

RIDE WITH ME
- Ride-sharing Application with Opportunistic AI
By
Teh Ming En

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

JAN 2023

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: _RIDE WITH ME – Ride- sharing Application with Opportunistic AI _

Academic Session: _Jan 2023 _

I _____ TEH MING EN _____

(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:

No.9 Jalan 2/8, Taman Bukit Rawang Jaya

_48000 Rawang, Selangor. _____

Aun Yichiet

Supervisor's name

Date: _28/4/2023 _

Date: 28/4/2023

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

**FACULTY/INSTITUTE* OF
INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN**

Date: 28/4/2023

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that TEH MING EN (ID No: 19ACB02716) has completed this final year project/ dissertation/ thesis* entitled “RIDE WITH ME – Ride-sharing Application with Opportunistic AI” under the supervision of Dr. Aun Yichiet (Supervisor) from the Department of Computer and Communication Technology, Faculty/Institute* of Information and Communication Technology.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,




TEH MING EN

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled “**RIDE WITH ME - Ride-sharing Application with Opportunistic AI**” 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 : _ TEH MING EN _____

Date : _ 28/4/2023 _____

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisors, Dr Aun Yichiet who has given me this golden opportunity to engage in an IT project. Appreciated all the knowledges that he shared with me enthusiastically. A million thanks to you.

Moreover, I really appreciated my parents and my family for their love, support, and continuous encouragement throughout the course.

Sincerely thanks to everyone and everything that build my knowledge and experiences throughout the journey. Hello world!

ABSTRACT

This project is to create an in-house ride sharing application, Ride with ME for UTAR-Kampar campus. It applies opportunistic AI to optimise the scheduling of riders to passenger efficiently. The app is developed by using Flutter and the opportunistic AI algorithm that developed will assist in predicting the user's daily schedule of location visit based on the user's Google Maps' data and thus, matching the driver and passengers intelligently. This will assist a lot of students from UTAR Kampar campus that indeed need a helping hand in transportation problem.

TABLE OF CONTENTS

TITLE PAGE	I
REPORT STATUS DECLARATION FORM	II
SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS.....	III
DECLARATION OF ORIGINALITY	IV
ACKNOWLEDGEMENTS	V
ABSTRACT.....	VI
TABLE OF CONTENTS	VII
LIST OF FIGURES	X
LIST OF TABLES	XI
CHAPTER 1 - PROJECT BACKGROUND.....	1
1.1 Problem Statement and Motivation	1
1.2 Project Objectives	2
1.3 Project Scope	2
1.4 Contributions.....	3
1.5 Report Organization	3
CHAPTER 2 - LITERATURE REVIEW.....	5
2.1 Previous Works on Ride sharing application and Driver Trajectory Patterns	5
2.1.1 A systematic literature review of ride-sharing platforms, user factors and barriers	5
2.1.2 Car Pooling based on Trajectories of Drivers and Requirements of Passengers	7
2.1.3 WeRide application.....	8
2.2 Limitation of Previous Studies.....	10
2.3 Proposed Solutions.....	11

CHAPTER 3 – SYSTEM METHODOLOGY /APPROACH.....	12
3.1 System Design Diagram.....	12
3.2 Use Case Diagram and Description.....	13
3.3 Ride-sharing process.....	14
CHAPTER 4 – SYSTEM DESIGN	16
4.1 Opportunistic AI Configuration Block Diagram	16
4.2 Data Pre-processing and Data Cleaning	17
4.2.1 Day of Week	17
4.2.2 Time	17
4.2.3 Destination details.....	18
4.3 Predefined Function.....	20
4.4 Data Training	24
4.5 Timetable of Destination	27
CHAPTER 5 - SYSTEM IMPLEMENTATION	28
5.1 Hardware Setup	28
5.2 Software Setup	29
5.3 Settings and Configuration	30
5.4 System Operation.....	31
5.4.1 Ride-sharing Application	31
5.4.2 User’s Google Sign in and Sign Out Function.....	33
5.4.3 Ride with ME User Interface	35
5.4.5 Firebase	38
5.5 Opportunistic AI	39
CHAPTER 6 - SYSTEM EVALUATION AND DISCUSSION	41
6.1 System Testing.....	41
6.2 Testing Setup and Result.....	43
6.3 Project Challenges	45
6.4 Objectives Evaluation	46
CHAPTER 7 - CONCLUSION AND RECOMMENDATION	47

7.1 Conclusion	47
7.2 Recommendation.....	47
REFERENCES.....	48
FINAL YEAR PROJECT WEEKLY REPORT	1
POSTER.....	3
PLAGIARISM CHECK RESULT	4
CHECKLIST FOR FYP2 THESIS SUBMISSION	6

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	WeRide	8
Figure 2.2	WeRide Trip Details	9
Figure 3.1	System Design Diagram	12
Figure 3.2	Use Case Diagram	13
Figure 3.3	Ride-sharing process	14
Figure 4.2.1	Day of Week	16
Figure 4.2.2	Time	16
Figure 4.2.3.1	Area of Kampar, Perak	17
Figure 4.2.3.2	Destination Classification	17
Figure 4.2.3.3	Destination Classification Code	18
Figure 4.3.1	Library	19
Figure 4.3.2	Get the day of week and classify time	19
Figure 4.3.3	Preprocess Google map data	20
Figure 4.3.4	getLatest	21
Figure 4.3.5	getlatlon and getkpr	21
Figure 4.3.6	getkpr	23
Figure 4.3.7	destination column	23
Figure 5.2.1	Google Cloud Platform	29
Figure 5.2.2	Flutter	29
Figure 5.2.3	Firebase	30
Figure 5.2.4	Jupyter Notebook	30
Figure 6.2.1	User interface of inputting ride request	42
Figure 6.2.3	Ride matching	43
Figure 6.3	Turn Off Location History	44

LIST OF TABLES

Table Number	Title	Page
Table 2.2	Comparison between the existing applications and proposed application	16
Table 5.1	Specifications of laptop	27
Table 5.2	Specifications of Android Phone	27

Chapter 1 - Project Background

Ride With ME is an in-house ride sharing app for UTAR Kampar campus that applies driver trajectory patterns by applying Opportunistic AI and Google Map's data. This distinctly optimizes the scheduling of riders and the efficiency of carpooling's processes. The university student may easily obtain an offer for carpooling by just one tap away.

1.1 Problem Statement and Motivation

In this globalization, the amount of usage of ride hailing apps increases every day such as Grab and MyCar and Uber. However, they are costly and unbearable for a university student who has no income and requires transportation to university rapidly. At the same time, the government had announced that the micro mobility vehicle such as e-scooter and e-bike are not allowed to be operated on the public roads on April 2022 recently. Hence, more students are concerned on the issue of transportation in Kampar. This is because there is a myriad of students rely on e-bike as their main transportation to the UTAR. Hence, they have lost their main transportation to school. Unfortunately, they have to push their e-bike along the main road to walk to the bicycle pathway for continuing ride on their e-bike to university. Moreover, most of the students lack of ability to own a car or have a car to drive on. There are also no bus services on weekend and for night shift. According to the above issue, clearly, a ride-sharing application is in a high demand among the students. However, there is still lack of any popular ride-sharing app that can be utilized, thus, they have to ask their friends for a ride by themselves. There is only some unpopular ride-sharing platform which required the students to make a posting about their requirement for a carpool. This is really not convenient, and they have to wait for the reply from the drivers unpredictably.

This project is aimed to provide a new application for students in Kampar campus to reach to their desired location by carpooling. Nowadays, people frequently utilize online maps to reach their destinations such as Google Maps and Waze. For your

information, Google Maps catches the user's current location from time to time in the background. With a ride-sharing application with driver trajectory model that is built based on the google maps data collected, the students may automatically assign a driver or student who has an identical desired destination as them effortlessly.

Through applying the driver trajectory model that will be developed, the passenger also will have less pressure on the transportation fee as the driver is having a similar destination as them. The students will not need to worry about the transportation problem especially at night when there is no UTAR bus service and it is extremely dangerous to cycle or walk to university in dark.

1.2 Project Objectives

- **To develop a ride-sharing mobile application.**

The mobile application should be user friendly and handy to be applied in daily life. The time taken to book a carpool have to be short. The application has to be introduced among possible users to maximize its advantages.

- **To develop an intelligent opportunistic AI to perform on ride-matching.**

The opportunistic AI helps to analysis the drivers' destination on a specific day and time based on their past visited location that stored in Google Map. This helps in improving the ride matching process. The opportunistic AI model will match the driver and rider automatically.

1.3 Project Scope

First of all, the Google Map's data is collected for the developing and classification process. The opportunistic AI will help in processing the data of location history of the driver and his time schedule at the location. Some of the popular classifiers like KNeighbors Classifier, Random Forest Classifier, Gradient Boosting Classifier, Decision Tree Classifier, SVM classifier and MLP classifier are being utilized to test out the most suitable algorithm to applied on this project. The algorithm that has the highest accuracy will be applied on opportunistic AI. This will vastly assist in the process of ride matching. Moreover, the ride-sharing mobile application will develop by using Flutter. The application should be handy and user-friendly. The users have to

CHAPTER 1

turn on the GPS location services while using the application. The system will capture his actual current location. The application allows the user to upload their google map data to the firebase to import to the opportunistic AI model and then to produce the user's daily destination timetable. When the user inputs a carpool request in the application, the opportunistic AI will perform a ride matching for the passenger based on the predicted driver's daily schedule to search for a perfect match.

1.4 Contributions

With this ride-sharing application with opportunistic AI successfully released, the students may easily carpool with the other students who have the identical destination as them. This will definitely help the students who encountered the transportation problem and may help to reduce their burden of transportation cost. This will also solve the parking problem in UTAR Kampar campus. Reducing the number of cars on the road will solve the traffic jams all the time. People find it hard to step out from their comfort zone, so they need to experience the ride sharing process in order to know more about its convenience and benefits. The car pollution problem will be relieved if carpooling and ride sharing are popularized among people.

The students also may easily make new friends with each other as students are having the concern of having less university friends during the covid-19 crisis. As they are having an almost identical destination to university at the time, they may be having the same course, but they have not known each other. This may help in finding their assignment group member too. This also may improve the communication skills of the users as they may communicate throughout the process of ride sharing. Last but not least, this particularly reduces the car pollution and air pollution problem which are the main concern in global.

1.5 Report Organization

The report contains a total of 7 chapters. The chapter 1 introduced the problem statements, motivations, objectives, project scope and contributions. In Chapter 2, the related project backgrounds are reviewed and their strengths and also weaknesses to be improved are commented. Next, the system methodology for the ride sharing

CHAPTER 1

application is presented in Chapter 3. After that, Chapter 4 describes the system design. Chapter 5 presents the system implementation and chapter 6 perform the system evaluation dan discussion based on the final result. Lastly, chapter 7 contains the conclusion and recommendation for future development.

Chapter 2 - Literature Review

2.1 Previous Works on Ride sharing application and Driver Trajectory Patterns

2.1.1 A systematic literature review of ride-sharing platforms, user factors and barriers

According to (Mitropoulos et al., 2021), ride sharing is the driver carpooling the others without gaining any profit, but they may share the costs with the passengers. Carpooling usually planned ahead of time by matching and connecting the driver and passengers [1]. Due to the organization, technology and economical constraints, ridesharing only present a limited uptake for now. The past research usually concentrated on the algorithm of ride-matching, pricing and its advantages only. The ride sharing platform should always allow user to input their feedback because the feedback from user is crucial to raise the quality of services and find out the safety threats. Next, user may evaluate the dependability of the driver. Some ride sharing platform allow the female user to input the choice to travel with female driver and passengers only to boost the security level. Some platforms allow users to plan the journeys in advance and also can be booked immediately.

Most of the ride sharing platform conduct destination-based matching methods. The driver may enter the pickup and drop off points and look for the passengers. The passengers may select the driver who best suits their requirements. Ride sharing services do not conduct a complex algorithm with a myriad of criteria to perform a ideal ride-match. They may made arrangements among themselves such as their meeting points and the way to recognise each other. However, TwoGo platform applied intelligent technology to analyse optimal ride match and also determine the accurate trajectories and travel time by real-time traffic.

To encourage ride sharing, some countries offer toll cost reduction, free or discounted parking fees, and discounted public transport tickets. Significantly, GoCarma platform offer toll discount by automatically detect the number of passengers in the vehicle through Bluetooth.

Apart from that, the user of ride sharing platform value the security, privacy and the behaviour of driver but not their background whereas the driver value the flexibility of driving time.

CHAPTER 2

The time cost while booking the ride is extremely important as most of the drivers will not want to wait for the passengers for too long. However, pre-plan ride sharing may be not convenient for some of the users who have to book the ride instantly and who value the flexibility. The time taken to book a ride should be shortened. Next, the increasing in fuel prices also vastly influence the ride sharing.

The ride sharing platform can linked with the social media like Facebook to integrate with the information database of driver or passenger in order to enhance the ride matching and also protect the safety of passengers. It also suggests providing the rating system for both drivers and passengers to enhance their user experiences. Due to the pandemic, users have to submit their vaccine certificate for carpooling. The application also should provide an alarm button for emergency condition, and it will share the user's real time location and information to help with.

The limitation presented is the matching algorithm and optimization available for now. The paper suggest that the development of matching algorithm should concentrate in future to increase the effectiveness. There is genuinely inadequate of ride sharing platforms.

2.1.2 Car Pooling based on Trajectories of Drivers and Requirements of Passengers

According to (Hsieh, 2017), it is significant to find out the routes of drivers to reduce the total of travelled distance but able to send all the pool users to the destination.

The past trajectory data is extremely important to match the drivers and passengers dynamically [2]. In the paper, it applied Google distance Matrix API to present the travel time and distance based on the calculation of Google Maps API. The Google Maps API is easy to conduct, and it provides a lot of function such as touch gestures and map display.

This paper contributed a dynamic carpooling algorithm. It will figure out the location of driver and passenger in the form of longitude and latitude. Then, it will analysis the needs of drivers and passengers. The original route and the ride sharing rote is being calculated by system. The first condition is identical ride sharing which both the driver and passenger have the common destination. Another situation is inclusive ridesharing which the pickup and drop off point of passenger is on the original route of driver. The next situation is partial ridesharing which is either the pick-up or drop off point is on the route of driver. The last condition is detouring ridesharing which one of or both the origin and destination of the user is not on the driver's journey. This condition should be avoided. The algorithm will complete the ride matching according to the distance calculated by Google Maps API. The final result of assigned driver and passengers is produced in the form of XML file, and it can be presented on a map.

The limitation of the system is it is only available on Android devices. This will disappoint the IOS user. The interface of the system is not user friendly and not tidy. Another from that, the experiment and testing that shown in the paper is limited, hence, it is still unpredictable when it opens to the public usage.

2.1.3 WeRide application

WeRide application is a platform for carpooling community [3]. The user is free to choose on the driver or passengers, payment method and any arrangements by themselves. The application is free of charge. The user may access to the images of traffic camera as WeRide is integrated with traffic camera in Malaysia and Singapore.

First of all, the user has to sign in their google account or Facebook account, then input their phone number to register themselves.

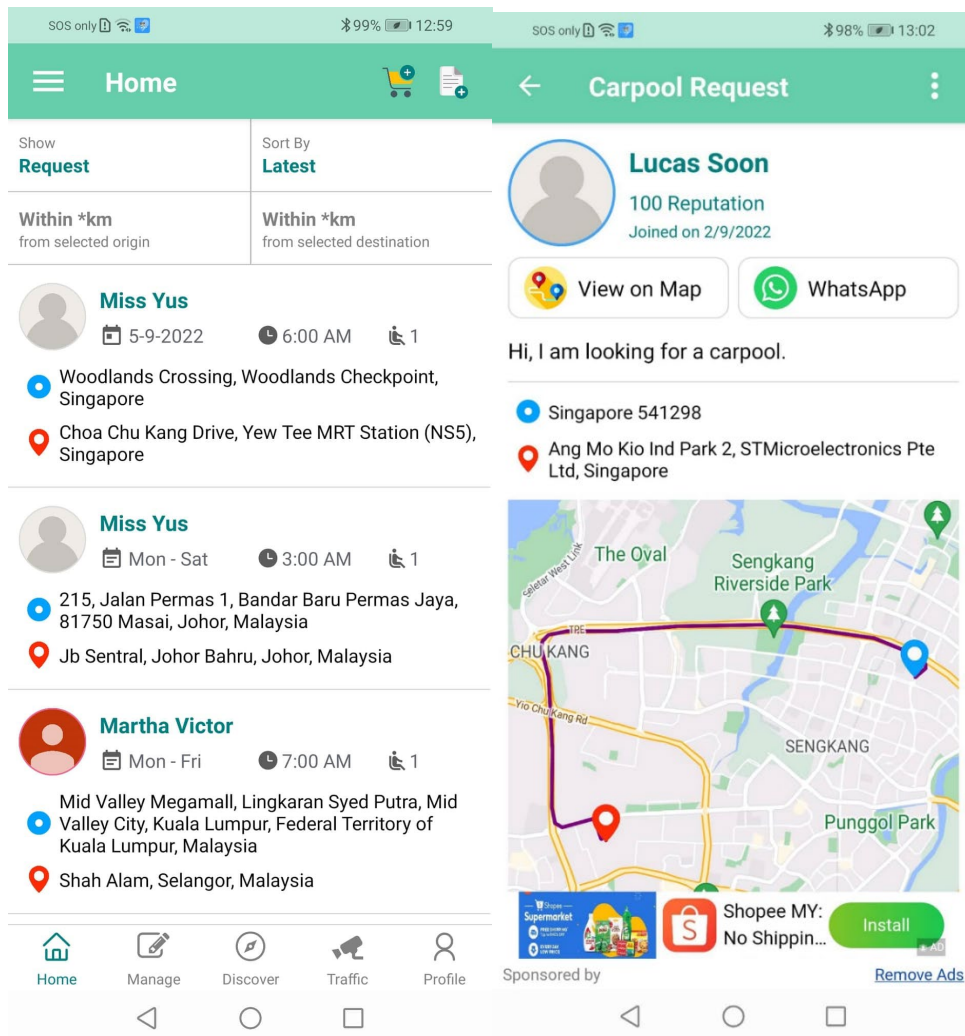


Figure 2.1 WeRide

The user is presented with all the posting of carpool request. They may click into it to review their pickup and drop off location. The user may opt to view on map. The driver may contact the passengers via WhatsApp.

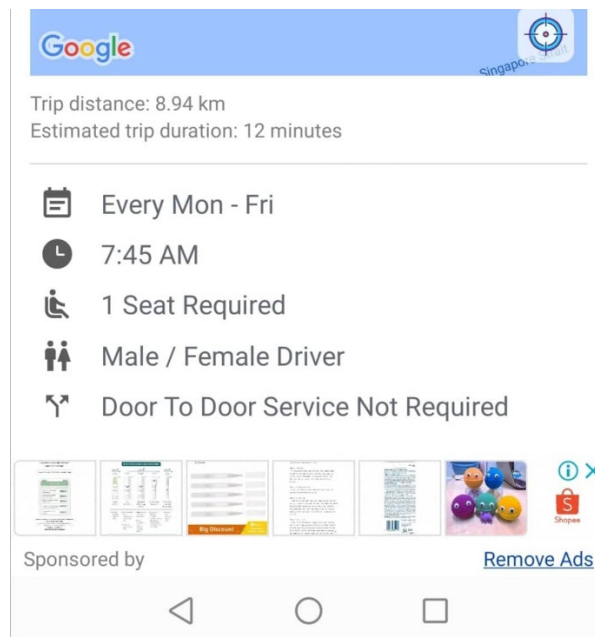


Figure 2.2 WeRide Trip Details

The application will show the trip distance and the estimated trip duration which will convenient the driver. The requirements and preferences of user also will present on the below of map interface.

The driver may search for their desired carpool request by entering the keywords such as the destination. Each user will have their own reputation points and it is presented on the profile and open to view by any users.

The application also provided a feedback form for the users. This help to improve the platform according to their honest feedback.

2.2 Limitation of Previous Studies

According to the previous studies, the time taken for booking a carpool is too long and it will affect on the willingness of the driver to offer a ride. The driver values the flexibility of the time schedule of carpooling. The user also has to made arrangement and discussion about the details of carpooling with the drivers themselves. This is extremely time consuming for both parties. Another from that, the cost of fuel consumption is one of the issues that reduce the drivers of carpooling.

The carpooling system by (Hsieh, 2017) only cover on Android devices. This will not benefit the IOS users, and this will reduce the drivers from IOS devices who may help in the ride sharing. The interface of system is not ideal and not user friendly. The dynamic carpooling algorithm is aimed to reduce the distance travelled but not its accuracy. As shown in the paper, the test and result provided are only examined on two drivers which is far from enough.

The WeRide application required the user to book the ride in advance by posting their requirements. This may not benefit the user who prefer to book the ride immediately. The driver also may miss the posting of the potential passengers easily.

In the journey of exploring the carpooling system, I realized that there is extremely lack of ride sharing application in Malaysia and most of the applications are from western countries. The ride sharing platform also exceedingly lack of users.

2.3 Proposed Solutions

This project aims to propose a ride sharing mobile application that applies opportunistic AI for real time ride-matching. The application is developed by using Flutter, and the design is delightful to enhance the user experiences in UI and UX. Through the application, the Google Maps data is uploaded by user to be stored in firebase. The firebase storage will trigger the google cloud function to run the python script of opportunistic AI to perform real time data preprocessing and thus the final result of algorithm prediction on the driver's schedule of destination. This will allow the application to perform the ride matching in real-time. The passengers may request and be offered with a carpool immediately through the application with opportunistic AI.

User safety will be enhanced by applying the user authentication system. The users will require to sign in through Google sign-in method in order to use the application.

To begin the journey of ride sharing platform and expand the possible user, the target user of the application is the students who are studying in UTAR Kampar campus. Ride sharing is undoubtedly in high demand among the students as most of the students require transportation to travel. Hence, we will achieve a sufficient amount of application users.

Chapter 3 – System Methodology /Approach

3.1 System Design Diagram

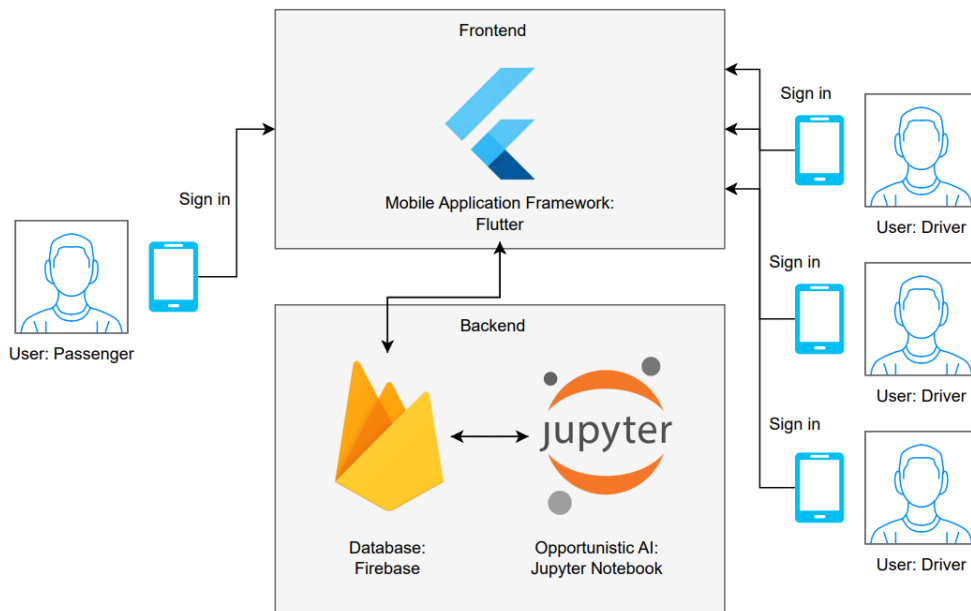


Figure 3.1 System Design Diagram

The UI framework will be developed by using flutter with the Google Maps API embedded. When the user signs in through Google Sign In authentication method, their information will automatically store into the firebase real-time database. When the user uploads the Google map data which is the Records.json file, the data will store in the firebase storage and then import to Jupyter Notebook in order to carry out the opportunistic AI model to process the data and build the users' timetable of their predicted destination. After that, the jupyter notebook will store the result to the firebase real-time database. When a passenger input their ride request, the jupyter notebook will get the request details from firebase and process it to get the output of the best match of driver with the passenger according to the daytime and destination. The result will upload to the firebase and then present to the mobile application.

3.2 Use Case Diagram and Description

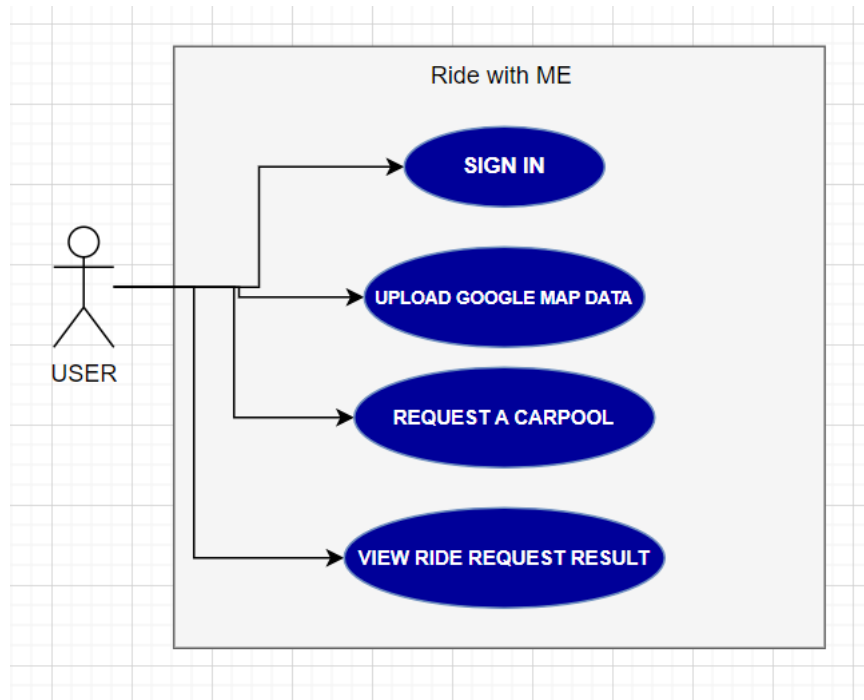


Figure 3.2 Use Case Diagram

The user may be a passenger or driver based on their preferences on the spot. Users must sign in their Google account to use the application. Users may upload their google map data. As a passenger, he may book a ride by inputting the details of ride request like datetime, origin and destination. As a driver, he may offer the carpool for the passenger by inputting the ride details such as the datetime, pickup and drop off point. The user may view the ride request result when the ride matching is successful.

3.3 Ride-sharing process

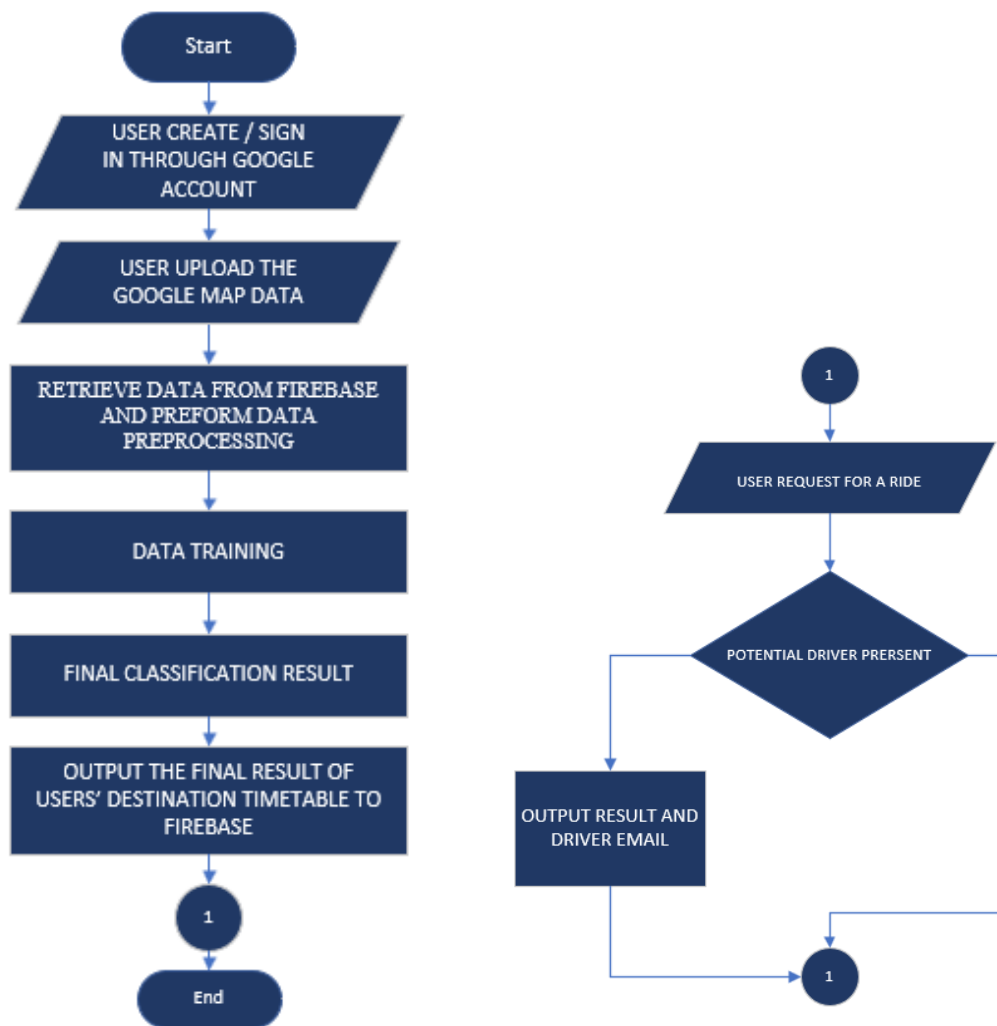


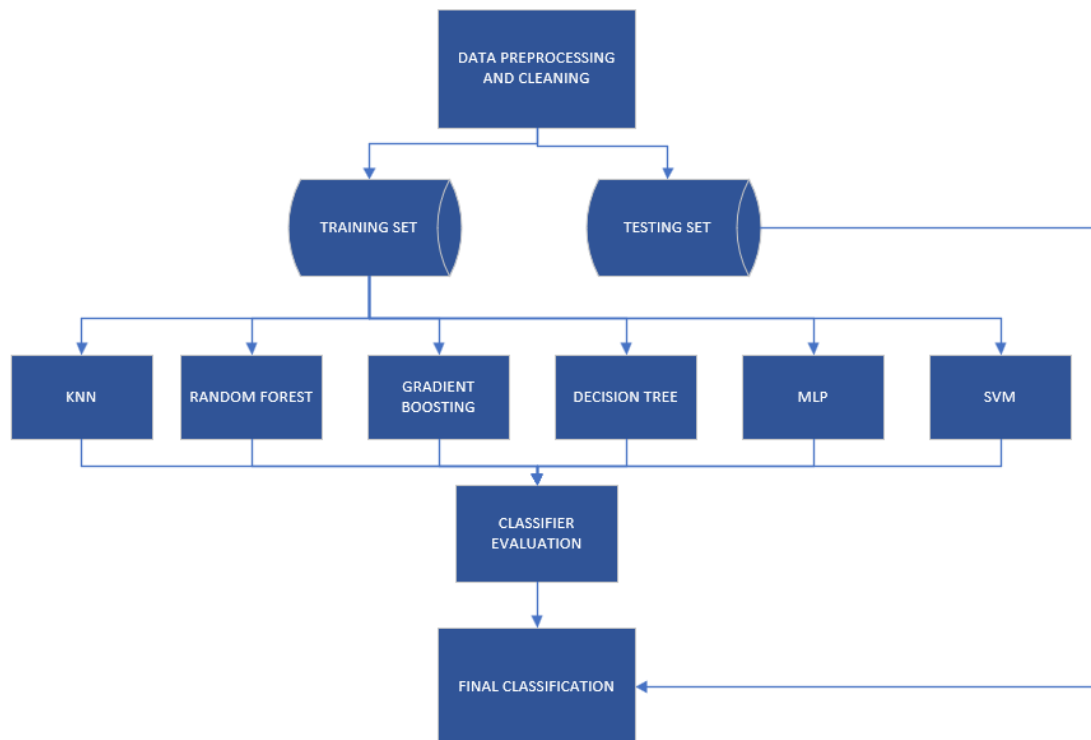
Figure 3.3 Ride-sharing process

The following are the steps of ride sharing process:

- i. Users create or sign in their Google account in the application.
- ii. Users upload their Google Maps data.
- iii. The system retrieves data from firebase and performs data preprocessing.
- iv. System performs data training.
- v. The system performs classification of data. The final result of the destination timetable will output.
- vi. The result is uploaded to firebase.
- vii. If the user requests a carpool and potential driver present, the ride-matching result will output to the user interface.

Chapter 4 – System Design

4.1 Opportunistic AI Configuration Block Diagram



First of all, data pre-processing and data cleaning is performed for the Google Maps data collected. It will split 70% of data to training set and 30% to testing set. The accuracy of the 6 classifier which are KNN, RandomForest, Gradient Boosting, Decision Tree, Multilayer Perceptron (MLP) and Support Vector Machine (SVM) is compared and hence, the classifier that performs the highest accuracy is chosen as the classifier conducted in this project. Finally, it will perform the final classification and output the user 's destination timetable as final result.

To ensure the accuracy of the prediction of the user's destination based on the day and time, data pre-processing is a very crucial part of the opportunistic AI model. All of the data collected from the Records.json which are String will be converted into float64 format to ease the process of the opportunistic AI classification.

Hence, the day of the week, time and destination are being classified to number accordingly beforehand.

4.2 Data Pre-processing and Data Cleaning

4.2.1 Day of Week

```
day_dict = {
    0: {'day': 'Sunday'},
    1: {'day': 'Monday'},
    2: {'day': 'Tuesday'},
    3: {'day': 'Wednesday'},
    4: {'day': 'Thursday'},
    5: {'day': 'Friday'},
    6: {'day': 'Saturday'},
}
```

Figure 4.2.1 Day of Week

As the Figure 4.2.1 shown as above, the day of the week is stored in a python dictionary to classify the day to an integer. For instance, 0 is Sunday, 1 is Monday and 6 is Saturday.

4.2.2 Time

```
time_ranges = {
    (pd.to_datetime('12:00 AM').time(), pd.to_datetime('5:59:59 AM').time()): 0,
    (pd.to_datetime('6:00 AM').time(), pd.to_datetime('6:59:59 AM').time()): 6,
    (pd.to_datetime('7:00 AM').time(), pd.to_datetime('7:59:59 AM').time()): 7,
    (pd.to_datetime('8:00 AM').time(), pd.to_datetime('8:59:59 AM').time()): 8,
    (pd.to_datetime('9:00 AM').time(), pd.to_datetime('9:59:59 AM').time()): 9,
    (pd.to_datetime('10:00 AM').time(), pd.to_datetime('10:59:59 AM').time()): 10,
    (pd.to_datetime('11:00 AM').time(), pd.to_datetime('11:59:59 AM').time()): 11,

    (pd.to_datetime('12:00 PM').time(), pd.to_datetime('12:59:59 PM').time()): 12,
    (pd.to_datetime('1:00 PM').time(), pd.to_datetime('1:59:59 PM').time()): 13,
    (pd.to_datetime('2:00 PM').time(), pd.to_datetime('2:59:59 PM').time()): 14,
    (pd.to_datetime('3:00 PM').time(), pd.to_datetime('3:59:59 PM').time()): 15,
    (pd.to_datetime('4:00 PM').time(), pd.to_datetime('4:59:59 PM').time()): 16,
    (pd.to_datetime('5:00 PM').time(), pd.to_datetime('5:59:59 PM').time()): 17,
    (pd.to_datetime('6:00 PM').time(), pd.to_datetime('6:59:59 PM').time()): 18,

    (pd.to_datetime('7:00 PM').time(), pd.to_datetime('7:59:59 PM').time()): 19,
    (pd.to_datetime('8:00 PM').time(), pd.to_datetime('8:59:59 PM').time()): 20,
    (pd.to_datetime('9:00 PM').time(), pd.to_datetime('9:59:59 PM').time()): 21,
    (pd.to_datetime('10:00 PM').time(), pd.to_datetime('10:59:59 PM').time()): 22,
    (pd.to_datetime('11:00 PM').time(), pd.to_datetime('11:59:59 PM').time()): 23,
}

time_dict = {
    0: {'time': 'Midnight'},
    6: {'time': '6 am'},
    7: {'time': '7 am'},
    8: {'time': '8 am'},
    9: {'time': '9 am'},
    10: {'time': '10 am'},
    11: {'time': '11 am'},
    12: {'time': '12 noon'},
    13: {'time': '1 pm'},
    14: {'time': '2 pm'},
    15: {'time': '3 pm'},
    16: {'time': '4 pm'},
    17: {'time': '5 pm'},
    18: {'time': '6 pm'},
    19: {'time': '7 pm'},
    20: {'time': '8 pm'},
    21: {'time': '9 pm'},
    22: {'time': '10 pm'},
    23: {'time': '11 pm'},
}
```

Figure 4.2.2 Time

The time data collected will be classify according to the specific time range as the Figure 4.2.2 shown. The time range from 12.00 am to 5.59am will be classified as midnight and will be removed in data cleaning as most of the users will be inactivated during the time and the destination in midnight is inconsistent.

4.2.3 Destination details

The location retrieved will be classified according to the Kampar area. For this project, only the students' active area in Kampar will be applied.



Figure 4.2.3.1 Area of Kampar, Perak

The python code in figure 4.2.3.1 is to pre-set to retrieve the location history in Kampar area only to avoid confusion of process of classification.

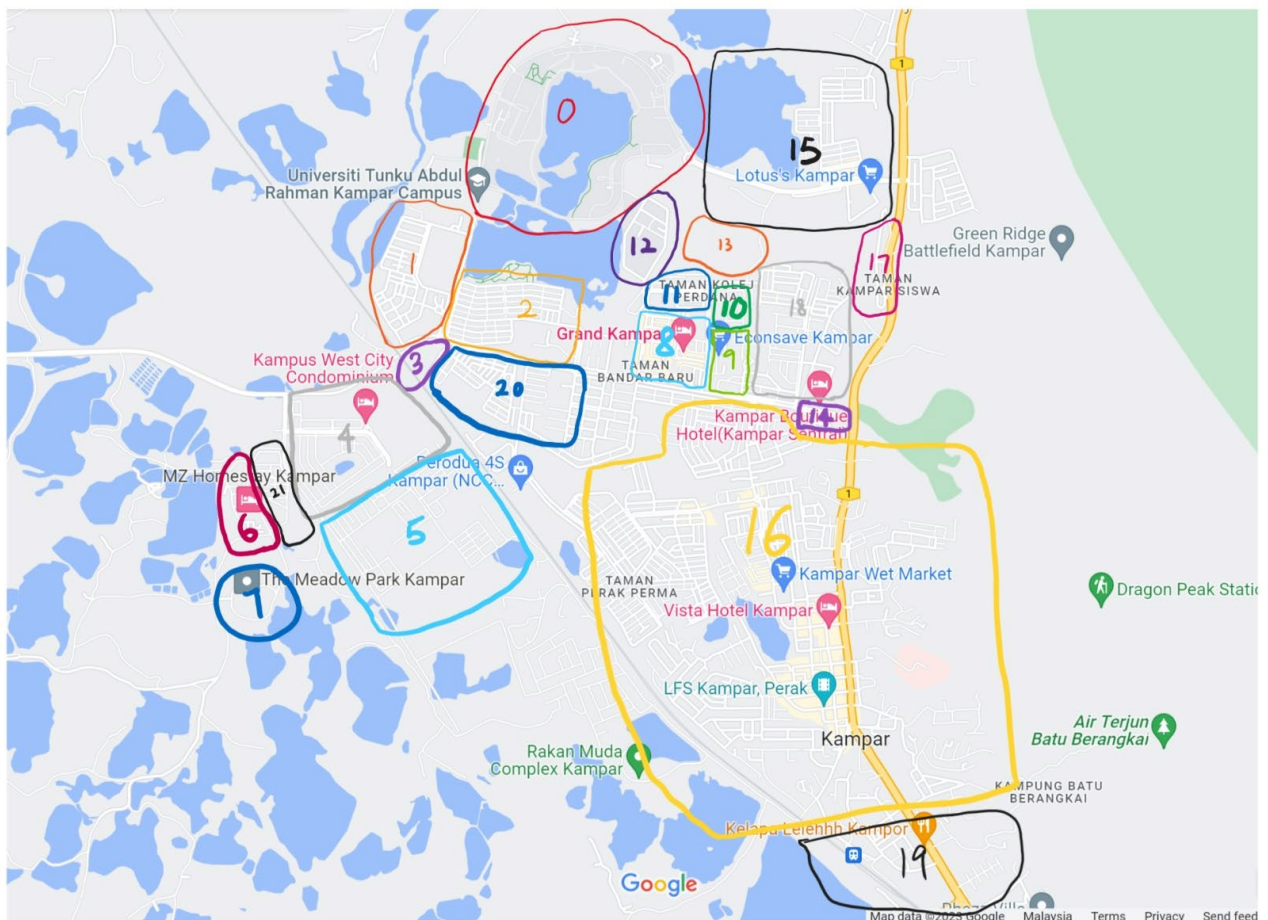


Figure 4.2.3.2 Destination Classification

```

dest_dict = {
    0: {'destination': 'UTAR Kampar Campus'},
    1: {'destination': 'Westlake Cambridge Kampar'},
    2: {'destination': 'Westlake Kampar'},
    3: {'destination': 'Westlake Villas Condominium Kampar'},
    4: {'destination': 'Westcity Kampar'},
    5: {'destination': 'Terminal Kampar'},
    6: {'destination': 'Trails of Kampar'},
    7: {'destination': 'Meadow Park Kampar'},
    8: {'destination': 'New Town Restaurant Area Kampar'},
    9: {'destination': 'Econsave Kampar'},
    10: {'destination': 'Ichiban Food Junction Row'},
    11: {'destination': 'Paris Mini Baker row + Sutera Apartment'},
    12: {'destination': 'TarUMT Kampar Campus'},
    13: {'destination': 'Kampar Lake Campus Condominium'},
    14: {'destination': 'Kampar KBS row'},
    15: {'destination': 'Eastlake Kampar'},
    16: {'destination': 'Old Town Kampar'},
    17: {'destination': 'Kampar MH Unilodge'},
    18: {'destination': 'Kampar Mahsuri'},
    19: {'destination': 'KTM Kampar Area'},
    20: {'destination': 'Jalan Suasa Housing Area'},
    21: {'destination': 'Disney Avenue'},
    88: {'destination': 'Unknown Kampar Area'},
}

```

```

def classify_coordinates(lat, lon):
    # Define your classification Logic here
    if 4.334054 <= lat <= 4.342920 and 101.132196 <= lon <= 101.145852:
        return ('UTAR Kampar Campus', 0)
    elif 4.3279 <= lat <= 4.334423 and 101.129 <= lon <= 101.94129185:
        return ('Westlake Cambridge Kampar', 1)
    elif 4.3267 <= lat <= 4.331186 and 101.1336 <= lon <= 101.142:
        return ('Westlake Kampar', 2)
    elif 4.326576 <= lat <= 4.3281 and 101.131 <= lon <= 101.133529:
        return ('Westlake Villas Condominium Kampar', 3)
    elif 4.317 <= lat <= 4.3269 and 101.125 <= lon <= 101.136:
        return ('Westcity Kampar', 4)
    elif 4.314996 <= lat <= 4.318965 and 101.129218 <= lon <= 101.134325:
        return ('Terminal Kampar', 5)
    elif 4.318255 <= lat <= 4.321332 and 101.122383 <= lon <= 101.124927:
        return ('Trails of Kampar', 6)
    elif 4.314002 <= lat <= 4.31887 and 101.122280 <= lon <= 101.125615:
        return ('Meadow Park Kampar', 7)
    elif 4.325797 <= lat <= 4.329371 and 101.143046 <= lon <= 101.146380:
        return ('New Town Restaurant Area Kampar', 8)
    elif 4.325478 <= lat <= 4.329007 and 101.146354 <= lon <= 101.148079:
        return ('Econsave Kampar', 9)
    elif 4.329069 <= lat <= 4.330771 and 101.146558 <= lon <= 101.147510:
        return ('Ichiban Food Junction Row', 10)
    elif 4.329640 <= lat <= 4.331299 and 101.143640 <= lon <= 101.146283:
        return ('Paris Mini Baker row + Sutera Apartment', 11)
    elif 4.330855 <= lat <= 4.334471 and 101.142146 <= lon <= 101.144718:
        return ('TarUMT Kampar Campus', 12)
    elif 4.331354 <= lat <= 4.332731 and 101.144645 <= lon <= 101.146429:
        return ('Kampar Lake Campus Condominium', 13)
    elif 4.323888 <= lat <= 4.3247 and 101.151582 <= lon <= 101.153104:
        return ('Kampar KBS row', 14)
    elif 4.3237 <= lat <= 4.349 and 101.145 <= lon <= 101.157:
        return ('Eastlake Kampar', 15)
    elif 4.301 <= lat <= 4.327 and 101.138 <= lon <= 101.162:
        return ('Old Town Kampar', 16)
    elif 4.328601 <= lat <= 4.330672 and 101.153872 <= lon <= 101.155176:
        return ('Kampar MH Unilodge', 17)
    elif 4.324632 <= lat <= 4.331825 and 101.147333 <= lon <= 101.153484:
        return ('Kampar Mahsuri', 18)
    elif 4.297891 <= lat <= 4.308494 and 101.150977 <= lon <= 101.156911:
        return ('KTM Kampar Area', 19)
    elif 4.3227 <= lat <= 4.3277 and 101.1325 <= lon <= 101.1389:
        return ('Jalan Suasa Housing Area', 20)
    elif 4.3188 <= lat <= 4.323 and 101.123 <= lon <= 101.1271:
        return ('Disney Avenue', 21)
    else:
        return ('Unknown Kampar Area', 88)

```

Figure 4.2.3.3 Destination Classification Code

The latitude and longitude that retrieved from the Records.json file will classify based on which area they are in. For example, UTAR Kampar campus has latitude that within 4.334054 and 4.342920, and longitude that between 101.132196 and 101.145852. Hence, the latitude and longitude of the destination that within the range will be classify into class 0 which is UTAR Kampar Campus.

4.3 Predefined Function

```

import json
import csv
import pandas as pd
from datetime import datetime
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score

from geopy.geocoders import Nominatim
from geopy.distance import geodesic
from geopy.geocoders import GoogleV3

# Replace 'YOUR_API_KEY' with your own Google Maps API key
geolocator = GoogleV3(api_key='AIzaSyCMCRCoVHpfu9F6Q191YYvtrLi8cbAiFYI')

import googlemaps
from datetime import datetime
#api = AIzaSyCMCRCoVHpfu9F6Q191YYvtrLi8cbAiFYI
gmaps = googlemaps.Client(key='AIzaSyCMCRCoVHpfu9F6Q191YYvtrLi8cbAiFYI')

```

4.3.1 library

First of all, all the needed library is imported.

```

def getDay(date):
    day_num = date.strftime("%w") # %w returns the day of the week (0 for Sunday, 6 for Saturday)
    return int(day_num) # convert day number to 1-7 range and return as string

def classifyTime(df, time_ranges):
    classified_time_series = pd.Series(index=df['time'].index, dtype='float64')

    for i in range(len(df['time'])):
        time = df['time'][i]
        classified_time = None

        # Check if the time falls within any of the defined ranges
        for key in time_ranges.keys():
            if time >= key[0] and time <= key[1]:
                classified_time = time_ranges[key]
                break

        if classified_time is None:
            classified_time = -1

        classified_time_series.iloc[i] = classified_time

    return classified_time_series

```

Figure 4.3.2 Get the day of week and classify time

The `getDay` function will get the date and return the result as day of week in integer for classification later. For example, it will classify Sunday as 0. The `classifyTime` function will classify the time to integer according to their time range.

```

# Loading the data FROM JSON TO CSV TO DF
def jsonRecordtoCSV(inputjson, outputcsv):
    with open(inputjson) as json_file:
        data = json.load(json_file)
        loc_records = data['locations']
        data_file = open(outputcsv, 'w')

        # create the csv writer object
        csv_writer = csv.DictWriter(data_file,
            fieldnames=['latitudeE7', 'longitudeE7', 'accuracy', 'activity', 'source', 'deviceTag', 'activeWifiScan', 'placeId', 'locationMet

        csv_writer.writeheader()

        for i in loc_records:
            if 'activity' in i.values():
                continue
            # Writing data of CSV file
            else:
                csv_writer.writerow(i)
        data_file.close()

#read the csv file
df = pd.read_csv (outputcsv, sep=",", on_bad_lines='skip', usecols=['latitudeE7', 'longitudeE7', 'timestamp'])

#add DateTime column
df["time"] = pd.to_datetime(df["timestamp"]).dt.time
df["date"] = pd.to_datetime(df["timestamp"]).dt.date
df["day"] = pd.to_datetime(df["timestamp"]).dt.day_name()

# Classify day in 0-6 0 is sunday
df["day_int"] = df["date"].apply(getDay)
# Classify the time
df["time_range"] = classifyTime(df, time_ranges)

return df

```

df

	latitudeE7	longitudeE7	timestamp	time	date	day	day_int	time_range
0	33986701	1015640994	2015-04-17T13:07:37.737Z	13:07:37.737000	2015-04-17	Friday	5	13.0
1	33986701	1015640994	2015-04-17T13:08:29.389Z	13:08:29.389000	2015-04-17	Friday	5	13.0
2	33986701	1015640994	2015-04-17T13:09:19.459Z	13:09:19.459000	2015-04-17	Friday	5	13.0
3	33986701	1015640994	2015-04-17T13:10:09.384Z	13:10:09.384000	2015-04-17	Friday	5	13.0
4	33986701	1015640994	2015-04-17T13:10:59.419Z	13:10:59.419000	2015-04-17	Friday	5	13.0
...
830665	43388002	1011368583	2023-04-13T07:32:50.908Z	07:32:50.908000	2023-04-13	Thursday	4	7.0
830666	43388002	1011368583	2023-04-13T07:35:25.431Z	07:35:25.431000	2023-04-13	Thursday	4	7.0
830667	43388002	1011368583	2023-04-13T07:39:42.149Z	07:39:42.149000	2023-04-13	Thursday	4	7.0
830668	43387471	1011368466	2023-04-13T07:40:21.498Z	07:40:21.498000	2023-04-13	Thursday	4	7.0
830669	43387471	1011368466	2023-04-13T07:43:01.226Z	07:43:01.226000	2023-04-13	Thursday	4	7.0

830670 rows × 8 columns

Figure 4.3.3 Preprocess Google map data

This function will retrieve the Google Map data and convert it into csv file to read the file locally in order to investigate the file information and then preprocess it into a readable DataFrame format in python.


```
def get_df_latest(df):
    today = pd.Timestamp.today() # Get today's date
    recent = today - pd.DateOffset(months=3)
    return df[df["date"] >= recent]
```

```
df_latest = get_df_latest(df)
df_latest
```

C:\Users\TEH\anaconda3\lib\site-packages\pandas\core\ops\array_ops.py:73: FutureWarning: Comparison date is deprecated in order to match the standard library behavior. In a future version these will e. Use 'ts == pd.Timestamp(date)' or 'ts.date() == date' instead.
result = libops.scalar_compare(x.ravel(), y, op)

	latitudeE7	longitudeE7	timestamp	time	date	day	day_int	time_range
794018	43305121	1011371685	2023-01-27T00:01:58.179Z	00:01:58.179000	2023-01-27	Friday	5	0.0
794019	43305121	1011371685	2023-01-27T00:03:58.220Z	00:03:58.220000	2023-01-27	Friday	5	0.0
794020	43305121	1011371685	2023-01-27T00:05:58.262Z	00:05:58.262000	2023-01-27	Friday	5	0.0
794021	43305121	1011371685	2023-01-27T00:09:33.108Z	00:09:33.108000	2023-01-27	Friday	5	0.0
794022	43305121	1011371685	2023-01-27T00:11:33.216Z	00:11:33.216000	2023-01-27	Friday	5	0.0
...
830665	43388002	1011368583	2023-04-13T07:32:50.908Z	07:32:50.908000	2023-04-13	Thursday	4	7.0
830666	43388002	1011368583	2023-04-13T07:35:25.431Z	07:35:25.431000	2023-04-13	Thursday	4	7.0
830667	43388002	1011368583	2023-04-13T07:39:42.149Z	07:39:42.149000	2023-04-13	Thursday	4	7.0
830668	43387471	1011368466	2023-04-13T07:40:21.498Z	07:40:21.498000	2023-04-13	Thursday	4	7.0
830669	43387471	1011368466	2023-04-13T07:43:01.226Z	07:43:01.226000	2023-04-13	Thursday	4	7.0

36652 rows x 8 columns

Figure 4.3.4 getLatest

The `get_df_latest` function is to get the recent 3 months of location visited only as the 1 semester of UTAR is only 3 months long and to avoid any confusion in classification.

```
def getLatLon(df):
    # define a lambda function to divide the value by 10000000 and format the result with all decimal places
    div_and_format = lambda x: '{:.13f}'.format(x / 10000000)

    # apply the lambda function to the column
    #df['latitude'] = df['latitudeE7'].apply(div_and_format).astype(float)
    #df['longitude'] = df['longitudeE7'].apply(div_and_format).astype(float)

    df.loc[:, 'latitude'] = pd.to_numeric(df['latitudeE7'].apply(div_and_format), errors='coerce')
    df.loc[:, 'longitude'] = pd.to_numeric(df['longitudeE7'].apply(div_and_format), errors='coerce')

    return df
```

```
df_latest = getLatLon(df_latest)
df_latest
```

	latitudeE7	longitudeE7	timestamp	time	date	day	day_int	time_range	latitude	longitude
794018	43305121	1011371685	2023-01-27T00:01:58.179Z	00:01:58.179000	2023-01-27	Friday	5	0.0	4.330512	101.137169
794019	43305121	1011371685	2023-01-27T00:03:58.220Z	00:03:58.220000	2023-01-27	Friday	5	0.0	4.330512	101.137169
794020	43305121	1011371685	2023-01-27T00:05:58.262Z	00:05:58.262000	2023-01-27	Friday	5	0.0	4.330512	101.137169
794021	43305121	1011371685	2023-01-27T00:09:33.108Z	00:09:33.108000	2023-01-27	Friday	5	0.0	4.330512	101.137169
794022	43305121	1011371685	2023-01-27T00:11:33.216Z	00:11:33.216000	2023-01-27	Friday	5	0.0	4.330512	101.137169
...
830665	43388002	1011368583	2023-04-13T07:32:50.908Z	07:32:50.908000	2023-04-13	Thursday	4	7.0	4.338800	101.136858

Figure 4.3.5 getlatlon and getkpr

The `getLatLon` is to correct the format of latitude and longitude in numeric form and divide by 10^7 .

```
def getKprArea(df):
    lat_range = (df['latitude'] >= 4.29) & (df['latitude'] <= 4.35)
    lon_range = (df['longitude'] >= 101.11) & (df['longitude'] <= 101.17)
    df_kpr = df.loc[lat_range & lon_range]

    return df_kpr
```

Figure 4.3.6 getkpr

The getKprArea function is to extract data frame to contain the data that located in Kampar area only.

```
def dest_column(df_old):
    # Keep the columns that you want to keep
    df_keep = df_old[['latitude', 'longitude', 'day_int', 'time_range']]

    df = df_old.groupby(['latitude', 'longitude']).first().reset_index(['latitude', 'longitude'])

    # Classify the coordinates and add the new column 'class'
    df['destination'], df['dest_code'] = zip(*df.apply(lambda x: classify_coordinates(x['latitude'], x['longitude']), axis=1))

    # Merge the new DataFrame with the old DataFrame
    df_new = pd.merge(df_keep, df, on=['latitude', 'longitude'], how='right')

    return df_new
```

```
: df_dest = dest_column(df_kpr)
df_dest.iloc[10]
```

```
: latitude          4.305467
longitude          101.156027
day_int            6
time_range         16.0
destination      Old Town Kampar
dest_code         16
Name: 10, dtype: object
```

Figure 4.3.7 destination column

The dest_column function is to obtain the column of the destination area name and code of it in integer to ease the classification later.

```
#drop unused column
df_latest = df_latest.drop(['latitudeE7', 'longitudeE7', 'timestamp', 'time', 'date'], axis=1)
```

```
df_latest
```

	day	day_int	time_range	latitude	longitude
794018	Friday	5	0.0	4.330512	101.137169
794019	Friday	5	0.0	4.330512	101.137169
794020	Friday	5	0.0	4.330512	101.137169
794021	Friday	5	0.0	4.330512	101.137169
794022	Friday	5	0.0	4.330512	101.137169
...
830665	Thursday	4	7.0	4.338800	101.136858
830666	Thursday	4	7.0	4.338800	101.136858
830667	Thursday	4	7.0	4.338800	101.136858
830668	Thursday	4	7.0	4.338747	101.136847
830669	Thursday	4	7.0	4.338747	101.136847

36652 rows × 5 columns

Figure 4.3.8 drop cloumn

Then, drop the used column and leave those are useful for the later classification.

```
# drop rows of unknown area and midnight
df_clear = df_dest[(df_dest['dest_code'] != 88 ) & (df_dest['time_range'] != 0 ) ]
df_clear
```

	latitude	longitude	day_int	time_range	destination	dest_code
7	4.301506	101.158745	5	8.0	Old Town Kampar	16
8	4.302737	101.157534	0	9.0	Old Town Kampar	16
9	4.305467	101.156027	6	16.0	Old Town Kampar	16
10	4.305467	101.156027	6	16.0	Old Town Kampar	16
11	4.305467	101.156027	6	16.0	Old Town Kampar	16
...
31834	4.342123	101.140298	3	8.0	UTAR Kampar Campus	0
31835	4.342154	101.138389	1	9.0	UTAR Kampar Campus	0
31836	4.342189	101.139280	1	7.0	UTAR Kampar Campus	0
31840	4.342449	101.139193	1	7.0	UTAR Kampar Campus	0
31841	4.342557	101.140093	4	6.0	UTAR Kampar Campus	0

21701 rows × 6 columns

Figure 4.3.9 drop row

Lastly, drop the unknown Kampar Area which are close to the edge of Kampar or the area that are not included in this project and also drop the midnight data as users are inactive and inconsistent schedule during midnight.

4.4 Data Training

```
def splitData(df):
    # Split the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(df[['day_int', 'time_range']],
                                                    df['dest_code'],
                                                    test_size=0.3, random_state=42)
    return X_train, X_test, y_train, y_test
```

Figure 4.4 splitData

The splitData function is to split the data to the training set and testing set accordingly to 70% and 30%.

```
X_train, X_test, y_train, y_test = splitData(df_clear)
```

```
# Train the KNN model
knn = KNeighborsClassifier(n_neighbors=150)
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_test)
print("KNN Accuracy: {:.2f}%".format(accuracy_score(y_test, knn_pred)*100))
```

KNN Accuracy: 84.57%

```
# Train the Random Forest model
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)
print("Random Forest Accuracy: {:.2f}%".format(accuracy_score(y_test, rf_pred)*100))
```

Random Forest Accuracy: 84.57%

```
# Train the Gradient Boosting model
gb = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb.fit(X_train, y_train)
gb_pred = gb.predict(X_test)
print("Gradient Boosting Accuracy: {:.2f}%".format(accuracy_score(y_test, gb_pred)*100))
```

Gradient Boosting Accuracy: 84.15%

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Create the decision tree classifier
dtc = DecisionTreeClassifier(random_state=42)

# Train the classifier
dtc.fit(X_train, y_train)

# Predict on the testing set
dtc_pred = dtc.predict(X_test)

# Calculate the accuracy score
print("DecisionTreeClassifier: {:.2f}%".format(accuracy_score(y_test, dtc_pred)*100))
```

DecisionTreeClassifier: 84.57%

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Create the SVM classifier
SVC = SVC(random_state=42)

# Train the classifier
SVC.fit(X_train, y_train)

# Predict on the testing set
SVC_pred = SVC.predict(X_test)

# Calculate the accuracy score
print("SVM classifier: {:.2f}%".format(accuracy_score(y_test, SVC_pred)*100))
```

SVM classifier: 82.10%

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Create the neural network classifier
MLP = MLPClassifier(random_state=42)

# Train the classifier
MLP.fit(X_train, y_train)

# Predict on the testing set
MLP_pred = MLP.predict(X_test)

# Calculate the accuracy score
print("MLPClassifier: {:.2f}%".format(accuracy_score(y_test, MLP_pred)*100))
```

MLPClassifier: 83.28%

Figure 4.4.2 Classifier

```
print("KNN Accuracy: {:.2f}%".format(accuracy_score(y_test, knn_pred)*100))
print("Random Forest Accuracy: {:.2f}%".format(accuracy_score(y_test, rf_pred)*100))
print("Gradient Boosting Accuracy: {:.2f}%".format(accuracy_score(y_test, gb_pred)*100))
print("DecisionTreeClassifier: {:.2f}%".format(accuracy_score(y_test, dtc_pred)*100))
print("SVM classifier: {:.2f}%".format(accuracy_score(y_test, SVC_pred)*100))
print("MLPClassifier: {:.2f}%".format(accuracy_score(y_test, MLP_pred)*100))
```

```
KNN Accuracy: 84.57%
Random Forest Accuracy: 84.57%
Gradient Boosting Accuracy: 84.15%
DecisionTreeClassifier: 84.57%
SVM classifier: 82.10%
MLPClassifier: 83.28%
```

Figure 4.4.2 Accuracy of Classifier

The figure above shows the accuracy of each classifier. The accuracy rate is satisfied with the result of above 80%. The highest accuracy classifiers are KNN, Random Forest and Decision Tree Classifier. After observing the other test case, I decided to use Random Forest Classifier as it has a higher stability in accuracy rate. Random Forest Classifier is suitable to compute large datasets and in combining multiple decision trees to avoid overfitting.

4.5 Timetable of Destination

```

df_result = pd.DataFrame()
day_list=[]
time_list=[]
dest_list=[]
proba_list=[]
cnt = 0
for i in range(7):
    for j in range(6, 24):
        print(i, j)
        result = knn.predict([[i, j]])
        dest = dest_dict[np.array(result)[0]]['destination']
        day = day_dict[i]['day']
        time = time_dict[j]['time']
        day_list.append(day)
        time_list.append(time)
        dest_list.append(dest)
        print("Destination: {}".format(dest))

        # print probability estimates
        proba = knn.predict_proba([[i, j]])
        print("Probability estimates: {:.2f}%".format(proba[0].max()*100))
        proba_list.append(proba)
        cnt+=1

df_result['day'] = day_list
df_result['time'] = time_list
df_result['destination'] = dest_list
df_result['probability'] = proba_list

0 6
Destination: Westlake Cambridge Kampar
Probability estimates: 96.00%
0 7
Destination: Westlake Cambridge Kampar
Probability estimates: 99.33%
0 8
Destination: Westlake Cambridge Kampar
Probability estimates: 97.33%
0 9

2 15
Destination: Westlake Cambridge Kampar
Probability estimates: 97.33%
2 16
Destination: Westlake Cambridge Kampar
Probability estimates: 98.67%
2 17
Destination: Westlake Cambridge Kampar
Probability estimates: 99.33%
2 18
Destination: Westlake Cambridge Kampar
Probability estimates: 100.00%
2 19
Destination: Westlake Cambridge Kampar
Probability estimates: 100.00%
2 20
Destination: Westlake Cambridge Kampar
Probability estimates: 100.00%
2 21
Destination: Westlake Cambridge Kampar
Probability estimates: 100.00%
2 22
Destination: Westlake Cambridge Kampar
Probability estimates: 97.33%
2 23
Destination: Westlake Cambridge Kampar
Probability estimates: 93.33%

```

df_result					
	day	time	destination	probability	
0	Sunday	6 am	Westlake Cambridge Kampar	84.000000	
1	Sunday	7 am	Westlake Cambridge Kampar	90.666667	
2	Sunday	8 am	Westlake Cambridge Kampar	85.333333	
3	Sunday	9 am	Westlake Cambridge Kampar	65.333333	
4	Sunday	10 am	Westlake Cambridge Kampar	75.333333	
...
121	Saturday	7 pm	Westlake Cambridge Kampar	100.000000	
122	Saturday	8 pm	Westlake Cambridge Kampar	98.666667	
123	Saturday	9 pm	Westlake Cambridge Kampar	90.000000	
124	Saturday	10 pm	Westlake Cambridge Kampar	96.000000	
125	Saturday	11 pm	Westlake Cambridge Kampar	96.666667	

126 rows × 4 columns

Figure 4.5 result

Finally, the result of the timetable of Destination will output to a data frame as the figure shown. It will store the day, time, destination area and the probability of the user will visit the place.

Chapter 5 - System Implementation

5.1 Hardware Setup

The hardware involved in this project is laptop and android mobile device. A laptop issued for the process of modelling the opportunistic AI and developing the ride sharing application. The android mobile devices are utilized for testing and deploying this ride sharing application to trial the method in real life ride sharing cases.

Table 5.1 Specifications of laptop

Description	Specifications
Model	Lenovo Legion S7 15IMH5
Processor	Intel Core i7-10705H
Operating System	Windows 10
Graphic	NVIDIA RTX 2060 Max-Q 6GBD6
Memory	16GB DDR4 RAM
Storage	1TB SSD

Table 5.2 Specifications of Android Phone

Description	Specifications
Model	Huawei Nova 4e
Processor	Hisilicon Kirin 710 (12 nm) CPU processor
Operating System	Android 9.0 (Pie), EMUI 9.0
Memory	6GB RAM
Storage	128GB

5.2 Software Setup

1. Google Cloud



Figure 5.2.1 Google Cloud Platform

From the Google Cloud platform, we may utilize the google map API in it. Google Map is extremely popular in the driver s' community. Hence, we may collect a myriad of drivers' driving patterns to be optimized and analysis. Google maps usually will collect the user's location history in the background and store it in the Records.json that can be retrieved in Google Map application. The file will be collected for the opportunistic AI's data processing. The Google Map's API will be applied in the mobile application. The Google Maps API performs a clear and informative map interface which accepts touch gestures from the users.

2. Flutter



Figure 5.2.2 Flutter

The Google Flutter framework will be utilized to develop the ride-sharing mobile application by using Dart language. Flutter is exceedingly powerful in developing and deploying mobile applications. It is widely used to create cross platform applications for both Android and iOS devices, and also other platforms like Windows. With Flutter, we can provide great user experiences, fast performances and immediate updates.

3. Firebase



Figure 5.2.3 Firebase

Firebase is automatically available to store all the NoSQL databases. All the user information and their Google account details will be stored in real time responsively. The user also will upload their Google Map data, location history to the firebase storage. The python file will process the data and return the predicted user's destination back to the real time firebase.

4. Jupyter Notebook



Figure 5.2.4 Jupyter Notebook

The Jupyter Notebook will be implemented to develop and train the opportunistic ai model. It is powerful to build, train and deploy the opportunistic AI in python language.

5.3 Settings and Configuration

Before starting to develop the Ride with ME application, there are three software require to be installed and downloaded in my laptop:

1. Visual Studio Code: 1.73
2. Flutter: 3.3.8 SDK
3. Anaconda: Python 3.9

5.4 System Operation

5.4.1 Ride-sharing Application



Figure 5.4.1.1 Ride with ME logo

The ride-sharing application is created by using Flutter. As shown in Figure 5.4.1, it is the logo of the Ride with ME application. A logo is really important to strengthen the first impression of users. The logo consisted of the application name and a car sharing icon to clarify the main usage and purpose of the application. It is presented in blue color as people feel the stability, confidence, trustiness and intelligence in it.

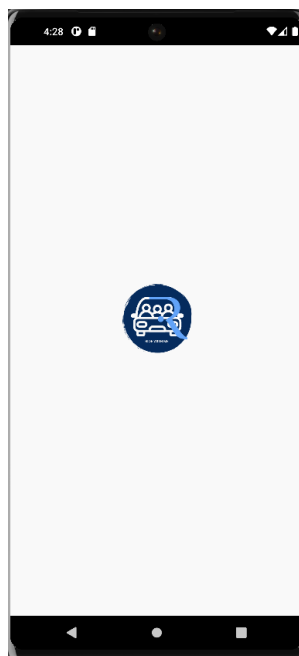


Figure 5.4.1.2 Splash Screen

A splash screen is developed in order to enhance the user experience. The splash screen improves the first impression of the users, and they may feel like their waiting time is reduced [6]. A crystal clean splash screen with the company logo located at the center is presented.

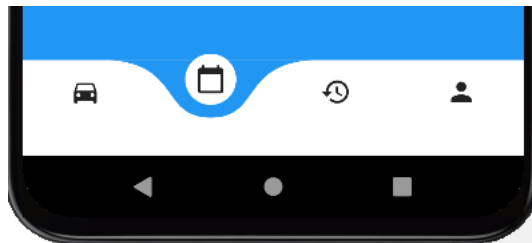


Figure 5.4.1.3 Bottom navigation bar

The first icon on the navigation bar below represents the main screen where the user will perform their request or offer for a carpool. The user may input their pickup point, destination and also the datetime if they are planning the carpool for another day. After the rider trajectory model output the suitable driver, the driver will get a notification to review the ride details. If the route is slightly different with the user, the driver may input a small charge for the passenger. The passenger will then receive the driver's details and he may accept or reject the carpool offer. Finally, the confirmed ride details will be presented on the screen of both users.

The third icon presented is the history of the ride-sharing that is completed or pending. It will store and present the past ride sharing cases of users to back up the details of the ride.

The fourth icon on the bottom navigation bar is a chat room for the users. This eases the communication between the passenger and the driver to solve some unexpected conditions such as the driver fails to locate the passenger.

5.4.2 User's Google Sign in and Sign Out Function

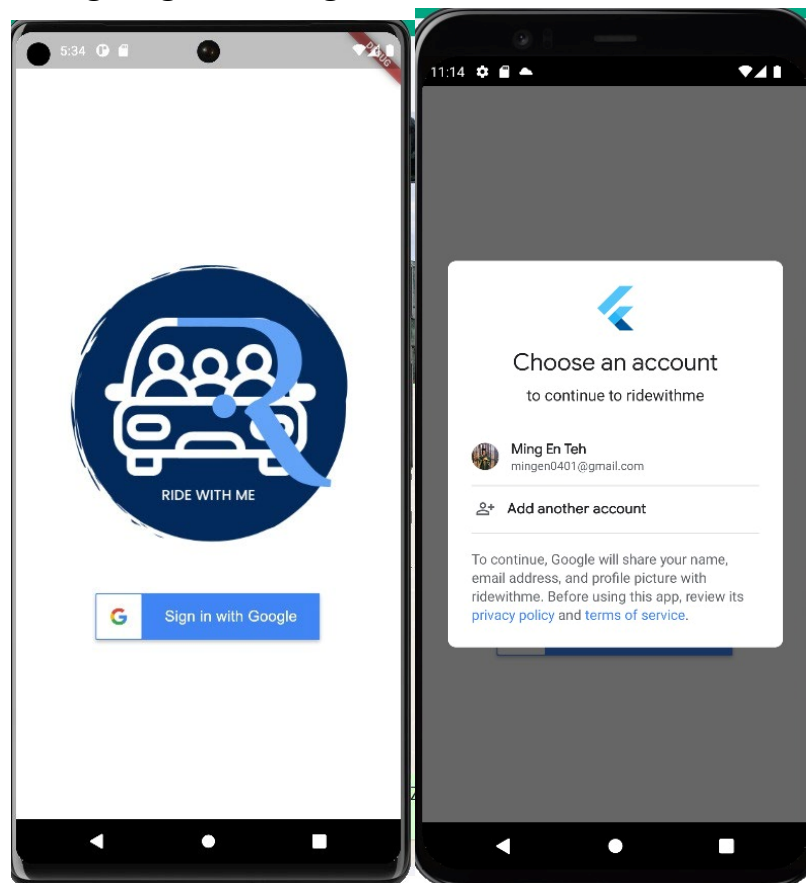


Figure 5.4.2.1 Google Sign in Screen

First of all, the users will need to sign into their Google account to utilize the application. The user authentication of this application is applying the Google sign in method. Google sign in method ease the process for users to sign in efficiently without the need of creating a new account. With that, the probability of the users that are utilizing the Google Map will be higher and we may access the user's Google maps information in future. Users have to press on the Google sign in button, and they will forward to a Google log in page immediately. If they have not sign up for any Google account, they may create a Google account on the spot. If user has a Google account, it will prompt the user to input his email and password to sign in. It is convenient and handy to use the Google authentication method to log in and their data will retrieve into the real time database responsively.

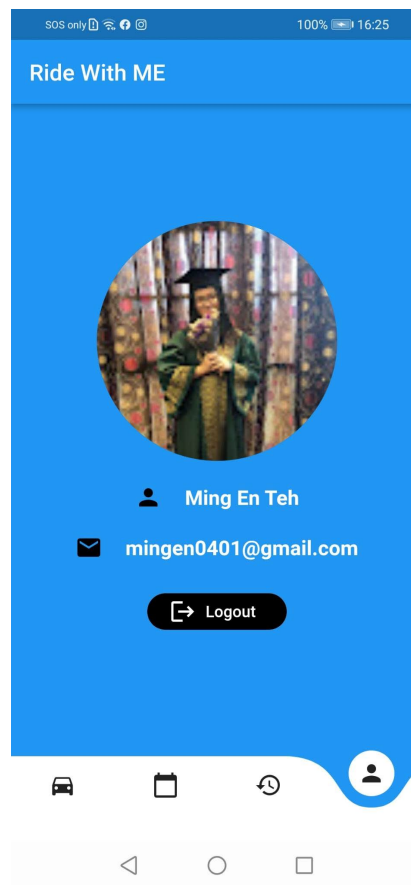


Figure 5.4.2.2 Account Information Page

After the user logged in, an account information page is presented with an interactive and appealing navigation bar on the bottom of the page. When the user presses on the last button which contains an account icon, the user's account information will pop out with the user's profile picture. The user may click the logout button to log out of their Google account and then sign into another Google account.

5.4.3 Ride with ME User Interface

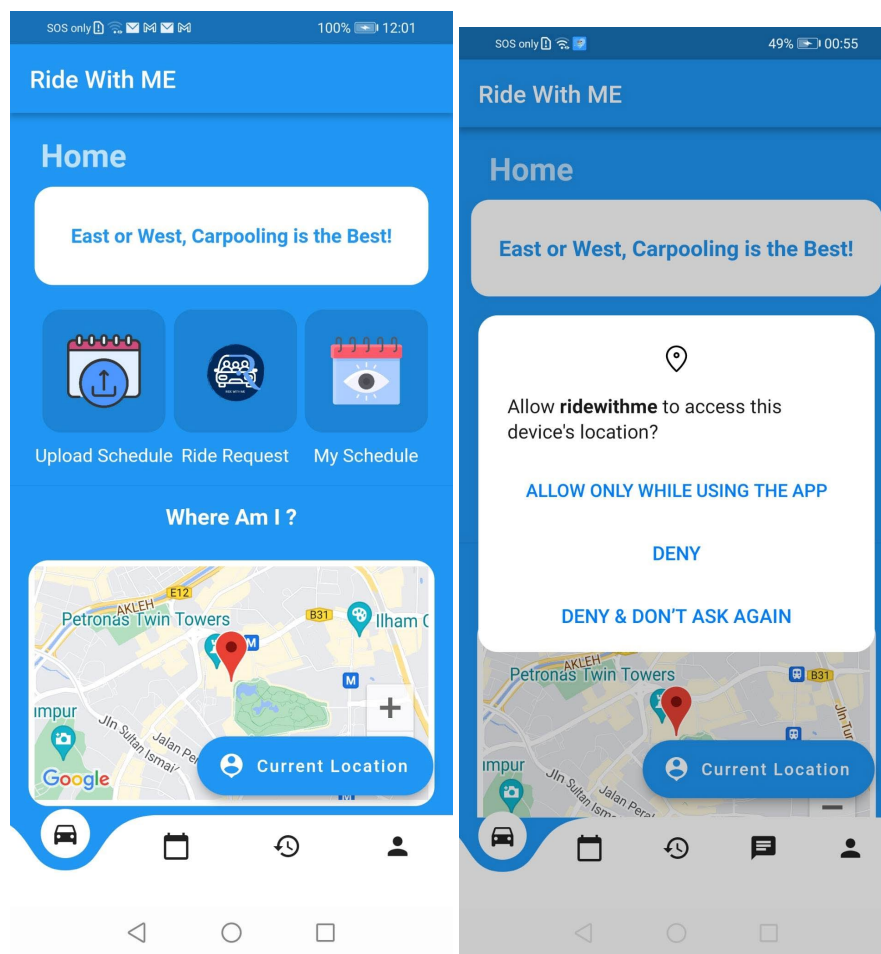


Figure 5.4.3.1 Main Screen

The main screen contains a quote to encourage users to implement carpooling in daily life. The 3 buttons which are upload schedule, ride request and my schedule is presented appealingly. The bottom navigation bar is included to enhance user experience. After the user click on the current location button, it will prompt out a request to access the user's real time location. After the user allow to access, it will present the user's current location on the Google Map shown at the bottom of page.

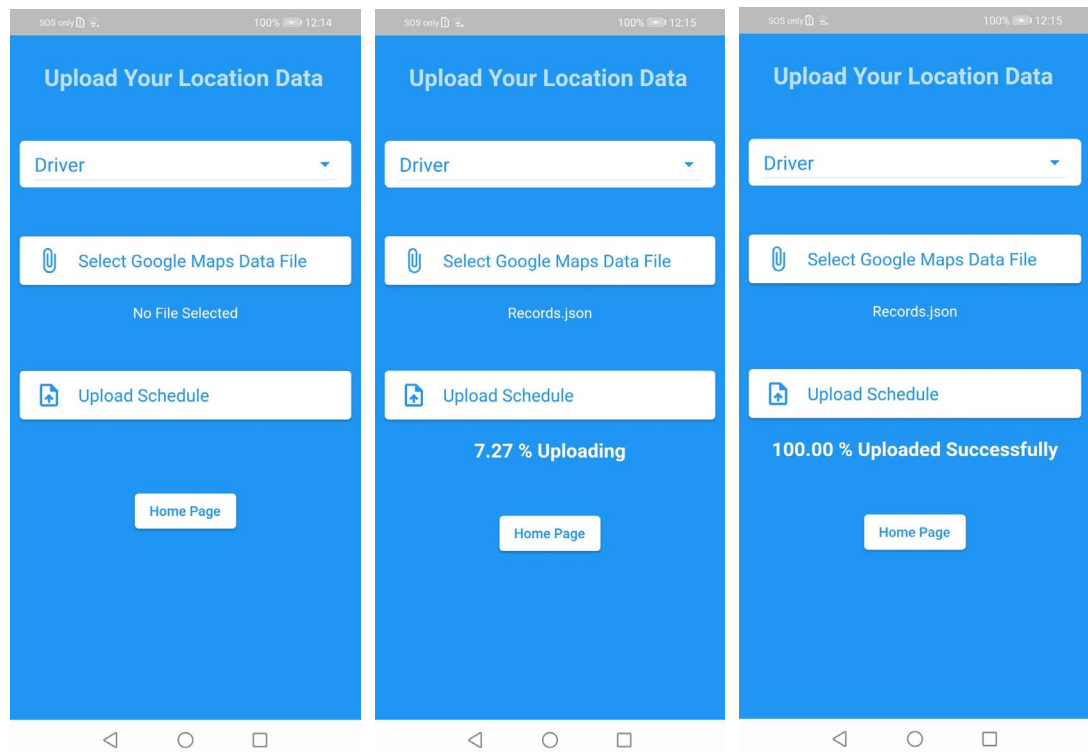


Figure 5.4.3.2 Upload Data

The first button, upload schedule is to direct user to a page to upload the user's Google Map data. After the user press on the select file button, it will prompt to ask for the permission to access to local storage. After the user select the file, the file name will present below of the button. When the user press on the upload schedule button, the uploading progress will show on the page. When the file is uploaded successfully, it will indicate the user. The user may press on the home page button to return to the main screen.

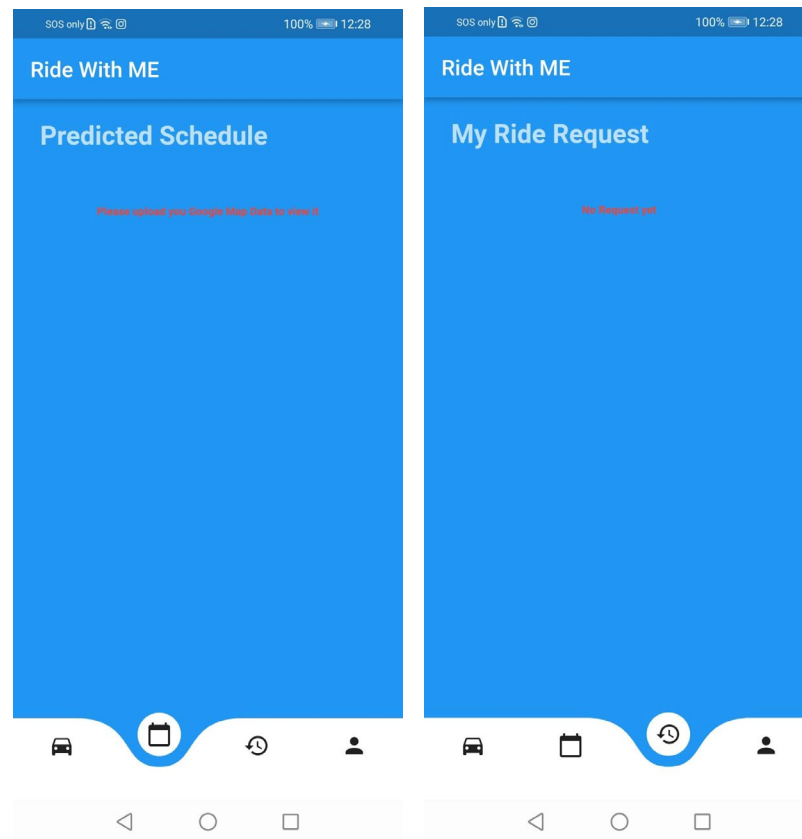


Figure 5.4.3.3 Pages

The second button in bottom navigation bar would show the user's predicted schedule of location visit, only if the user uploaded the Google map data. The third button in bottom navigation bar will show the user's ride request status if the system successfully matches the passenger request with a potential driver.

5.4.5 Firebase

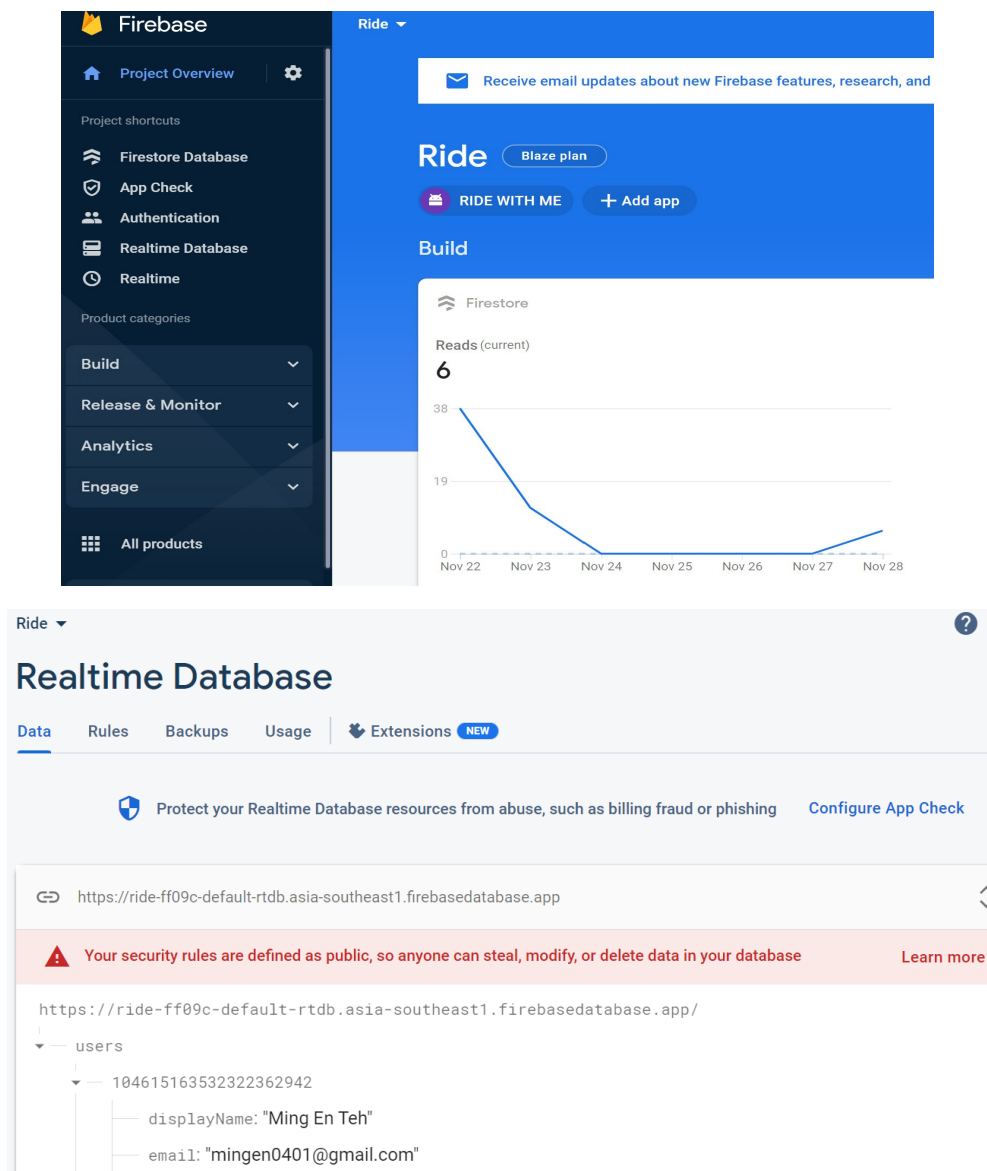


Figure 5.4.5.1 Firebase Project

A firebase account is activated. The flutter application is registered into the firebase account. The real-time database also connected to it flourishingly. Hence, the database in firebase will store and update the user information responsively.

After the user signs in through Google Sign in method, the user information will upload to the Firebase Real-time Database. It will be stored under the collection of users and inside the user's Google user ID accordingly as their unique user ID.

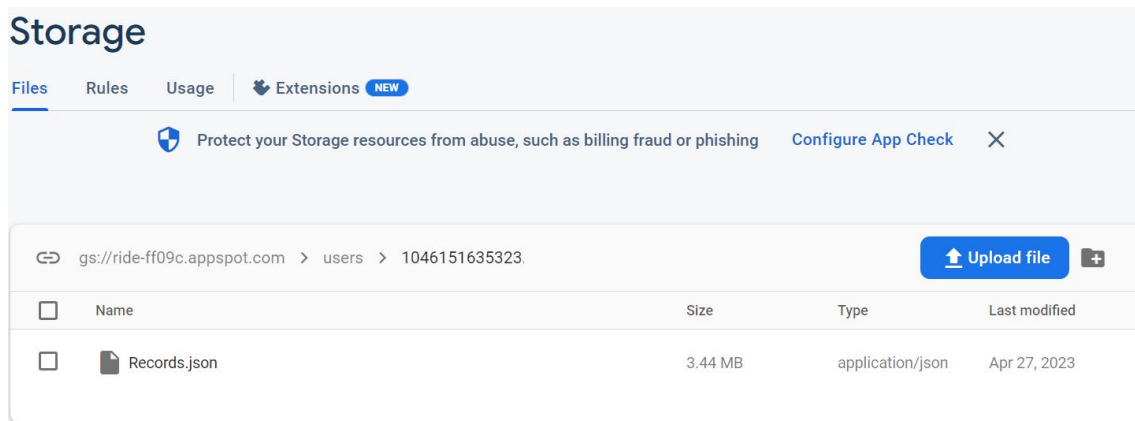


Figure 5.4.5.2 Firebase Storage

After the user uploaded the file, it will store inside the firebase storage according to their user ID.

5.5 Opportunistic AI

```
import firebase_admin
from firebase_admin import credentials, storage

cred = credentials.Certificate(r"C:\Users\TEH\Downloads\ride-ff09c-7e5309c094cd.json")
app = firebase_admin.initialize_app(cred, {
    'storageBucket': 'ride-ff09c.appspot.com'
})
```

```
bucket = storage.bucket()
bucket_name = 'ride-ff09c.appspot.com'
blob = bucket.blob(f'users/{userID}/Records.json')
blob.download_to_filename('C:/Users/TEH/Downloads/Records.json')
```

```
df = jsonRecordtoCSV('C:/Users/TEH/Downloads/Records.json', "inputRecords.csv")
df
```

	latitudeE7	longitudeE7	timestamp	time	date	day	day_int	time_range
0	28606399	1016904850	2019-12-22T07:23:40.957Z	07:23:40.957000	2019-12-22	Sunday	0	7.0
1	28606475	1016904781	2019-12-22T07:23:56Z	07:23:56	2019-12-22	Sunday	0	7.0
2	28058752	1017034491	2019-12-22T07:31:38.486Z	07:31:38.486000	2019-12-22	Sunday	0	7.0
3	28053334	1017018956	2019-12-22T07:31:54Z	07:31:54	2019-12-22	Sunday	0	7.0
4	28054006	1017054067	2019-12-22T07:32:00Z	07:32:00	2019-12-22	Sunday	0	7.0

Figure 5.5.1 Retrieve from firebase

To perform the opportunistic AI, we will first retrieve the data from Firebase storage. After extracting the file from Firebase Storage, we will download the file to the local storage.

```

df = jsonRecordtoCSV('C:/Users/TEH/Downloads/Records.json', "inputRecords.csv")

df_latest = get_df_latest(df)
df_latest = getLatLon(df_latest)
df_latest = df_latest.drop(['latitudeE7', 'longitudeE7', 'timestamp', 'time', 'date'], axis=1)
df_kpr = getKprArea(df_latest)
df_dest = dest_column(df_kpr)
df_clear = df_dest[(df_dest['dest_code'] != 88) & (df_dest['time_range'] != 0)]
X_train, X_test, y_train, y_test = splitData(df_clear)
# Train the Random Forest model
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
rf_pred = rf.predict(X_test)
print("Random Forest Accuracy: {:.2f}%".format(accuracy_score(y_test, rf_pred)*100))

df_result = pd.DataFrame()
day_list=[]
time_list=[]
dest_list=[]
proba_list=[]
cnt = 0
for i in range(7):
    for j in range(6, 24):
        print(i, j)
        result = knn.predict([[i, j]])
        dest = dest_dict[np.array(result)[0]]['destination']
        day = day_dict[i]['day']
        time = time_dict[j]['time']
        day_list.append(day)
        time_list.append(time)
        dest_list.append(dest)
        print("Destination: {}".format(dest))

        # print probability estimates
        proba = knn.predict_proba([[i, j]])
        print("Probability estimates: {:.2f}%".format(proba[0].max()*100))
        proba_list.append(proba)
        cnt+=1

df_result['day'] = day_list
df_result['time'] = time_list
df_result['destination'] = dest_list
df_result['probability'] = proba_list

result = df_result.to_json()

#update database
predictRef.set(result)
print("Database updated successfully.")

```

Figure 5.5.2 Opportunistic AI

Then, it will perform data preprocessing, data training with random forest model and output the final classification result. Finally, it will produce the destination timetable of user into the json file format and upload to the Firebase Realtime database.

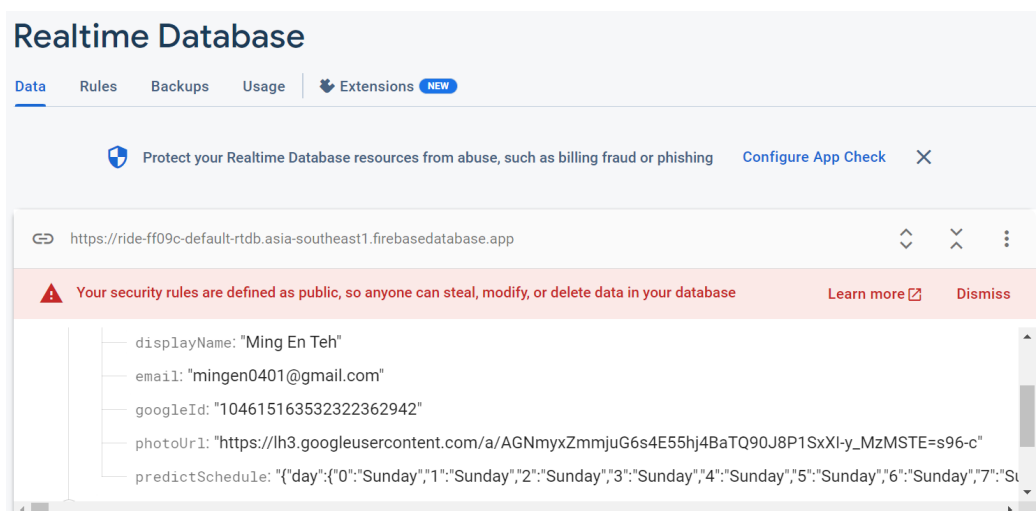


Figure 5.5.3 Result in Realtime Database

CHAPTER 6 - SYSTEM EVALUATION AND DISCUSSION

6.1 System Testing

```

def rideMatching(df_user, date, time, destination):
    proba = 0
    driver = None
    #get the time
    time = datetime.strptime(time, "%H:%M:%S").time()
    for key in time_ranges.keys():
        if time >= key[0] and time <= key[1]:
            classified_time = time_ranges[key]
            time = time_dict[classified_time]['time']
            break

    #get the day
    date = datetime.strptime(date, "%d/%m/%Y")
    day = getDay(date)
    day = day_dict[day]['day']
    print('Your search for carpool to ', destination, ' on ', day, date, ' at ', time)
    print('\n')
    print('Potential Driver List: ')
    for i in range(df_user.shape[0]):
        uData = df_user.values[i][1]
        userName = df_user.values[i][0]
        for _, row in uData.iterrows():
            if (row['destination'] == destination and (row['day'] == day) and (row['time'] == time):
                result = uData.loc[(uData['destination'] == destination) & (uData['day'] == day) & (uData['time'] == time)]

                user_prob = result['probability'].iloc[0]
                print(userName, destination, day, time, user_prob)

                if user_prob > proba:
                    driver = userName
                    proba = user_prob
                    destination = destination

    if driver is None:
        return None, None, None
    else:
        return destination, driver, proba

```

```

destination = input("Where you want to go? ")
date = input('Input the date in the format of DD/MM/YYYY: (e.g. 01/04/2023) ')
time = input('Input the time in the format of HH:MM:SS: (e.g. 13:05:00) ')

destination, driver, proba = rideMatching(df_user, date, time, destination)
print('Result :')
print("The Driver is ", driver)
print("Probability estimates: {:.2f}%".format(proba))

```

```

Where you want to go? UTAR Kampus
Input the date in the format of DD/MM/YYYY: (e.g. 01/04/2023) 11/04/2023
Input the time in the format of HH:MM:SS: (e.g. 13:05:00) 13:30:00
Your search for carpool to UTAR Kampus on Tuesday 2023-04-11 00:00:00 at 1 pm

```

```

Potential Driver List:
u6 UTAR Kampus Tuesday 1 pm 80.66666666666666
Result :
The Driver is u6
Probability estimates: 80.67%

```

Figure 6.1.1 system testing

The system is test run in python beforehand. It will output the highest probability among the potential drivers that have common destination with the passenger on the daytime in the predicted timetable.

CHAPTER 6

```
destination, driver, proba = rideMatching(df_user, datetime_str, destination)
print('Result :')
print("The Driver is ", driver)
```

Your search for carpool to UTAR Kampar Campus on Tuesday 2023-05-02 at 4 pm

Potential Driver List:

Result :

The Driver is None

Figure 6.1.2 no driver

If no driver has the same destination as the passenger at the specific daytime, it will output a null result.

6.2 Testing Setup and Result

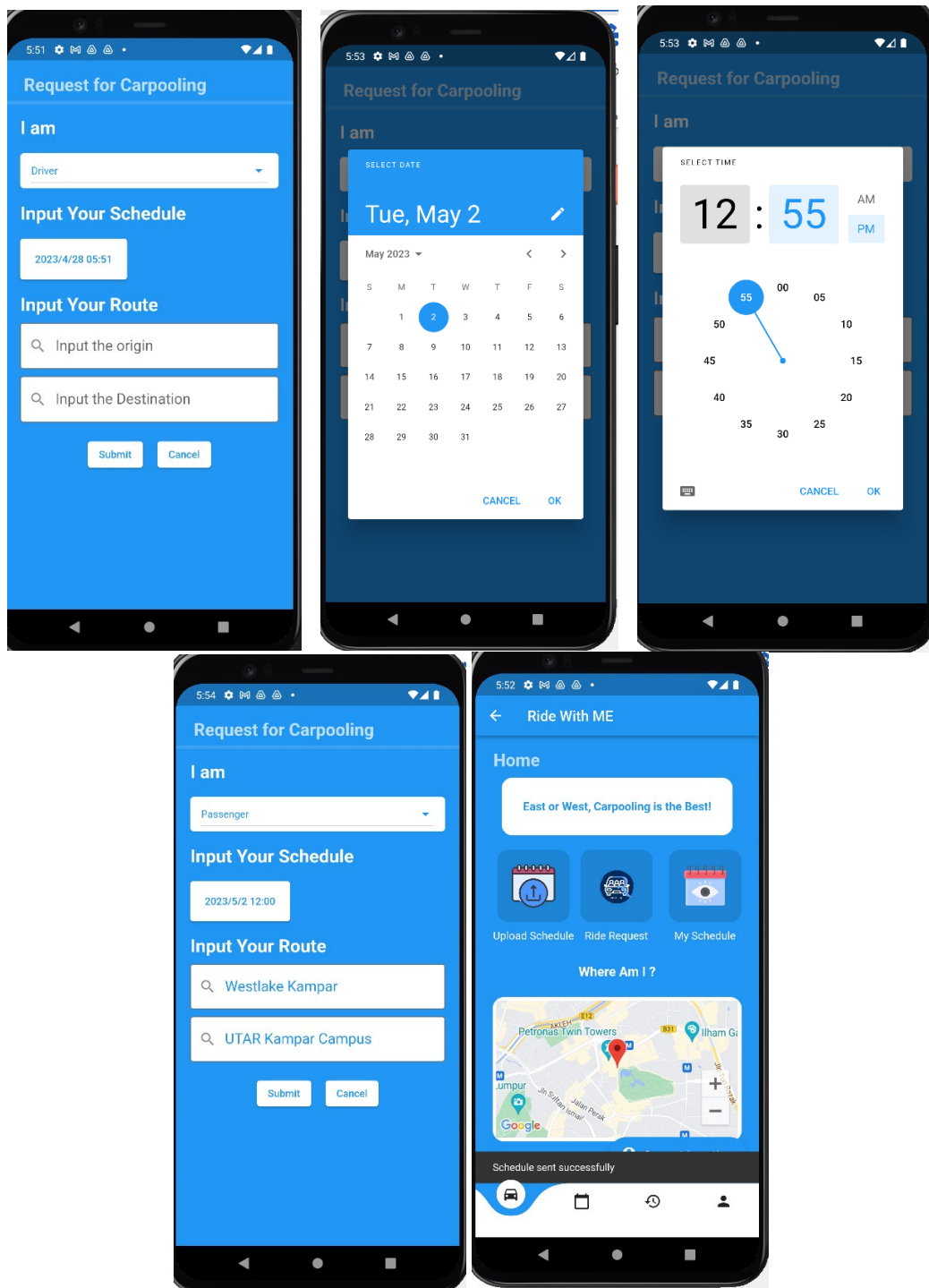


Figure 6.2.1 User interface of inputting ride request

When the user clicks on the ride request button, it will prompt out a form to let user to input his carpool details such as the date, time, origin and destination. The user interface for inputting is user-friendly. After submitting the ride request, if the request is uploaded to the firebase successfully, it will prompt out a snack bar to indicate the user that the request is sent successfully.

```

db = firebase.database()
passengerDateTime = db.child('users').child(userID).child('rideRequest').child('DateTime').get().val()
passengerDestination = db.child('users').child('userID').child('rideRequest').child('Destination').get().val()

```

Figure 6.2.2 Get ride request

First of all, retrieve the ride request details from the Firebase Realtime database.

```

import datetime
def rideMatching(df_user, datetime_str, destination):
    proba = 0
    driver = None
    destination = destination

    # Convert datetime string to datetime object
    dt = datetime.datetime.strptime(datetime_str, '%Y-%m-%d %H:%M:%S')
    # Extract date and time objects from datetime object
    date = dt.date()
    time = dt.time()

    # Classify time into time ranges
    for key in time_ranges.keys():
        if time >= key[0] and time <= key[1]:
            classified_time = time_ranges[key]
            time = time_dict[classified_time]['time']
            break

    # Classify day into weekdays/weekend
    day = getDay(date)
    day = day_dict[day]['day']

    print('Your search for carpool to ', destination, ' on ', day, date, ' at ', time)
    print('\n')
    print('Potential Driver List: ')

    for i in range(df_user.shape[0]):
        uData = df_user.values[i][1]
        userName = df_user.values[i][0]

        for _, row in uData.iterrows():
            if (row['destination'] == destination) and (row['day'] == day) and (row['time'] == time):
                result = uData.loc[(uData['destination'] == destination) & (uData['day'] == day) & (uData['time'] == time)]
                user_prob = result['probability'].iloc[0]
                print(userName, destination, day, time, user_prob)

                if user_prob > proba:
                    driver = userName
                    proba = user_prob
                    destination = destination

    if driver is None:
        return None, None, None
    else:
        return destination, driver, proba

```

```

destination, driver, proba = rideMatching(df_user, datetime_str, destination)
print('Result :')
print("The Driver is ", driver)
print("Probability estimates: {:.2f}%".format(proba))

```

Your search for carpool to UTAR Kampar Campus on Tuesday 2023-05-02 at 9 am

```

Potential Driver List:
evis@gmail.com UTAR Kampar Campus Tuesday 9 am 84.0
qianhui@gmail.com UTAR Kampar Campus Tuesday 9 am 84.0
Result :
The Driver is evis@gmail.com
Probability estimates: 84.00%

```

```

destination, driver, proba = rideMatching(df_user, datetime_str, destination)
print('Result :')
print("The Driver is ", driver)
print("Probability estimates: {:.2f}%".format(proba))

```

Your search for carpool to Westlake Cambridge Kampar on Saturday 2023-04-29 at 7 pm

```

Potential Driver List:
jarod9394@gmail.com Westlake Cambridge Kampar Saturday 7 pm 100.0
evis@gmail.com Westlake Cambridge Kampar Saturday 7 pm 100.0
paul2001@gmail.com Westlake Cambridge Kampar Saturday 7 pm 100.0
qianhui@gmail.com Westlake Cambridge Kampar Saturday 7 pm 100.0
leo08@gmail.com Westlake Cambridge Kampar Saturday 7 pm 100.0
Result :
The Driver is jarod9394@gmail.com
Probability estimates: 100.00%

```

Figure 6.2.3 Ride matching

After the analysis of the ride request, it will configure the driver who has the identical destination at the daytime mentioned by the passenger in the destination timetable. It will output the first highest probability among the potential driver list who has the same destination of the daytime.

```
db = firebase.database()
rideRequestRef = db.child('users').child(userID).child('rideRequeststatus')
rideRequestRef.set(result)
```

Figure 6.2.4 Store Result

Finally, it will store the potential driver email for the ride request back to the firebase.

6.3 Project Challenges

It is really hard to predict a human schedule for 24 hours basis. Most of the people will have inconsistent places to visit during non-working hours. For example, the accuracy of a test user on Sunday is quite low as he will visit different places during his weekend and during lunch hours students will visit different places to have lunch. Next, the extraction of Google Map data has to be educated properly and it is not efficient.

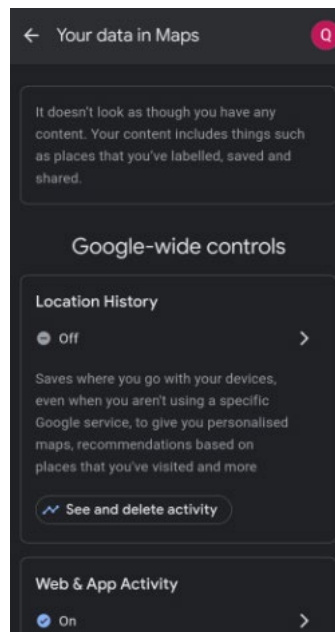


Figure 6.3 Turn Off Location History

Some of the users may turn off their GPS or turn off the location history of the Google Map app. Moreover, more users are needed in order to smoothen the ride-sharing process and to test out the real-life cases.

6.4 Objectives Evaluation

With that, a Ride with ME application with interactive and delightful interface is successfully developed. An opportunistic AI model which will predict the user's daily timetable of location visit is developed with accuracy more than 80%. The objectives are achieved victoriously.

Chapter 7 - Conclusion and Recommendation

7.1 Conclusion

In conclusion, the opportunistic AI developed able to achieve the average of accuracy that above of 80%. It will analysis the driver's location history and datetime to predict the driver's destination on different day and time in order to match the driver and passenger that are having common destination by just one tap away.

The Ride with ME application with opportunistic AI is developed with a user-friendly interface and with much consideration with the user experience. Users may act as a passenger or driver at the moment in order to join or offer a carpool to the others. The mobile application will be further developed in future to complete and achieve all the functionality.

In the future, the application will definitely help the students or people with transportation problems and further assists in promoting the car-pooling application to the world.

7.2 Recommendation

In future, the application may collab with Google to obtain the user Google map data instead of user upload it manually to ensure the efficiency.

Ideally, the passengers should have the identical destination with the driver, or their pickup point is near to the driver. If not, the system will allow the driver to request a small charge to cover the fuel consumption if the pickup point or destination of the passengers is not on their planned route. The passenger may accept the ride offered by the driver or reject it. This may increase the willingness of drivers to offer a carpool. The rating system may include to enhance the user experience during the ride sharing process. Both the drivers and the passengers can rate each other accordingly based on their ride sharing experiences such as their attitude and punctuality.

REFERENCES

- [1] L. Mitropoulos, A. Kortsari, and G. Ayfantopoulou, "A systematic literature review of ride-sharing platforms, user factors and barriers," *European Transport Research Review*, vol. 13, no. 1, 2021. [Online]. Available: https://www.researchgate.net/publication/356834141_A_systematic_literature_review_of_ride-sharing_platforms_user_factors_and_barriers
- [2] F. -S. Hsieh, "Car Pooling Based on Trajectories of Drivers and Requirements of Passengers," 2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA), 2017, pp. 972-978, doi: 10.1109/AINA.2017.41.
- [3] "Malaysia and Singapore Carpool & Traffic Camera," WeRide. [Online]. Available: <https://weride.my/>. [Accessed: 30-Aug-2022].
- [4] "Bus or carpool? your pick of rides at low prices. | Blablacar." [Online]. Available: <https://www.blablacar.co.uk/>. [Accessed: 01-Sep-2022].
- [5] S. Moosavi, R. Ramnath, and A. Nandi, "Discovery of driving patterns by trajectory segmentation," *Proceedings of the 3rd ACM SIGSPATIAL PhD Symposium*, 2016.
- [6] "Splash screens: Designing a welcome experience users adore," *Inside Design Blog*. [Online]. Available: <https://www.invisionapp.com/inside-design/splash-screens/>. [Accessed: 20-Nov-2022].

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: 3, 3	Study week no.: 2
Student Name & ID: Teh Ming En	
Supervisor: Dr. Aun Yichiet	
Project Title: RIDE WITH ME - Ride-sharing Application with Opportunistic AI	

1. WORK DONE

Collect user data.

2. WORK TO BE DONE

Data pre-processing and data cleaning.

3. PROBLEMS ENCOUNTERED

4. SELF EVALUATION OF THE PROGRESS

Ok



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT
(Project II)

Trimester, Year: 3, 3	Study week no.: 10
Student Name & ID: Teh Ming En	
Supervisor: Dr. Aun Yichiet	
Project Title: RIDE WITH ME - Ride-sharing Application with Opportunistic AI	

1. WORK DONE Opportunistic AI
2. WORK TO BE DONE Flutter user interface
3. PROBLEMS ENCOUNTERED
4. SELF EVALUATION OF THE PROGRESS Ok



Supervisor's signature



Student's signature

POSTER

.....

RIDE WITH ME



RIDE-SHARING APPLICATION WITH OPPORTUNISTIC AI

 **DEVELOPER: TEH MING EN**
SUPERVISOR: DR. AUN YICHIE

OBJECTIVES:

-  TO DEVELOP A RIDE-SHARING MOBILE APPLICATION.
-  TO DEVELOP AN INTELLIGENT OPPORTUNISTIC AI TO PERFORM ON RIDE-MATCHING.



 **APPLIES THE OPPORTUNISTIC AI TO PREDICT THE DRIVER'S DAILY SCHEDULE OF LOCATION**

PERFORM RIDE-MATCHING FOR THE PASSENGER AND THE DRIVER IN REAL TIME 

 **NICE UI AND UX RIDE-SHARING MOBILE APPLICATION**

.....



PLAGIARISM CHECK RESULT

[Document Viewer](#)

Turnitin Originality Report

Processed on: 28-Apr-2023 16:25 +08
 ID: 2078077683
 Word Count: 5681
 Submitted: 2

RIDE WITH ME - Ride-sharing Application with ... By TEH MING EN

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

include quoted	include bibliography	excluding matches < 8 words	mode: quickview (classic) report	print	download
<1% match (student papers from 01-Sep-2022) Submitted to Universiti Tunku Abdul Rahman on 2022-09-01					
<1% match (student papers from 29-Sep-2022) Submitted to CSU, San Jose State University on 2022-09-29					
<1% match (Internet from 10-Jul-2022) https://tel.archives-ouvertes.fr/tel-03522384/document					
<1% match (Internet from 01-Oct-2022) https://www.coursehero.com/file/127697913/Project-11pdf/					
<1% match (Internet from 08-Nov-2022) http://www.tonsindia.org					
<1% match (Thai Quang Le, Davar Pishva. "Optimization of convenience stores' distribution system with web scraping and Google API service", 2015 17th International Conference on Advanced Communication Technology (ICACT), 2015) Thai Quang Le, Davar Pishva. "Optimization of convenience stores' distribution system with web scraping and Google API service", 2015 17th International Conference on Advanced Communication Technology (ICACT), 2015					
<1% match (Internet from 16-Jan-2023) https://ir.lib.uwo.ca/cgi/viewcontent.cgi?article=4681&context=etd					
<1% match () Daniel Niguse Mamo, Tesfahun Melese Yijma, Makida Fekadie, Yakub Sebastian et al. "Machine learning to predict virological failure among HIV patients on antiretroviral therapy in the University of Gondar Comprehensive and Specialized Hospital, in Amhara Region, Ethiopia, 2022". BMC Medical Informatics and Decision Making					

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.: 1 of 1



FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	TEH MING EN
ID Number(s)	19ACB02716
Programme / Course	CS
Title of Final Year Project	RIDE WITH ME - Ride-sharing Application with Opportunistic AI

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u> 1 </u> % Similarity by source Internet Sources: <u> 1 </u> % Publications: <u> 0 </u> % Student Papers: <u> 1 </u> %	
Number of individual sources listed of more than 3% similarity: <u> 0 </u>	
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 <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

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

Name: Aun Yichiet

Date: 28/4/2023

Signature of Co-Supervisor

Name: _____

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	19ACB02716
Student Name	TEH MING EN
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 Plastic Cover (for hardcopy)
✓	Title Page
✓	Signed Report Status Declaration Form
✓	Signed FYP Thesis Submission Form
✓	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)
✓	Weekly Log
✓	Poster
✓	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
✓	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*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: 28/4/23

APPENDIX