OPPORTUNISTIC RIDE-SHARING USING GOOGLE MAPS ANALYTICS

BY

CHEONG CHERN SIAN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OFINFORMATION TECHNOLOGY (HONS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2019

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: ____

____

____

Academic Session: _____________

I ____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

	Verified by,
_________________________	_________________________
(Author's signature)	(Supervisor's signature)
Address:	
_________________________	
_________________________	_________________________
_________________________	Supervisor's name
Date: ____________________	**Date**: ____________________

OPPORTUNISTIC RIDE-SHARING USING GOOGLE MAPS ANALYTICS

By

Cheong Chern Sian

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2019

DECLARATION OF ORIGINALITY

I declare that this report entitled "**OPPORTUNISTIC RIDE-SHARING USING GOOGLE MAPS ANALYTICS**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature	:	_________________________
Name	:	_________________________
Date	:	_________________________

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr. Aun Yichiet who has given me all the support and guidance and this an opportunity to involve in a data analytics project. He had given me all the guidance when I faced problems during this project. It is my first step that will help me to establish a career in data analytics field. A million thanks to you.

Finally, I would like to appreciate my friends and family who had given all the encouragements and mental support during this project. These support and continuous encouragements helped throughout in the project.

ABSTRACT

This project is titled “Opportunistic Ride-Sharing using Google Maps Analytics”. This is an automated ride sharing model that can enhance the effective of the matchmaking of rides that populistically share similar itineraries and schedules. The GPS navigation in the smartphone is used as an indicator that able to track user visited location. User require to turn on the GPS Location services and mobile data in the smartphone and the system will able start to track and capture the visited location and departure date and time, the collected data will upload to the Firebase database. In traditional ride sharing or hailing, driver within some predefined area of internet are assign to rider based on proximity distance. Instead of rider driven recommendation, this project propose to matchmake driver to rider based on some common trajectory opportunistically. Firstly, a rider or driver pathing behaviour is modelled using HMM based on location into denned from an in-house location tracker app. Secondly, a. recommendation system is designed to infer on the pathing model to suggest a driver there share common destination to the rider. For end to end coverage, transfer learning is used on multi user pathing for transit aware recommendation. This automated ride sharing system is an user friendly system that user just required open the system one time and the system will keep tracking the location that you visit.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

GPS	Global Positioning System
CDR	Call Detail Record
LDA	Latent Dirichlet Allocation
B2B	Businesses to businesses
C2C	Customer to Customer
APP	Application
JSON	JavaScript Object Notation
CSV	Comma-separated values
GUI	Graphical user interface
2D	2-dimension
API	Application programming interface
iOS	iPhone OS
KNN	K-nearest neighbors
AWS	Amazon Web Services
WHO	World Health Organization
SDK	Software Development Kit

Chapter 1: Introduction

1.1 Problem Statement

Nowadays, there are some on-demand ride sharing commercial services are already in place such as Grab, Uber and Lyft. These services are smartphone application based that to match the drivers and users, thereby provide a taxi-like services into commuters at a fraction of the cost. The problem of the existing ride sharing services is the time consuming for matching drivers to users. The current ride sharing services model using a method that is assign the nearest driver to the user, if that is no driver nearby and the system need to take longer time to match the driver for the user, and again if there is more users request a ride in the same area and there is no drivers nearby, the time consuming will be more longer. Other than the time consuming, assigning the closest driver to the user will be a non-effective matching method. If user A request a ride from point A to point B, and the system just assign a driver A that is the closest to users A, but the destination of the driver A is not point B and the driver still need to pick up the users A to point B, in this situation will occur an in-effective matching case. As mention above, the current ride sharing services is a taxi-like services, which means user need to perform a book session to request a ride, user need to enter the time to pick-up, pick-up and drop-off point as a booking services then only request a ride to the system, this will be no different between ride sharing and taxi services. Hence, the current ride sharing services is not efficiency, effective and time consuming.

1.2 Project Background

Nowadays, Internet has revolutionized the computer and communication world like nothing before. Widely usage of Internet, evolution in network infrastructure and mobile devices that had exploit a new form of businesses. This phenomenon was made transactions among companies in the beginning of Internet access which exploit the electronic B2B system. In the recent years every people can use Internet connection by using the personal smartphones. Thus C2C electronic market platform has been emerge and widely used by worldwide customers such as Airbnb, Ebay, Gearcommons.

According to the (International Business Times 2017), for those large metropolitan areas such as Beijing (China), Gwalior and Patna (India) are the places where serious pollution that caused by vehicular traffic and industry, and the air pollution has raise to 20 times limit of the hazardous levels that had endorsed by the WHO. At the same time, traffic congestion also continuously rising and this issue will become a crucial factor that can harm the citizen's quality life. Ride sharing is one of the actions that can most benefit to reduce the urban traffic.

Therefore, the ride sharing services is growing rapidly in the market, here are some on-demand ride sharing commercial services are already in place such as Grab, Uber and Lyft. The reason of ride sharing service growth rapidly is because ride sharing can provide a discovery services for the both of the drivers and riders which have a similar destination and arrival point. This ride sharing services was allow drivers to share their extra vacant seats of their vehicles, this phenomenon is bringing a several of benefits for all the driver and people who don't have their own vehicle which can share travel costs and extended social circle.

However, the opportunistic of ride sharing services still exist in-efficiency performance when the matching process for the driver and user. Meanwhile, the most of the ride sharing services are using a taxi like services to match the driver with user, and this kind of method will affect the opportunistic of ride sharing that cannot optimize the best performance to match the driver and user who have the same itineraries and schedules.

1.3 Motivation

As mention the scenarios above, the current ride sharing services is a not effective, efficiency and time consuming. Thus, it will be a crucial to ensure the efficiency to match driver and user process and this is also an important to ensure the efficiency of matching process that able to match the suitable driver to the user. Let's having an assumption, user A is heading to point C from point A, at the same time if there is a driver A is also heading from point A to point C. Why not just match them together in a ride? This will able to solve the in-efficient matching method a provide more effective ride for both of driver and user. Therefore, to solve the problem stated above and enhance the efficiency, this project is being proposed to develop an opportunistic ride sharing using google maps analytics. This automated ride sharing system can match the driver and user who have the similar itineraries and schedules and reduce the time consume problem. This propose system is based on an opportunistic algorithm to analysis the user visited location in which day and time by using their smartphone's GPS and mobile data to collect the data and upload to the database. By the collected data, export the data can convert to csv file then, upload to the AWS SageMaker to train a model. This trained model will be used to run the predictions to determine the most suitable driver to the user.

1.4 Project Scope

In this project, an opportunistic ride sharing using google maps analytics will be develop a system to solve the problem statement that mention in section 1.1, which is to improve the efficiency, accuracy and effective during the matching process. To develop this system, smartphone is needed in this project, turn on the GPS location services and mobile data, the system will start to track and capture the location data. The location data will process by the main program and the data will be stored in the database, which is Firebase database. These location data will proceed to the next step to perform the data processing to do the data classification and data labeling then export the dataset into a CSV dataset. These CSV datasets will upload to the Amazon SageMaker and do the model training, when the model training job is done the model will hosted as an endpoint at the SageMaker and ready to use for predictions job. The model will use to prediction the which day, what time and where the driver will go. Then base on the prediction then it will suggest a most compatible driver to the user.

However, there is a difficulty to collect the data in a large population within a limited time. In this project, we will shrink the population area to Kampar area, and pick a student to collect the data in every day. Student needed to register an account in the mobile application, an user id will be an argument for data collection. In this proposed system, the GPS location services and mobile data are required to keep turn on to ensure the accuracy and reliability of the data collection.

1.5 Project Objective

The main objective of this project:

- **To develop a pervasive location tracker for unsupervised data collection.**

The mobile application is provided a GUI interface that is more user friendly for users. This mobile application is able to automatic collect the data without any human intervention when the GPS location services and mobile data in the smartphone is turning on. Before the user start to use this application, user are required to register an account for the authentication section in the beginning stage in the application. When user successfully registered an account the user will get an unique user id as a validation tools for user identity. Every time the data collection, the data will automatically upload and store to Firebase real time database.

- **To model driver or rider trajectory behavior using HMM on AWS SageMaker.**

By using the collected data, these data will be used to train a model of driver or rider to identify the trajectory behavior in daily life. This model will help the system to trace the similar itineraries and schedules. The model is used the collected data and make a probability to forecast the outcomes. When the model is made, the variables that will be likely to influence the future output.

- **To design an end-to-end path discovery for opportunistic transit using time slicing and compensation.**

This end-to-end path discovery is designed to analysis the opportunities of the transit that will using time slicing method and compensation way. The time slicing method will be used because it can reduce the margin of the time to track the transit opportunities, then it can be province the time during the matchmaking. While the compensation also can lead the opportunistic of the transit rapidly, it can offer both of the driver or rider with some benefits that can help to attract drive to provide a ride share, at the same time rider will have more opportunities to get the ride share services.

1.6 Report Organization

This report has consisted 6 sections. In the Chapter 1, the problem statements, project background, motivation, project scope and objectives as the project contribution will be discussed. For the Chapter 2 will discussed the existing work and the comparison of existing work with the proposed system. In the Chapter 3 will discussed the system design. Next, the methodology and tools, system requirement will be discussed in Chapter 4. For the Chapter 5, the implementation and testing will be discussed. Lastly, The Chapter 6 as a conclusion of the project, limitation and future work of the project.

Chapter 2: Literature Review

2.1 Review of existing works

vHike Ride-Sharing Service for Smartphones

In this paper (Stach & Brodt 2011) proposed a vHike, an user friendly dynamic ride-sharing system that can be used in smartphone. Web 2.0 social discomfort and networks emerged by ride sharing were used in this vHike. This system used Bluetooth to locate each other within a limited range that up to 100 meters then pair to another Bluetooth device either synchronous traffic or asynchronous data communication. GPS also used in this system to determine both of the driver and the current position of the rider by the signal that timing and redirect by the satellites.

This system required 4 non-functional requirements which are community-based, smartphone-based, trust-based and web-based. The vHike used community-based to close the community of registered User exclusively, it was help to balance between the supply and demand. Moreover, vHike initially implement the service for Android-based smartphones, so it can handle the service in easy and fast that can minimize the inconveniences. While vHike had implement the trust-based into the system that only available to the close community of the members of vHike, this implement offer an emergency button to trigger for an emergency call immediately. Moreover, the vHike had made a rating system that is similar to the Amazon Marketplace rating system. Thus, the user can vote and write their opinion with a short description of comment of the ride. Then the following the rating result will be recorded to ensure the ride is closed and then a new ride can start again. In the project also provides a choice to organize the ride whether for long-term ride or periodic ride. Hence, each of the user can offer a ride to can get the intention in the Web-based forum that approachable in the registration page

Limitation and weakness

- Bluetooth is not capable of locating other in a longer and larger range above 100 meters.
- High energy consumption for smartphone.
- Inflexible searching technique as it only can search nearest vehicle instead of travel the same destination point.

Campus Ride Sharing Platform for Academic Institution

(Chowdhury et al. 2016) proposed a campus ride sharing platform that implemented Proximity Search panel, Ride List (Circle Priority), Rating Option panel and Notification Panel. The proximity search panel is allowed student to select a departure or arrival point when search for a ride. For the Ride List, user is used to add a private ride that only visible for user in their circle list, they also can see both of the public and the private post when the user are inside the circle list. Rating Option panel of this system can let user to rate the driver and this rating will save into the involved drivers, then others user is able to decide to pick up a ride share offer a ride. Moreover, the notification panel is the initial launch page in the system, and user click the notification button, a notification will pop in the chart, and user can decide to accept or decline the offers. There is one more Suggestion list panel in their platform that suggest a ride for user when both of them have a same destination or home location.
For this system, user was required to use E-mail address that under University or ID and password to login to this platform. Thus, the student safety had been protected, because the outsider has no other ways to use this ride sharing system.

Weaknesses of this system:

- Students will facing the time wasting issue if the drivers or passengers are late comers, because there are no GPS tracking in this platform.
- Real traditional way to find a ride share services.

A Dynamic Vehicle Pooling and Ride-Sharing System Framework

(Farin et al. 2016) proposed a dynamic vehicle pooling and ride sharing system framework that provided security, accessibility, and identification of user. Moreover their system had combined with the Information Communication and Technology based solution in both of the security of the peculiar payment system and dynamic resource allocation for all kinds of vehicles.

In this system, they implemented and designed a dynamic system and this system is able to feasible on different types of vehicles. Therefore, user can send a request to the vehicle owner. Top-up and pre-paid method is implemented and allow user to make the payment through this system. Thus, passenger can pay to the driver with bank account or bKash account. During the transaction, the whole transaction process is under a safe condition and the process had been secured. Certainly, this system had an authentication system that required to login before proceed to request a ride share. This system validate user by the National Identification Number (NID) with two factor authentication key, which is cellular phone number must be provided. In addition, the searching technique was used GPS to track the user location and then the nearest car will suggest to the passenger. By measuring the Euclidean distance to determine the nearest vehicle.

Weaknesses of this system:

- Inflexible searching technique as it only can search nearest vehicle instead of travel the same destination point.
- Real traditional way to find a ride share services.

Opportunistic Ride Sharing via Whereabouts Analysis

According to (Bicocchim et al. 2015) proposed an opportunistic ride sharing by Whereabouts Analysis. Based on the analysis they used to extract the information from mobility traces that suitable for the project to identify the same path that manageable for ride sharing. They collect all the Call Detail Record (CDR) event, then the event will clustered to each other into coherent groups based on their specific geographical areas. A weight will replying on commonsense assumption to define a time window associated home or work place. Thus, the work place and home will be consider as the heaviest cluster of the center which identify by Thresholding.
A work-home was generated by the topic modeling technique. This topic modeling is used to spot the user mobility routines and recurring whereabouts pattern to characterize the typical behavior of each user. A word will created as first and second cell for two location and a time stamp of transition hold the places. Thus, each of the word will keep as a guide to confirm the indicate both of the location and distance forced be street geology from the geolocation coordinates. Furthermore, by using a probability generator model which is call LDA that to cluster the data according to the term pattern that contain by each of them. In addition, they implement the library Mallet toperform the computation in there model. Once the parameter of the model was found, Bayesian will proceed subtraction to extract the topics that is the best to describe the routines of given day, and this the model that they apply in novel way and spot the foundation of the information that in the opportunities of ride sharing.

Weaknesses of this system:

- Difficult to constraint the size of cluster that to be identified.
- Need to depend a complicated mechanism to compel the cluster size.
- The ground-truth information is not accuracy.

Smart Ride Share with Flexible Route Matching

(Chen, Shallcross et al. 2011) proposed a Smart Ride Share system that adopted ride-share scheduling algorithm that combined the greedy method and the bipartite matching algorithms to reduce the quantity of vehicles needed and the total trip distance for an input set of dispatch tasks.

The algorithms introduced in the project are simple O(n2m) algorithm and the O(n((m+n) log n)) algorithm which is more complicated, where the n is the nodes number, and the m is the edges number. They took Kuhn attributed algorithm which is a simple algorithm and the algorithm also similar to the algorithm of maximal cardinality bipartite matching. Thus, the maximal cardinality bipartite matching algorithm run in first time is to get k, which is the minimum of vehicles that needed during the implementation. Afterward, it will run the maximum-weight matching algorithm second time that is using k as an input to determine the lowest trip distance with the schedule among all applicants with size k.

In the previous case, they proposed a 2-phase approach solution which is more effective in practice of ride sharing scheduling. Greedy method is used in first phase to identify the tasks that can be shared in the same ride and then split them into a new task. Then the new set of task will be the input of the bipartite algorithm as discussed before to generate the final scheduling. Assumption will made for each vehicle that only can carry two passengers at the same time, then the assumption can lighten in out algorithm. Then greedy method will integrate again new task A and B to new one, the extra delay will affected by either passenger due to exceed a pre-defined system threshold.

Weaknesses of this system:

- Longer time consume than other algorithms for big case of problem, because it’s just search the best choice to reach from the current state.
- Parameters is very sensitive in changes.

2.2 Comparison. Of Existing Works with Proposed System

Existing Systems	Accuracy to locate driver	Flexibility	Time Consuming for matching
vHike – Dynamic Ride-Sharing Service for Smartphones	Low	✗	✓
Campus Ride Sharing System	Low	✗	✓
Dynamic Vehicle Pooling and Ride-Sharing System	Moderate	✗	✗
Whereabouts Analysis	High	✓	✗
Flexible Route Matching	Moderate	✓	✓
Opportunistic Ride-Sharing Using Google Maps Analytics	High	✓	✗

Table 2-1 Comparison of Existing Works with the proposed system

The relate works are compared in terms of the accuracy tracking, flexibility and time consuming for searching and matching a ride share. Firstly, the vHike – Dynamic Ride-Sharing Services for smartphones has a less accuracy to locate each other, due to the Bluetooth technique was used the system only capable locate others within a limited range that only up to 100 meters. However, the arrangement of the vHike system via Bluetooth Chat, user still need to find and start a chart with the driver to confirm the ride, then the system is an inflexibility searching technique in ride sharing. Due the inflexibility searching technique, the time consuming will become longer to search a ride share.

While Campus Ride Sharing Platform is also have a low accuracy to locate the driver for a rider, because this system was used the post method to search a driver for a ride, when a passenger add a new post for a ride, user need to wait others to check and accept their request. Therefore, this method will affected the flexibility for searching process in the system. Moreover, the time consume for searching a ride and wait for driver to accept will required more time to complete a ride, then time consuming will belonger. (Farin et al. 2016) proposed the framework for the system, the accuracy is better compare to vHike and Campus Ride Sharing Platform, by using GPS and implementation Euclidean algorithm will suggest the nearest driver for the user who request a ride. Although the accuracy had improve compare with previous system, but the system still inflexibility, because the system just only suggest the nearest driver to passenger. What if the driver has no same destination or pass by path with passenger? So this situation will affect the flexibility of the system. This system has no time consuming issues, due the nearest driver will suggest to passenger, then the time for searching process will be more effective.

The Whereabouts Analysis has a low accuracy because of it lack of accuracy ground-truth information. However, by the methodology they used is automatically to get the flexibility traces information that to determine ride sharing opportunities, the system is become more flexible to use. While a good searching and extract method, this system has perform well in time consuming for matching.

Smart Ride system has a moderate accuracy to locate a driver for ride is depending the number of the vehicle. The bipartite matching algorithms and greedy method implemented in this system to reduce the number of vehicles needed and generate the flexible route matching. In addition, the algorithm used in this system will take longer time consume when the case is getting bigger.

For our proposed system, opportunistic ride-sharing using Google Maps Analytics can outperform than other systems in terms of the accuracy tracking, flexibility and time consuming for searching and matching a ride share. In our proposed system will works by Google Maps Analytics to search and identify a ride share services for a use, user can login to mobile application or web service to search a ride for their travel. Furthermore, user can travel to final destination with multiple point with different driver when the travel distance is long. By this implement this methodology can increase the accuracy to locate a driver for a ride, flexibility to searching andmatching in a system. Thus the time consuming for searching and matching will decrease drastically.

Chapter 3: System Design

3.1 Overview of System Design

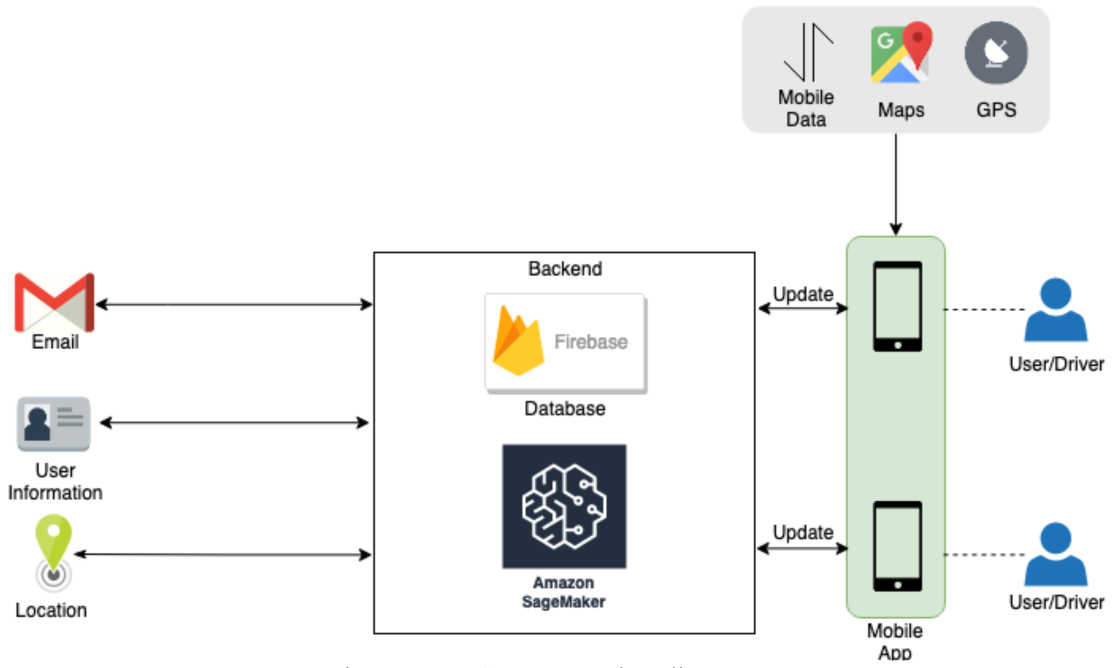


Figure 3.1.1 System Design diagram

The system will be implemented by smartphone. The mobile app will be installed in the smartphone. While user start to use the mobile app, it will start to track and collect the location data and the collected data will store into the database. Whenever the data had been collected and store in database, the data will export to JSON file, then a program will run to process the data then convert the JSON file into CSV file format. Then the converted CSV file will upload to SageMaker and perform the job training, when the job training is successful, a model will be created and deploy on the SageMaker hosting services. Lastly, these hosted models will ready for call to run the predictions.

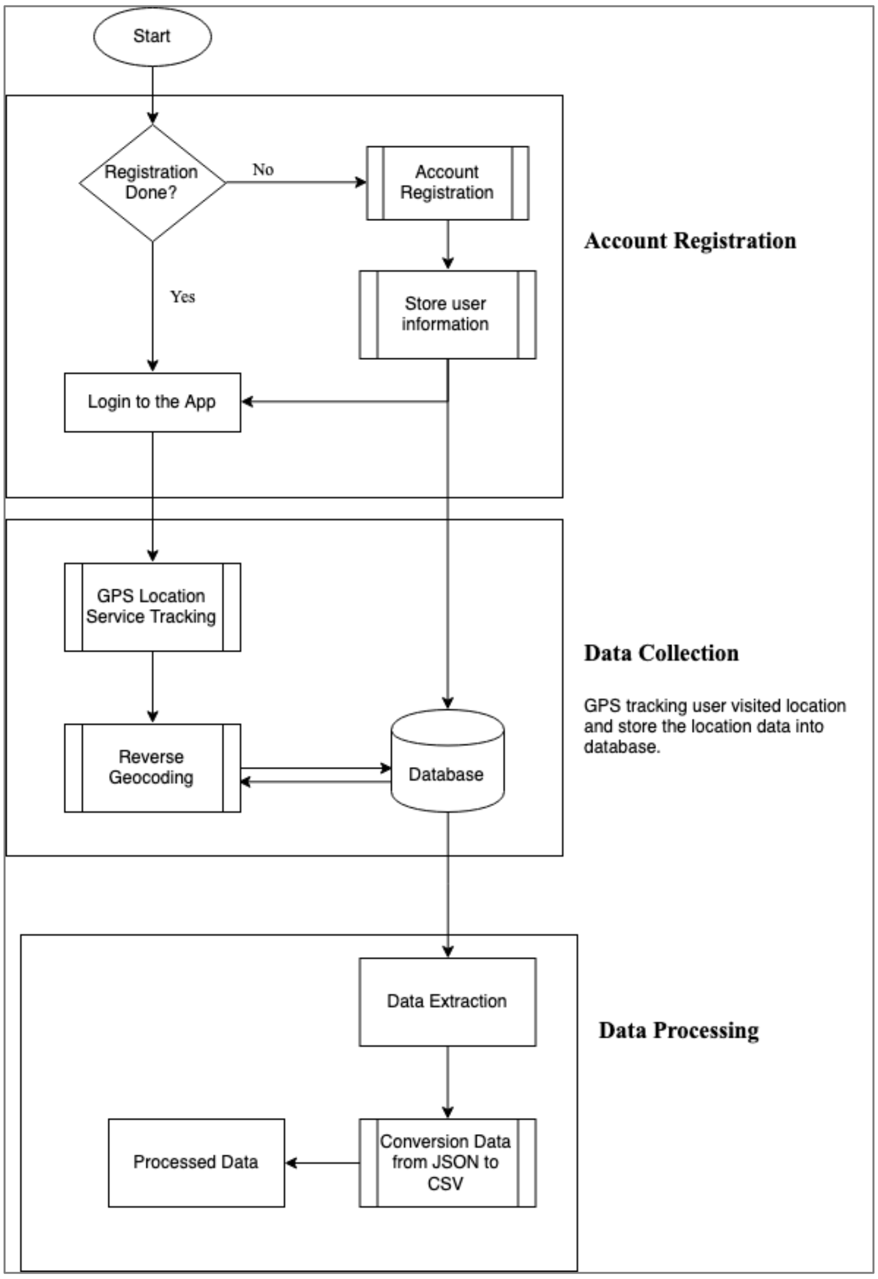


Figure 3.1.2 Location Tracker Flowchart

From the figure above show the flow of the location tracker system. Before the system begins to capture data, user are required to proceed for an account registration that to verify the identity. If the registration is done, user will login to the system and it will automatically start to track the user visited location. Once the visited location had been tracked, the reverse geocoding will reverse the 2D coordinates location into a normal location point. Afterward, the location will just upload and store into database. The GPS location tracking services will keep tracking in the location. Lastly, the collected data will export from the database into JSON file, it will proceed to extract the data and start to convert the data from JSON to CSV format. Due to the limited of the input data type for attributes of the Amazon Machine Learning, data labeling was to use label the data from string into numeric during the conversion of from JSON CSV file format.

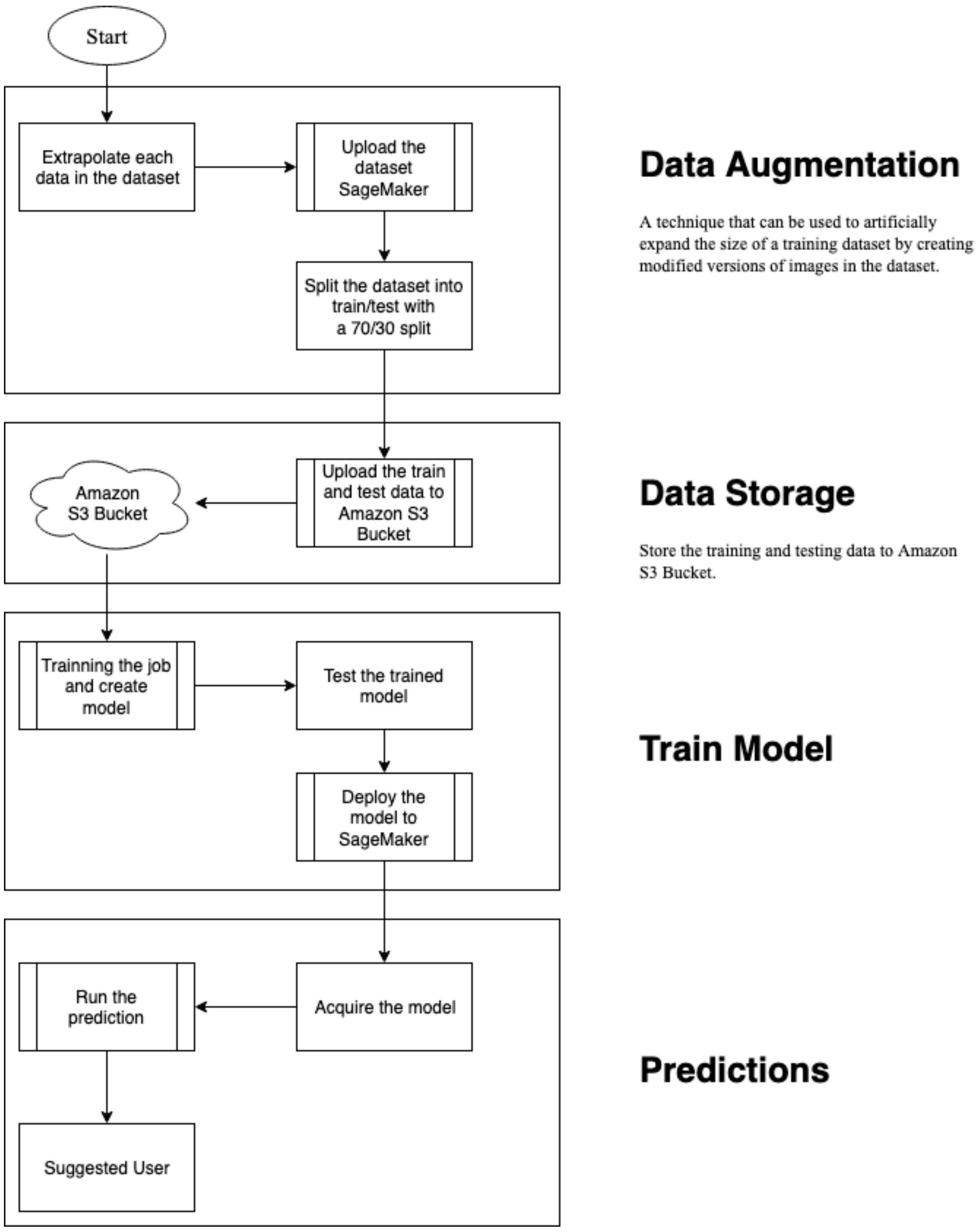

Start
Extrapolate each data in the dataset
Upload the dataset SageMaker
Split the dataset into train/test with a 70/30 split
Data Augmentation
A technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.
Amazon S3 Bucket
Upload the train and test data to Amazon S3 Bucket
Data Storage
Store the training and testing data to Amazon S3 Bucket.
Trainning the job and create model
Test the trained model
Deploy the model to SageMaker
Train Model
Run the prediction
Acquire the model
Suggested User
Predictions

Figure 3.1.3 Predictions Flow chat

From the figure 3.1.3 show the flow of the predictions system. At the beginning step of the predictions system, the data augmentation will do, in which to extrapolate each of the data in dataset that to extend a known sequence data that able to helps to increase the accuracy of the model and predict the future. After the done the data extrapolation, these datasets will upload to the SageMaker. Next the dataset will split the into train and test with the ratio of 70/30 split, then these data will upload to the Amazon S3 Bucket that ready to use for job training. After done the uploading, these data will be used and start the job training process and create into a model, there will be a testing process after done the model training that to validate the accuracy of the model. Then an endpoint will be created and deploy the model to the SageMaker hosting site. By acquiring the endpoint from the SageMaker, the system is able to run the prediction to get the result of the user, and determine which user will be suggested.

3.2 Main Program

There will be three part of main program, which is data collection program, data processing program, model training and building, and the last is the prediction program.

3.3 Data Collection Program

The location tracker is developed in Swift programming language. This location tracker is use to collect the location of user with the specific day and time by using their GPS of their smartphone. This program will be used in the beginning of the project.

3.3.1 User Registration

This algorithm is use to allow user to create an account and bind a unique user id to every user with their account, and obtain the user information then store into the Firebase Database that can use for future information tracking. In the account registration period, this is the first step that user need to proceed. If user doesn't have account of this mobile app, then user required to register an account before login to the mobile app.

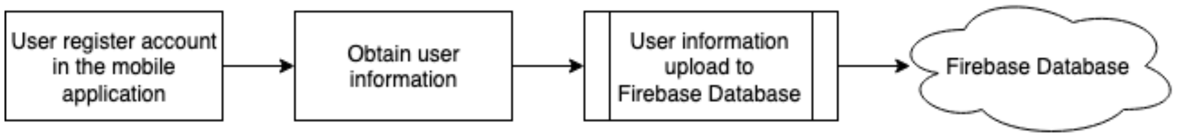

User register account in the mobile application
Obtain user information
User information upload to Firebase Database
Firebase Database

Figure 3.3.1.1 Flow of User Registration Algorithm

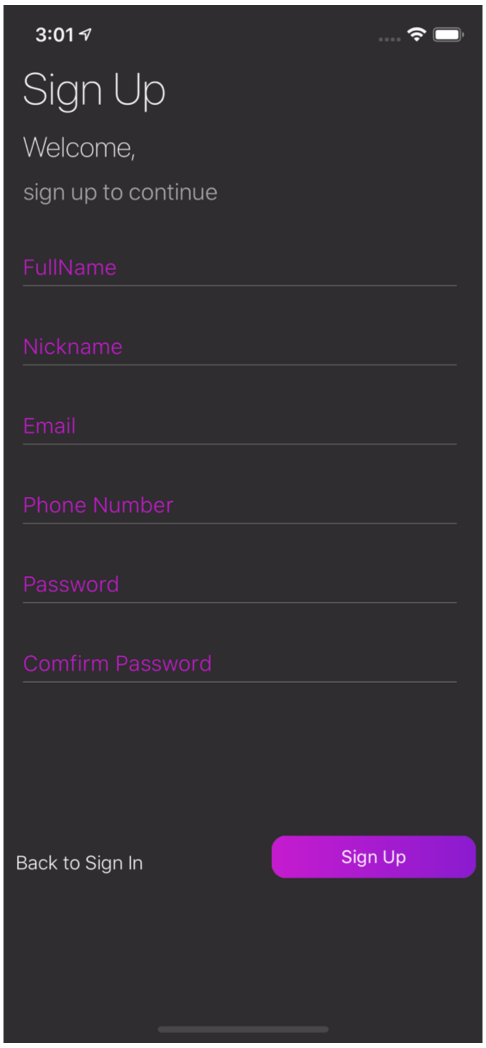
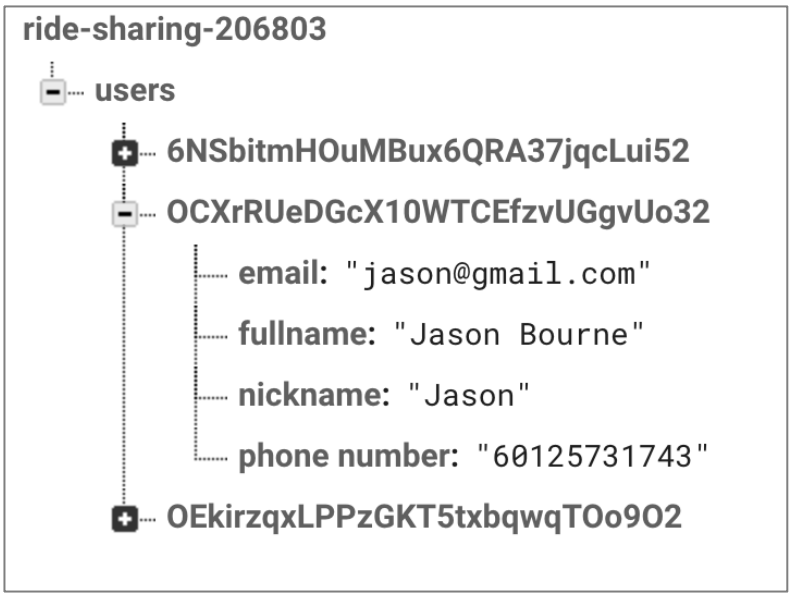


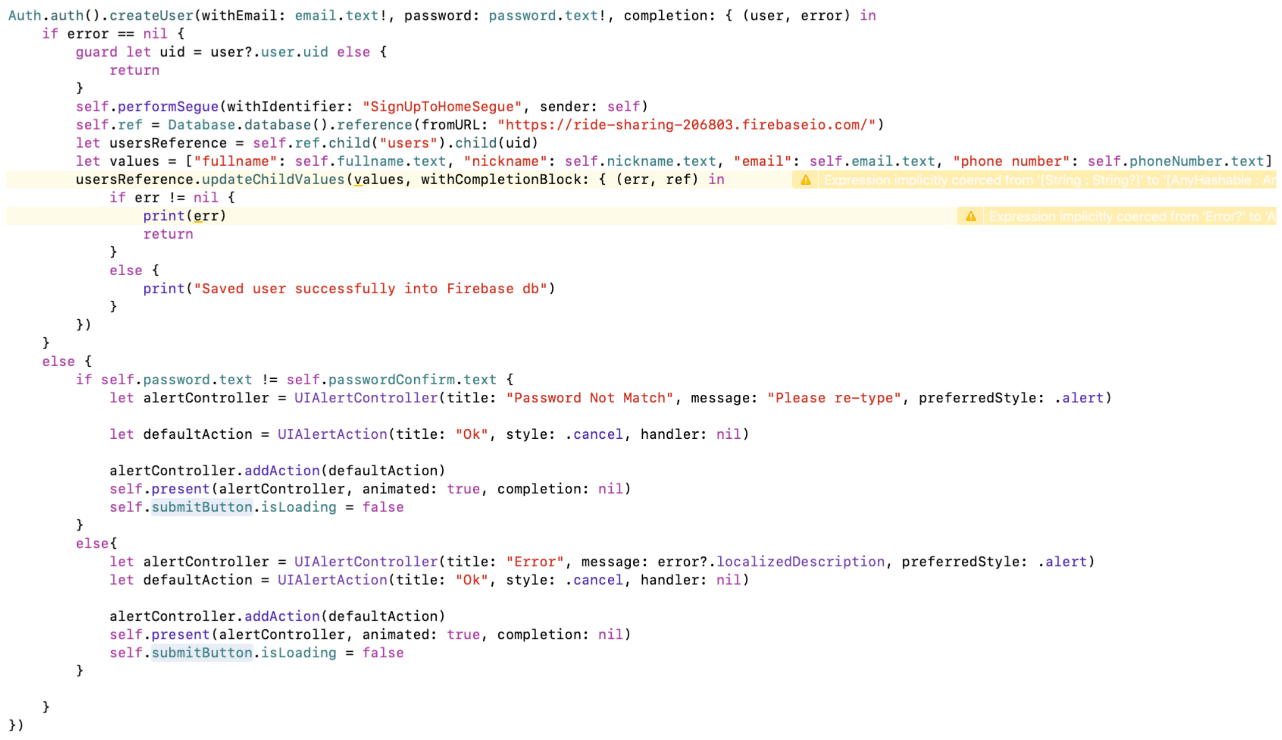
Figure 3.3.1.2 Sign up page of the location tracker app

The figure above shows the interface of the mobile application for user registration part, this design is user friendly that allow users to fill their information. User are required fill up all the field start from full name, nickname, email, phone number and password. Then an unique user ID will assign to the user. When the user done the registration of the mobile app, the user information will store into the database which is Firebase realtime database and it will be logged into the mobile app home screen.



```
ride-sharing-206803
  users
    6NSbitmHOuMBux6QRA37jqcLui52
    OCXrRUeDGcX10WTCEfzvUGgvUo32
      email: "jason@gmail.com"
      fullname: "Jason Bourne"
      nickname: "Jason"
      phone number: "60125731743"
    OEkirzqxLPPzGKT5txbqwqTOo9O2
```

Figure 3.3.1.3 User information store in realtime database.



```
Auth.auth().createUser(withEmail: email.text!, password: password.text!, completion: { (user, error) in
    if error == nil {
        guard let uid = user?.user.uid else {
            return
        }
        self.performSegue(withIdentifier: "SignUpToHomeSegue", sender: self)
        self.ref = Database.database().reference(fromURL: "https://ride-sharing-206803.firebaseio.com/")
        let usersReference = self.ref.child("users").child(uid)
        let values = ["fullname": self.fullname.text, "nickname": self.nickname.text, "email": self.email.text, "phone number": self.phoneNumber.text]
        usersReference.updateChildValues(values, withCompletionBlock: { (err, ref) in        ⚠ Expression implicitly coerced from '[String : String?]' to '[AnyHashable : Ar
            if err != nil {
                print(err)                                                                    ⚠ Expression implicitly coerced from 'Error?' to 'A
                return
            }
            else {
                print("Saved user successfully into Firebase db")
            }
        })
    }
    else {
        if self.password.text != self.passwordConfirm.text {
            let alertController = UIAlertController(title: "Password Not Match", message: "Please re-type", preferredStyle: .alert)

            let defaultAction = UIAlertAction(title: "Ok", style: .cancel, handler: nil)

            alertController.addAction(defaultAction)
            self.present(alertController, animated: true, completion: nil)
            self.submitButton.isLoading = false
        }
        else{
            let alertController = UIAlertController(title: "Error", message: error?.localizedDescription, preferredStyle: .alert)
            let defaultAction = UIAlertAction(title: "Ok", style: .cancel, handler: nil)

            alertController.addAction(defaultAction)
            self.present(alertController, animated: true, completion: nil)
            self.submitButton.isLoading = false
        }

    }
})
```

Figure 3.3.1.4 User registration algorithm

From the source code that shows in figure 3.3.1.4 above, the algorithm will create an user account that use user email and password to login to the mobile application. The algorithm will point the Firebase Database by using URL as a reference that to save the user information later on. During the registration, if the email address is blank, password is blank and password mismatch then the algorithm will prompt an error message box to remind the user to insert again the correct value to the textbox. While there is no any error, an user account will created successfully and all the user information will upload and store into the Firebase Database, and it will automatically login into the location tracker application.

3.3.2 Location Data Collection

After done the registration part, the system will proceed to the next stage that is the data collection stage. During the location data collection process, user is required to turn on the GPS location services and mobile data in the smartphone. When the mobile app meets these requirements, the mobile app will run the GPS location tracking algorithm and start to capture the location data information, the algorithm will run as a background app in the smartphone to keep tracking the user visited location. If the mobile app had tracked a visited location coordinate, the reverse geocoding algorithm will process the coordinate into a location name. Afterward, this location data will straight away upload and store inside the database. User also able get to know the visited location in the mobile app. The figure 3.3.2.1 shows the interface of the main screen of the location tracker, and the GPS is in the tracking mode.

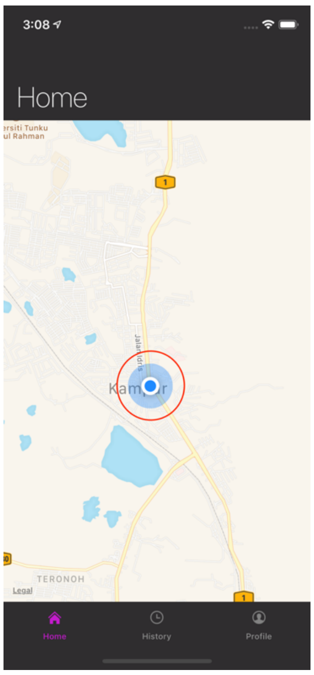
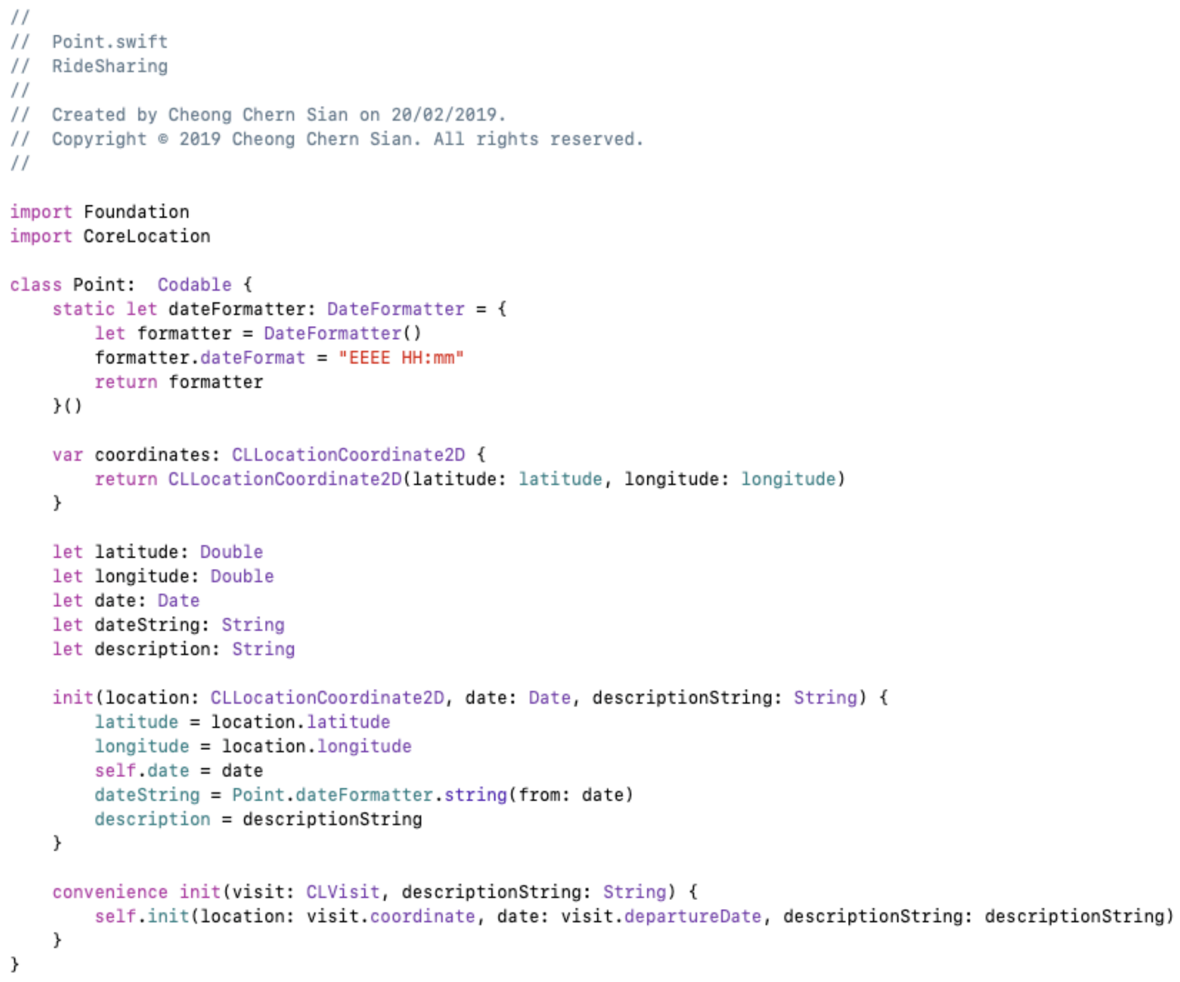


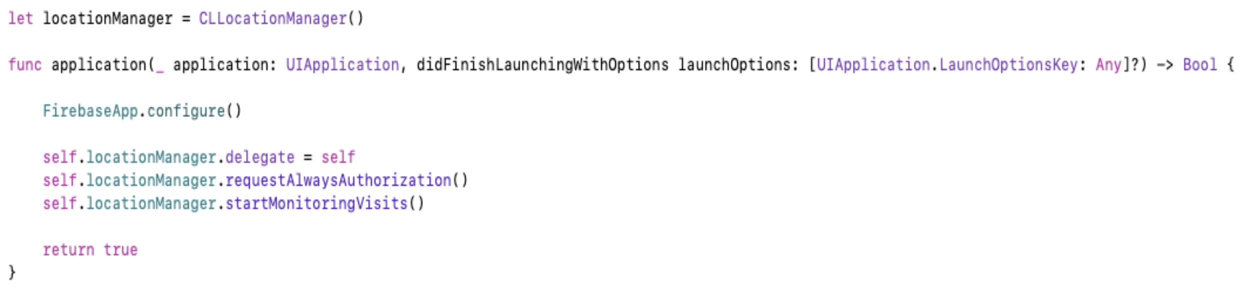
Figure 3.3.2.1 Red circle shows the GPS is in the tacking mode



```swift
//
//  Point.swift
//  RideSharing
//
//  Created by Cheong Chern Sian on 20/02/2019.
//  Copyright © 2019 Cheong Chern Sian. All rights reserved.
//

import Foundation
import CoreLocation

class Point:  Codable {
    static let dateFormatter: DateFormatter = {
        let formatter = DateFormatter()
        formatter.dateFormat = "EEEE HH:mm"
        return formatter
    }()

    var coordinates: CLLocationCoordinate2D {
        return CLLocationCoordinate2D(latitude: latitude, longitude: longitude)
    }

    let latitude: Double
    let longitude: Double
    let date: Date
    let dateString: String
    let description: String

    init(location: CLLocationCoordinate2D, date: Date, descriptionString: String) {
        latitude = location.latitude
        longitude = location.longitude
        self.date = date
        dateString = Point.dateFormatter.string(from: date)
        description = descriptionString
    }

    convenience init(visit: CLVisit, descriptionString: String) {
        self.init(location: visit.coordinate, date: visit.departureDate, descriptionString: descriptionString)
    }
}
```

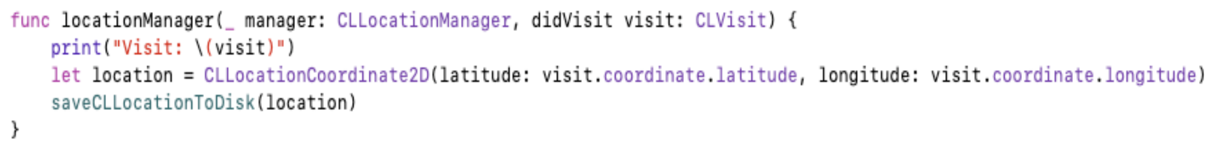
Figure 3.3.2.2 Struct of the point

Based pm the figure 3.3.2.2 shows that is a struct of a point that composite the record declaration that will be defined in a physical grouped list of variables to be group as a one name in a block of memory. The data that will be group in the struct are coordinate of the location point, departure date to the location point and the description of the location point. The date format will be designed in day, hours and minutes format, and the coordinate is used the CLLocationCoordinate2D to combine the latitude and longitude of the point.



```
let locationManager = CLLocationManager()

func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

    FirebaseApp.configure()

    self.locationManager.delegate = self
    self.locationManager.requestAlwaysAuthorization()
    self.locationManager.startMonitoringVisits()

    return true
}
```

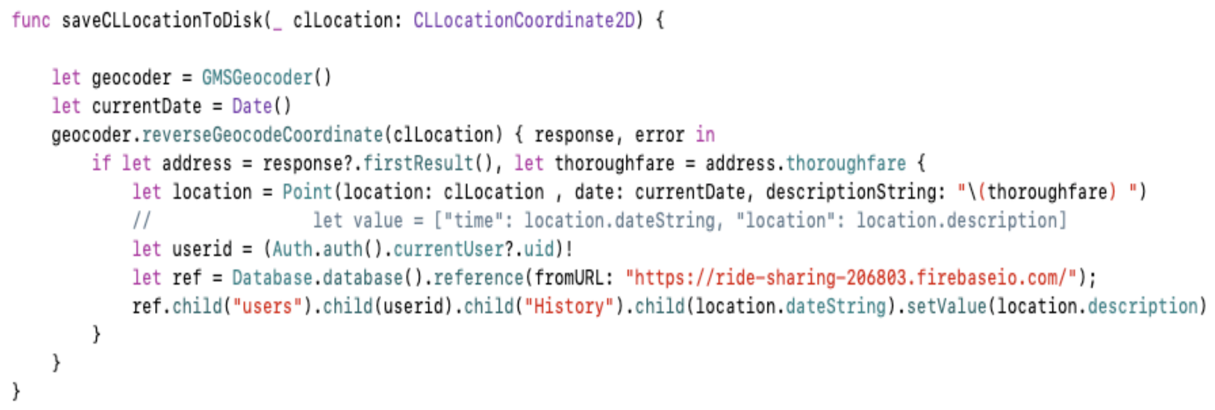
Figure 3.3.2.3 Location Manager of the location tracker



```
func locationManager(_ manager: CLLocationManager, didVisit visit: CLVisit) {
    print("Visit: \(visit)")
    let location = CLLocationCoordinate2D(latitude: visit.coordinate.latitude, longitude: visit.coordinate.longitude)
    saveCLLocationToDisk(location)
}
```

Figure 3.3.2.4 didVisit function in the locationManager

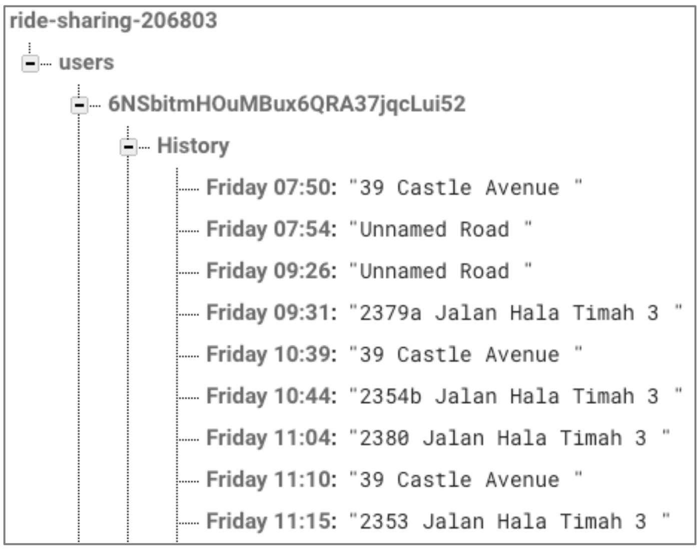
From the figure 3.3.2.3 shows that the location manager is used for the object that to start and stop the delivery of the location-related events in the location tracker. This location manager will run background in the smartphone, it will keep reporting the changes from the GPS and deferring the delivery of location updates in the location tracker. When the smartphone is stop at location, the didVisit function in the location manager will collect the latitude and longitude of the location that shown in the figure 3.3.2.4.



```
func saveCLLocationToDisk(_ clLocation: CLLocationCoordinate2D) {

    let geocoder = GMSGeocoder()
    let currentDate = Date()
    geocoder.reverseGeocodeCoordinate(clLocation) { response, error in
        if let address = response?.firstResult(), let thoroughfare = address.thoroughfare {
            let location = Point(location: clLocation , date: currentDate, descriptionString: "\(thoroughfare) ")
            //              let value = ["time": location.dateString, "location": location.description]
            let userid = (Auth.auth().currentUser?.uid)!
            let ref = Database.database().reference(fromURL: "https://ride-sharing-206803.firebaseio.com/");
            ref.child("users").child(userid).child("History").child(location.dateString).setValue(location.description)
        }
    }
}
```

Figure 3.3.2.5 Upload and save the location data into Firebase Database

After the location coordinate tracked, by using the Google Maps Services API and use the function Google Maps Services Geocoder to do the reverseGeocodeCoordinate from coordinate into a readable string location name. Then this location data will automatically upload to Firebase and store into the database by differentiate with the user Id.



```
ride-sharing-206803
  users
    6NSbitmHOuMBux6QRA37jqcLui52
      History
        Friday 07:50: "39 Castle Avenue "
        Friday 07:54: "Unnamed Road "
        Friday 09:26: "Unnamed Road "
        Friday 09:31: "2379a Jalan Hala Timah 3 "
        Friday 10:39: "39 Castle Avenue "
        Friday 10:44: "2354b Jalan Hala Timah 3 "
        Friday 11:04: "2380 Jalan Hala Timah 3 "
        Friday 11:10: "39 Castle Avenue "
        Friday 11:15: "2353 Jalan Hala Timah 3 "
```

Figure 3.3.2.6 Collected visited location data store in realtime database

Based on the figure 3.3.2.6 above shows that location data that store in the Firebase Realtime Database. This database is a cloud-hosted No-SQL database, the location data that we collect will stored as JSON object into the JOSN tree.

3.4 Data Processing

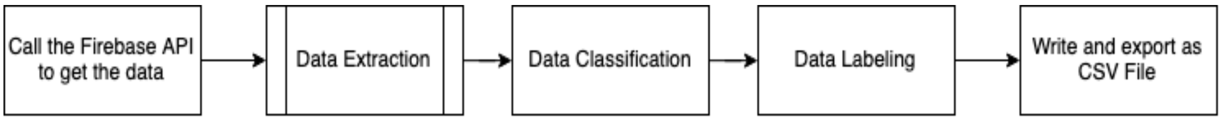
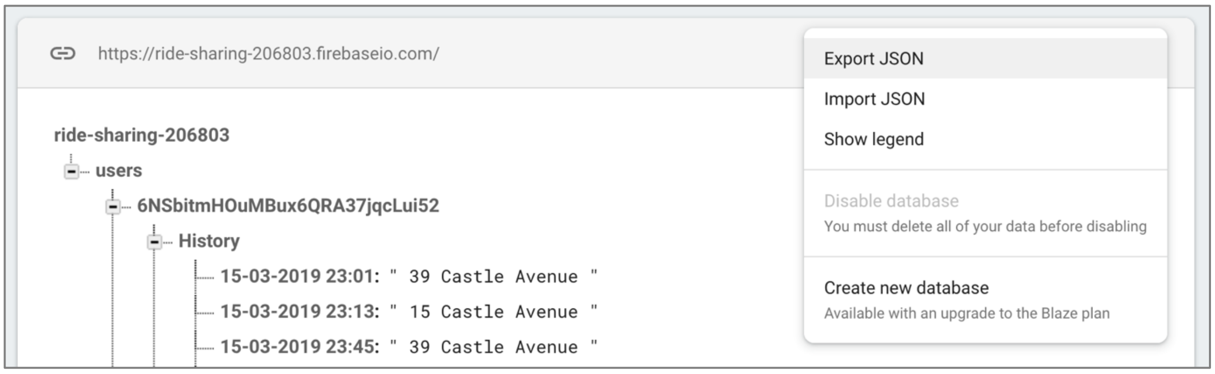

Call the Firebase API to get the data
Data Extraction
Data Classification
Data Labeling
Write and export as CSV File

Figure 3.4.1 Flow Data Processing

While the JSON to CSV converter is developed in Python programming language. In the previous in last semester, it was a manual way to export the data into JSON file by click the button in the Firebase database that shown in figure 3.4.2.



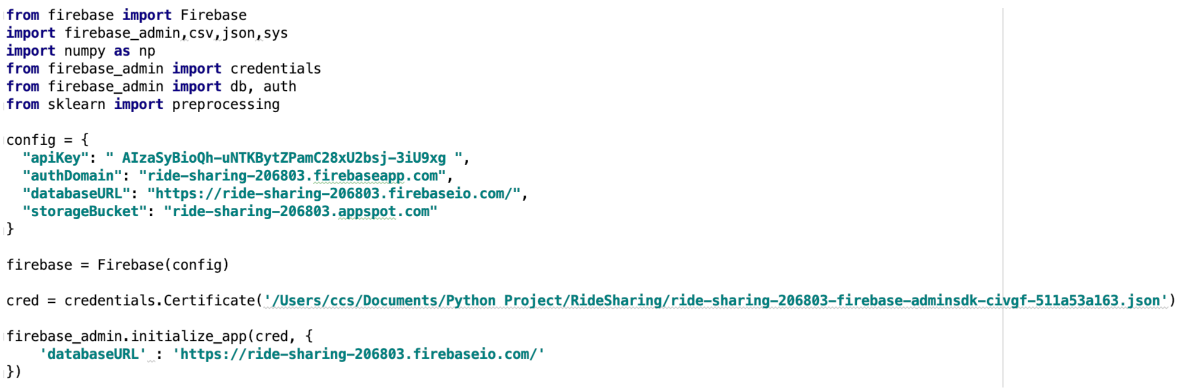
https://ride-sharing-206803.firebaseio.com/

ride-sharing-206803
users
6NSbitmHOuMBux6QRA37jqcLui52
History
15-03-2019 23:01: " 39 Castle Avenue "
15-03-2019 23:13: " 15 Castle Avenue "
15-03-2019 23:45: " 39 Castle Avenue "

Export JSON
Import JSON
Show legend
Disable database
You must delete all of your data before disabling
Create new database
Available with an upgrade to the Blaze plan

Figure 3.4.2 Export data from database into JSON file in manual way

By using the manual way to export the data from Firebase and download into a JSON file, it is not a flexible and efficiency solution to export the data. Therefore, instead of using the manual way to export the data, there is an improvement that in the data processing main program, which is using the Firebase-admin API that call in the main program. By using this API that can easily retrieve the data from the Firebase database, this will be more flexible and effective.



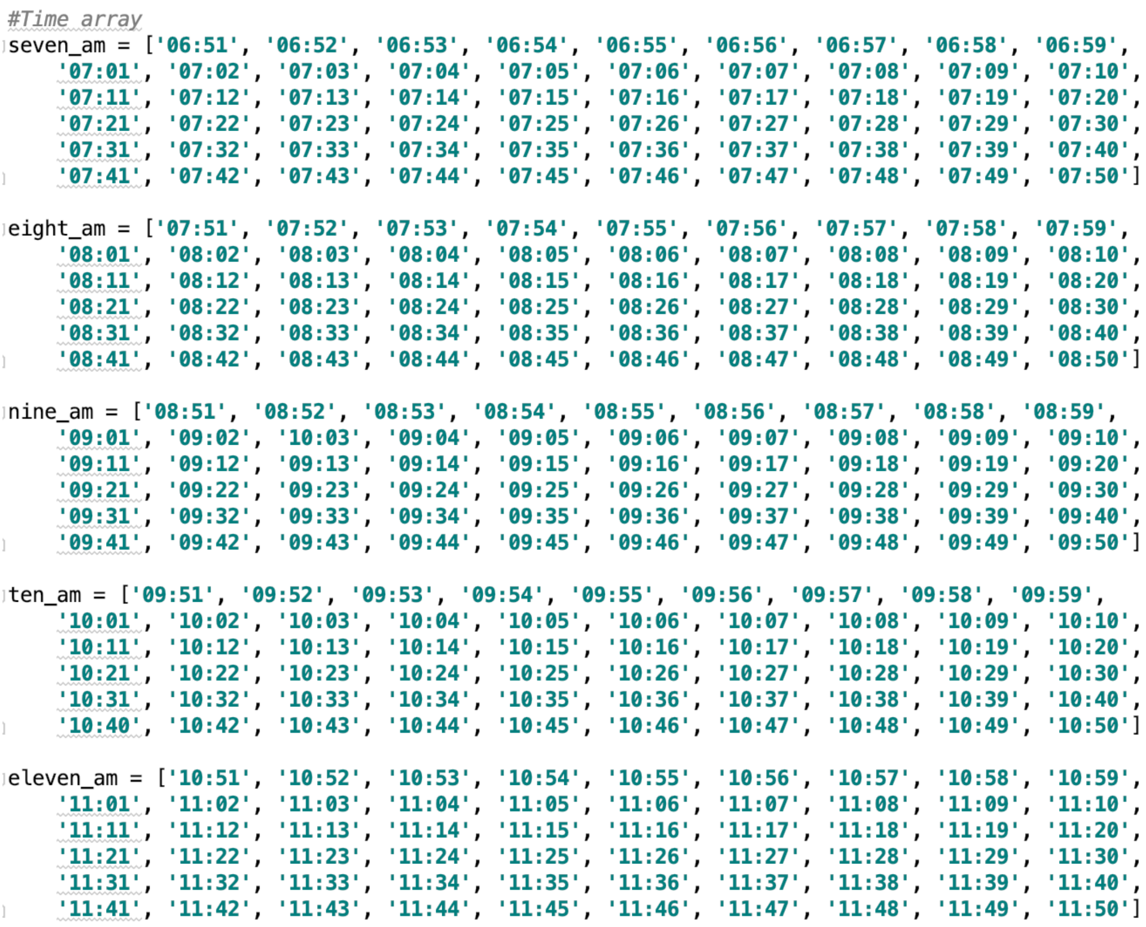
```
from firebase import Firebase
import firebase_admin,csv,json,sys
import numpy as np
from firebase_admin import credentials
from firebase_admin import db, auth
from sklearn import preprocessing

config = {
  "apiKey": " AIzaSyBioQh-uNTKBytZPamC28xU2bsj-3iU9xg ",
  "authDomain": "ride-sharing-206803.firebaseapp.com",
  "databaseURL": "https://ride-sharing-206803.firebaseio.com/",
  "storageBucket": "ride-sharing-206803.appspot.com"
}

firebase = Firebase(config)

cred = credentials.Certificate('/Users/ccs/Documents/Python Project/RideSharing/ride-sharing-206803-firebase-adminsdk-civgf-511a53a163.json')

firebase_admin.initialize_app(cred, {
    'databaseURL' : 'https://ride-sharing-206803.firebaseio.com/'
})
```

Figure 3.4.3 Configuration of Firebase Admin SDK

In the program there will be some algorithms to convert the data from JSON object format into string format, and write the convert data and export as CSV file. During the conversion the data classification and labeling will do, this is because to make the training job more compatible in the AWS SageMaker.

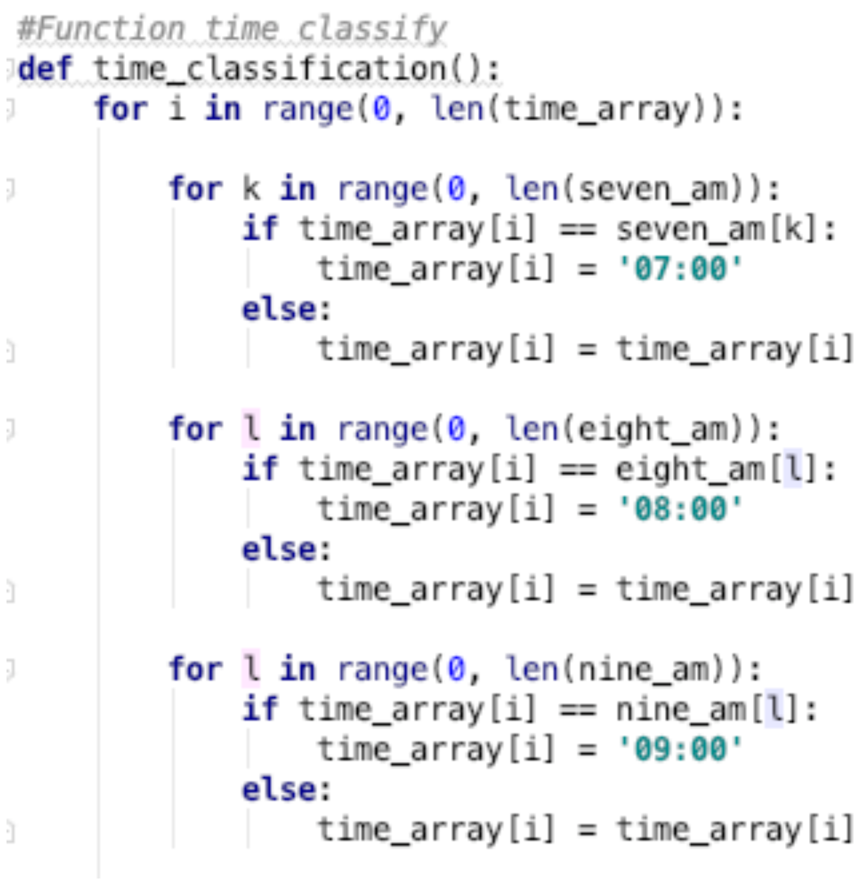
3.4.1 Data Classification

In this project, data classification will do, because to make the data more compatible when the model training jobs, the time and day will do the classification in this project. For the time classification part, from the figure 3.4.1.1 shown the time array will be defined that classify all the hours and minutes, the from



```
#Time array
seven_am = ['06:51', '06:52', '06:53', '06:54', '06:55', '06:56', '06:57', '06:58', '06:59',
    '07:01', '07:02', '07:03', '07:04', '07:05', '07:06', '07:07', '07:08', '07:09', '07:10',
    '07:11', '07:12', '07:13', '07:14', '07:15', '07:16', '07:17', '07:18', '07:19', '07:20',
    '07:21', '07:22', '07:23', '07:24', '07:25', '07:26', '07:27', '07:28', '07:29', '07:30',
    '07:31', '07:32', '07:33', '07:34', '07:35', '07:36', '07:37', '07:38', '07:39', '07:40',
    '07:41', '07:42', '07:43', '07:44', '07:45', '07:46', '07:47', '07:48', '07:49', '07:50']

eight_am = ['07:51', '07:52', '07:53', '07:54', '07:55', '07:56', '07:57', '07:58', '07:59',
    '08:01', '08:02', '08:03', '08:04', '08:05', '08:06', '08:07', '08:08', '08:09', '08:10',
    '08:11', '08:12', '08:13', '08:14', '08:15', '08:16', '08:17', '08:18', '08:19', '08:20',
    '08:21', '08:22', '08:23', '08:24', '08:25', '08:26', '08:27', '08:28', '08:29', '08:30',
    '08:31', '08:32', '08:33', '08:34', '08:35', '08:36', '08:37', '08:38', '08:39', '08:40',
    '08:41', '08:42', '08:43', '08:44', '08:45', '08:46', '08:47', '08:48', '08:49', '08:50']

nine_am = ['08:51', '08:52', '08:53', '08:54', '08:55', '08:56', '08:57', '08:58', '08:59',
    '09:01', '09:02', '10:03', '09:04', '09:05', '09:06', '09:07', '09:08', '09:09', '09:10',
    '09:11', '09:12', '09:13', '09:14', '09:15', '09:16', '09:17', '09:18', '09:19', '09:20',
    '09:21', '09:22', '09:23', '09:24', '09:25', '09:26', '09:27', '09:28', '09:29', '09:30',
    '09:31', '09:32', '09:33', '09:34', '09:35', '09:36', '09:37', '09:38', '09:39', '09:40',
    '09:41', '09:42', '09:43', '09:44', '09:45', '09:46', '09:47', '09:48', '09:49', '09:50']

ten_am = ['09:51', '09:52', '09:53', '09:54', '09:55', '09:56', '09:57', '09:58', '09:59',
    '10:01', '10:02', '10:03', '10:04', '10:05', '10:06', '10:07', '10:08', '10:09', '10:10',
    '10:11', '10:12', '10:13', '10:14', '10:15', '10:16', '10:17', '10:18', '10:19', '10:20',
    '10:21', '10:22', '10:23', '10:24', '10:25', '10:26', '10:27', '10:28', '10:29', '10:30',
    '10:31', '10:32', '10:33', '10:34', '10:35', '10:36', '10:37', '10:38', '10:39', '10:40',
    '10:40', '10:42', '10:43', '10:44', '10:45', '10:46', '10:47', '10:48', '10:49', '10:50']

eleven_am = ['10:51', '10:52', '10:53', '10:54', '10:55', '10:56', '10:57', '10:58', '10:59',
    '11:01', '11:02', '11:03', '11:04', '11:05', '11:06', '11:07', '11:08', '11:09', '11:10',
    '11:11', '11:12', '11:13', '11:14', '11:15', '11:16', '11:17', '11:18', '11:19', '11:20',
    '11:21', '11:22', '11:23', '11:24', '11:25', '11:26', '11:27', '11:28', '11:29', '11:30',
    '11:31', '11:32', '11:33', '11:34', '11:35', '11:36', '11:37', '11:38', '11:39', '11:40',
    '11:41', '11:42', '11:43', '11:44', '11:45', '11:46', '11:47', '11:48', '11:49', '11:50']
```

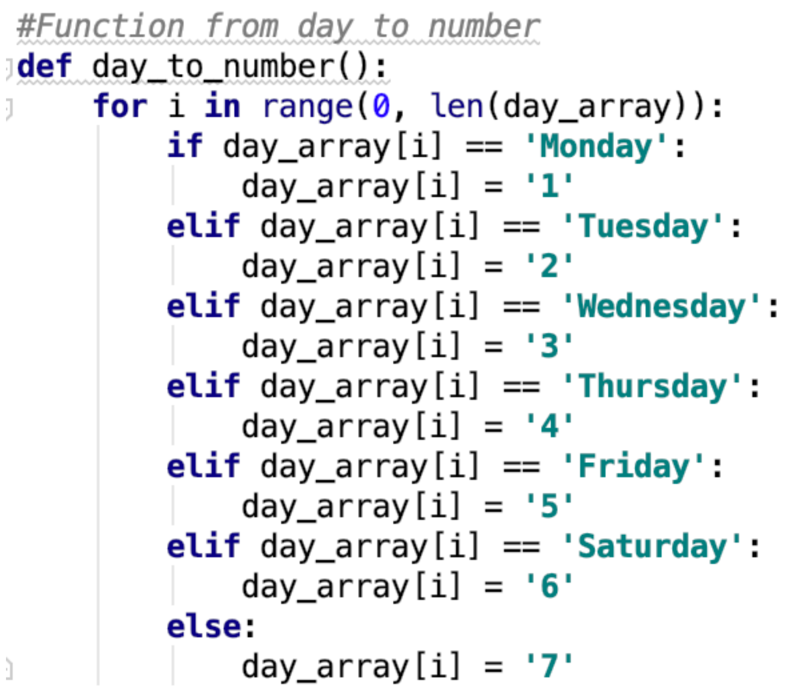
Figure 3.4.1.1 Array declaration of each time



```python
#Function time classify
def time_classification():
    for i in range(0, len(time_array)):

        for k in range(0, len(seven_am)):
            if time_array[i] == seven_am[k]:
                time_array[i] = '07:00'
            else:
                time_array[i] = time_array[i]

        for l in range(0, len(eight_am)):
            if time_array[i] == eight_am[l]:
                time_array[i] = '08:00'
            else:
                time_array[i] = time_array[i]

        for l in range(0, len(nine_am)):
            if time_array[i] == nine_am[l]:
                time_array[i] = '09:00'
            else:
                time_array[i] = time_array[i]
```

Figure 3.4.1.2 Function of time classification

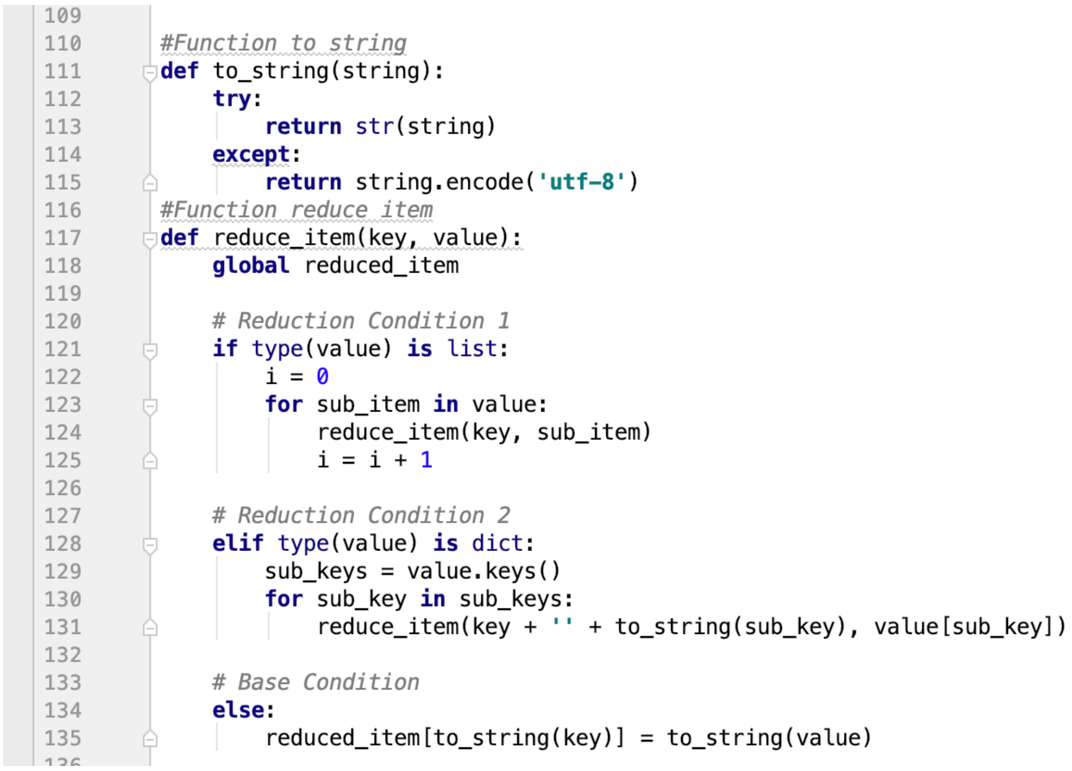
From the figure 3.4.1.2 shows that is a defined function of the time classification, this function will run when the main program will call this function and this function will load the time data that collected and store into an array that call time array, and start to loop the time array to determine all the time value and then classify them into the relevant value for each time format. Like an example if the time format is 8:15, when the function is called, and it will classify the time into 8:00. The reason to perform the time classification is to categorize the data into various class, and to improve the effective of determination in project work later.



```
#Function from day to number
def day_to_number():
    for i in range(0, len(day_array)):
        if day_array[i] == 'Monday':
            day_array[i] = '1'
        elif day_array[i] == 'Tuesday':
            day_array[i] = '2'
        elif day_array[i] == 'Wednesday':
            day_array[i] = '3'
        elif day_array[i] == 'Thursday':
            day_array[i] = '4'
        elif day_array[i] == 'Friday':
            day_array[i] = '5'
        elif day_array[i] == 'Saturday':
            day_array[i] = '6'
        else:
            day_array[i] = '7'
```

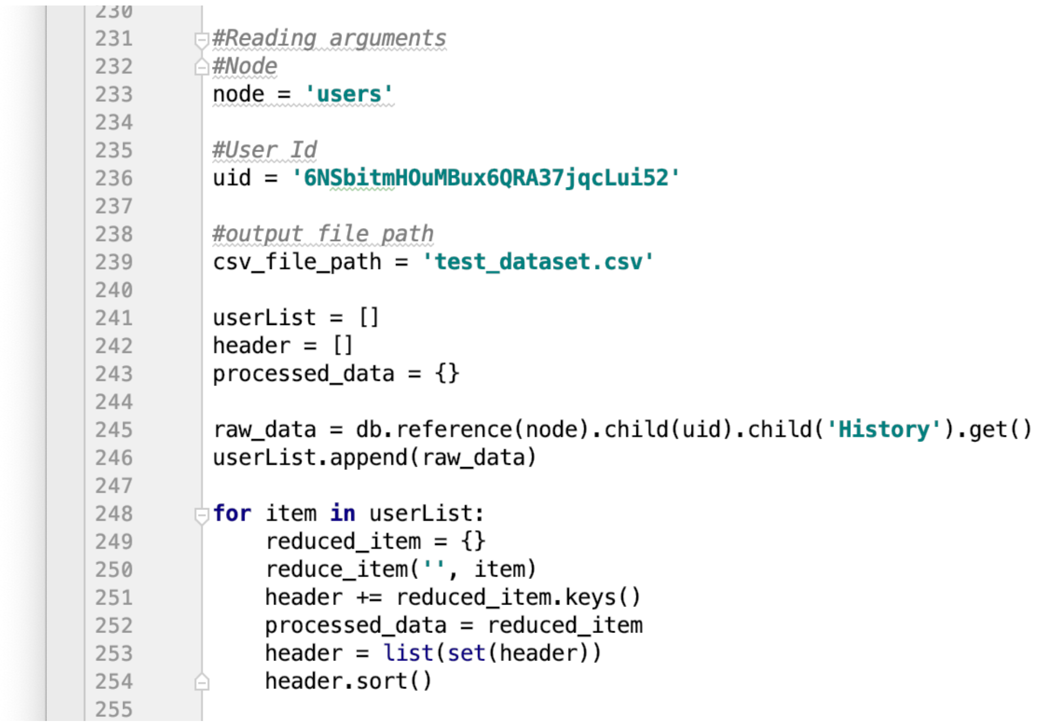
Figure 3.4.1.3 Function of conversion from day to number

Based on the figure 3.4.1.3 shows that is the function to convert the from day to number, due to the day data that collected is in string format, and the string format is not able to perform to train during the model training. Hence, the day classification will do, which is to classify all the data day that collected and store in an array, when this function is called, then this function will perform the conversion from day in string into number.



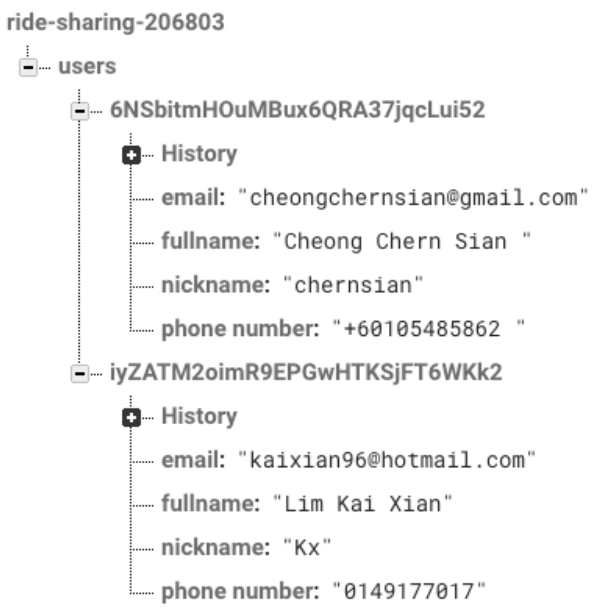
```python
#Function to string
def to_string(string):
    try:
        return str(string)
    except:
        return string.encode('utf-8')
#Function reduce item
def reduce_item(key, value):
    global reduced_item

    # Reduction Condition 1
    if type(value) is list:
        i = 0
        for sub_item in value:
            reduce_item(key, sub_item)
            i = i + 1

    # Reduction Condition 2
    elif type(value) is dict:
        sub_keys = value.keys()
        for sub_key in sub_keys:
            reduce_item(key + '' + to_string(sub_key), value[sub_key])

    # Base Condition
    else:
        reduced_item[to_string(key)] = to_string(value)
```

Figure 3.4.1.4 Function declaration



```python
#Reading arguments
#Node
node = 'users'

#User Id
uid = '6NSbitmHOuMBux6QRA37jqcLui52'

#output file path
csv_file_path = 'test_dataset.csv'

userList = []
header = []
processed_data = {}

raw_data = db.reference(node).child(uid).child('History').get()
userList.append(raw_data)

for item in userList:
    reduced_item = {}
    reduce_item('', item)
    header += reduced_item.keys()
    processed_data = reduced_item
    header = list(set(header))
    header.sort()
```

Figure 3.4.1.5 Main program of the data processing part 1



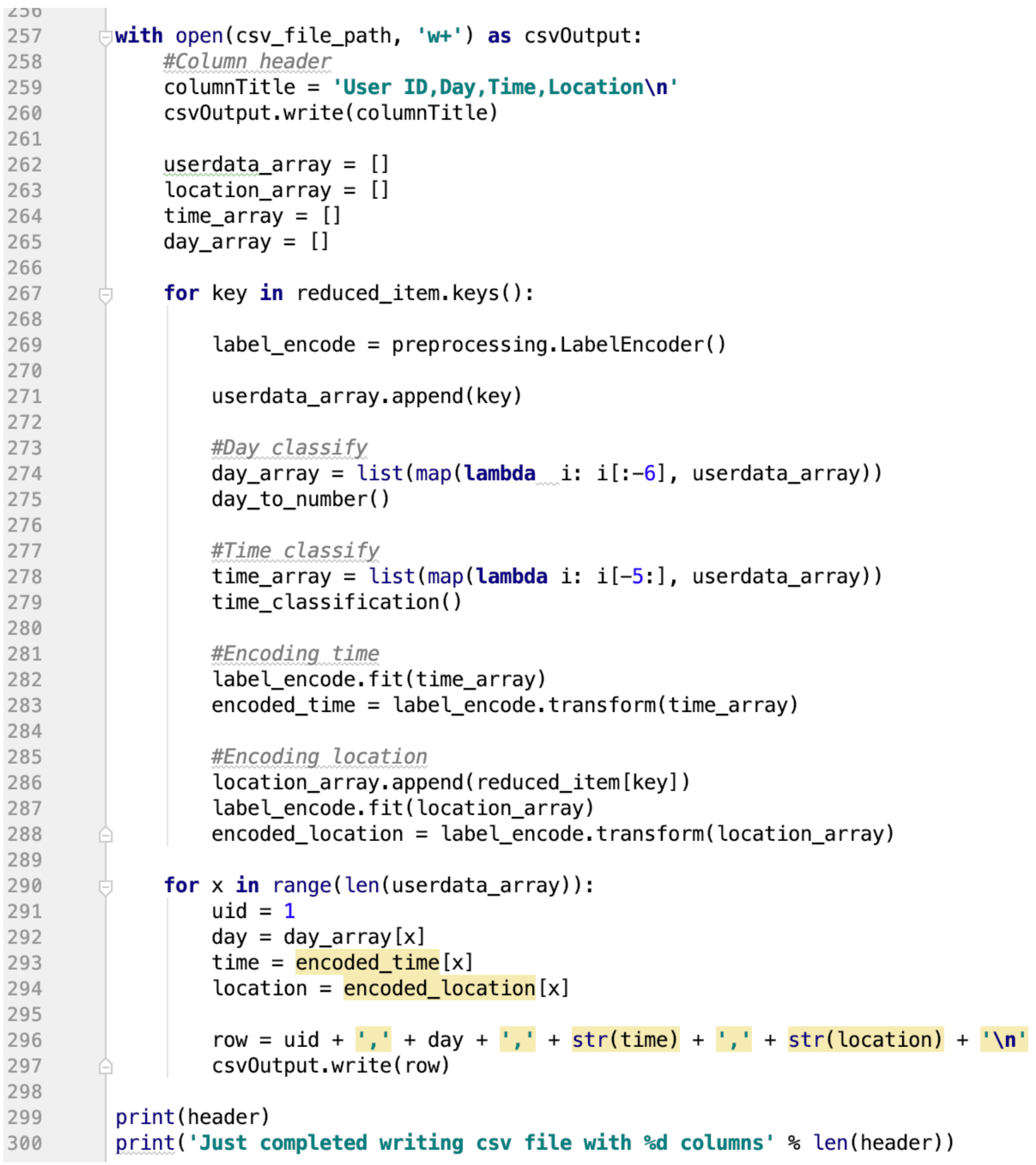
```
ride-sharing-206803
 users
  6NSbitmHOuMBux6QRA37jqcLui52
   History
   email: "cheongchernsian@gmail.com"
   fullname: "Cheong Chern Sian "
   nickname: "chernsian"
   phone number: "+60105485862 "
  iyZATM2oimR9EPGwHTKSjFT6WKk2
   History
   email: "kaixian96@hotmail.com"
   fullname: "Lim Kai Xian"
   nickname: "Kx"
   phone number: "0149177017"
```

Figure 3.4.1.6 Data structure of the database

Based on the figure 3.4.1.5 shows that is the first part of the main program, the three arguments will be declared, which is node, uid as user ID and csv_file_path as the output file name. Then the two arrays will be declared which is userList as the list item of user that to store the and header of the list item. To retrieve the data from the Firebase database, firebase API will be called, and this API will directly determine which database that shown in figure 3.4.3 and the figure 3.4.1.6 shown, the data structure of the Firebase database is in JSON tree structure, ride-sharing-206803 is the name of the database and the users will be the node of the database. Thus, the format that to retrieve the data is ride-sharing-206803/users/userID/History, in the main program the API will reference from the node, user ID and History to get the data, and these data will store into userList array. From the userList, will loop the userList and call the functions that shown in figure 3.4.1.1 to bind each of the all data into format the key with value and store in to a processed data list and the key also will store into the header array.

3.4.2 Data Labeling

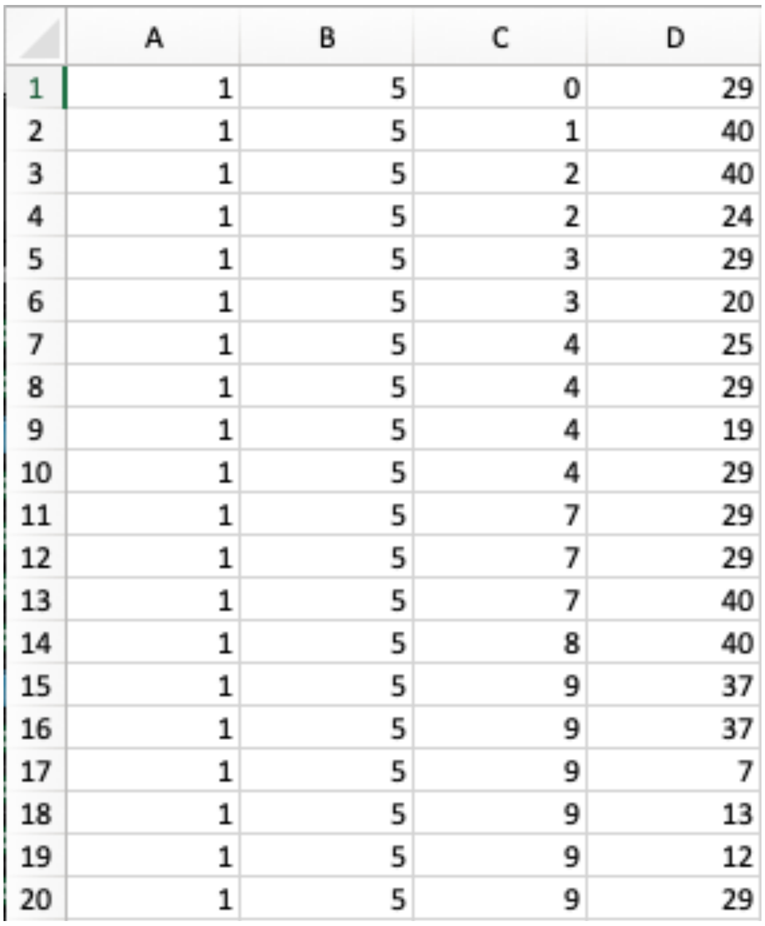
In this project, data labeling will do, because during the model training, data in string type is not able to use to train and perform the predictions and accuracy calculation. Thus data labeling is to make the data more compatible when the model training jobs.



```python
with open(csv_file_path, 'w+') as csvOutput:
    #Column header
    columnTitle = 'User ID,Day,Time,Location\n'
    csvOutput.write(columnTitle)

    userdata_array = []
    location_array = []
    time_array = []
    day_array = []

    for key in reduced_item.keys():

        label_encode = preprocessing.LabelEncoder()

        userdata_array.append(key)

        #Day classify
        day_array = list(map(lambda  i: i[:-6], userdata_array))
        day_to_number()

        #Time classify
        time_array = list(map(lambda i: i[-5:], userdata_array))
        time_classification()

        #Encoding time
        label_encode.fit(time_array)
        encoded_time = label_encode.transform(time_array)

        #Encoding location
        location_array.append(reduced_item[key])
        label_encode.fit(location_array)
        encoded_location = label_encode.transform(location_array)

    for x in range(len(userdata_array)):
        uid = 1
        day = day_array[x]
        time = encoded_time[x]
        location = encoded_location[x]

        row = uid + ',' + day + ',' + str(time) + ',' + str(location) + '\n'
        csvOutput.write(row)

print(header)
print('Just completed writing csv file with %d columns' % len(header))
```

Figure 3.4.2.1 Main program of the data processing part 2

Based in the figure 3.4.2.1 shows the second part of the main program of the data processing, with open the csv file as output file and w+ is able to write and read the file. Next, will defined the column title which is the row title in the csv file, and write the column title into the file. Before the preprocessing part, four arrays will be defined for the data storing uses. Next, a loop will be performed to do the data labeling, by using the scikit-learn libraries and import the preprocessing to use the function of label encoder. User data array will store the keys of the data, these keys are the days and times of the location data, and the location data also will store in the location array. By using the lambda function, to segment into day into day array and time into time array. Then each of the day and time array will call the day_to_number and time_classification function to perform the data classification. The label encoder will start to encoding the time and location. Lastly, these data will write row by row into the CSV file.



	A	B	C	D
1	1	5	0	29
2	1	5	1	40
3	1	5	2	40
4	1	5	2	24
5	1	5	3	29
6	1	5	3	20
7	1	5	4	25
8	1	5	4	29
9	1	5	4	19
10	1	5	4	29
11	1	5	7	29
12	1	5	7	29
13	1	5	7	40
14	1	5	8	40
15	1	5	9	37
16	1	5	9	37
17	1	5	9	7
18	1	5	9	13
19	1	5	9	12
20	1	5	9	29

Figure 3.4.2.2 Processed data export in CSV file.

3.4.3 Reference of data

Day	Number
Monday	1
Tuesday	2
Wednesday	3
Thursday	4
Friday	5
Saturday	6
Sunday	7

Table 3-4-3-1 Table of label number to day

Time	Number
0	7:00 a.m.
1	8:00 a.m.
2	9:00 a.m.
3	10:00 a.m.
4	11:00 a.m.
5	12:00 p.m.
6	1:00 p.m.
7	2:00 p.m.
8	3:00 p.m.
9	4:00 p.m.
10	5:00 p.m.
11	6:00 p.m.

Table 3-4-3-2 Table of label number to time

In the project, the target users are UTAR student or lecturer. Therefore the time duration that we collect the data is only weekdays and based on the working hours of the UTAR from 7:00 a.m. to 6:00 p.m. Hence, the starting point will be UTAR.

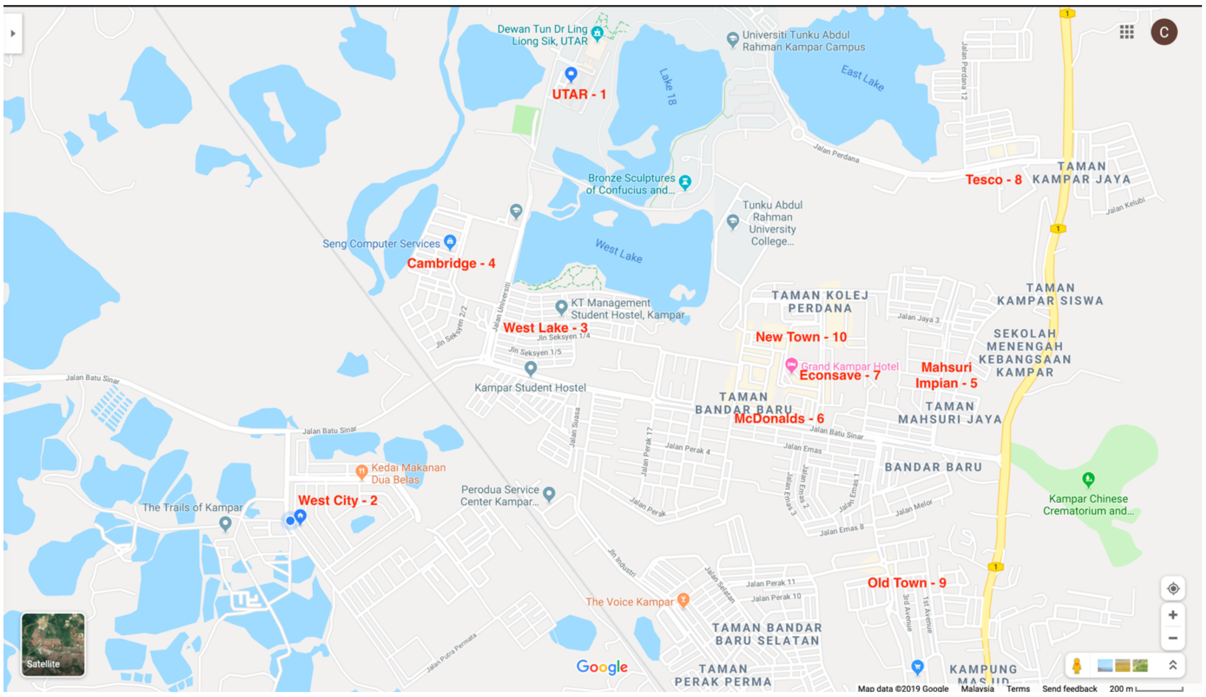
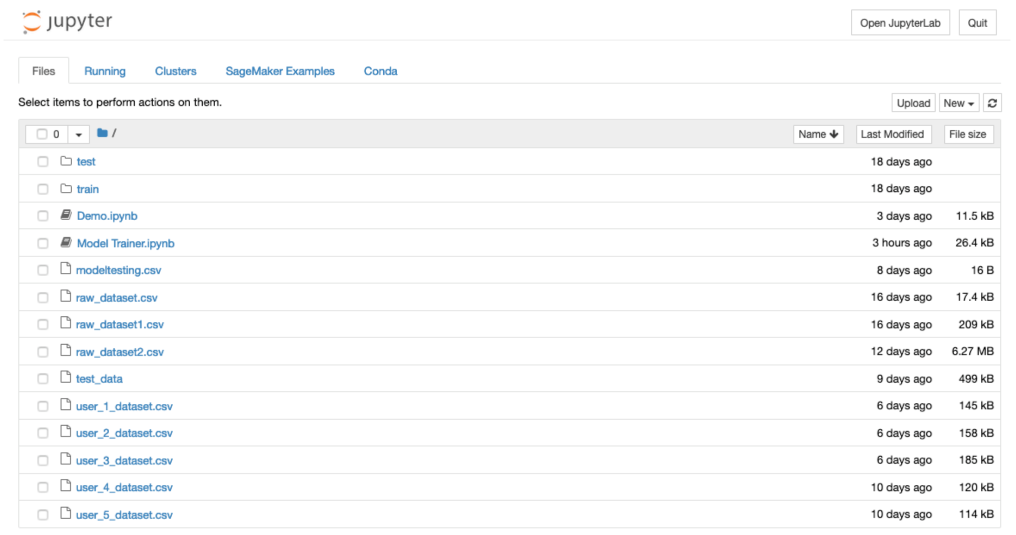


Figure 3.4.3.1 Map labeling

Location	Number
UTAR	1
West City	2
West Lake	3
Cambridge	4
Mahsuri Impian	5
McDonald’s	6
Econsave	7
Tesco	8
Kampar Old Town	9
Kampar New Town	10

Table 3-4-3-3 Table of label number to location

3.5 Model Training

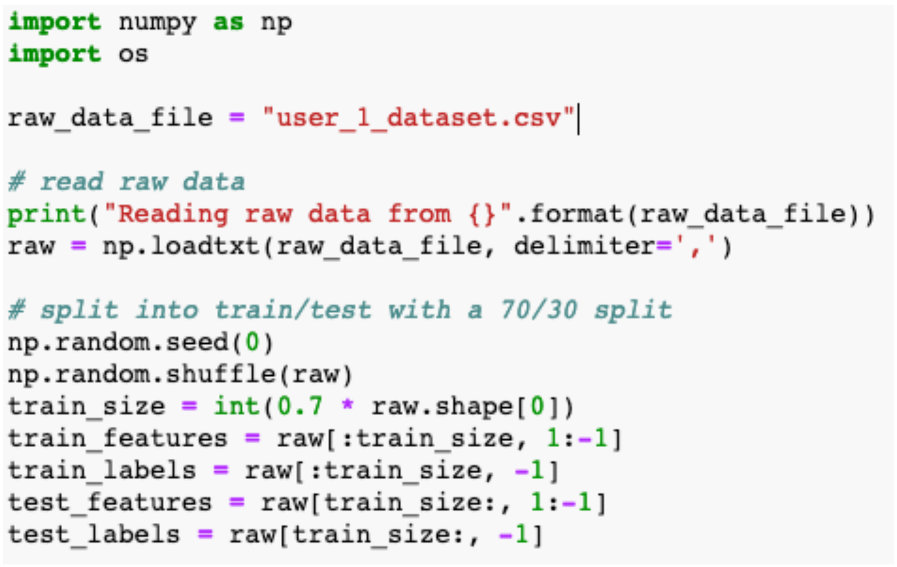


Select items to perform actions on them.

Name	Last Modified	File size
test	18 days ago	
train	18 days ago	
Demo.ipynb	3 days ago	11.5 kB
Model Trainer.ipynb	3 hours ago	26.4 kB
modeltesting.csv	8 days ago	16 B
raw_dataset.csv	16 days ago	17.4 kB
raw_dataset1.csv	16 days ago	209 kB
raw_dataset2.csv	12 days ago	6.27 MB
test_data	9 days ago	499 kB
user_1_dataset.csv	6 days ago	145 kB
user_2_dataset.csv	6 days ago	158 kB
user_3_dataset.csv	6 days ago	185 kB
user_4_dataset.csv	10 days ago	120 kB
user_5_dataset.csv	10 days ago	114 kB

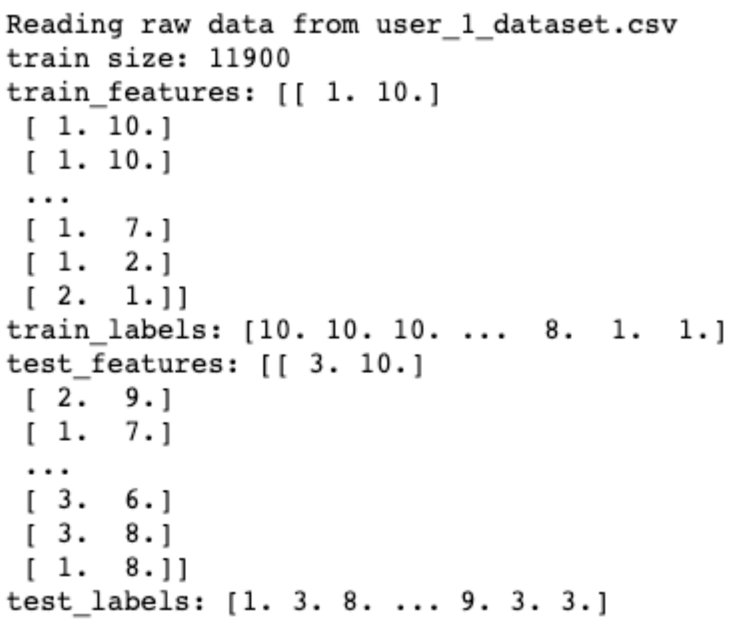
Figure 3.5.1 Jupyter notebook instance

For the modeling training, building and predictions program is developed by Python, and Jupyter Notebook instance will be used in the model training and predictions. Next due to the time limitation to collect the huge amount data for the data training, data augmentation will be processed that to increase the diversity of data available for training models. Thus, the dataset of each user will be extrapolated to fit the amount for model train. After the data augmentation is done, the dataset will upload to the notebook instance, then the dataset will split into 70% for train 30 % for test.



```
import numpy as np
import os

raw_data_file = "user_1_dataset.csv"

# read raw data
print("Reading raw data from {}".format(raw_data_file))
raw = np.loadtxt(raw_data_file, delimiter=',')

# split into train/test with a 70/30 split
np.random.seed(0)
np.random.shuffle(raw)
train_size = int(0.7 * raw.shape[0])
train_features = raw[:train_size, 1:-1]
train_labels = raw[:train_size, -1]
test_features = raw[train_size:, 1:-1]
test_labels = raw[train_size:, -1]
```

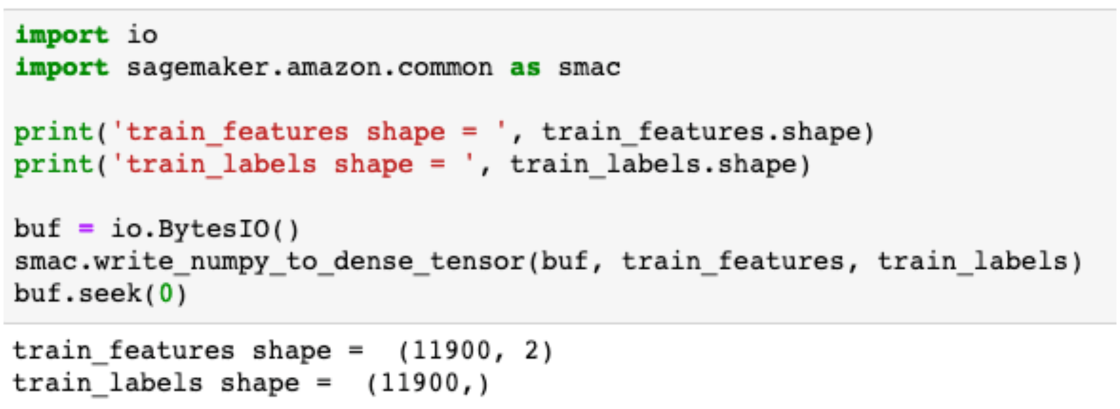
Figure 3.5.2 Read dataset and split into train and test



```
Reading raw data from user_1_dataset.csv
train size: 11900
train_features: [[ 1. 10.]
 [ 1. 10.]
 [ 1. 10.]
 ...
 [ 1.  7.]
 [ 1.  2.]
 [ 2.  1.]]
train_labels: [10. 10. 10. ...  8.  1.  1.]
test_features: [[ 3. 10.]
 [ 2.  9.]
 [ 1.  7.]
 ...
 [ 3.  6.]
 [ 3.  8.]
 [ 1.  8.]]
test_labels: [1. 3. 8. ... 9. 3. 3.]
```

Figure 3.5.3 Console result of the data splitting

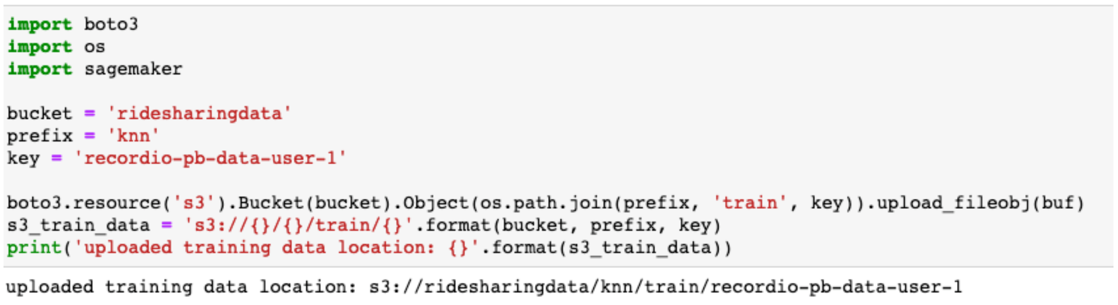
From the figure 3.5.1 shows the dataset will read and define a seed that to make the random number predictable, and shuffle the data. Next, the data will be split into ratio of 70:30, 70% for train and 30% for test. After split the data, the day and times in the data will become the features and the location will become the labels for model training.



```
import io
import sagemaker.amazon.common as smac

print('train_features shape = ', train_features.shape)
print('train_labels shape = ', train_labels.shape)

buf = io.BytesIO()
smac.write_numpy_to_dense_tensor(buf, train_features, train_labels)
buf.seek(0)
```

```
train_features shape =  (11900, 2)
train_labels shape =  (11900,)
```

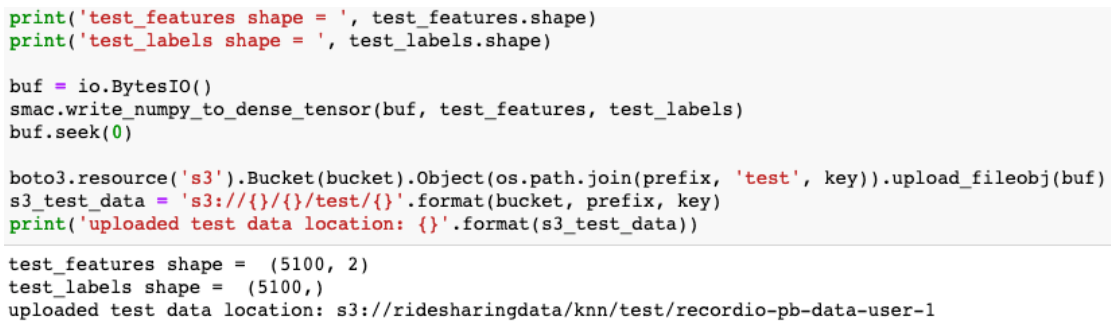
Figure 3.5.4 Using the IO buffer as dataset



```
import boto3
import os
import sagemaker

bucket = 'ridesharingdata'
prefix = 'knn'
key = 'recordio-pb-data-user-1'

boto3.resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'train', key)).upload_fileobj(buf)
s3_train_data = 's3://{}/{}/train/{}'.format(bucket, prefix, key)
print('uploaded training data location: {}'.format(s3_train_data))
```

```
uploaded training data location: s3://ridesharingdata/knn/train/recordio-pb-data-user-1
```

Figure 3.5.5 Save the train dataset to the S3 bucket

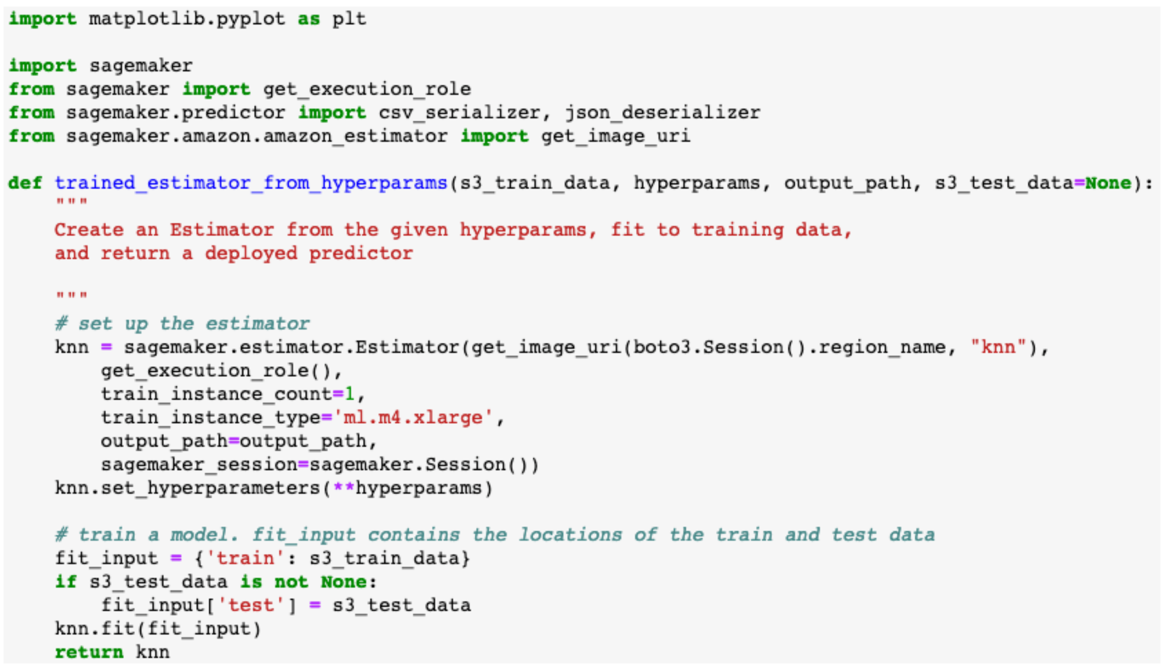


```
print('test_features shape = ', test_features.shape)
print('test_labels shape = ', test_labels.shape)

buf = io.BytesIO()
smac.write_numpy_to_dense_tensor(buf, test_features, test_labels)
buf.seek(0)

boto3.resource('s3').Bucket(bucket).Object(os.path.join(prefix, 'test', key)).upload_fileobj(buf)
s3_test_data = 's3://{}/{}/test/{}'.format(bucket, prefix, key)
print('uploaded test data location: {}'.format(s3_test_data))
```

```
test_features shape =  (5100, 2)
test_labels shape =  (5100,)
uploaded test data location: s3://ridesharingdata/knn/test/recordio-pb-data-user-1
```

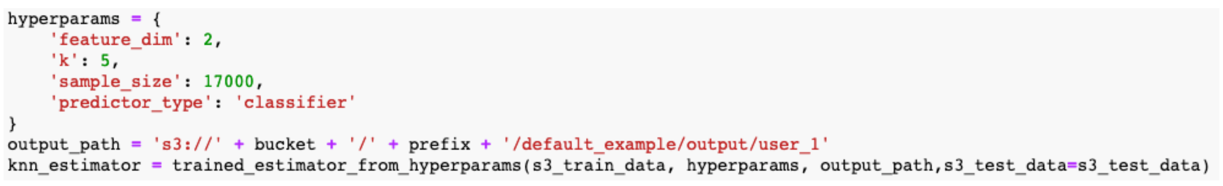
Figure 3.5.6 Save the test dataset to the S3 Bucket

From the figure 3.5.4 to 3.5.6 shows, the train features and labels is using the IO buffer to write the data into the memory buffer instead of writing the data to a file. At the same time, the test features and labels also will do the same procedure that write the data into memory buffer. Then the buffer will upload and store the S3 bucket that ready for use later. In the train model program the train data will be rename a s3_train_data and the test data is s3_test_data.



```
import matplotlib.pyplot as plt

import sagemaker
from sagemaker import get_execution_role
from sagemaker.predictor import csv_serializer, json_deserializer
from sagemaker.amazon.amazon_estimator import get_image_uri

def trained_estimator_from_hyperparams(s3_train_data, hyperparams, output_path, s3_test_data=None):
    """
    Create an Estimator from the given hyperparams, fit to training data,
    and return a deployed predictor

    """
    # set up the estimator
    knn = sagemaker.estimator.Estimator(get_image_uri(boto3.Session().region_name, "knn"),
        get_execution_role(),
        train_instance_count=1,
        train_instance_type='ml.m4.xlarge',
        output_path=output_path,
        sagemaker_session=sagemaker.Session())
    knn.set_hyperparameters(**hyperparams)

    # train a model. fit_input contains the locations of the train and test data
    fit_input = {'train': s3_train_data}
    if s3_test_data is not None:
        fit_input['test'] = s3_test_data
    knn.fit(fit_input)
    return knn
```

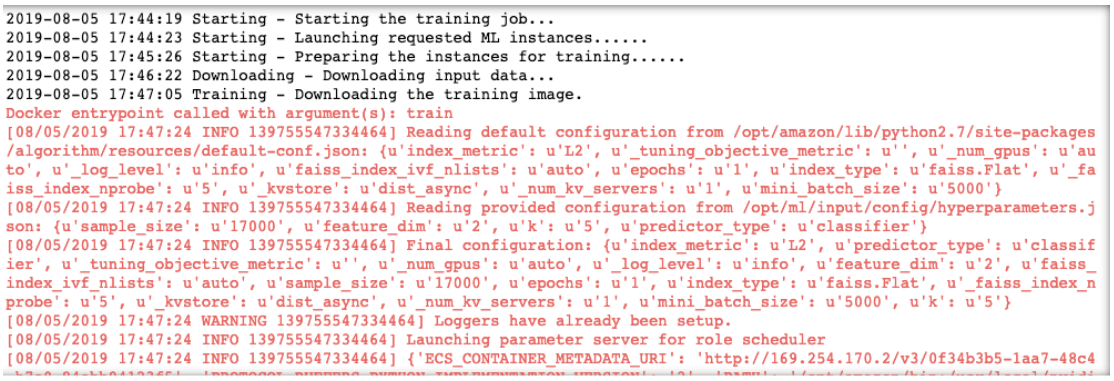
Figure 3.5.7 Function of model trainer



```
hyperparams = {
    'feature_dim': 2,
    'k': 5,
    'sample_size': 17000,
    'predictor_type': 'classifier'
}
output_path = 's3://' + bucket + '/' + prefix + '/default_example/output/user_1'
knn_estimator = trained_estimator_from_hyperparams(s3_train_data, hyperparams, output_path,s3_test_data=s3_test_data)
```

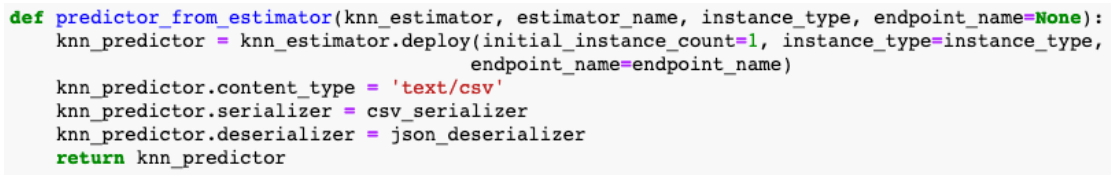
Figure 3.5.8 Hyperparameters and start to train a model

Based on the Figure 3.5.7, a function has been defined that to use for call to start to train a model, the KNN will be chosen as the algorithm that to use in the job training . Next, s3_train_data is use to train and become model, while the s3_test_data is use to fit in input contains the locations of the train and test data. The hyperparameters basic settings in the model training are feature_dim, k, sample_size and predictor_type. From the figure 3.5.4 shown that the features shape is 2, then the feature_dim in the whole model training is using 2. In the user 1 dataset there has 5 class and the k will set at 5, the k value will be different that according to the dataset. If the user 2 dataset has 6 class then the k will be 6. For the predictor type in entire model training program, the classifier type will be used. Lastly the job will starting train after the function in figure 3.5.7 has called and the model will save the S3 bucket.



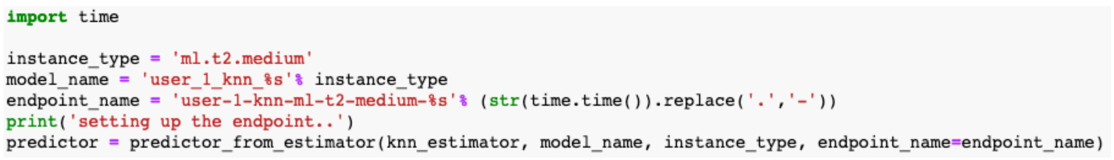
```
2019-08-05 17:44:19 Starting - Starting the training job...
2019-08-05 17:44:23 Starting - Launching requested ML instances......
2019-08-05 17:45:26 Starting - Preparing the instances for training......
2019-08-05 17:46:22 Downloading - Downloading input data...
2019-08-05 17:47:05 Training - Downloading the training image.
Docker entrypoint called with argument(s): train
[08/05/2019 17:47:24 INFO 139755547334464] Reading default configuration from /opt/amazon/lib/python2.7/site-packages
/algorithm/resources/default-conf.json: {u'index_metric': u'L2', u'_tuning_objective_metric': u'', u'_num_gpus': u'au
to', u'_log_level': u'info', u'faiss_index_ivf_nlists': u'auto', u'epochs': u'1', u'index_type': u'faiss.Flat', u'_fa
iss_index_nprobe': u'5', u'_kvstore': u'dist_async', u'_num_kv_servers': u'1', u'mini_batch_size': u'5000'}
[08/05/2019 17:47:24 INFO 139755547334464] Reading provided configuration from /opt/ml/input/config/hyperparameters.j
son: {u'sample_size': u'17000', u'feature_dim': u'2', u'k': u'5', u'predictor_type': u'classifier'}
[08/05/2019 17:47:24 INFO 139755547334464] Final configuration: {u'index_metric': u'L2', u'predictor_type': u'classif
ier', u'_tuning_objective_metric': u'', u'_num_gpus': u'auto', u'_log_level': u'info', u'feature_dim': u'2', u'faiss_
index_ivf_nlists': u'auto', u'sample_size': u'17000', u'epochs': u'1', u'index_type': u'faiss.Flat', u'_faiss_index_n
probe': u'5', u'_kvstore': u'dist_async', u'_num_kv_servers': u'1', u'mini_batch_size': u'5000', u'k': u'5'}
[08/05/2019 17:47:24 WARNING 139755547334464] Loggers have already been setup.
[08/05/2019 17:47:24 INFO 139755547334464] Launching parameter server for role scheduler
[08/05/2019 17:47:24 INFO 139755547334464] {'ECS_CONTAINER_METADATA_URI': 'http://169.254.170.2/v3/0f34b3b5-1aa7-48c4
```

Figure 3.5.9 The progress of model training



```python
def predictor_from_estimator(knn_estimator, estimator_name, instance_type, endpoint_name=None):
    knn_predictor = knn_estimator.deploy(initial_instance_count=1, instance_type=instance_type,
                                         endpoint_name=endpoint_name)
    knn_predictor.content_type = 'text/csv'
    knn_predictor.serializer = csv_serializer
    knn_predictor.deserializer = json_deserializer
    return knn_predictor
```

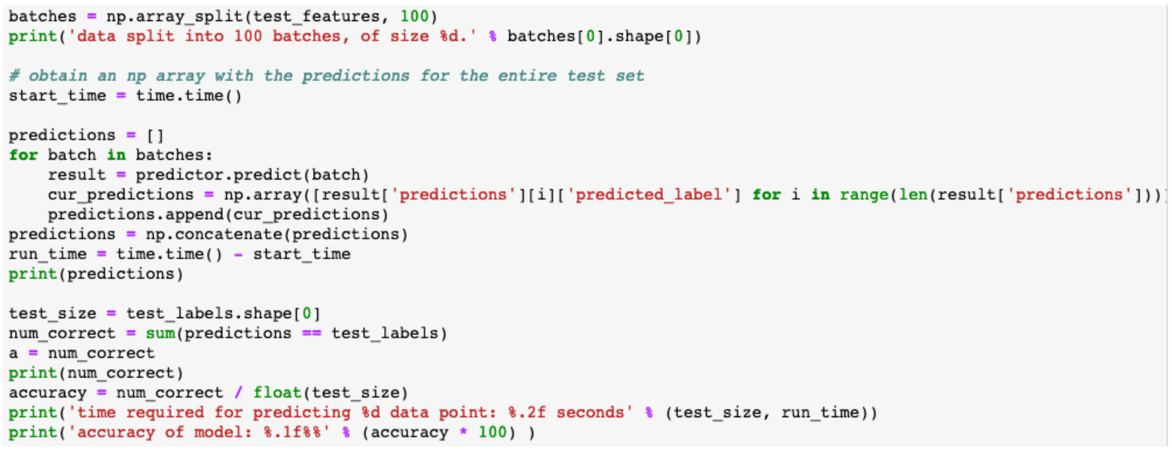
Figure 3.5.10 Function of model predictor



```python
import time

instance_type = 'ml.t2.medium'
model_name = 'user_1_knn_%s'% instance_type
endpoint_name = 'user-1-knn-ml-t2-medium-%s'% (str(time.time()).replace('.','-'))
print('setting up the endpoint..')
predictor = predictor_from_estimator(knn_estimator, model_name, instance_type, endpoint_name=endpoint_name)
```

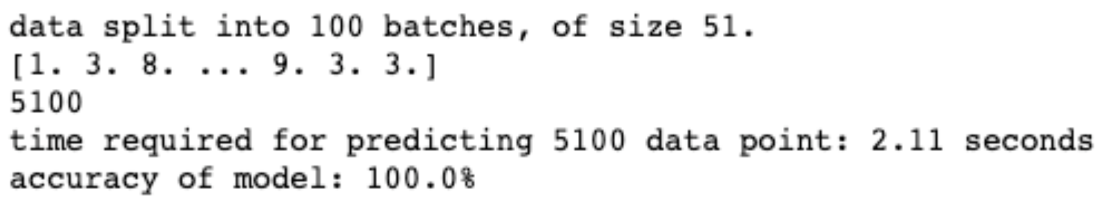
Figure 3.5.11 Model deployment

After done the model training job, the model will deploy to SageMaker as an endpoint and ready to use for the prediction jobs later. In the figure 3.5.10, a function is defined that to setup the content type, serializer and deserializer of the model, the content type of the model is in text/csv, the serializer is csv_serializer and the deserializer is json_deserializer. Next, the instance type that to use in deployment is ml.t2.medium, t is the type for general purpose use and burstable, good for changing workloads. While the medium stand for the size. Create the model name and the endpoint name as a variable that easy to fit in the function. Lastly, the model will be created and the model will deploy to SageMaker at the same time.



```
batches = np.array_split(test_features, 100)
print('data split into 100 batches, of size %d.' % batches[0].shape[0])

# obtain an np array with the predictions for the entire test set
start_time = time.time()

predictions = []
for batch in batches:
    result = predictor.predict(batch)
    cur_predictions = np.array([result['predictions'][i]['predicted_label'] for i in range(len(result['predictions']))])
    predictions.append(cur_predictions)
predictions = np.concatenate(predictions)
run_time = time.time() - start_time
print(predictions)

test_size = test_labels.shape[0]
num_correct = sum(predictions == test_labels)
a = num_correct
print(num_correct)
accuracy = num_correct / float(test_size)
print('time required for predicting %d data point: %.2f seconds' % (test_size, run_time))
print('accuracy of model: %.1f%%' % (accuracy * 100) )
```

Figure 3.5.12 Model testing



```
data split into 100 batches, of size 51.
[1. 3. 8. ... 9. 3. 3.]
5100
time required for predicting 5100 data point: 2.11 seconds
accuracy of model: 100.0%
```

Figure 3.5.13 Result of model

Based on the figure 3.5.12 shows that the model will be test to calculate the accuracy of the model. The test features data will split 100 as batches, then the loop the batches. In the loop, each batch will use to run the predict with train model, then the result will store in the predictions array that to use for calculation later. While loop has done, a sum will do, by comparing how many test labels is equal with the predictions and sum the total number of correct. Then the number of correct will divide by test size to calculate the accuracy of the model.

3.6 Prediction

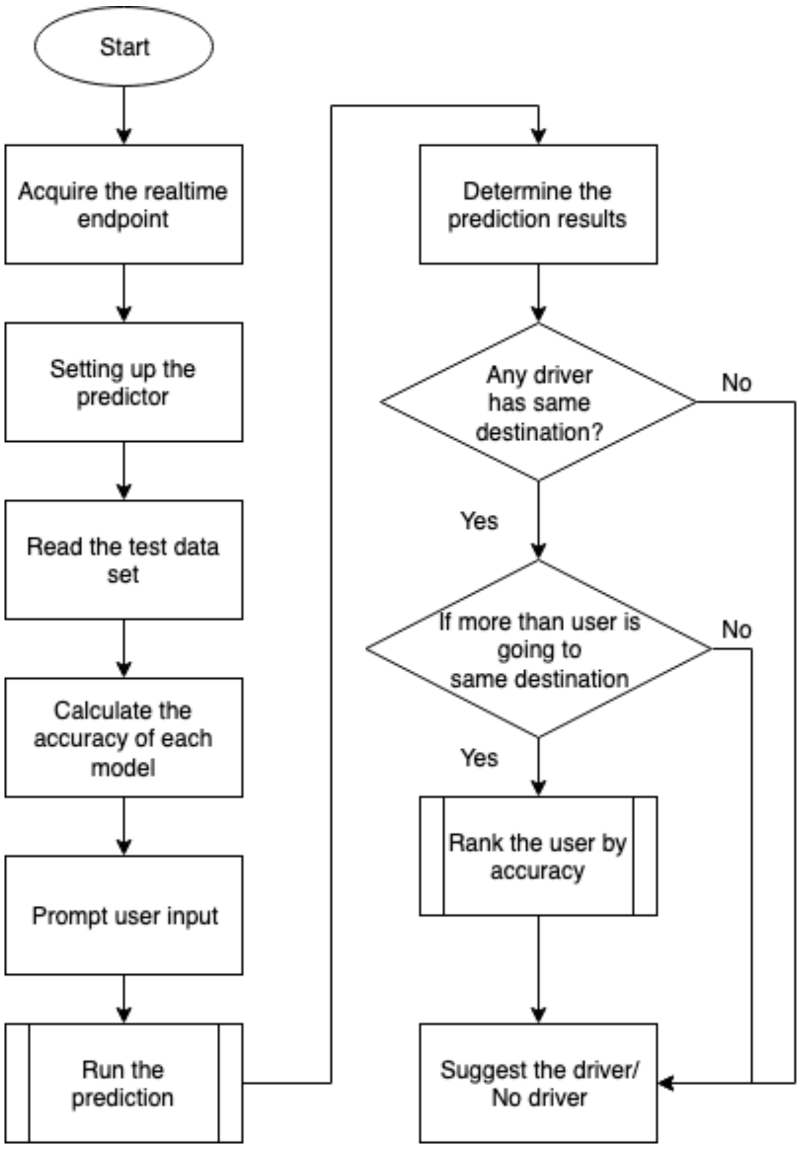
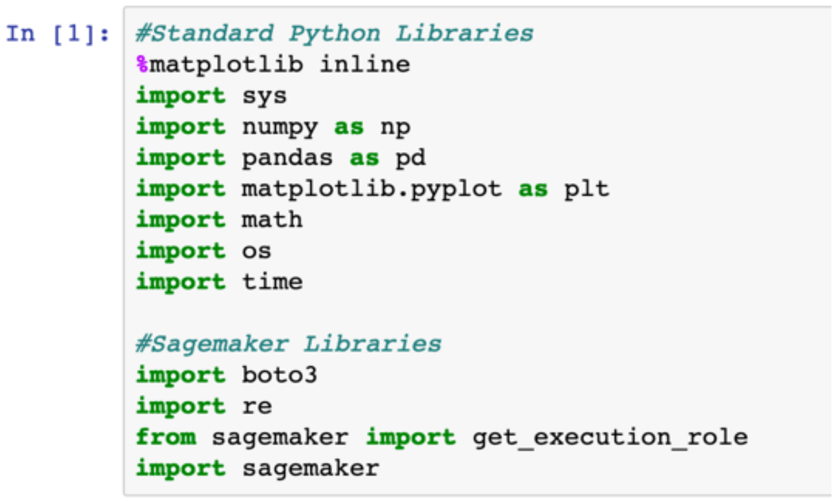

Start
Acquire the realtime endpoint
Setting up the predictor
Read the test data set
Calculate the accuracy of each model
Prompt user input
Run the prediction
Determine the prediction results
Any driver has same destination?
No
Yes
If more than user is going to same destination
No
Yes
Rank the user by accuracy
Suggest the driver/ No driver

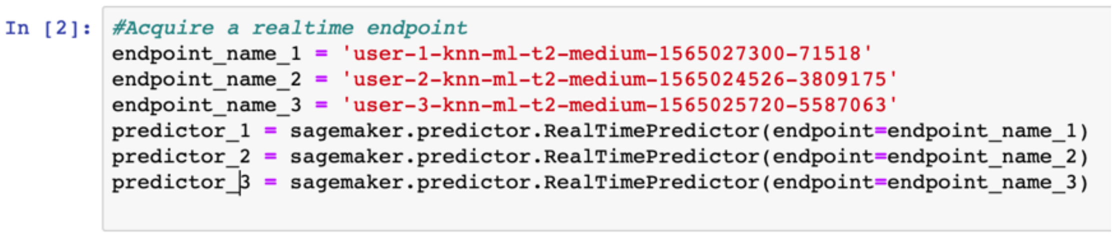
Figure 3.6.1 Flow of the prediction process

Based on the figure 3.6.1 shows the overall flow of the prediction process, the first step is to acquire the endpoint as realtime predictor, and then setting up the predictor detail. Next read the test data that to use for calculation of the accuracy of the each user model, the accuracy model that calculated will be used for ranking later if meet the condition. User will input the date, time and location and the prediction process will run to determine the predictions result for each user model. After the prediction result is done, and the result meet the condition that more than one user are going to the same destination location, the it will perform the ranking process and determine the user to be suggested. If the condition is only one user is going to the same place, then the user directly suggested.



```
In [1]: #Standard Python Libraries
        %matplotlib inline
        import sys
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import math
        import os
        import time

        #Sagemaker Libraries
        import boto3
        import re
        from sagemaker import get_execution_role
        import sagemaker
```

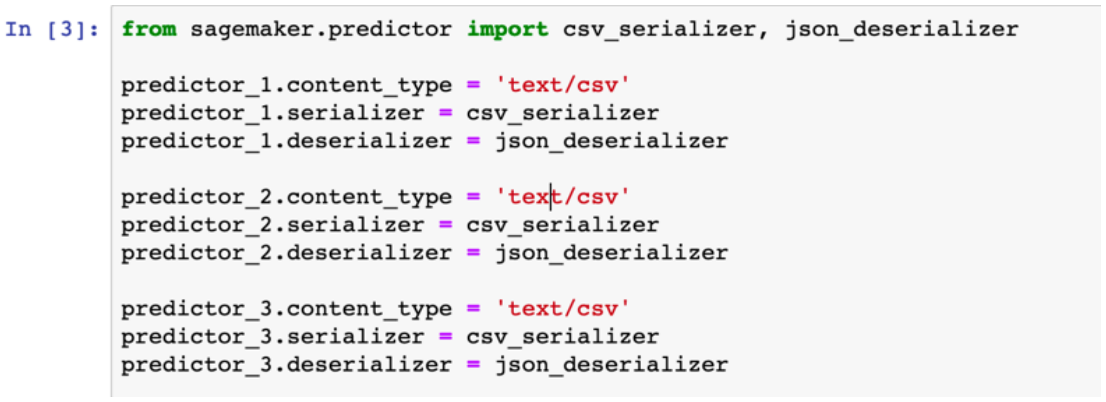
Figure 3.6.2 Python and SageMaker libraries used



```
In [2]: #Acquire a realtime endpoint
        endpoint_name_1 = 'user-1-knn-ml-t2-medium-1565027300-71518'
        endpoint_name_2 = 'user-2-knn-ml-t2-medium-1565024526-3809175'
        endpoint_name_3 = 'user-3-knn-ml-t2-medium-1565025720-5587063'
        predictor_1 = sagemaker.predictor.RealTimePredictor(endpoint=endpoint_name_1)
        predictor_2 = sagemaker.predictor.RealTimePredictor(endpoint=endpoint_name_2)
        predictor_3 = sagemaker.predictor.RealTimePredictor(endpoint=endpoint_name_3)
```

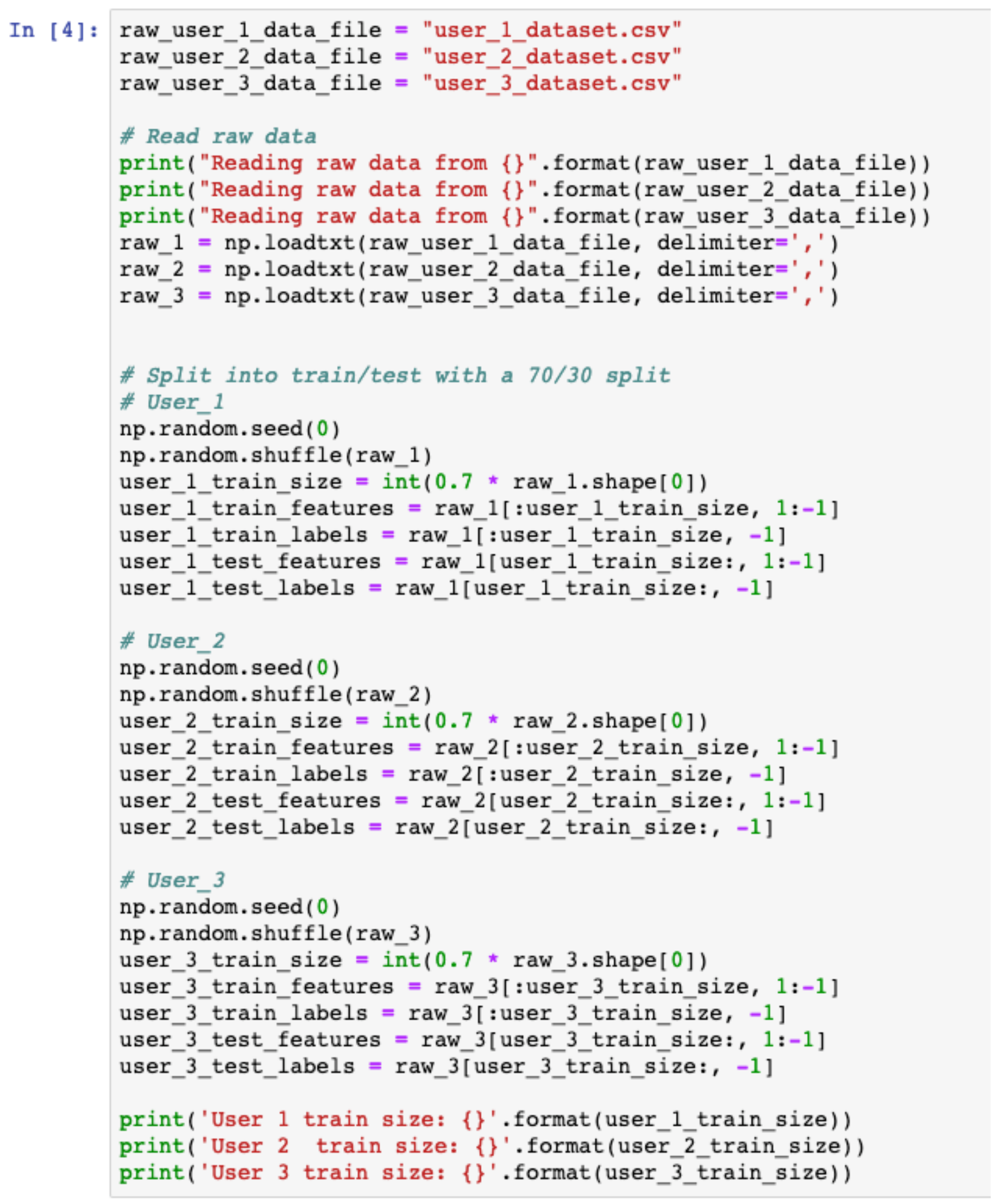
Figure 3.6.3 Access the endpoint

At the beginning stage, the libraries that need to use in the program will import that shown in the figure 3.6.2. Next, declare the endpoint name that must be same with the endpoint name that hosting in the SageMaker. Using the realtime predictor function to call the endpoint to use in the predictor later.



```
In [3]: from sagemaker.predictor import csv_serializer, json_deserializer

        predictor_1.content_type = 'text/csv'
        predictor_1.serializer = csv_serializer
        predictor_1.deserializer = json_deserializer

        predictor_2.content_type = 'text/csv'
        predictor_2.serializer = csv_serializer
        predictor_2.deserializer = json_deserializer

        predictor_3.content_type = 'text/csv'
        predictor_3.serializer = csv_serializer
        predictor_3.deserializer = json_deserializer
```

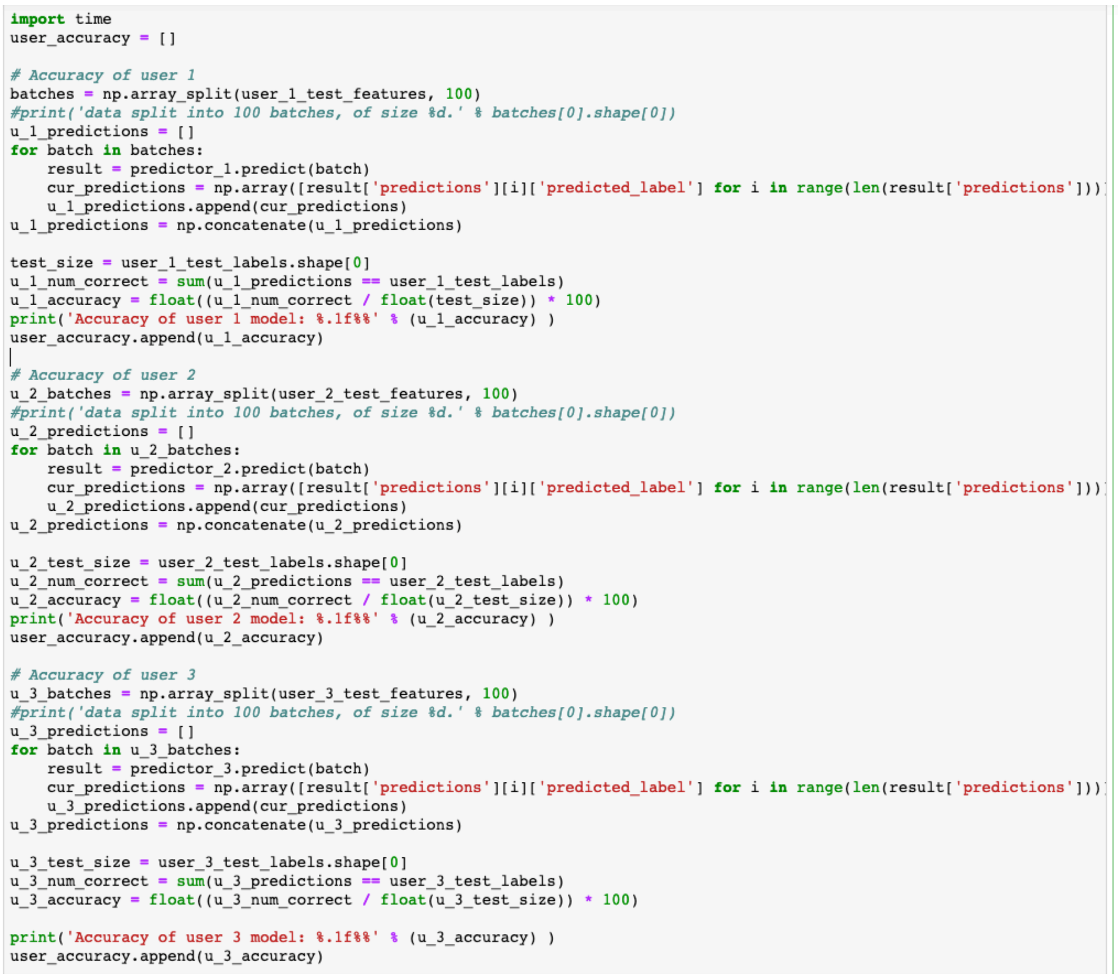
Figure 3.6.4 Setting up the predictor type



```
In [4]: raw_user_1_data_file = "user_1_dataset.csv"
        raw_user_2_data_file = "user_2_dataset.csv"
        raw_user_3_data_file = "user_3_dataset.csv"

        # Read raw data
        print("Reading raw data from {}".format(raw_user_1_data_file))
        print("Reading raw data from {}".format(raw_user_2_data_file))
        print("Reading raw data from {}".format(raw_user_3_data_file))
        raw_1 = np.loadtxt(raw_user_1_data_file, delimiter=',')
        raw_2 = np.loadtxt(raw_user_2_data_file, delimiter=',')
        raw_3 = np.loadtxt(raw_user_3_data_file, delimiter=',')


        # Split into train/test with a 70/30 split
        # User_1
        np.random.seed(0)
        np.random.shuffle(raw_1)
        user_1_train_size = int(0.7 * raw_1.shape[0])
        user_1_train_features = raw_1[:user_1_train_size, 1:-1]
        user_1_train_labels = raw_1[:user_1_train_size, -1]
        user_1_test_features = raw_1[user_1_train_size:, 1:-1]
        user_1_test_labels = raw_1[user_1_train_size:, -1]

        # User_2
        np.random.seed(0)
        np.random.shuffle(raw_2)
        user_2_train_size = int(0.7 * raw_2.shape[0])
        user_2_train_features = raw_2[:user_2_train_size, 1:-1]
        user_2_train_labels = raw_2[:user_2_train_size, -1]
        user_2_test_features = raw_2[user_2_train_size:, 1:-1]
        user_2_test_labels = raw_2[user_2_train_size:, -1]

        # User_3
        np.random.seed(0)
        np.random.shuffle(raw_3)
        user_3_train_size = int(0.7 * raw_3.shape[0])
        user_3_train_features = raw_3[:user_3_train_size, 1:-1]
        user_3_train_labels = raw_3[:user_3_train_size, -1]
        user_3_test_features = raw_3[user_3_train_size:, 1:-1]
        user_3_test_labels = raw_3[user_3_train_size:, -1]

        print('User 1 train size: {}'.format(user_1_train_size))
        print('User 2  train size: {}'.format(user_2_train_size))
        print('User 3 train size: {}'.format(user_3_train_size))
```

Figure 3.6.5 Read the user dataset and split into train and test

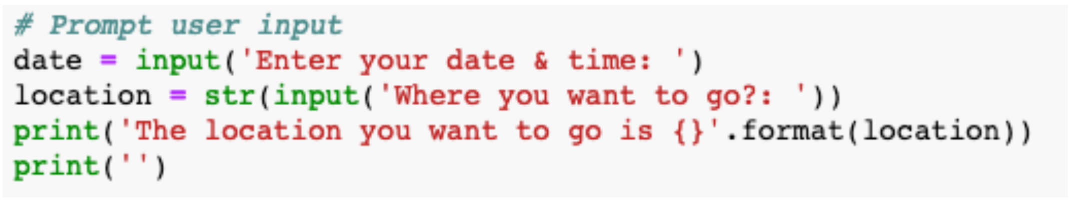
From the figure 3.6.5 shows the 3 user datasets will read and define a seed that to make the random number predictable, and shuffle the data. Next, the data will be split into ratio of 70:30, 70% for train and 30% for test. After split the data, the day and times in the data will become the features and the location will become the labels for model training, these features and labels will be used to calculated the accuracy for each user model later.



```
import time
user_accuracy = []

# Accuracy of user 1
batches = np.array_split(user_1_test_features, 100)
#print('data split into 100 batches, of size %d.' % batches[0].shape[0])
u_1_predictions = []
for batch in batches:
    result = predictor_1.predict(batch)
    cur_predictions = np.array([result['predictions'][i]['predicted_label'] for i in range(len(result['predictions']))
    u_1_predictions.append(cur_predictions)
u_1_predictions = np.concatenate(u_1_predictions)

test_size = user_1_test_labels.shape[0]
u_1_num_correct = sum(u_1_predictions == user_1_test_labels)
u_1_accuracy = float((u_1_num_correct / float(test_size)) * 100)
print('Accuracy of user 1 model: %.1f%%' % (u_1_accuracy) )
user_accuracy.append(u_1_accuracy)
|
# Accuracy of user 2
u_2_batches = np.array_split(user_2_test_features, 100)
#print('data split into 100 batches, of size %d.' % batches[0].shape[0])
u_2_predictions = []
for batch in u_2_batches:
    result = predictor_2.predict(batch)
    cur_predictions = np.array([result['predictions'][i]['predicted_label'] for i in range(len(result['predictions']))
    u_2_predictions.append(cur_predictions)
u_2_predictions = np.concatenate(u_2_predictions)

u_2_test_size = user_2_test_labels.shape[0]
u_2_num_correct = sum(u_2_predictions == user_2_test_labels)
u_2_accuracy = float((u_2_num_correct / float(u_2_test_size)) * 100)
print('Accuracy of user 2 model: %.1f%%' % (u_2_accuracy) )
user_accuracy.append(u_2_accuracy)

# Accuracy of user 3
u_3_batches = np.array_split(user_3_test_features, 100)
#print('data split into 100 batches, of size %d.' % batches[0].shape[0])
u_3_predictions = []
for batch in u_3_batches:
    result = predictor_3.predict(batch)
    cur_predictions = np.array([result['predictions'][i]['predicted_label'] for i in range(len(result['predictions']))
    u_3_predictions.append(cur_predictions)
u_3_predictions = np.concatenate(u_3_predictions)

u_3_test_size = user_3_test_labels.shape[0]
u_3_num_correct = sum(u_3_predictions == user_3_test_labels)
u_3_accuracy = float((u_3_num_correct / float(u_3_test_size)) * 100)

print('Accuracy of user 3 model: %.1f%%' % (u_3_accuracy) )
user_accuracy.append(u_3_accuracy)
```

Figure 3.6.6 Accuracy calculation for each user model

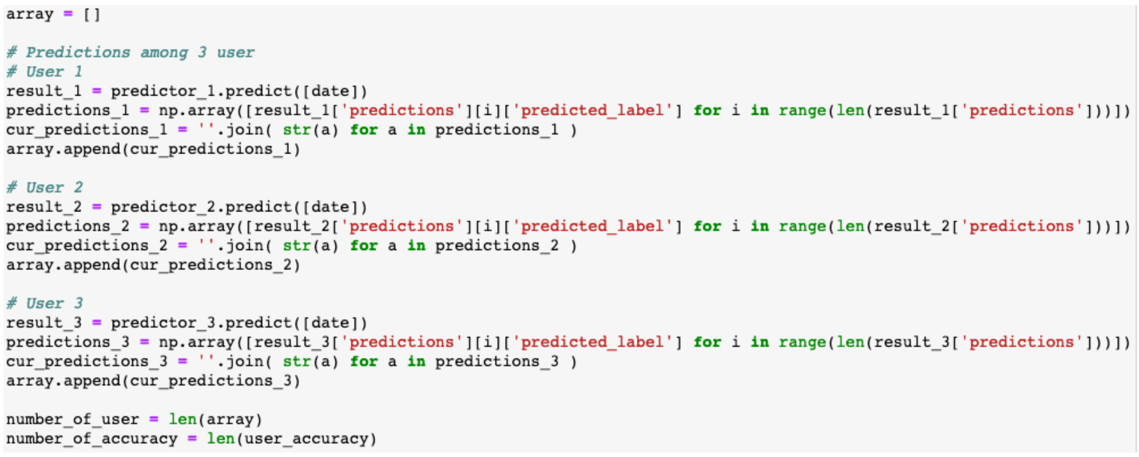
Based on the figure 3.6.6 shows that the each of the 3 users model will be test to calculate the accuracy of the model. The test features data will split 100 as batches, then the loop the batches. In the loop, each batch will use to run the predict with train model, then the result will store in the predictions array that to use for calculation later. While loop has done, a sum will do, by comparing how many test labels is equal with the predictions and sum the total number of correct. Then the number of correct will divide by test size to calculate the accuracy of the model.



```
# Prompt user input
date = input('Enter your date & time: ')
location = str(input('Where you want to go?: '))
print('The location you want to go is {}'.format(location))
print('')
```

Figure 3.6.7 User prompt input

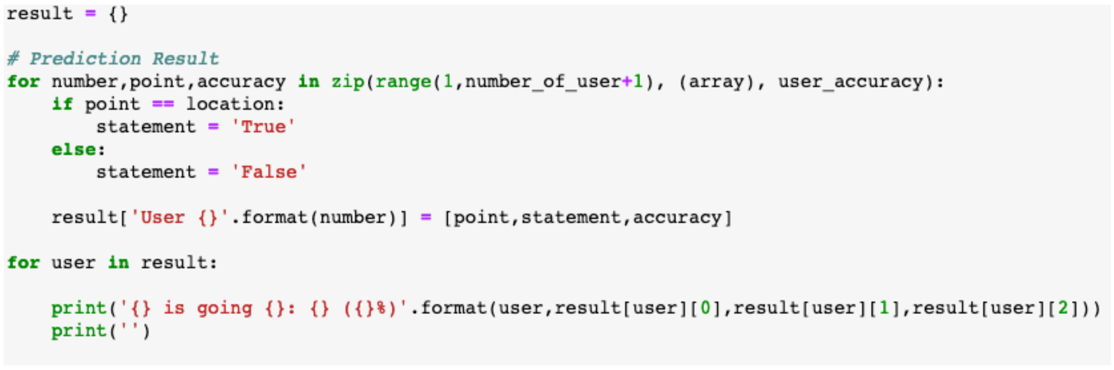
In the figure 3.6.7 shows the user prompt input, this is allowed user to select which of the day, what time and location.



```
array = []

# Predictions among 3 user
# User 1
result_1 = predictor_1.predict([date])
predictions_1 = np.array([result_1['predictions'][i]['predicted_label'] for i in range(len(result_1['predictions']))])
cur_predictions_1 = ''.join( str(a) for a in predictions_1 )
array.append(cur_predictions_1)

# User 2
result_2 = predictor_2.predict([date])
predictions_2 = np.array([result_2['predictions'][i]['predicted_label'] for i in range(len(result_2['predictions']))])
cur_predictions_2 = ''.join( str(a) for a in predictions_2 )
array.append(cur_predictions_2)

# User 3
result_3 = predictor_3.predict([date])
predictions_3 = np.array([result_3['predictions'][i]['predicted_label'] for i in range(len(result_3['predictions']))])
cur_predictions_3 = ''.join( str(a) for a in predictions_3 )
array.append(cur_predictions_3)

number_of_user = len(array)
number_of_accuracy = len(user_accuracy)
```

Figure 3.6.8 Prediction process

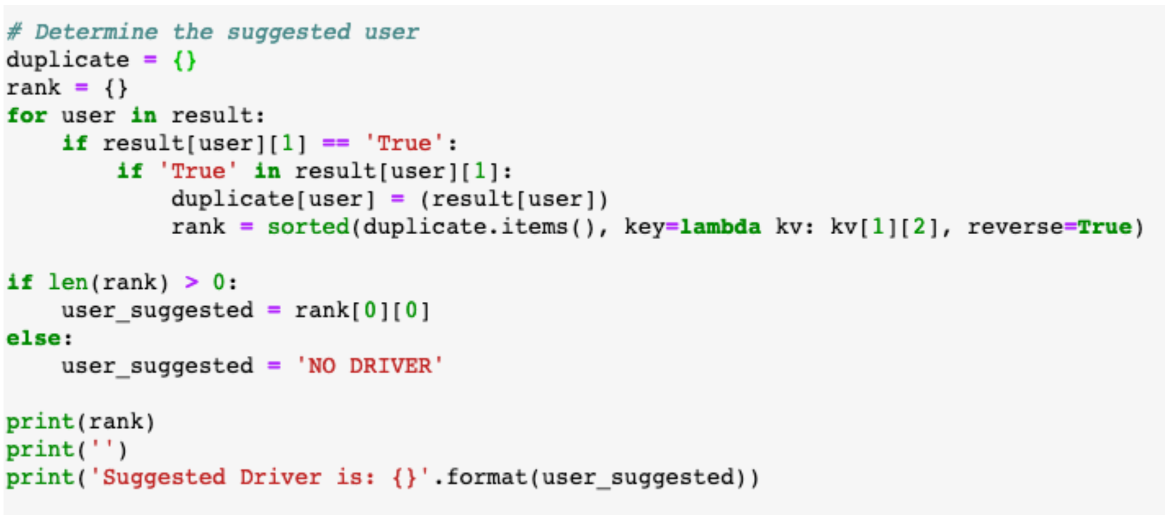
After the user enter the day, time and location, the three predictors of each user will start to run the prediction. From the figure 3.6.8, the result of the prediction will present "{predictions: predicted label: 2.0}". To convert the result format in "2.0", using the np.array function to retrieve the value and the value become "{2.0}". But that's not the result format that needed, so using then join function to remove the curly bracket and the result will store into an array.



```
result = {}

# Prediction Result
for number,point,accuracy in zip(range(1,number_of_user+1), (array), user_accuracy):
    if point == location:
        statement = 'True'
    else:
        statement = 'False'

    result['User {}'.format(number)] = [point,statement,accuracy]

for user in result:

    print('{} is going {}: {} ({}%)'.format(user,result[user][0],result[user][1],result[user][2]))
    print('')
```

Figure 3.6.9 Prediction Result

Based on the Figure 3.6.9 shows the for loop of prediction result, in this for loop it will determine which driver has the same destination location with user. If the statement driver has same destination with user then the statement is true, else the user doesn't have same destination with user then the statement will be false. Each of the point, statement and accuracy of driver will be stored in a list, and the list will use for next step.



```
# Determine the suggested user
duplicate = {}
rank = {}
for user in result:
    if result[user][1] == 'True':
        if 'True' in result[user][1]:
            duplicate[user] = (result[user])
            rank = sorted(duplicate.items(), key=lambda kv: kv[1][2], reverse=True)

if len(rank) > 0:
    user_suggested = rank[0][0]
else:
    user_suggested = 'NO DRIVER'

print(rank)
print('')
print('Suggested Driver is: {}'.format(user_suggested))
```

Figure 3.6.10 Determination of driver suggestion

From the figure above 3.6.10 shows the for loop that to determine the suggestion driver to user, there is a duplicate list will be defined. During the for loop, each drivers will be compared to determine which driver has the true statement and that will be the suggested driver to the user. While there have two driver going to same destination that same the user at the same time, the drivers will store in the list and perform the lambda function to determine the high accuracy, and sort the highest accuracy driver at the first space in the list, then the driver will be ranked by the accuracy and with the highest accuracy will be suggested to the user.

Chapter 4: System Methodology

4.1 Methodologies and General Work Procedures

This proposed system provides a ride sharing services by Google Maps Analytics. Thus, passenger will pick their source point and destination point, then the driver who overlap on the path selected by passenger will suggest to the passenger.

Therefore, user can using the interface applications to request a ride to archive their travel destination. Authentication is the very first step in the project, in this process user will get an account with a unique user id to validate and classify the user identity. In the second stage of the project, the mobile app will run as background app refresh, and it will start to track and collect the data which is the user visited location. Every time the collected data will automatically upload and store into database, the location data will classify be the datetime. Next, once the data are collected, the data will export into JSON file format and the data will extract from the JSON file and proceed a conversion from JSON to CSV file format. After the conversion is done, the csv dataset will proceed the data augmentation process which is extrapolate the data in the data set, then upload these dataset to the AWS SageMaker and the split the dataset into train and test file. Next, these train and test data will upload and store into AWS S3 Bucket for ready to use in further steps. By using these data will start to training the job and create model, then use the model and deploy as an endpoint in the SageMaker. Lastly, the hosted endpoint will be as an API that use to run the predictions to predict the driver is going to which location at the specific day and time, and it will be determined the most compatible driver to the user.

4.2 Tools to use

Xcode

Xcode is an integrated development environment for macOS that contain a set of development tools that will be used to develop the main program of this project. The main program will be develop using the Swift, it's designed to give developers for building apps for iOS, Mac, Apple Watch and Apple TV. In this project, a mobile app will be developed and the main of program function such as the user authentication, 2D location coordinates extraction from map, location data collection and reverse geocoding will develop in the Xcode IDE. In addition, the program will also be used to call the API key to connect with database, which is Firebase realtime database, to update and retrieve the user information from the database.

Pycharm

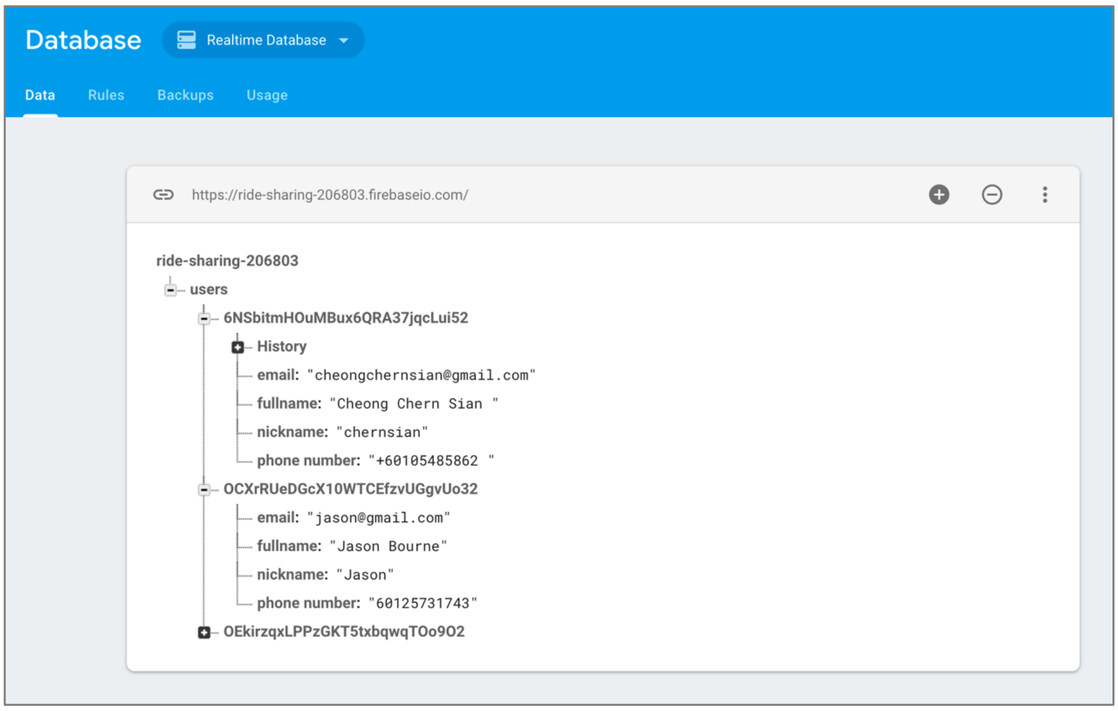
Pycharm is an integrated development environment used in computer programming tool that also will be used to develop the program of this project. This program will be develop using the Python, it's an object-oriented programming language, it's supporting all kind of different data formats and it can use comma-separated value document or JSON sourced from the web. Thus, the algorithms in this project are JSON data extraction and conversion data from JSON into CSV format.

Amazon SageMaker

Amazon SageMaker is a managed machine learning service. Data scientists and developers are able to build and train models, and then deploy the model into a production-ready hosted environment. With the integrated Jupyter notebook instance to access the data that collected and perform exploration and analysis. By using the algorithms that provided by SageMaker, k-nearest neighbors (KNN) algorithm is used in this project to analysis and train the collected data. After the training job is done, the model will be hosted in the SageMaker that ready used for prediction.

Firebase Database

Firebase Realtime Database a hosted NoSQL Database to store and synchronize to mobile app and website data. Data is stored as JSON and synchronized in realtime to every connected client. Firebase Realtime database is the database that will be used in the project. It used to store the user's information and the visited location. Each user will have an unique user id as a key to identify each user. The user information will consist of the history, fullname, nickname, email and phone. The most important data will store in to the history, and inside the history will contain the visited location and the datetime will be the key of each visited location.



```
ride-sharing-206803
 users
   6NSbitmHOuMBux6QRA37jqcLui52
     History
     email: "cheongchernsian@gmail.com"
     fullname: "Cheong Chern Sian "
     nickname: "chernsian"
     phone number: "+60105485862 "
   OCXrRUeDGcX10WTCEfzvUGgvUo32
     email: "jason@gmail.com"
     fullname: "Jason Bourne"
     nickname: "Jason"
     phone number: "60125731743"
   OEkirzqxLPPzGKT5txbqwqTOo9O2
```

Figure 4.1 Firebase realtime database

4.3 User Requirements

The user requirement of this project:

- **GPS location service in smartphone must be turned on.**

GPS location service will become the core element in this project. This is because the whole project need the GPS location services to track the user location at all the time, if the service is not turned on and it will not able to track the user location. Thus, this is the main requirement in this project.

- **Ensure the mobile data is turned on.**

Mobile data also a of the main element in this project. Due to the type of database that used in the project, mobile data is needed to upload and store the data into the database when the data is collected.

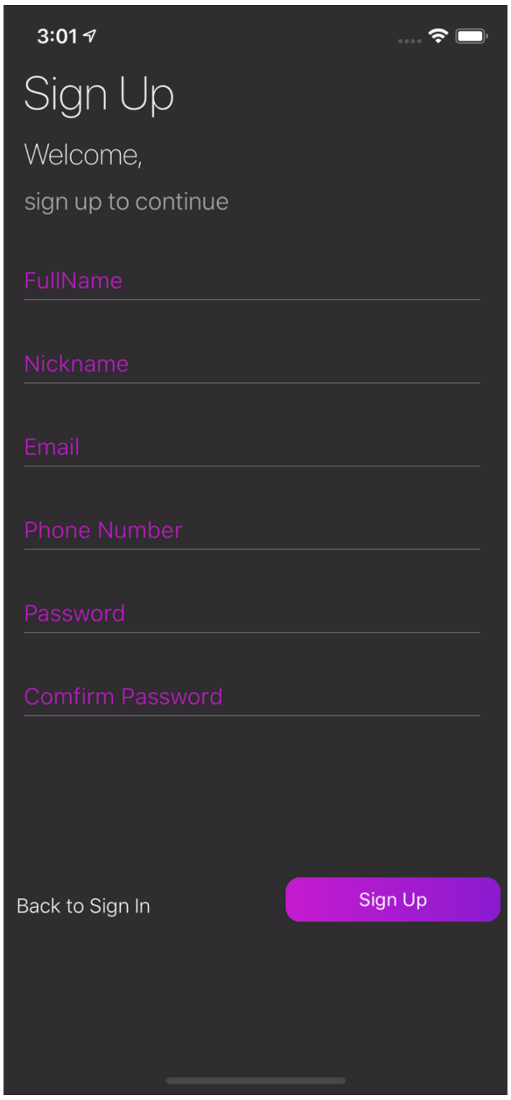
- **User required to create an account.**

Mobile authentication is the verification of a user's identity through the use a device and one or more authentication methods for secure access. Thence, user is required to register an account, an unique user id will provided for each user. Every time user log in to the mobile app, in the app will give an access token which it will be uses to identify the user.

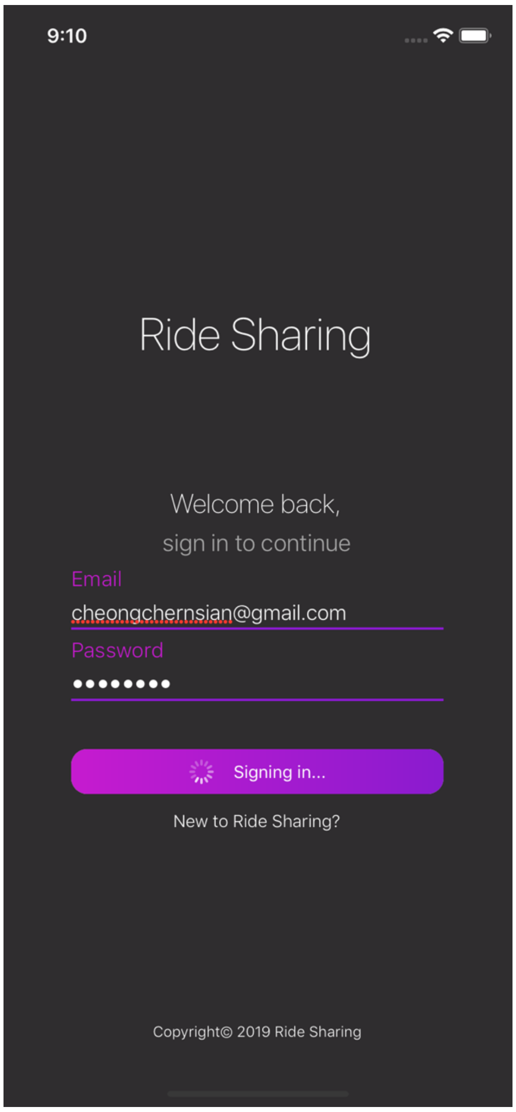
Chapter 5: Implementation and Testing

5.1 User Registration

In the account registration period, this is the first step that user need to proceed. If user doesn't have account of this mobile app, then user required to register an account before login to the mobile app. User are required fill up all the field start from fullname, nickname, email, phone number and password.



(a)

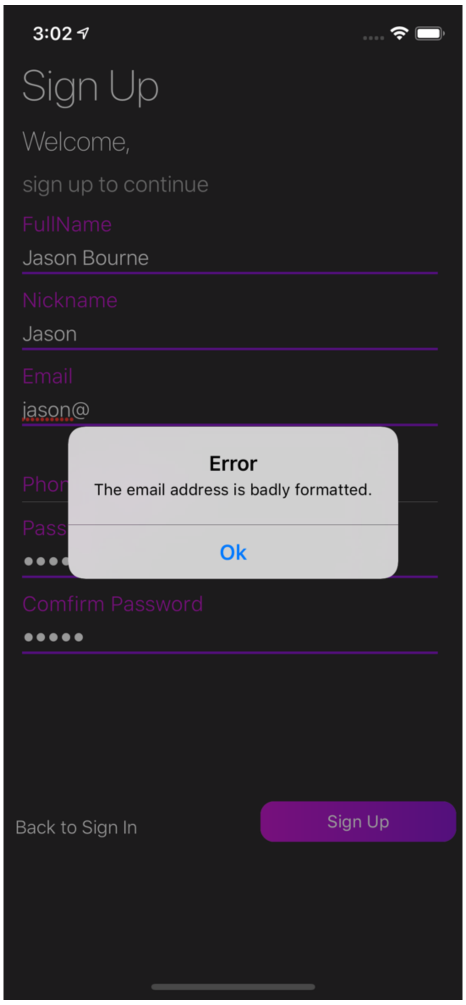


(b)

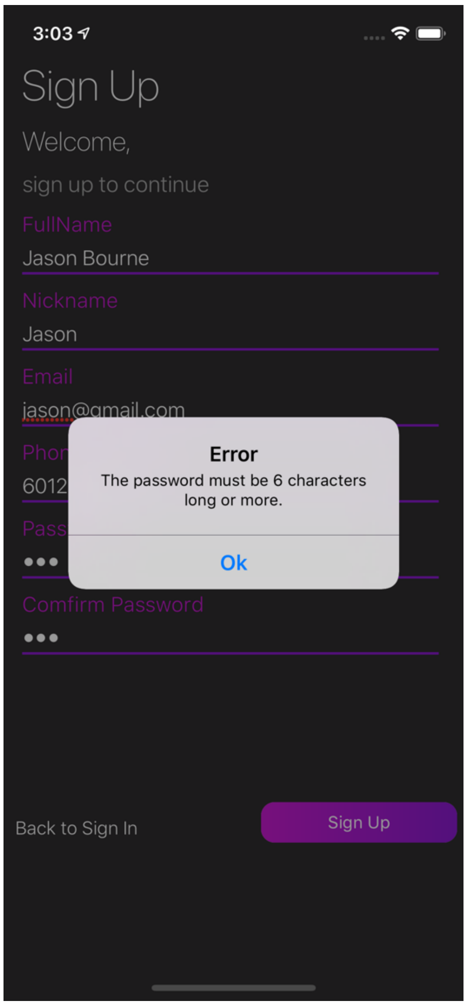
(a) Figure 5.1.1 Sign up page of the location tacker.

(b) Figure 5.1.2 Login page of the location tracker

In addition, there some text field validation during the registration process such as email address format validation and password length validation.



Error
The email address is badly formatted.
Ok



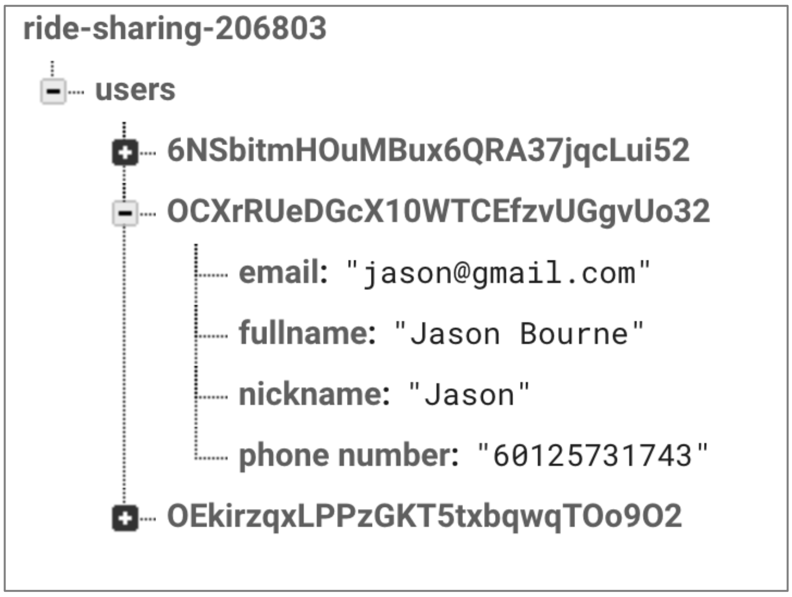
Error
The password must be 6 characters long or more.
Ok

(a) (b)

(a) Figure 5.1.3 Email address format validation

(b) Figure 5.1.4 Password length validation

When the user done the registration of the mobile app, the user information will store into the database which is Firebase realtime database and it will logged into the mobile app home screen.



```
ride-sharing-206803
  users
    6NSbitmHOuMBux6QRA37jqcLui52
    OCXrRUeDGcX10WTCEfzvUGgvUo32
      email: "jason@gmail.com"
      fullname: "Jason Bourne"
      nickname: "Jason"
      phone number: "60125731743"
    OEkirzqxLPPzGKT5txbqwqTOo9O2
```

Figure 5.1.5 User information store in realtime database.

5.2 Location Data Collection

After done the registration part, the system will proceed to the next stage that is the data collection stage. During the location data collection process, user is required to turn on the GPS location services and mobile data in the smartphone. When the mobile app meets these requirements, the mobile app will run the GPS location tracking algorithm and start to capture the location data information, the algorithm will run as a background app in the smartphone to keep tracking the user visited location. If the mobile app had tracked a visited location coordinate, the reverse geocoding algorithm will process the coordinate into a location name. Afterward, this location data will straight away upload and store inside the database. User also able get to know the visited location in the mobile app.

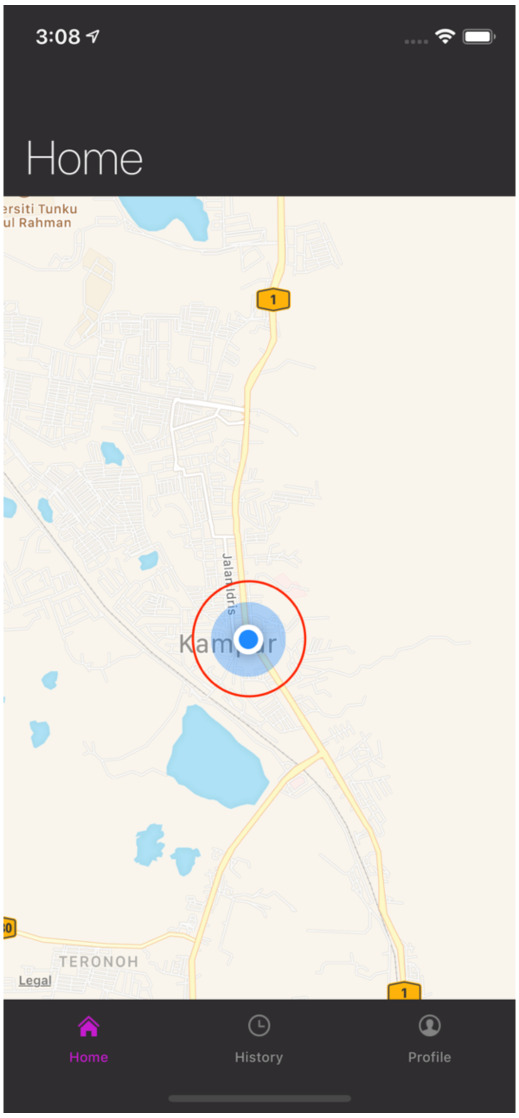
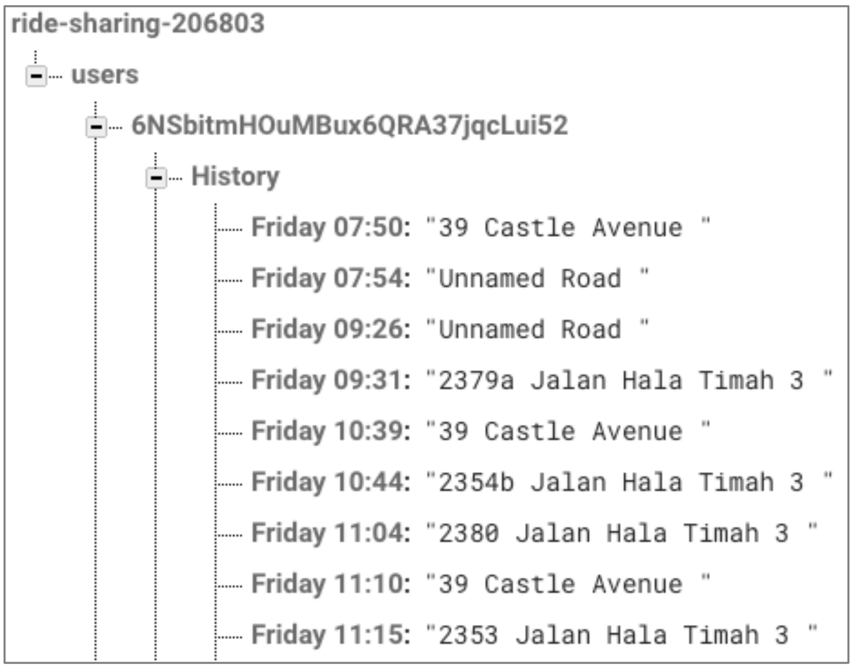
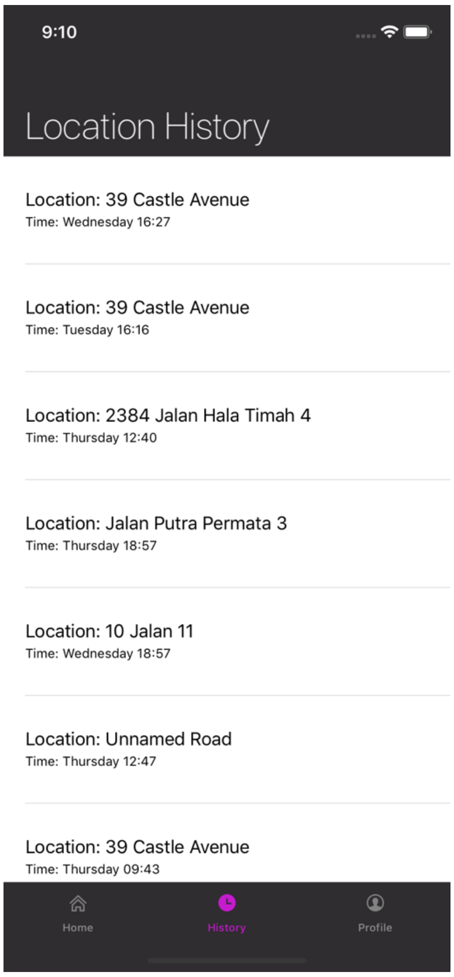


Figure 5.2.1 Red circle shows the GPS is in the tacking mode



```
ride-sharing-206803
  users
    6NSbitmHOuMBux6QRA37jqcLui52
      History
        Friday 07:50: "39 Castle Avenue "
        Friday 07:54: "Unnamed Road "
        Friday 09:26: "Unnamed Road "
        Friday 09:31: "2379a Jalan Hala Timah 3 "
        Friday 10:39: "39 Castle Avenue "
        Friday 10:44: "2354b Jalan Hala Timah 3 "
        Friday 11:04: "2380 Jalan Hala Timah 3 "
        Friday 11:10: "39 Castle Avenue "
        Friday 11:15: "2353 Jalan Hala Timah 3 "
```

Figure 5.2.2 Collected visited location data store in realtime database



9:10

Location History

Location: 39 Castle Avenue
Time: Wednesday 16:27

Location: 39 Castle Avenue
Time: Tuesday 16:16

Location: 2384 Jalan Hala Timah 4
Time: Thursday 12:40

Location: Jalan Putra Permata 3
Time: Thursday 18:57

Location: 10 Jalan 11
Time: Wednesday 18:57

Location: Unnamed Road
Time: Thursday 12:47

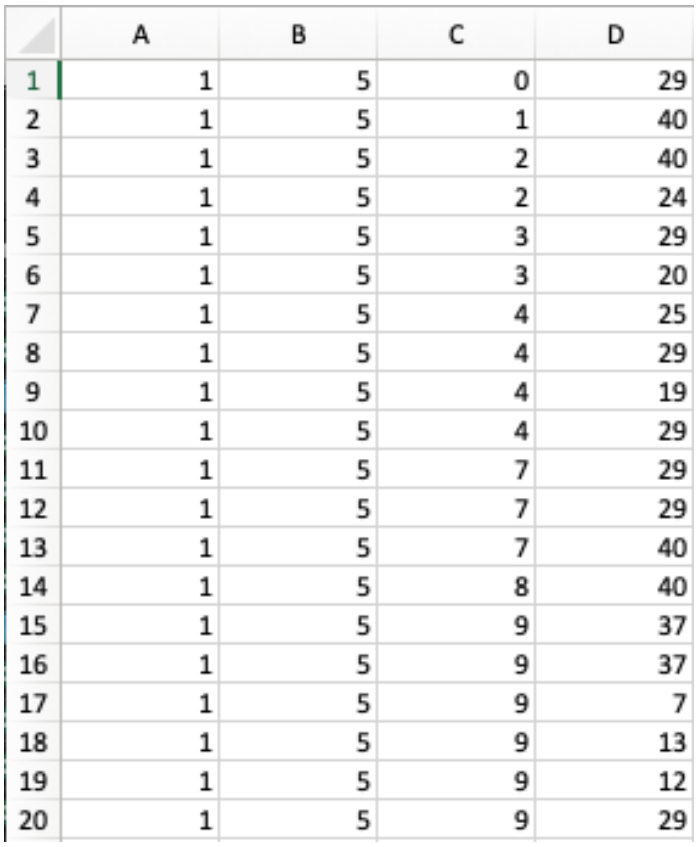
Location: 39 Castle Avenue
Time: Thursday 09:43

Figure 5.2.3 Collected visited location data able check in mobile app

In summary, this location tracker will be chosen and used during the project compare to the other location trackers such as Google Maps or Waze. Google Maps powered by Google Search and with a simple interface, but due to the background feature in the Google Maps, and this will become one of the reasons that will draining more battery energy in the smartphone. Moreover, it can be slow to load when the traffic is in a busy time. Other than that, it will occur some inaccurate condition due to the GPS problems. Secondly, Waze is a crowdsourced and community-based location tracker. It only have the car choice for the mode of transportation. However, the advertisement will pop up when the car is in stop condition which is an annoying matter to the users. Waze also have the problem of the energy consumption because it will use lot of the battery energy. In the end, this location tracker is providing a simple interface, available for walking, biking and driving mode during the location tracking. This location tracker consist a didVisit function and it can track what location that user visited recently order by timestamp. Lastly, it is known to use less mobile data and battery energy.

5.3 Data Processing

During the data processing stage, the first step is to export a JSON file from the Firebase database. By using the Firebase-admin API that call in the main program, this API that can easily retrieve the data from the Firebase database, this will be more flexible and effective. The JSON data extraction algorithm will run to extract the data from the JSON. Meanwhile, the conversion algorithm will start to convert the JSON data and save into a CSV file.



	A	B	C	D
1	1	5	0	29
2	1	5	1	40
3	1	5	2	40
4	1	5	2	24
5	1	5	3	29
6	1	5	3	20
7	1	5	4	25
8	1	5	4	29
9	1	5	4	19
10	1	5	4	29
11	1	5	7	29
12	1	5	7	29
13	1	5	7	40
14	1	5	8	40
15	1	5	9	37
16	1	5	9	37
17	1	5	9	7
18	1	5	9	13
19	1	5	9	12
20	1	5	9	29

Figure 5.3.1 Processed dataset export in CSV file

The sample size in the project will be above 15000, this is because during the model training jobs, it will need a huge amount of data to feed in the model. While the sample size is too small, it will not a demand to valid the results. However, the value of the k is too large, the margin of the error will be larger and it will take a lot of time to analysis. Thus, the large k vale will lead time consuming problem. In addition, k size is around 5 to 10 is enough for a demand to valid the results and have a good time consumption during the analysis.

5.4 Model Training

After done the data collection of each user, the next procedure will proceed to data augmentation to reduce the location that just went with extremely low frequency, then extrapolate each data in the dataset to full the amount to train a model, these data will upload to SageMaker a ready for model training. The dataset that uploaded to SageMaker will split into 70:30 ratio for train : test. In the train and test data it also need to decide the features and labels before start to train model. Next the model will start training by using the KNN algorithm and the parameters will be based on the dataset it's need, and the model will use the test data to calculate the accuracy of the model. The trained model will deploy as an endpoint in the SageMaker that will be used for predictions in the next section.

5.5 Prediction

By using the hosted endpoint, it will be used as realtime predictor, and then setting up the predictor of each user models. The accuracy will be calculated for each user model and it will be used if meet the condition. Then, user will input the date, time and location and the prediction process will run to determine the predictions result for each user model. After the prediction result is done, and the result meet the condition that more than one user are going to the same destination location, the it will perform the ranking process and determine the user to be suggested. If the condition is only one user is going to the same place, then the user directly suggested.

5.6 System Testing

When the system is completed, there some testing is needed to run to ensure the system is working in correctly. In the testing part, tester will select the different inputs to the system and determine the accuracy of the models, algorithms and predictions.

5.6.1 Test Parameter

In the test, 3 users will involve in the predictions. The student information and the unique user is stored in the Firebase Database. The testing information of the parameter used in the test is shown below:

User	Name	User ID
User 1	Cheong Chern Sian	6NSbitmHOuMBux6QRA37jqcLui52
User 2	Lim Kai Xian	yZATM2oimR9EPGwHTKSjFT6WKk2
User 3	Wong Zheng Yan	PXKYDf7VL1N1JnnIUS0GmAX0CFs1

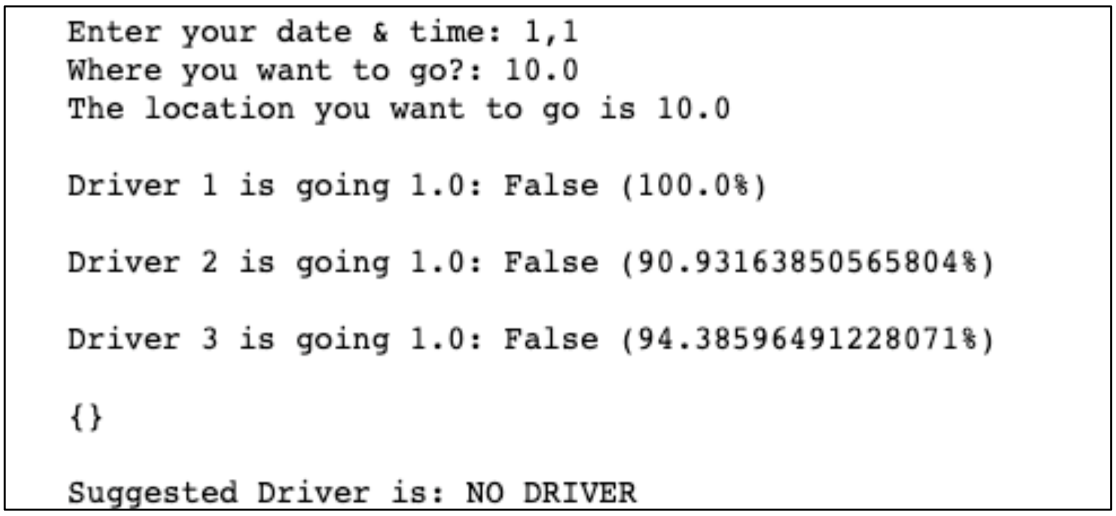
Table 5-6-1-1 Table of test parameter

5.6.2 Scenario 1: No Driver is suggested

In the first scenario, among the three drivers have no the same destination point with the user.

Test Case	Test Description	Theoretical Result	Actual Result
1	All the drivers don't have same destination point with the user.	No driver suggested	No driver suggested

Table 5-6-2-1 Table of scenario 1



```
Enter your date & time: 1,1
Where you want to go?: 10.0
The location you want to go is 10.0

Driver 1 is going 1.0: False (100.0%)

Driver 2 is going 1.0: False (90.93163850565804%)

Driver 3 is going 1.0: False (94.38596491228071%)

{}

Suggested Driver is: NO DRIVER
```

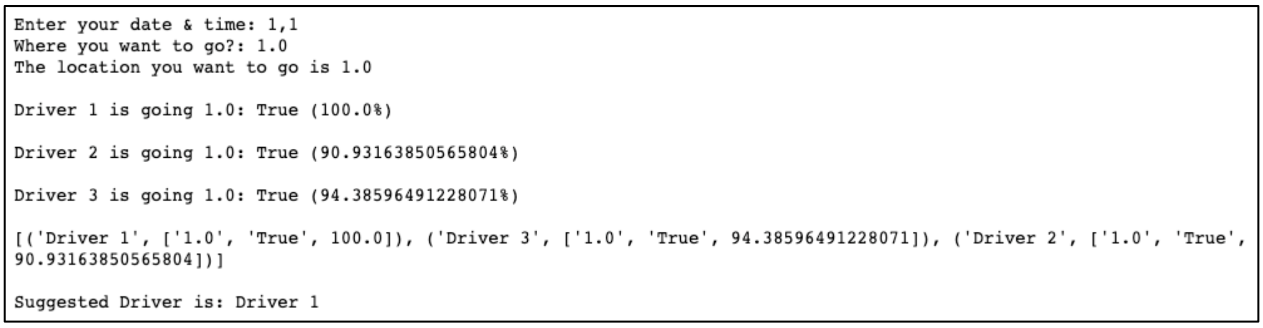
Figure 5.6.2.1 Result of scenario 1

5.6.3 Scenario 2: All drivers are going to the same destination

In the second scenario, among the three drivers are going to the same destination point with the user, but it only one driver will be suggested to user ranked by accuracy. In this case, driver 1 got the highest rank among the rest of the driver, so driver 1 will be the suggested driver.

Test Case	Test Description	Theoretical Result	Actual Result
2	All drivers are going to the same destination point.	Driver 1	Driver 1

Table 5-6-3-1 Table of scenario 2



```
Enter your date & time: 1,1
Where you want to go?: 1.0
The location you want to go is 1.0

Driver 1 is going 1.0: True (100.0%)

Driver 2 is going 1.0: True (90.931638505658048)

Driver 3 is going 1.0: True (94.38596491228071%)

[('Driver 1', ['1.0', 'True', 100.0]), ('Driver 3', ['1.0', 'True', 94.38596491228071]), ('Driver 2', ['1.0', 'True',
90.93163850565804])]

Suggested Driver is: Driver 1
```

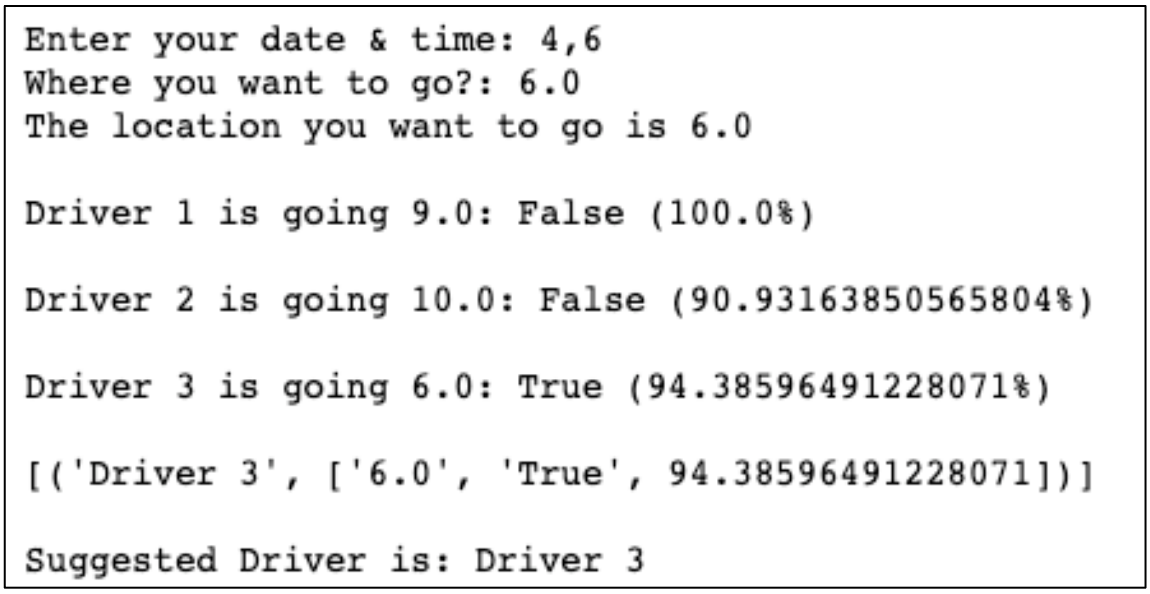
Figure 5.6.3.1 Result of scenario 2

5.6.4 Scenario 3: Only one driver going to the same destination

In the third scenario, among only one driver is going to the same destination point with the user. Hence, the driver who going to the same will be suggested. In this case, only driver 3 going to the same destination with the user, so driver 1 will be the suggested driver.

Test Case	Test Description	Theoretical Result	Actual Result
3	Only driver is going to the same destination point.	Driver 3	Driver 3

Table 5-6-4-1 Table of scenario 3



```
Enter your date & time: 4,6
Where you want to go?: 6.0
The location you want to go is 6.0

Driver 1 is going 9.0: False (100.0%)

Driver 2 is going 10.0: False (90.93163850565804%)

Driver 3 is going 6.0: True (94.38596491228071%)

[('Driver 3', ['6.0', 'True', 94.38596491228071])]

Suggested Driver is: Driver 3
```

Figure 5.6.4.1 Result of scenario 3

5.6.5 Summary of Test Result

Test Case	**Description**	**Expected Result**	**Actual Result**	**Validity of Result**
1	All the drivers doesn't have same destination point with the user.	No Driver	No Driver	✓
2	All drivers are going to the same destination point.	Driver 1	Driver 1	✓
3	Only driver is going to the same destination point.	Driver 3	Driver 3	✓

Table 5-6-5-1 Table of summary of test result

In the table 5-6-5-1 shows summary of the test result, and the system had passed all the test cases.

5.6.6 Discussion of Result

From the table 5-6-5-1 shown the result that obtained from the test cases, the accuracy of the system is 100 percent, means the system had pass all the test case. For the test case one, the system had predicted none of the driver is going the same destination that user had request, and the predictions result is exactly same with expected result, so the result of the test case 1 is valid. Next, the test case is when all drivers are going to the same destination that user request, then a classification will do and the driver will ranked by the accuracy of the model, with the highest accuracy driver will suggested to the user. Thus, the result of the test case 2 is valid. Lastly, in the test case 3, if there is only one driver going to the same destination with the user request, the only driver will suggest to the user, in this condition the actual result is same with expected result, so the result of the test case 3 is valid. From the result can shows that the system is able to match the driver to user by effective, accurately and time efficiency.

In short, the system is able to predict and determine accurately the matchmaking between driver and user with the model that had trained and built. Comparing to the current existing ride sharing services, this system is able to provide an accurate, flexible and effective matchmaking. For this system is had an unsupervised location tracker, the location tracker may help the location tracking with minimal human interaction, user are just to open the location tracker and it will keep running background in their smartphones. By using these data collect, process the data and train the data into a model, then use this model to run the predictions. Compare with the Campus Ride Sharing system and Dynamic Vehicle Pooing and Ride Sharing System Framework, the system need user create a post or poll and wait for the driver accept their request, this matchmaking method will encounter with the time consuming problem, which is not effective at all.

Chapter 6: Conclusion

6.1 Project Review and Conclusion

In conclusion, the final deliverable of this project is opportunistic ride-sharing using Google Maps Analytics. The opportunistic ride-sharing using Google Maps Analytics is to provide a more effective and efficiency to match the driver and user during the match process. With the implementation of this system, the matching process will be more time effective, and it also can enhance the flexibility and accurately while the matching process. Not only that it can be reduce unnecessary resources waste between driver and user.

As mention in chapter one, the motivation of this project is to solve the problem of the current ride sharing services that using the suggest the nearest driver to the user that is not effective, efficiency, accurately and time consuming. In this project can be a huge contribution to the ride sharing system. This is because nowadays all the demand of the ride sharing is increasing rapidly, and if every driver has not the same destination point and assign the user, it will be a resource wasting such as time, petrol and mind. This system is consisting of four part of main program to complete the system. These main programs are data collection by using a location tracker, data processing, model training and model predictions.

The main objectives of the project are to develop a pervasive location tracker for unsupervised data collection, the data that collected is used to train a model driver or rider trajectory behavior using HMM on Amazon SageMaker and design an end to end path discovery for opportunistic transit using time slice and compensation as state in chapter one. Furthermore, this system able to run in a good performance in term of accuracy, effective and time effective. This is because the system will predict the user behavior to determine the ride sharing services.

6.2 Limitation

The limitation of this project:

- **Poor network signal in Kampar area.**

This will become a issues in this project, when the mobile data network is weak and the visited location is not able to upload and store into database. Thus, the mobile data signal will affect the accuracy in data collection.

- **Time consuming when run the testing of GPS location tracking.**

Running a testing of GPS location tracking will become a challenge in this project. Every time modify or update the GPS location tracking algorithm, the testing was needed. When testing the functionality of the modified algorithm, it need to run the testing in outdoor, and the testing will consuming a lot of time.

- **Limited of collected data.**

Due to the time that only have 3 months to collect the user data, there is not enough time to collect the data in a huge volume to fit into the model for model training. When the data is not enough then the accuracy of the model will become lower, it may affect the prediction results.

- **Different user has different behaviors.**

In this system, user will become an important role that because of the user will carry their smartphone to anywhere, sometime a place that user go may not frequently, and if this place increase then it may affect the accuracy of the model for predictions.

6.3 Future Work

As the discussion in the previous section, there are some limitation in system that can be improved to enhance the usability and accuracy of the system. A functional website or mobile application can be developed to utilized the algorithm of the system. With a website or mobile application can enhance more efficiency of the system and it can become more commercialization value of the system.

Bibliography

Agatz, N., Erera, A.L., Savelsbergh, M.W. and Wang, X., 2011, Dynamic ride-sharing: A simulation study in metro Atlanta, *Procedia-Social and Behavioral Sciences, 17*, pp.532-550.

Amasyali, M.B. and Gul, E., 2017, November. VoIP integration for mobile ride-sharing application. In *2017 7th International Conference on Communication Systems and Network Technologies (CSNT)* (pp. 44-47).

Bicocchi, N., Mamei, M., Sassi, A. and Zambonelli, F., 2015, September. Opportunistic ride sharing via whereabouts analysis. In *Intelligent Transportation Systems (ITSC2015 IEEE 18th International Conference on* (pp. 875-881). IEEE.

Chen, C.M., Shallcross, D., Shih, Y.C., Wu, Y.C., Kuo, S.P., Hsi, Y.Y., Holderby, Y. and Chou, W., 2011, February. Smart ride share with flexible route matching. In *Advanced Communication Technology (ICACT), 2011 13th International Conference on* (pp. 1506-1510). IEEE.

Chowdhury, A., Jamal, A., Alam, R. and Palit, R., 2016, December. Campus Ride: An Environment-Friendly Ride Sharing Platform for Academic Institutions. In *Computer and Information Technology (CIT), 2016 IEEE International Conference on* (pp. 120-124). IEEE.

de Lira, V.M., Perego, R., Renso, C., Rinzivillo, S. and Times, V.C., 2018. Boosting Ride Sharing With Alternative Destinations. *IEEE Transactions on Intelligent Transportation Systems*, *19*(7), pp.2290-2300.

Farahat, A.K., Ghodsi, A. and Kamel, M.S., 2011, December. An efficient greedy method for unsupervised feature selection. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on* (pp. 161-170). IEEE.

Farin, N.J., Rimon, M.N.A.A., Momen, S., Uddin, M.S. and Mansoor, N., 2016, December. A framework for dynamic vehicle pooling and ride-sharing system. In *Computational Intelligence (IWCI), International Workshop on* (pp. 204-208). IEEE.

International Business Times UK. 2017. *The 10 most polluted cities in the world*. [online] Available at: <https://www.ibtimes.co.uk/world-environment-day-10-most-polluted-cities-world-1504260> [23 Jan. 2017].

Kuhn, H.W., 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, *2*(1-2), pp.83-97.

MaRS Discovery District. *Ride-sharing: The rise of innovative transportation services - MaRS Discovery District*. 2019. Available from: <https://www.marsdd.com/news/ride-sharing-the-rise-of-innovative-transportation-services/> [31 Mar. 2019].

Stach, C. and Brodt, A., 2011, June. vHike-a dynamic ride-sharing service for smartphones. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on* (Vol. 1, pp. 333-336). IEEE.

Poster

UNIVERSITI TUNKU ADBUL RAHMAN

Opportunistic Ride-Sharing using Google Maps Analytics

Project Developer: Cheong Chern Sian
Project Supervisor: Dr. Aun Yichiet

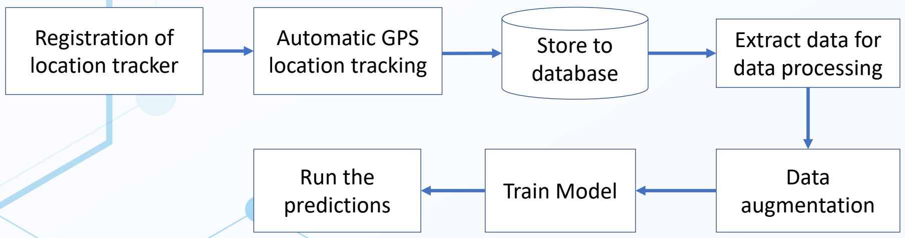
Introduction

Nowadays, there are some on-demand ride sharing commercial services are already in place such as Grab, Uber and Lyft. These services are smartphone application based that to match the drivers and users, thereby provide a taxi-like services into commuters at a fraction of the cost. This project is being proposed to develop an opportunistic ride sharing using google maps analytics that can match the driver and user who have the similar itineraries and schedules and reduce the time consume problem.

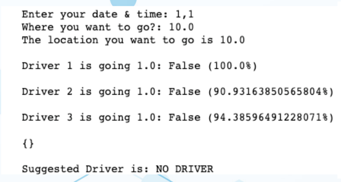
Objectives

- To develop a pervasive location tracker for unsupervised data collection.
- To model driver or rider trajectory behavior using HMM on AWS SageMaker.
- To design an end-to-end path discovery for opportunistic transit using time slicing and compensation.

Description and Methodology

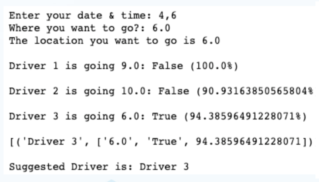

Registration of location tracker
Automatic GPS location tracking
Store to database
Extract data for data processing
Data augmentation
Train Model
Run the predictions

Result



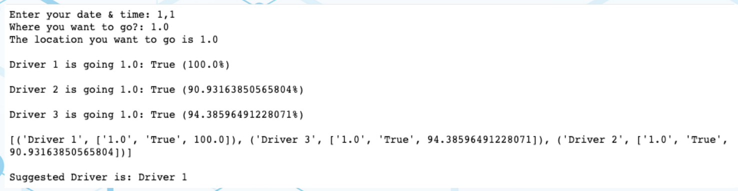
```
Enter your date & time: 1,1
Where you want to go?: 10.0
The location you want to go is 10.0

Driver 1 is going 1.0: False (100.0%)

Driver 2 is going 1.0: False (90.93163850565804%)

Driver 3 is going 1.0: False (94.38596491228071%)

{}

Suggested Driver is: NO DRIVER
```

All drivers don't have same destination point with user, no driver will suggest.



```
Enter your date & time: 4,6
Where you want to go?: 6.0
The location you want to go is 6.0

Driver 1 is going 9.0: False (100.0%)

Driver 2 is going 10.0: False (90.93163850565804%

Driver 3 is going 6.0: True (94.38596491228071%)

[('Driver 3', ['6.0', 'True', 94.38596491228071])

Suggested Driver is: Driver 3
```

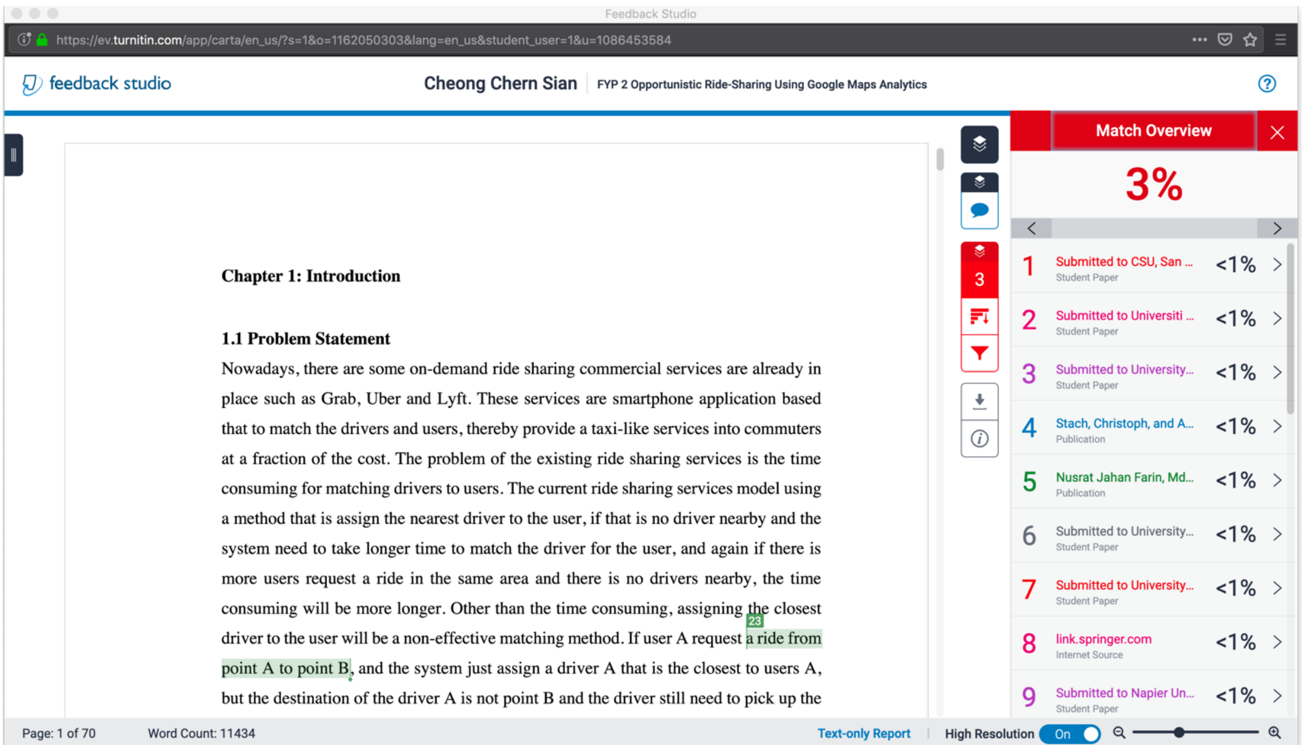
Only suggest the driver to user when one of the driver going to the same destination with user.



```
Enter your date & time: 1,1
Where you want to go?: 1.0
The location you want to go is 1.0

Driver 1 is going 1.0: True (100.0%)

Driver 2 is going 1.0: True (90.93163850565804%)

Driver 3 is going 1.0: True (94.38596491228071%)

[('Driver 1', ['1.0', 'True', 100.0]), ('Driver 3', ['1.0', 'True', 94.38596491228071]), ('Driver 2', ['1.0', 'True',
90.93163850565804])]

Suggested Driver is: Driver 1
```

When all the drivers are going to same destination with user, suggest the driver with highest accuracy.

Plagiarism check result


Feedback Studio

https://ev.turnitin.com/app/carta/en_us/?s=1&o=1162050303&lang=en_us&student_user=1&u=1086453584

feedback studio

Cheong Chern Sian | FYP 2 Opportunistic Ride-Sharing Using Google Maps Analytics

Chapter 1: Introduction

1.1 Problem Statement

Nowadays, there are some on-demand ride sharing commercial services are already in place such as Grab, Uber and Lyft. These services are smartphone application based that to match the drivers and users, thereby provide a taxi-like services into commuters at a fraction of the cost. The problem of the existing ride sharing services is the time consuming for matching drivers to users. The current ride sharing services model using a method that is assign the nearest driver to the user, if that is no driver nearby and the system need to take longer time to match the driver for the user, and again if there is more users request a ride in the same area and there is no drivers nearby, the time consuming will be more longer. Other than the time consuming, assigning the closest driver to the user will be a non-effective matching method. If user A request a ride from point A to point B, and the system just assign a driver A that is the closest to users A, but the destination of the driver A is not point B and the driver still need to pick up the

Match Overview

3%

1	Submitted to CSU, San ... Student Paper	<1%	>
2	Submitted to Universiti ... Student Paper	<1%	>
3	Submitted to University... Student Paper	<1%	>
4	Stach, Christoph, and A... Publication	<1%	>
5	Nusrat Jahan Farin, Md... Publication	<1%	>
6	Submitted to University... Student Paper	<1%	>
7	Submitted to University... Student Paper	<1%	>
8	link.springer.com Internet Source	<1%	>
9	Submitted to Napier Un... Student Paper	<1%	>

Page: 1 of 70 Word Count: 11434 Text-only Report | High Resolution On

Plagiarism check result

Turnitin Originality Report

FYP 2 Opportunistic Ride-Sharing Using Google Maps Analytics by Cheong Chern Sian

From FYPII2019-May (FYPII2019-May)

Processed on 21-Aug-2019 23:45 +08
ID: 1162050303
Word Count: 11434

Similarity Index	Similarity by Source	
3%	Internet Sources:	1%
	Publications:	0%
	Student Papers:	2%

sources:

1 < 1% match (student papers from 12-Apr-2019)
Submitted to CSU, San Jose State University on 2019-04-12

2 < 1% match (publications)
Stach, Christoph, and Andreas Brodt. "vHike - A Dynamic Ride-Sharing Service for Smartphones", 2011 IEEE 12th International Conference on Mobile Data Management, 2011.

3 < 1% match (student papers from 18-Apr-2019)
Submitted to University of Ulster on 2019-04-18

4 < 1% match (student papers from 08-Jul-2018)
Submitted to Universiti Teknologi MARA on 2018-07-08

5 < 1% match (publications)
Nusrat Jahan Farin, Md. Nur Ahsan Ali Rimon, Sifat Momen, Mohammad Shorif Uddin, Nafees Mansoor. "A framework for dynamic vehicle pooling and ride-sharing system", 2016 International Workshop on Computational Intelligence (IWCI), 2016

6 < 1% match (student papers from 05-Feb-2013)
Submitted to University of Glamorgan on 2013-02-05

7 < 1% match (student papers from 27-Apr-2017)
Submitted to University of East London on 2017-04-27

Plagiarism check result

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	
ID Number(s)	
Programme / Course	
Title of Final Year Project	

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: _________% **Similarity by source** Internet Sources: _______________% Publications: _______________% Student Papers: _______________ %	
Number of individual sources listed of more than 3% similarity: _________	
Parameters of originality required and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

______________________________ ______________________________

Signature of Supervisor Signature of Co-Supervisor

Name: __________________________ Name: __________________________

Date: ___________________________ Date: ___________________________

Check Lists

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	
Student Name	
Supervisor Name	

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Cover
	Signed Report Status Declaration Form
	Title Page
	Signed form of the Declaration of Originality
	Acknowledgement
	Abstract
	Table of Contents
	List of Figures (if applicable)
	List of Tables (if applicable)
	List of Symbols (if applicable)
	List of Abbreviations (if applicable)
	Chapters / Content
	Bibliography (or References)
	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
	Poster
	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report. ________________________ (Signature of Student) Date:	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction. ________________________ (Signature of Supervisor) Date: