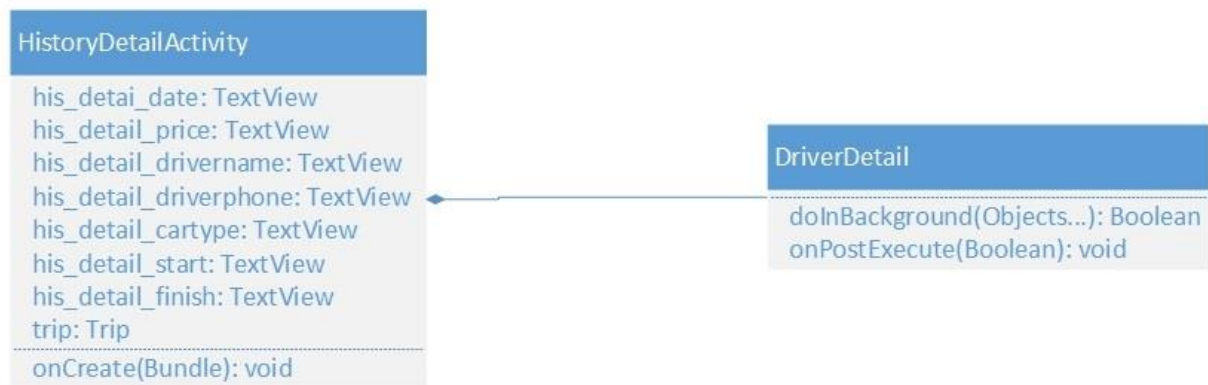


Figure 17 - HistoryDetailActivity Class Diagram

HistoryDetailActivity Class control activity showing detail of a trip when user choose in history list include date (his_detail_date), price (his_detail_price), driver's name (his_detail_drivename), driver's phone number (his_detail_driverphone), type of car (his_detail_cartype), address of start point (his_detail_start) and end point (his_detail_finish) of a trip.

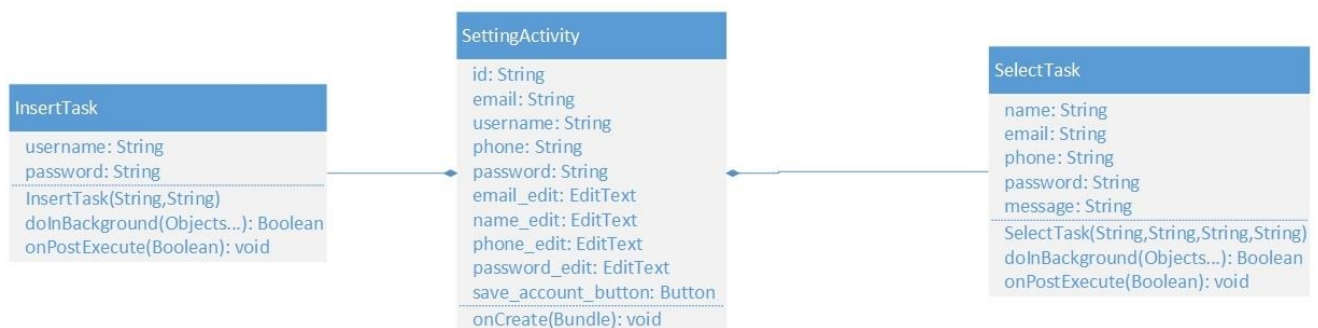


Figure 18 - SettingActivity Class Diagram

SettingActivity Class is used to change information of passenger such as email, username, phone and password. SelectTask Class will get data of passenger and display on screen. InsertTask Class will update new information of passenger to database server.

3.10.2 Driver App Class Diagram

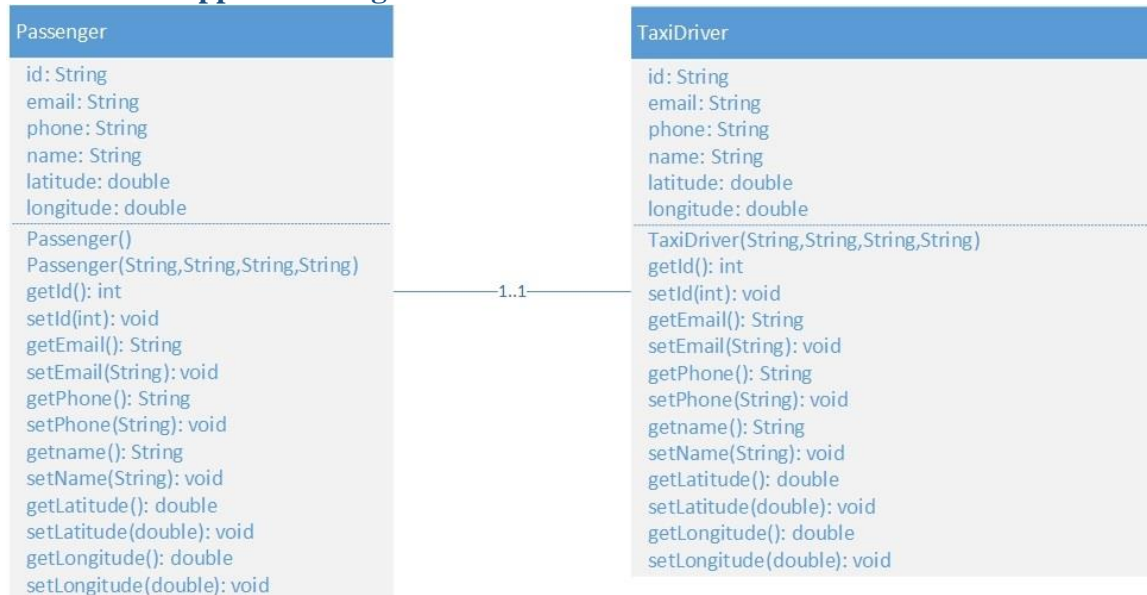


Figure 19 – Driver App Passenger and TaxiDriver Classes

Between Taxi Driver and Passenger is association relation, this relation only appear when Passenger book a trip, so a Passenger just can call one Taxi Driver, and a Taxi Driver is only called by one Passenger. Attributes of them are personal information of Passengers, Taxi Drivers and information of the Trips between these two classes same as in the Passenger App.

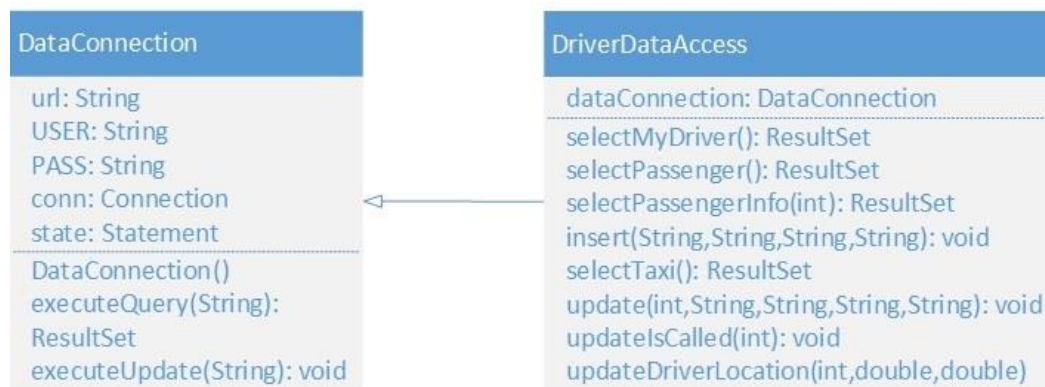


Figure 20 – Driver App DataConnection and DriverDataAccess Classes

DataConnection is class use JDBC to create connection to connect Database Server. `url` is attribute declare Server IP, `USER` is username and `PASS` is password of MySQL Server. **DriverDataAccess** Class inherit from **DataConnection** contains queries to get data concern to drivers to use in the app.

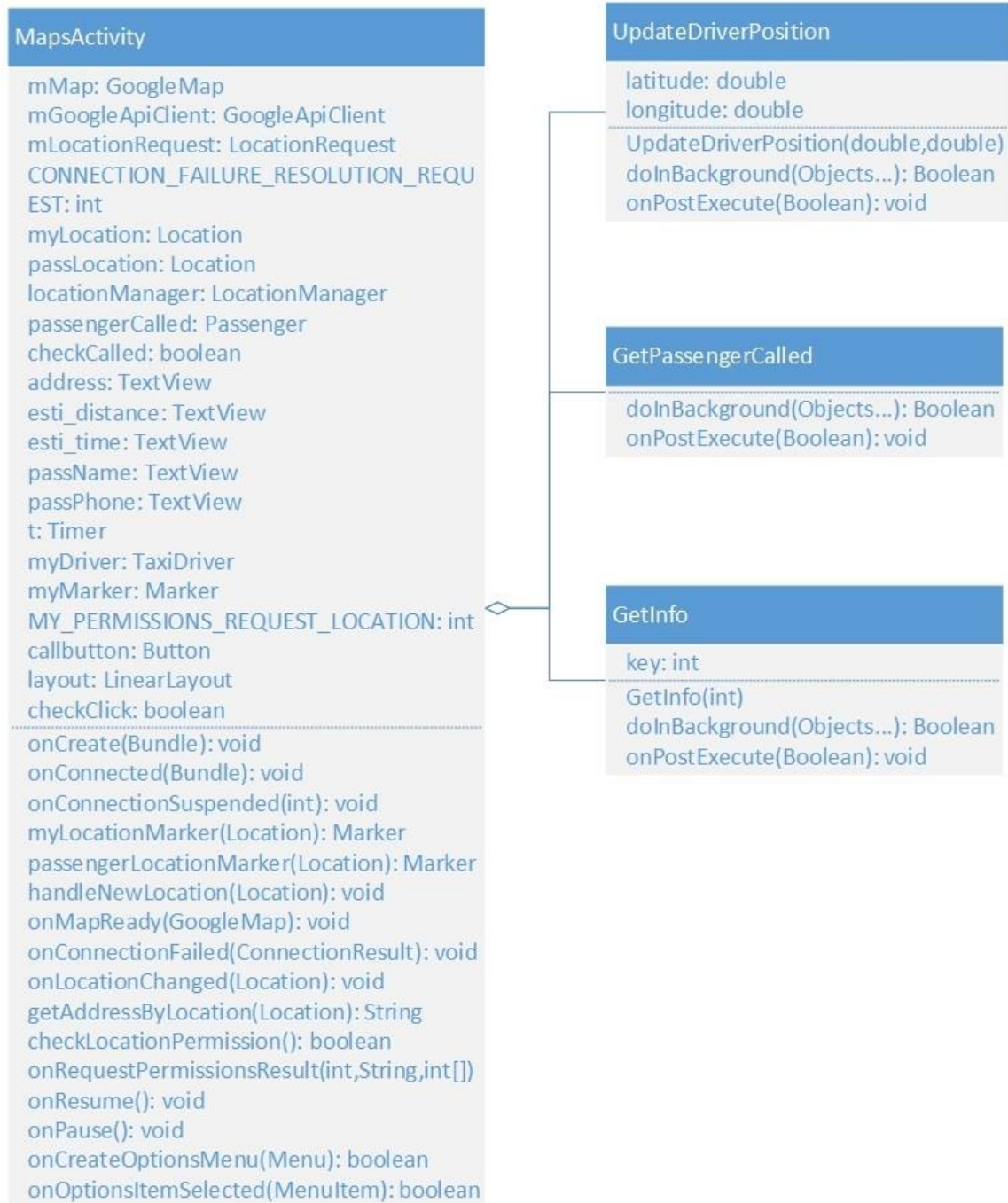


Figure 21 – Driver App MapsActivity Class Diagram

MapsActivity Class control main activity of the driver app. CONNECTION_FAILURE_RESOLUTION_REQUEST and MY_PERMISSIONS_REQUEST_LOCATION attributes is key to call some functions in Android. UpdateDriverPosition Class will update current coordinates of driver to database server. GetPassengerCalled Class will get data such as id, coordinates of passenger who just call driver. And GetInfo Class will get all personal information of passenger such as name, phone number to help user contact with passenger.

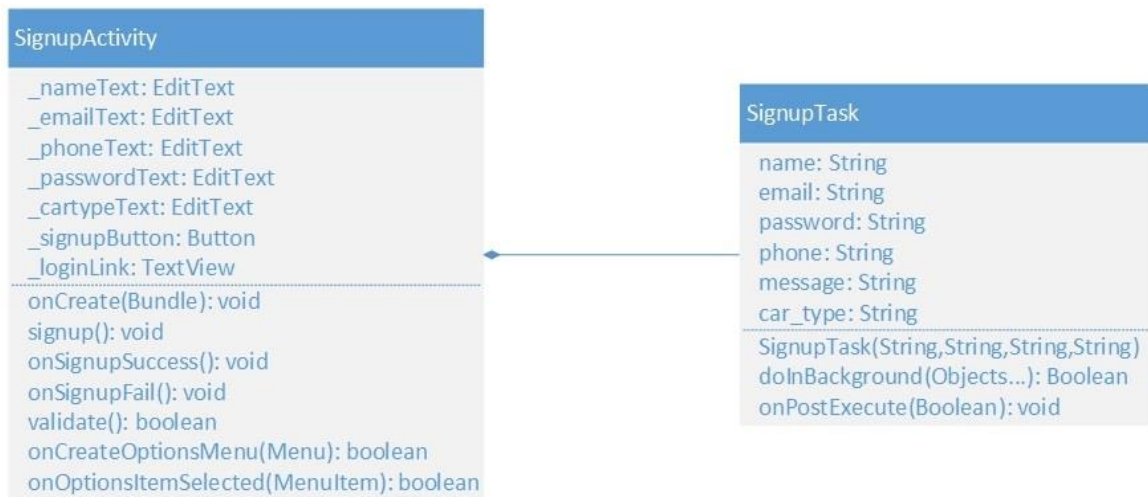


Figure 22 - Driver App SignupActivity Class Diagram

SignupActivity Class is used for drivers to register an account. `_nameText`, `_emailText`, `_phoneText`, `_passwordText`, `_cartypeText` attributes are used to get values of name, email, phone, password and type of car from driver, and these values will be used in SignupTask to insert to database. message attribute in SignupTask is message driver receive after a sign up activity.

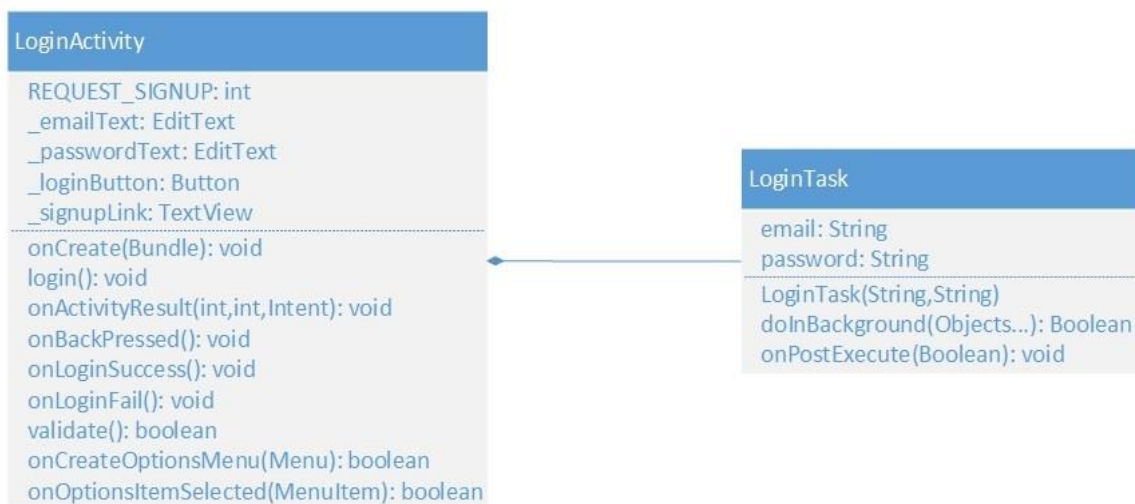


Figure 23 - Driver App Login Activity

Same as Passenger App, LoginActivity Class also control sign in activity. `_emailText` and `_passwordText` attribute are values of email and password in driver's account. REQUEST_SIGNUP is a key to let the function know when driver want to register. Email and password receive from driver will be used in LoginTask to connect to database and check account input of driver.

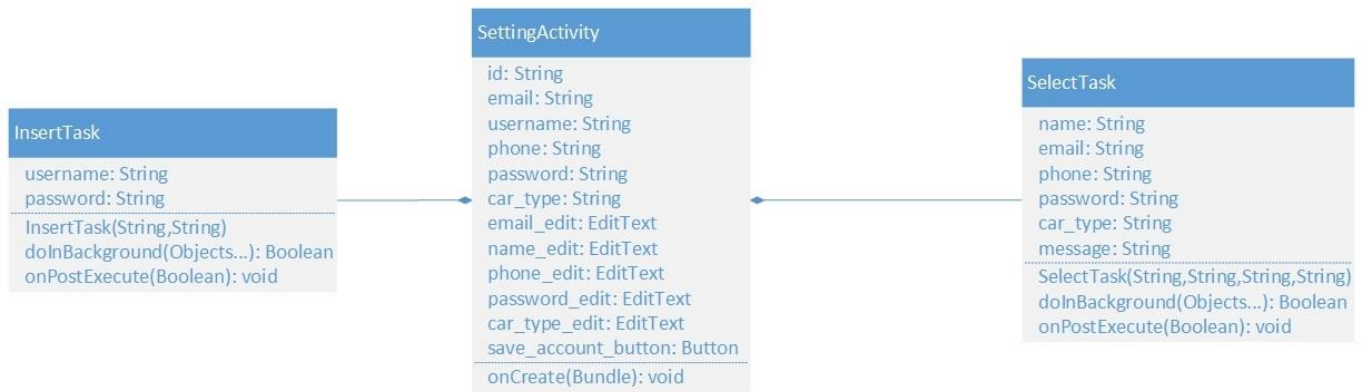


Figure 24 - Driver App Setting Activity Class Diagram

SettingActivity Class can change driver's information such as email, username, phone, password and type of car. SelectTask Class select data of driver and show on screen. InsertTask Class will update new information that driver changed to database server.

3.11 State Diagram

3.11.1 Load map and identify user position process:

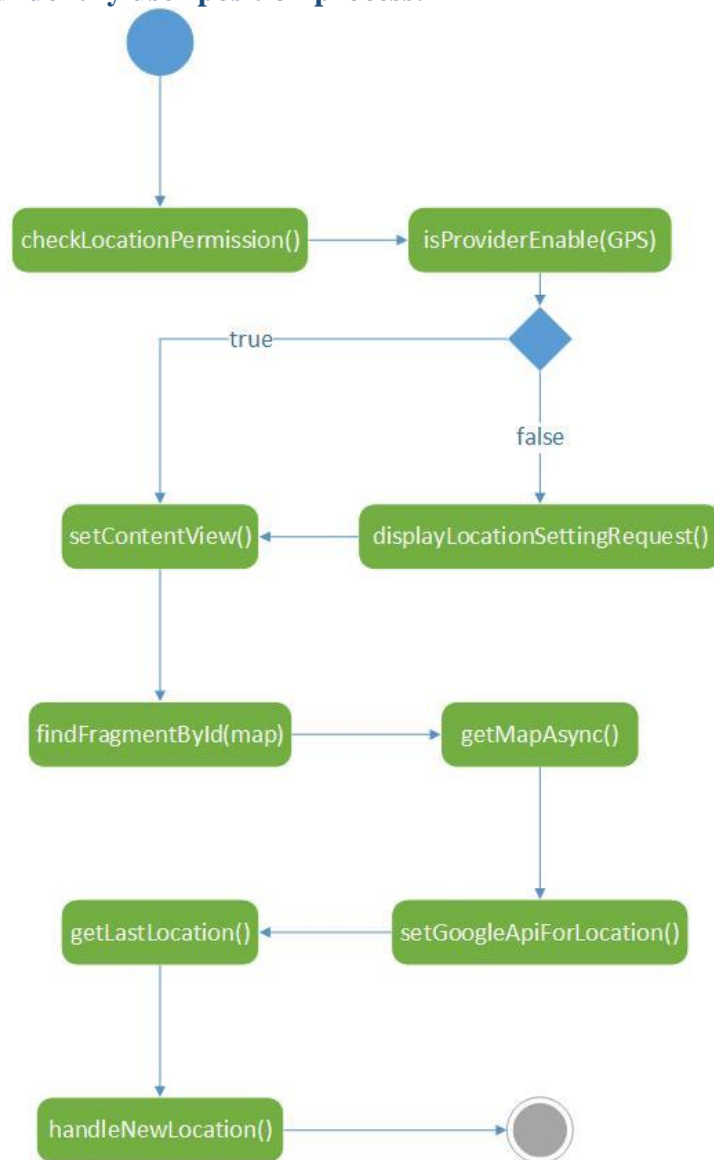


Figure 25 - Load Map and Identify User Position Process Diagram

When user open the app, the device will check permission allowed by user to run this app. Then function `isProviderEnable(GPS)` will check GPS of the device and request to turn on by `displayLocationSettingRequest()` if GPS is off. Then the app will load the layout by `setContentView()` and `findFragmentById(map)` will get value of map on layout and then `getMapAsync()` function will allow to use `onMapReady()` method. The `setGoogleApiForLocation()` function will declare value to use Google API to get current location of user and `getLastLocation()` will use the values from `setGoogleApiForLocation()` to return current location of user. The `handleNewLocation()` is a function used to display a marker, in this situation, that is the marker of user.

3.11.2 Display all taxis around process:

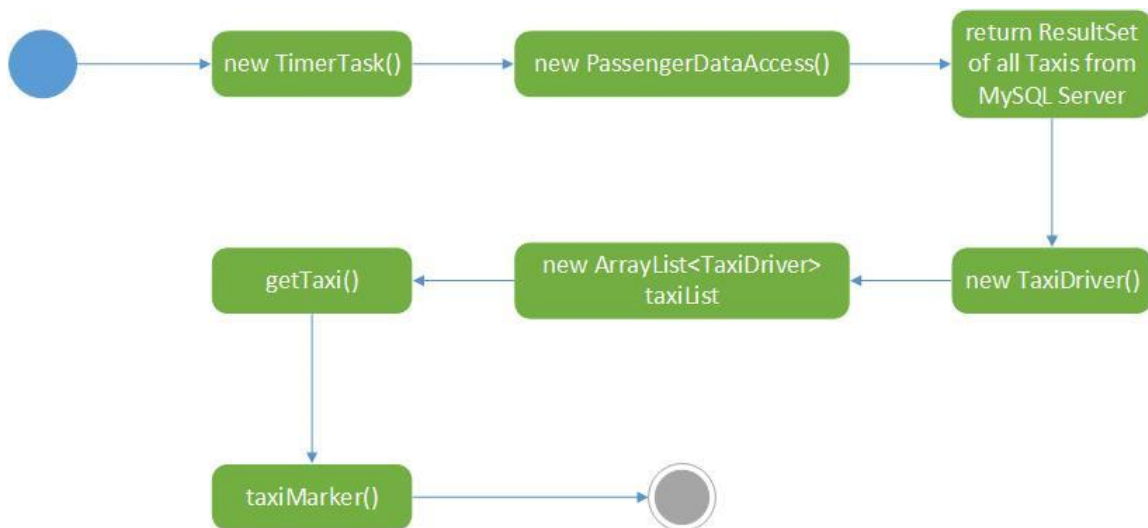


Figure 26 - Display All Nearby Taxis Process Diagram

To display taxis around passenger, first, the main function of activity **onCreate()** will create a new object **TimerTask()**, this is a class can create a method and run the code inside repeatedly after a time, in here I set the time is 2 second. For each 2 seconds, the app will create new **PassengerDataAccess()** to access to MySQL Server, and the MySQL Server will return a **ResultSet** contain data of all taxis. Then the app will add these data to **TaxiDriver()** object and add all objects **TaxiDriver()** to an **ArrayList** named **taxiList**. The **taxiMarker()** function will add marker for each taxi in **getTaxi()** function.

3.11.3 Choose destination and display estimate values process:

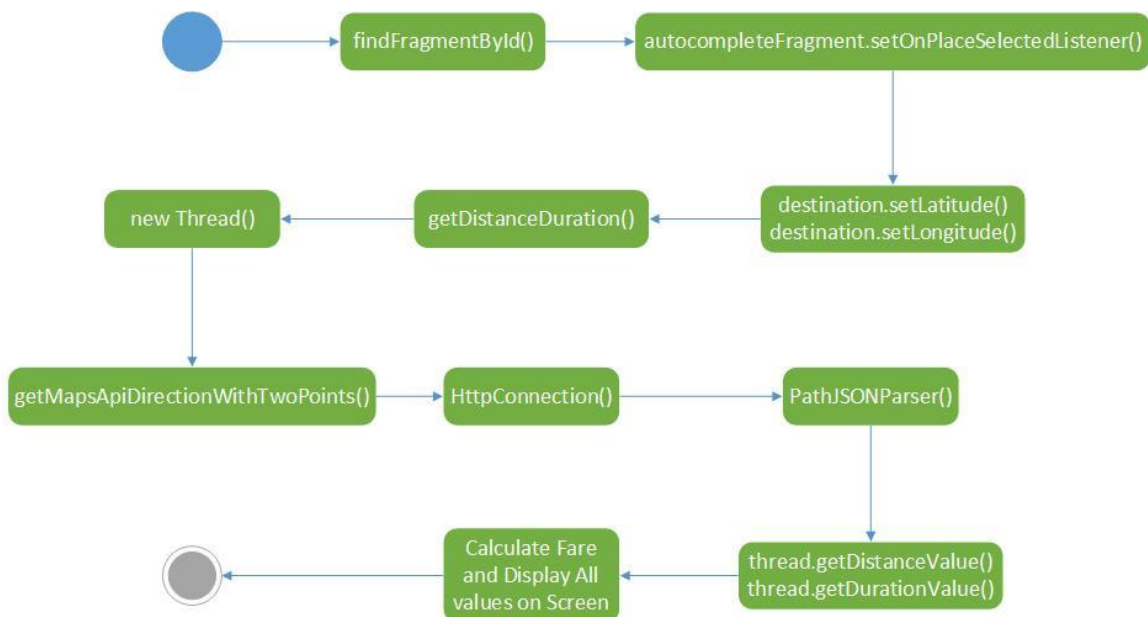


Figure 27 - Choose Destination and Display Estimate Values Process Diagram

First, **findFragmentById()** will find layout of Google Place Autocomplete Search Bar to use it. When user choose destination on autocomplete search bar, the app will get values of latitude and longitude for destination. And then **getDistanceDuration()** function will use these latitude and longitude to get distance and duration from passenger to the destination. The **getDistanceDuration()** function will create a new **Thread()** to make the app can connect to network by **HttpConnection()** class, it will use the url create by **getMapsapiDirectionWithTwoPoints()** to get a JSON data that contain distance and duration. **PathJSONParser()** is class which can translate JSON data to use on the app.

An example of JSON data will be returned:

```
{
  "geocoded_waypoints" : [
    {
      "geocoder_status" : "OK",
      "place_id" : "ChIJFfqFHKTYdDERhRYKVJ084yo",
      "types" : [ "route" ]
    },
    {
      "geocoder_status" : "OK",
      "place_id" : "ChIJW0BYtDgvdTERz8joZGptT4s",
      "types" : [ "street_address" ]
    }
  ],
  "routes" : [
    {
      "bounds" : {
        "northeast" : {
          "lat" : 10.877503,
          "lng" : 106.8084432
        },
        "southwest" : {
          "lat" : 10.7728475,
          "lng" : 106.6980971
        }
      },
      "copyrights" : "Dữ liệu bản đồ ©2017 Google",
      "legs" : [
        {
          "distance" : {
            "text" : "19,7 km",
            "value" : 19668
          },
          "duration" : {
            "text" : "39 phút",
            "value" : 2350
          }
        }
      ]
    }
  ]
}
```

Figure 28 - An Example of JSON Data Returned by Google [17]

We can see in JSON Data Example has values such as “distance”, “duration”, “lat” referring to latitude and “long” referring to longitude. By JSON Parser, these JSON data can be translate into and use in Java.

3.11.4 Call a taxi process:

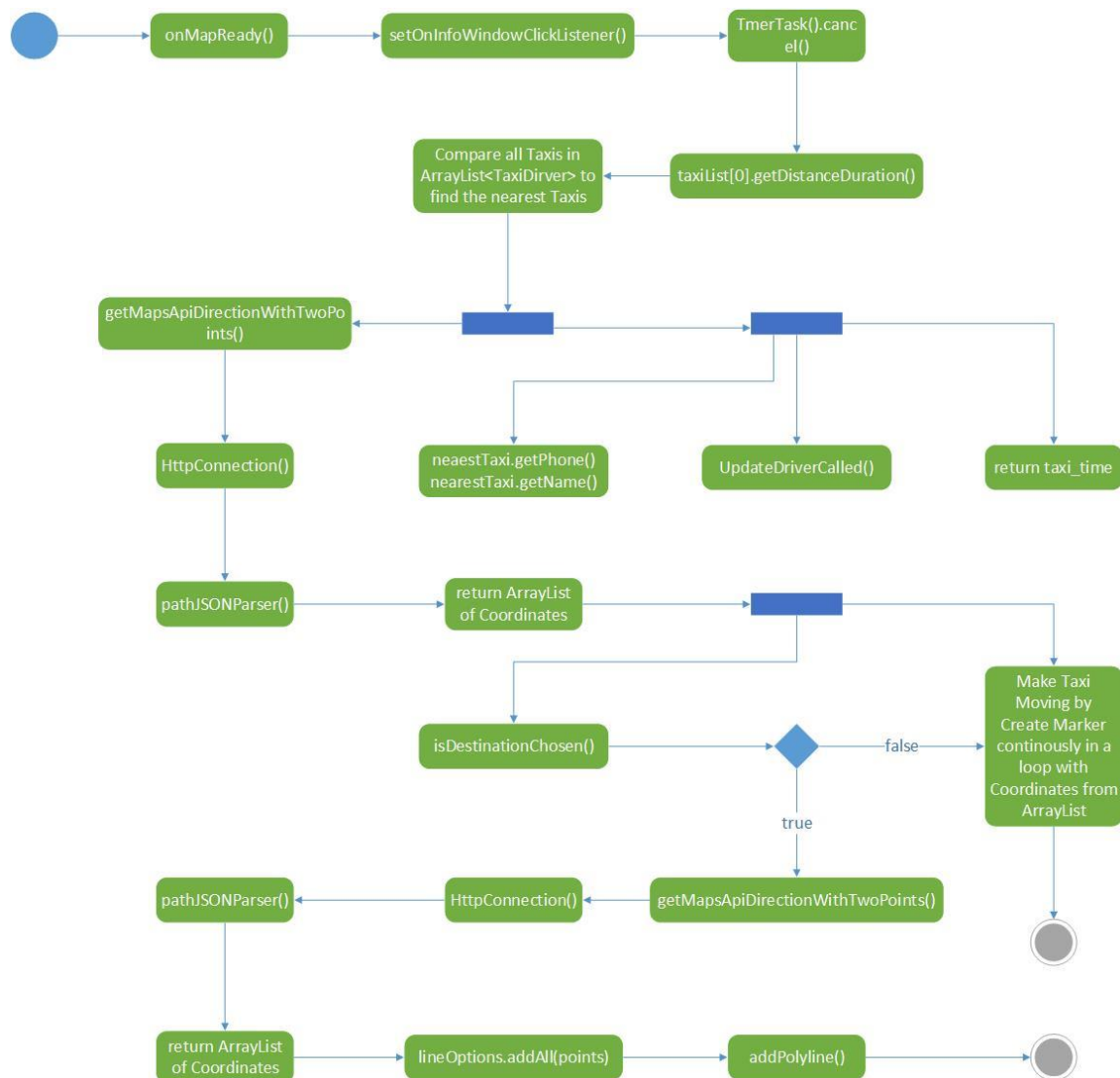


Figure 29 - Call a Taxi Process Diagram

The method **onMapReady()** will let some functions active when map is already loaded. The “CALL TAXI” button is an Info Window of Google Map, so function **setOnInfoWindowClickListener()** will set the action when user click on “CALL TAXI” button. When the button is clicked **TimerTask()** class used to display taxis position each 2 second will cancel, the first **TaxiDriver** object in **ArrayList taxiList** will be used to compare with each other **TaxiDriver** object in **taxiList** to find out the nearest taxi to passenger. When the nearest **TaxiDriver** object is found out, the app can get values as time from taxi to the passenger, phone and name of driver to show on screen, the id of passenger and trip request notification will update to MySQL Server to let driver know by **UpdateDriverCalled()**. To make the taxi moving, the app will send a request by url created by **getMapsapiDirectionWithTwoPoints()** to get the JSON data through **HttpConnection()** to

send request and **PathJSONParser()** to get the all coordinates of the path from driver to the passenger. From this moment, there are two action will execute, one is making taxi moving and the other is draw path from passenger to the destination if passenger chose. The app will get the coordinates data got from JSON above and create marker repeatedly on map to make taxi moving. To draw a path from passenger to destination, the way is same as mentioned above by using **getMapsapiDirectionWithTwoPoints()**, through **HttpConnection()** to send request and **PathJSONParser()** to get all coordinates from passenger to the destination, and **Polyline** will draw a line between each couple of coordinates and display the path, just **addAll(points)** and **addPolyline()** to use this function.

3.11.5 MATS Driver process

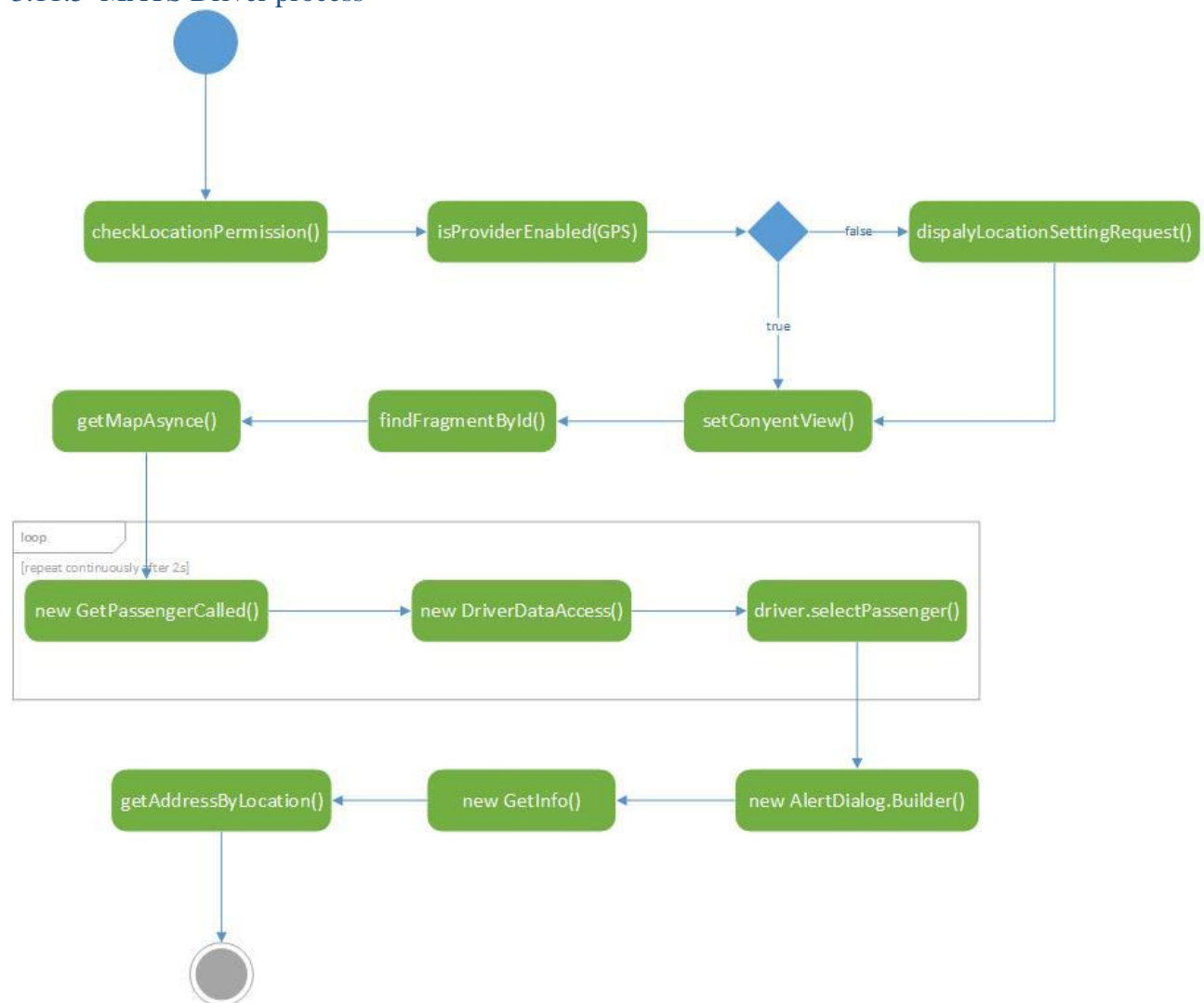


Figure 30 - MATS Driver Process Diagram

Same as the application for passenger, when user open MATS Driver app, the app will check permission by **checkLocationPermission()** and user need allow permission to open location of device to use the app (this function is only work on Android version from

6.0). After that, **isProviderEnabled(GPS)** will check GPS of the device is active or is not active. If GPS of device is off **displyLocationSettingRequest()** will send a notification to request user to open GPS or **setContentView()** will show main layout of the app if GPS is already on. Then a loop will run and repeat continuously after 2 seconds to identify trip request from passenger. **GetPassengerCall()** class create **DriverDataAccess()** object to execute **driver.selectPassenger()** function to check on server if there is any trip call driver receive. And if driver receive a trip request, the loop will stop, **AlertDialog.Builder()** will create a notification to notify driver and **GetInfo()** class will get information of passenger such as name, phone number, latitude, longitude. **getAddressByLocation()** function is used to find pick-up address of passenger rely on latitude and longitude.

Chapter 4 . Implementation

4.1 Login and Registration Activity

To use this app and the app can control user's information, an account to sign in is required.

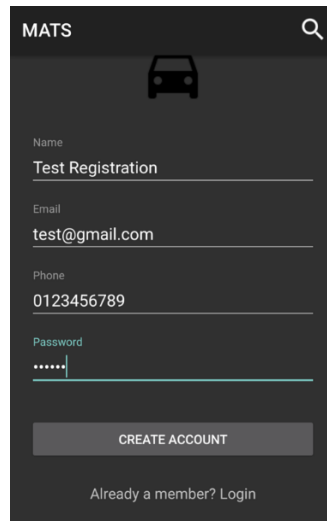
A screenshot of the registration screen in a mobile app. The screen has a dark background. At the top, there's a header with the text "MATS" on the left and a magnifying glass icon on the right. Below the header is a car icon. The form contains four input fields: "Name" with the text "Test Registration", "Email" with "test@gmail.com", "Phone" with "0123456789", and "Password" with masked characters ".....". Below these fields is a button labeled "CREATE ACCOUNT". At the bottom, there's a link that says "Already a member? Login".

Figure 31 - Registration Activity

Users can create an account by fill all information such as name for driver to know, email as username, phone for driver to contact to passenger and password to protect the account and personal information, and then click on “CREATE ACCOUNT” button, when have some problems such as email or phone number are already existed, user will receive a message.

After create an account, user can use that account to login and use main activity of the app on Google Map. Email and password are required to login.

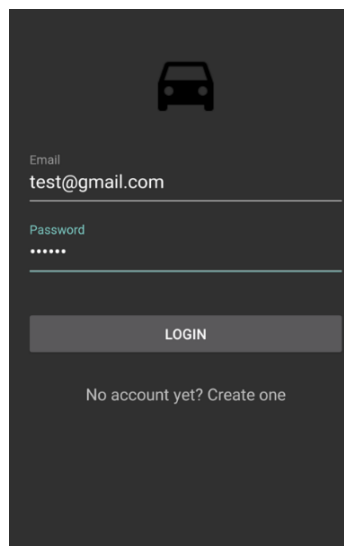
A screenshot of the login screen in a mobile app. The screen has a dark background. At the top, there's a car icon. Below it are two input fields: "Email" with the text "test@gmail.com" and "Password" with masked characters ".....". Below these fields is a button labeled "LOGIN". At the bottom, there's a link that says "No account yet? Create one".

Figure 32 - Login Activity

Users also can edit their account when they login the app successfully by “Settings” function in menu on the left side of the device screen.

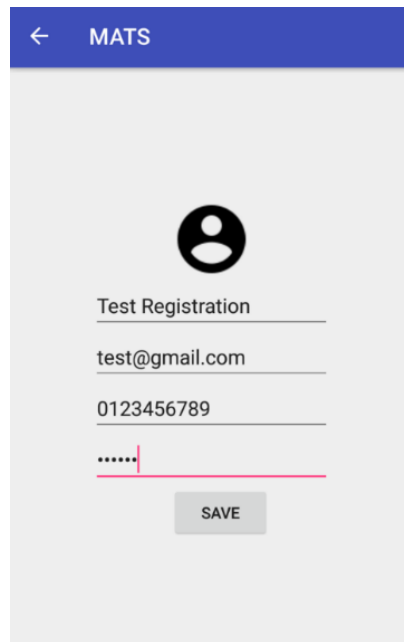


Figure 33 - Edit Account

4.2 Main Activity

After login successfully, the system will move to the main activity, in this activity, passenger user can use some main functions such as call a taxi, find destination, estimate time, fare. To use these functions, the first thing passenger has to do is open the phone location for the app can get position of user. If phone location does not open, the app will request user to open it.

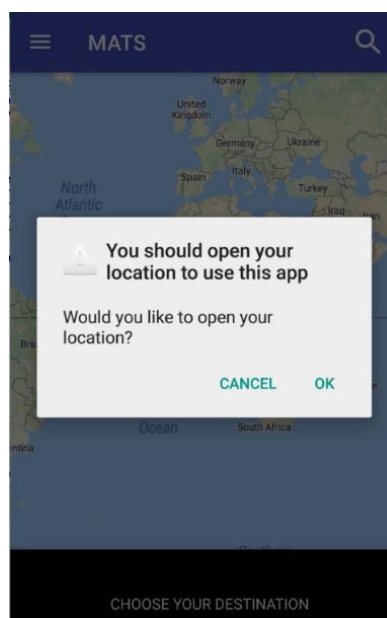


Figure 34 - Phone Location Opening Request

And after open phone location, user can access to main activity of the app and can see the main layout of the app.

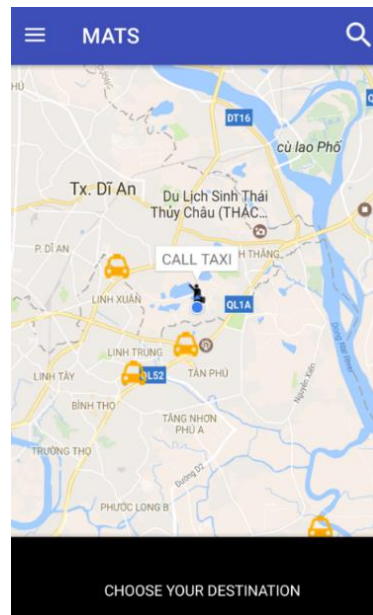


Figure 35 - Main Layout

In the Figure 35, passenger current location (📍) is at International University with latitude 10.8777777 and longitude 106.7994082, this coordinates will be updated to database and send to drivers when the passenger use calling taxi function. And beside to showing the passenger location, all of taxis (🚕) around passenger in that area are also displayed.

To call a taxi, passenger have to follow two steps. First, choose the destination or pick-up place, this step is optional. Passengers slide up the bottom bar and a search bar will appear. Then they can click on that, type the destination to search bar and get the estimation.

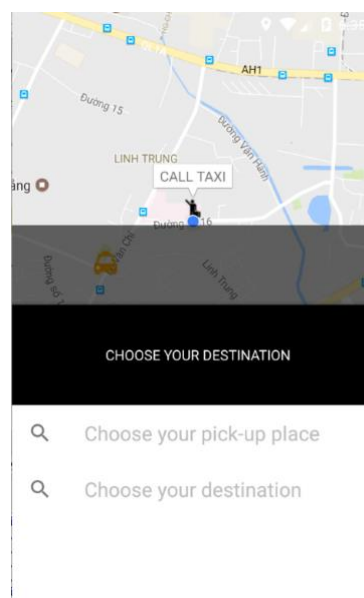


Figure 36 - Choose the Destination

A list of locations will appear to support passengers choose to enter destination or pick-up place when they click on the search bar.

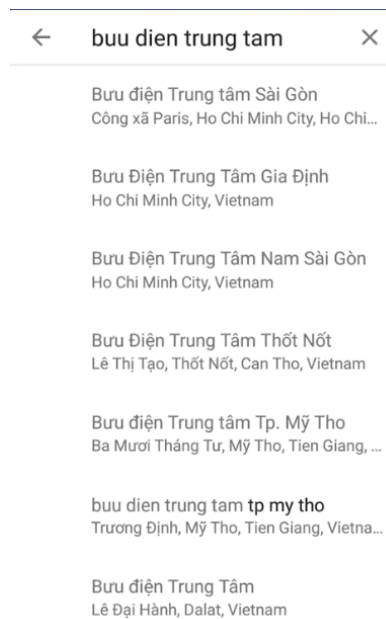


Figure 37 - List of Locations

When a destination is determined, information about passenger ride will be displayed on screen.

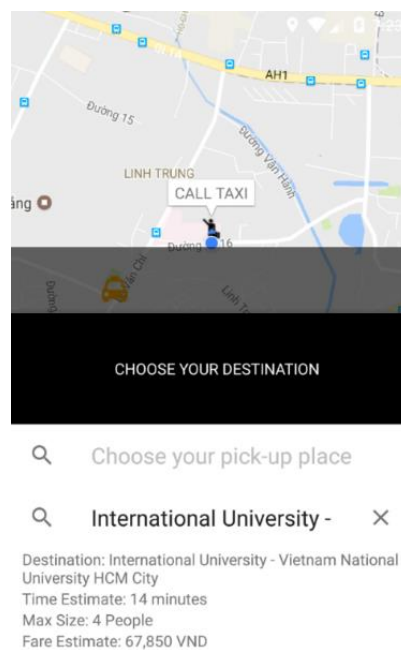


Figure 38 - Ride Estimation

If passenger choose pick-up place, information about the ride will be calculated again, and the passenger marker on map will move to the position of pick-up place.

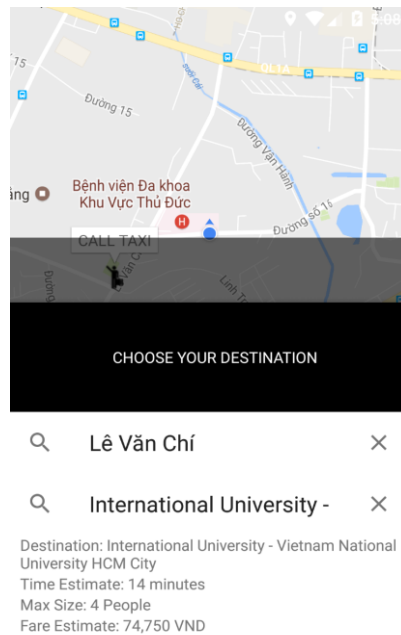


Figure 39 - Ride Pick-up Place

Besides choosing destination, passenger also can choose pick-up place by the same way. The “Time Estimate” value can be collected from Google Direction API, and we have a calculation to calculate the “Fare Estimate” value. A fare on one trip is calculate by the sum of time taxi in minutes come to passenger multiplied by 600 VND/MIN and the distance from passenger location to destination location multiplied by 11,500 VND/KM. That calculation rely on fare calculation of Uber (This fare is fare of Uber in Vietnam and rely on the fare shown on the Uber app).

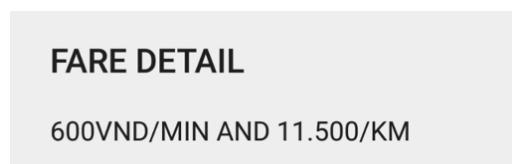


Figure 40 - Uber Fare Calculation

Then after chose a destination, the next step is click on “CALL TAXI” button to send request to taxi driver. After that, the nearest taxi will be chosen and the taxi driver will have information of passenger about location and phone number, then taxi driver can call their client and come to passenger position. The shortest path to passenger destination is also displayed for passenger can control the ride.

Passenger can cancel trip by click on “CANCEL” button and then click “Yes” to accept the question to cancel a trip.

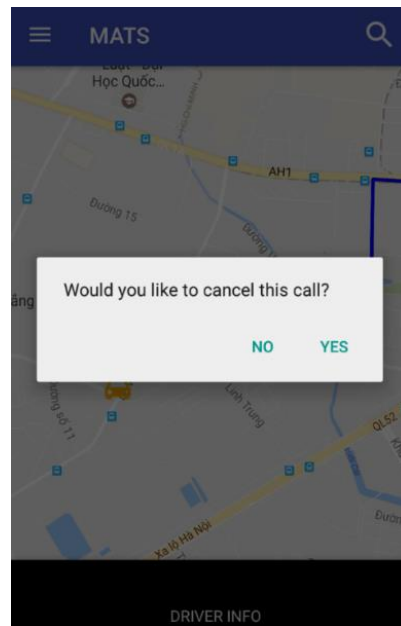


Figure 43 - Cancel a Trip

Or Taxi will continue come to passenger. And when taxi is at pick-up place, passenger will receive a notification to start the trip. If passenger want to start this trip, the passenger can click on “OK” to agree with the question.

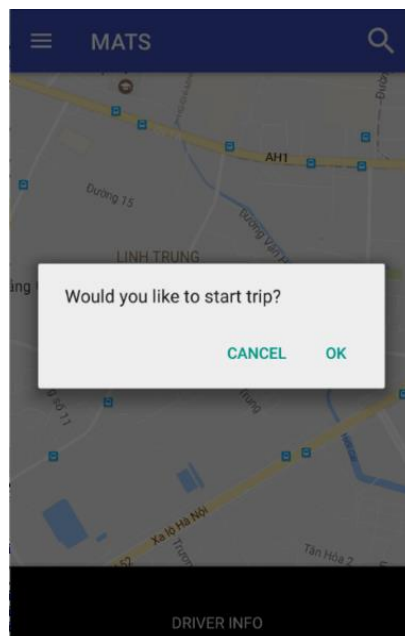


Figure 44 - Start Trip Notification

Taxi will take passenger to their place and when the trip finish, a notification is send to passenger to rate taxi driver. Passengers can choose number of stars as point for driver if they satisfy or not do not satisfy with their trip.

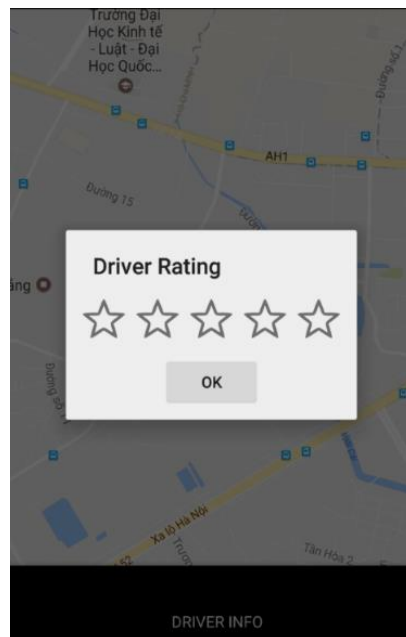


Figure 45 - Taxi Driver Rating When the Trip Finish

When passengers would like to see all trip that they called in the past, the passengers can view all their previous trips by click on “History” in menu on the left side of the app.

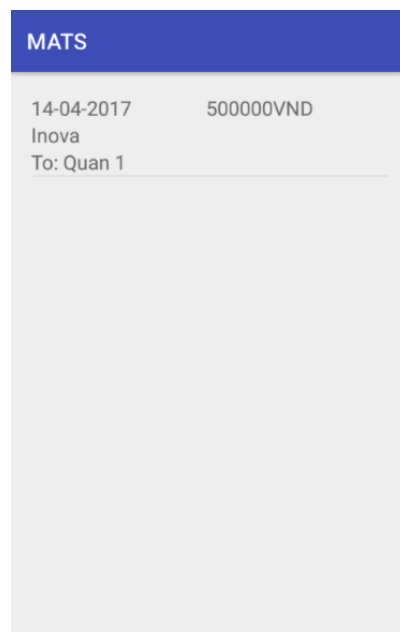


Figure 46 - Trip History

And to view the detail of each trips, passengers can click on the trip they want to see detail.

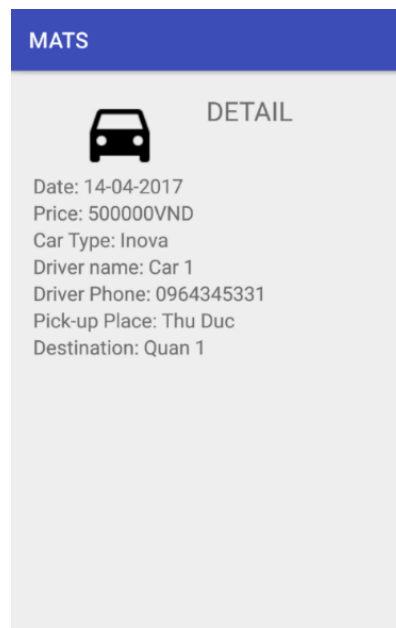


Figure 47 - Trip History Detail

4.3 Driver Application

To show the working of the app on server and service side, I built an application for drivers called MATS Driver. This application will show activities of the service side when passengers click on “CALL TAXI” button to call a taxi.

First, driver have to open GPS to let the app identify position of driver and driver will receive a notification to open GPS if GPS on the device is not active.

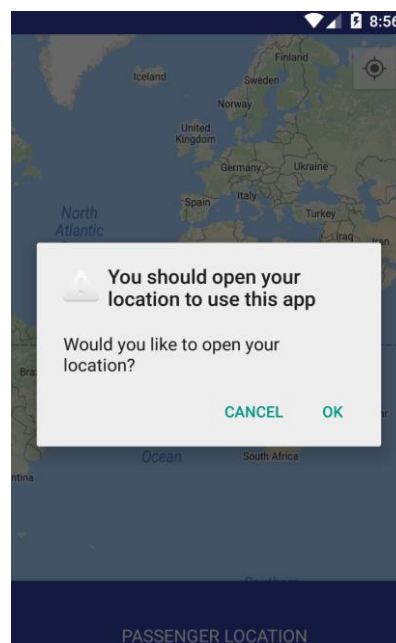


Figure 48 - Notification to Open GPS on MATS Driver

When driver open the app and GPS of device is ready, the app will request driver enter email and password to login to use the app.

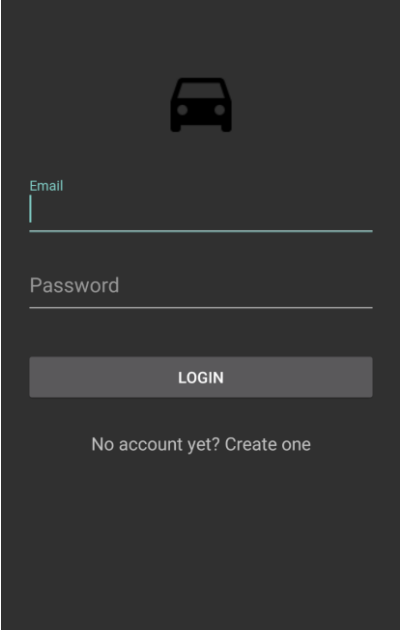
The image shows a login screen for a driver app. At the top, there is a dark grey header with a white car icon. Below the header, there are two input fields: 'Email' and 'Password'. The 'Email' field has a light blue cursor. Below the input fields is a dark grey button with the text 'LOGIN' in white. At the bottom, there is a link that says 'No account yet? Create one'.

Figure 49 - Login for Driver App

If driver do not have account, driver can register an account. In the app for drivers, users also fill the information such as name, email, phone, password, the different information drivers have to fill is type of car that drivers use.

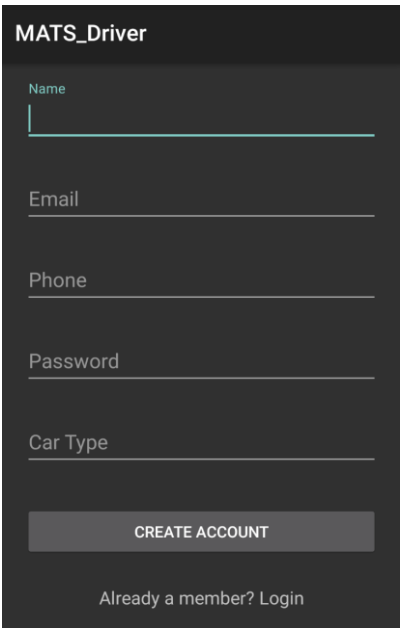
The image shows a registration screen for a driver app. At the top, there is a dark grey header with the text 'MATS_Driver' in white. Below the header, there are five input fields: 'Name', 'Email', 'Phone', 'Password', and 'Car Type'. The 'Name' field has a light blue cursor. Below the input fields is a dark grey button with the text 'CREATE ACCOUNT' in white. At the bottom, there is a link that says 'Already a member? Login'.

Figure 50 - Registration for Driver App

After login, driver can open main layout of the driver app, and use the app. On the right top corner of the screen, driver can see “Edit Account” icon and “Logout” icon.

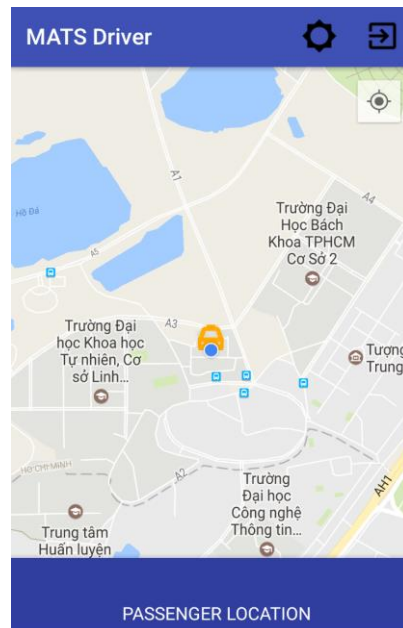


Figure 51 - Driver's Position on Main Layout of MATS Driver

By click on “Edit Account” icon (the left icon on the right top corner of the screen), Drivers also can change the information of their account as the passenger app.

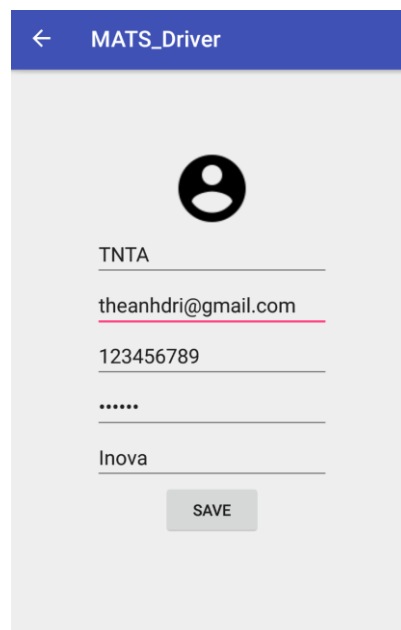


Figure 52 - Edit Account for Driver App

If there is any passenger call a taxi, and if driver is the nearest driver to the passenger, driver will receive a notification.

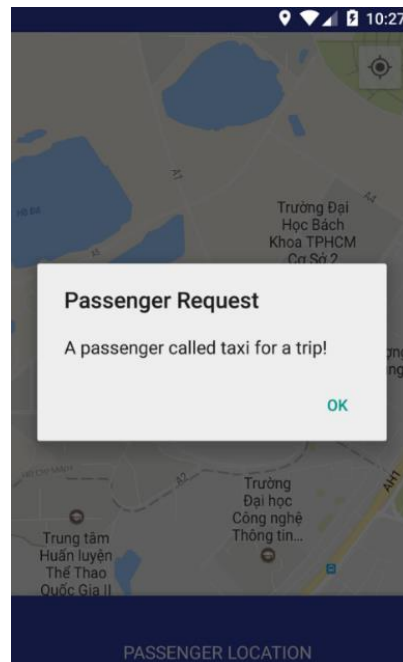


Figure 53 - New Taxi Call Notification for Driver

Driver can not cancel a taxi call, so driver have to click “OK”, after that the driver will see position of passenger who called taxi.

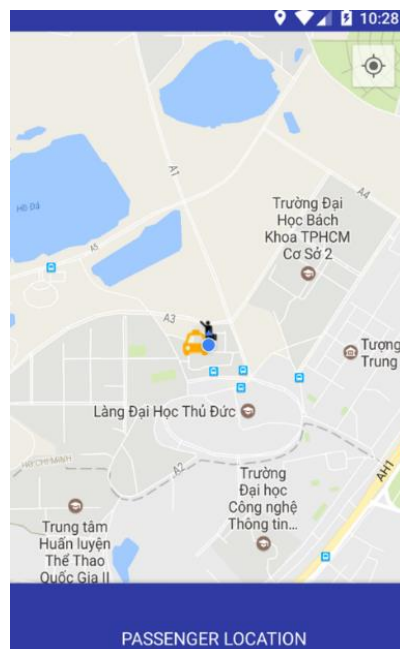


Figure 54 - Passenger's Position on MATS Driver

Driver also can know detail information of passenger by sliding up from the bottom of device screen.

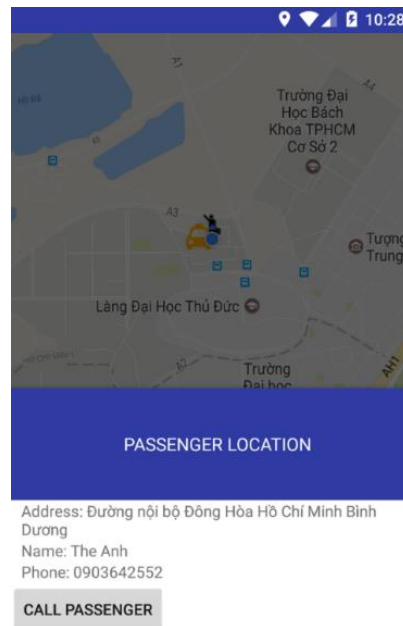


Figure 55 - Passenger Detail Information on MATS Driver

To contact to passenger to confirm the trip, driver can click on “CALL PASSENGER” button on device screen blow passenger’s phone number.

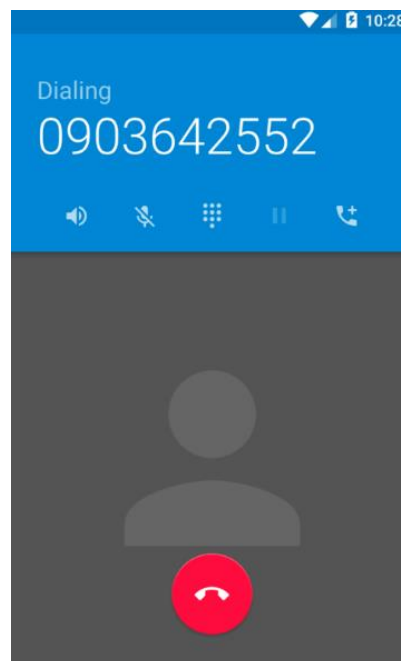


Figure 56 - Driver Can Make a Phone Call to Passenger

Chapter 5 . Conclusion and Future Work

5.1 Conclusion

The objective of this thesis is research and implement an Android taxi service demo app. With support from Google Maps API, we can load map from Google, edit many functions on map and find out the shortest way. In this thesis, I have completed to load map on mobile screen, connect to MySQL for users to control their account, get the current location of passenger and all taxis around passenger, find shortest path to destination expected by user and identify the nearest taxi to request taxi service for the passenger. While building app duration, I realized there are three things should be considered. First, to implement all functions concern to Google Maps API, we have to create an API Key, API Key can be applied for all Map Service of Google. Second, some functions such as connect to MySQL Server or request HTTP Service, we have to execute them on new thread. And finally, in Android Studio, to load map on real devices beside virtual devices, API Key must be added to both two files “google_maps_api.xml” on debug and release version. Building a mobile application app concern to map is not a too difficult, but it would be take time to research some tasks and test the accuracy of the app in real life.

5.2 Future Work

In the future, there may be many things to improve to build this app completely. First, about the motion of taxi on screen, I will research the way to make the animation to be more flexible and all of taxi can move at the same time without affect to the performance of the app. Then, the app will have more functions. Passengers do not only call taxi, but they also can call other vehicles such as motorbikes or coaches. I will also update function for passengers to pay by ATM by using Braintree applied to the app. Features to make a call to taxi driver and cancel the called ride will also be researched and implemented. And finally, the source code will be optimized to increase speed of the system. At present, there is not information or accurate evidence about algorithm that used by Google to find out the shortest path, therefore, in the future, I will try to research and update from many forums to know what the algorithm is.

References:

[1]: Uber History

<https://www.uber.com/en-VN/our-story/>

[2]: Three basic types of mobile apps

<http://blog.omnipress.com/2012/11/basic-types-mobile-event-apps/>

[3]: Android Operating System

[https://en.wikipedia.org/wiki/Android_\(operating_system\)](https://en.wikipedia.org/wiki/Android_(operating_system))

[4]: Popularity of Android

<http://www.androidauthority.com/develop-apps-for-android-rather-than-ios-607219/>

[5]: Popularity of Google Maps

<https://www.programmableweb.com/news/top-10-mapping-apis-google-maps-microsoft-bing-maps-and-mapquest/analysis/2015/02/23>

[6]: Two services from Google Maps for Android

<https://developers.google.com/maps/android/?hl=vi>

[7]: Web services of Google Maps API

<https://developers.google.com/maps/web-services/?hl=vi>

[8]: Using web services

<https://developers.google.com/maps/documentation/directions/web-service-best-practices>

[9]: Google Developers Console Website

<https://console.developers.google.com>

[10]: Example link using Google Web Services

https://maps.googleapis.com/maps/api/directions/json?origin=10.8777,106.8013833&destination=10.782557299999997,106.70536949999999&key=AIzaSyCQFAyfzpvU7j5Kz7kyXiPEBRhe4_61kgo

[11]: Technology behind Uber

<https://www.quora.com/What-is-the-technology-stack-behind-Uber>

[12]: Features of Uber

<http://pixsterstudio.com/much-cost-build-app-like-uber-time-cost.html>

[13]: Grab History

[https://en.wikipedia.org/wiki/Grab_\(application\)](https://en.wikipedia.org/wiki/Grab_(application))

[14]: MySQL Definition

<http://www.freesevers.com/WebHosting101/WhatIsMySQL.html>

[15]: Android Studio

<https://android-developers.googleblog.com/2016/04/android-studio-2-0.html>

[16]: Programming Languages of Android

<http://www.androidauthority.com/want-develop-android-apps-languages-learn-391008/>

[17]: Example of JSON Data returned by Google

https://maps.googleapis.com/maps/api/directions/json?origin=10.8777767%2C106.8016017&destination=10.7725916%2C106.69803879999999&key=AIzaSyCQFAyfpvU7j5Kz7kyXiPEBRhe4_61kgo