The University of Hong Kong

2018/19 COMP4801

Final Year Project

Interim Report

Building an Easy-to-use Ride-sharing App for HK

Supervisor Dr. Huang Z.Y.

Members Leung Hon Man, Anthony 3035278400
Lau Chi Ho, Eric 3035326049
Lau Chun Yin, Elven 3035294715

Abstract

In recent years, ride-sharing has been an alternative to public transportation or private car rental for commuters in countries around the world. It is believed that ride-sharing is a way to alleviate the traffic congestion and reduce the air pollution caused by vehicles [1]. However, Hong Kong, as a world-class city which is suffering from traffic congestion, has not exploited the benefits of ride-sharing. One of the reasons may be that there is no ride-sharing mobile application being extensively spread in Hong Kong until now.

In an effort to increase the popularity of ride-sharing in Hong Kong, this project aims to design and implement a full-featured handy ride sharing mobile application and related web services adapting matching algorithms from cutting-edge research for people in Hong Kong.

Contents

List of Figures

List of Tables

Abbreviations

DBMS	Database Management System
DevOps	Software Development and Information Technology Operations
JWT	JSON Web Tokens
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
UI	User Interface
UX	User Experience

1. Introduction

1.1 Motivation

Since Hong Kong is one of the densest cities in the world, traffic congestion has been a problem for commuters for a long time; despite the transportation network in Hong Kong is highly efficient and fully exploited, the traffic congestion problem is escalating [1]. However, ride-sharing may be a solution to the dilemma of Hong Kong's transportation. Studies demonstrate that adapting ride-sharing can significantly reduce the demand for vehicles, and thus it is a potential solution to traffic congestion [2, pp. 25-26]. Ride-sharing can reduce the number of empty seats in private vehicles and thus can alleviate traffic congestion.

Ride-sharing has been available in Hong Kong in recent years, but the usage of ride-sharing is not popular. A survey illustrates that there are only around 33 per cent non-car owners have ever used ride-sharing services before [3, pp. 40-41]. One possible reason of this situation is the fact that there is no free-of-charge, full-featured, safe, and easy-to-use ride-sharing mobile application which can encourage public to attempt the ride-sharing services for the first time in Hong Kong.

1.2 Concepts and Definitions

The term ride-sharing can be ambiguous since it is not stated in the laws of Hong Kong [3]. In this section, the definitions of ride-sharing and related concepts will be clarified.

Ride-sharing (or car-sharing) refers to any means of transportation in which a car owner (an individual or a company) provides a shared or non-shared ride to another individual or a small group of people [3]. In this project, only two types of ride-sharing are concerned, they are car hailing and carpooling.

Car hailing (or ride-hailing) refers to a service that a driver with or without a Taxi license provides a ride to an individual or a small group of people to their destination with or without prior appointment [3]. Traditional Taxi service is one of the car hailing by definition, other examples are UberX and Lyft which use digital the platform to assign the ride requests to drivers.

Carpooling refers to a service which is the same as car hailing except that the driver will pick up new passengers with a similar direction during the ride if the capacity of the car allows.

1.3 Previous Works

Many research related to ride-sharing have been conducted and attempted to provide solutions to the dynamic matching problem appearing in the online digital platform of ride-sharing. There have been at least 38 academic papers related to ride-sharing since 2006, and most of the papers presented matching algorithms which make it possible to assign ride requests to suitable drivers instantaneously [4]. Two academic papers published in 2018 were selected for implementing the matching algorithms in this

project ([5] and [6]). After investigating those papers, at least two cutting-edge matching algorithms from those papers will be implemented, evaluated, and compared subsequently.

There are several ride-sharing mobile applications available on the market such as Uber, LYFT, GRAB, etc. However, it seems the source code and technical details of those applications are not shared with the public. On the other hand, there are no open-source ride-sharing mobile applications are observed at this moment. However these apps have been taken as references in the UI design of the mobile application.

1.4 Objective and Scope

This project aims to deliver a free-of-charge, full-featured, safe, and easy-to-use ride-sharing digital platform to people in Hong Kong for the purpose of improving the transportation system. Only mobile devices are chosen as the medium of this digital platform. Therefore, the deliverables consist of two components, a front-end mobile application and back-end web services.

For the front-end mobile application, both Android and iOS platforms are targeted. The application will also include features other than ride-sharing function, such as users account management (i.e. login, signup, password reset, etc.), map view for easy locating the driver or passenger, and in-app communications between users and drivers, except the e-payment system.

The back-end web services will have all necessary functionalities (i.e. web server, database, and matching algorithms) that support the features presenting in the mobile application.

In addition, at least two matching algorithms for riders-drivers matching will be implemented, analyzed, and compared.

1.5 Outline

This report will first present the design, implementation details, and testing approach of the ride-sharing mobile application and the back-end web server in the methodology section. Next, a review of the progress, difficulties, limitations and future plan of this project will be delivered in the evaluation section. Finally, in the conclusion section, a summary of this report will be made.

2. Methodology

In this section, the abstract-level design and implementation details of the ride-sharing digital platform are introduced.

2.1 Design

This subsection explains the overall architecture of the system, the communication method for different parties in this system, and the main features of the mobile applications.

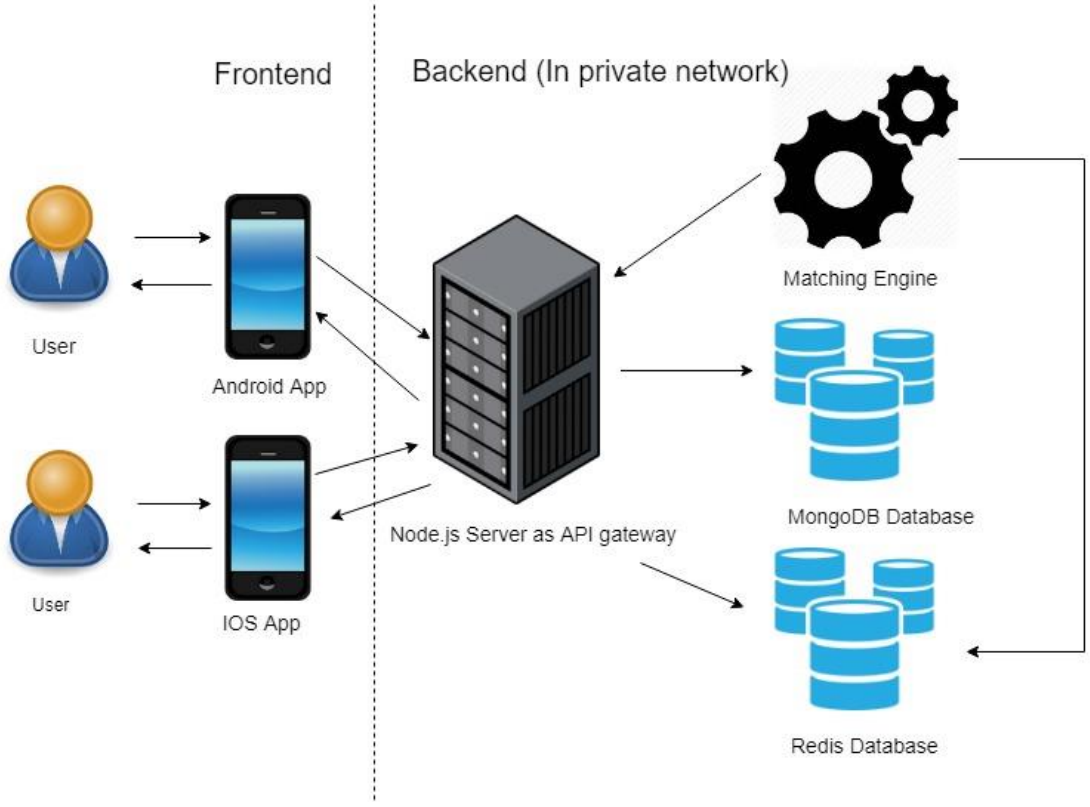

Fig.1: System Architecture of ride-sharing digital platform

The ride-sharing digital platform consists of two major parts as demonstrated in Figure 1, the front-end side and the back-end side. On the front-end side, the mobile application in each user's mobile device directly interacts with the user to provide ride-sharing services and communicates with the Node.js server on the back-end side to deliver users' information or ride requests. On the back-end side, there are four running services, they are web server, matching engine, and two databases, which provide different functionalities and communicate with each other within the private network. The architecture we use in our project is API gateway architecture. The Node.js Server in the middle of the figure act as an API gateway to encapsulates the internal structure of the backend. The mobile app only needs to 'talk' to the Node.js Server instead of connecting to the services and sending requests.

The Node.js server also provides user authentication service to ensure security.

This architecture can make the system become more flexible, scalable, and has a higher performance. In addition, it can reduce the coding work. If there is no API gateway, extra code for user authentication is needed for each service.

Workflow

The workflow of the ride-sharing service begins with a user initiating a ride request which indicates the desired departure location and destination. First, the mobile application will send the request to the web server. Then the web server will add a record of the ride request to the database before the request are delivered to the matching engine. When the matching engine has found a suitable driver for this ride

request, it first saves the details of matching in the database, then sends an acknowledgment of "match found" to the web server. Finally, the web server sends

those matching details to the mobile application, and those details (for example, driver's current location, expected waiting time, specification and appearance of the car, etc.) are presented to the user.

Main Features of the Mobile Application

In the user perspective, there are in total 6 main features in the mobile application:

1. Account management for users (i.e. sign up, log in, change and reset password)
2. Profile management (i.e. upload avatar images, edit bio, save favorite locations)
3. As a rider, finding a driver in real time
4. As a driver, finding riders to initiate a ride
5. As a driver, finding new riders with similar direction during a ride
6. In-app communication between riders and drivers

2.2 Implementation

This subsection provides the details of the current implementation of the design demonstrated in the last subsection (i.e. section 2.1), including technologies choices and user authentication process.

Technology choices

	Mobile Application	Web Server	Database	DevOps
Programming Language	JavaScript	JavaScript	N/A	N/A
Frameworks (or libraries)	React Native, NativeBase	Node.js, Express.js	N/A	N/A
Tools	Expo	N/A	MongoDB, Redis database	Google Cloud, Git, GitHub

Table 1 - List of programming languages, frameworks, and tools used in development

There are many tools and technologies available for implementing the design described in the previous subsection, yet, not all of them are suitable for this project. Thus, choices have been made carefully to ensure the development process will run smoothly, and no compatibility conflict among software libraries will occur. The selected programming languages, frameworks, and tools for this project are shown in table 2.

For the web server, JavaScript is used as the programming language with Node.js runtime environment. JavaScript is a popular choice for building web servers for mobile applications, because it has better performance, lower learning curve, more support on the Internet, compared to other choices like Python and Java.

For the database, MongoDB and redis database are chosen to be the DBMSs for the platform. MongoDB is used to store the data that will not change frequently such as the users' information. Redis is used to store dynamic data such as the locations of the riders and drivers.

This project uses two databases because each has their own advantage. Redis is a database that stores data in the memory, and it can provide a much faster response time compared to MongoDB so it is good for storing dynamic data (Fig.2) [7]. However, Redis is not good for storing complex and large data. Redis use memory to store the data which has a much more expensive cost than storing the data in hard disk. As a result, large and complex data should be better to be stored in a harddisk-based database and MongoDB is a good candidate as it provides a fair response time.

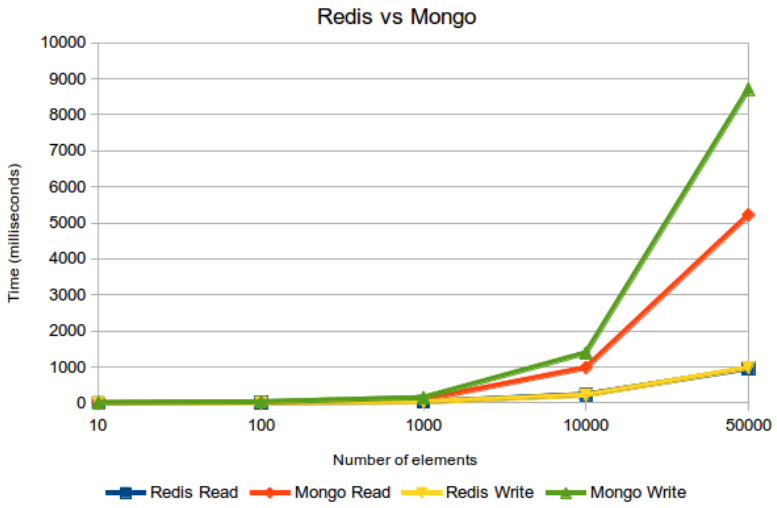

Fig.2: Table showing the response time of the read and write operation of MongoDB and Redis in different numbers of elements

In addition, compared to other available DBMSs such as PostgreSQL, MySQL, Oracle NoSQL Database, etc., MongoDB is much more suitable for this project since it is easy to learn, flexible, stable, and free-to-use.

For the mobile application, JavaScript will be the programming language with React Native and NativeBase framework. Using JavaScript for mobile application development is beneficial. Not only the development skill set can be aligned with the web server development to reduce the learning cost, but also that it is supported for both Android and iOS mobile platform development, and thus only one code base needs to be written.

Since the development of matching engine can only be done after fully understood the matching algorithms mentioned in section 1.3, there is no implementation detail for the matching engine yet, attributed to the academic papers regarding the matching algorithms are not fully explored.

User authentication process

The user authentication process in software requires special attention, because this process involved users' sensitive information which are their passwords. For this project, users' privacy and server-side security will be well handled by adopting industry standard for implementing user authentication feature.

When a user signs up in the mobile application, his or her email and password are sent to the web server via SSL and there is no third party can obtain those data. After the web server received those data, instead of storing the password into database in plain text, the web server keeps the hashed value, which is calculated by the password received and a random value, into database to ensure nobody can derive the original password from the database record.

When the user logins in the mobile application, the email and password entered will again be sent to the web server and used to generate another hashed value. By comparing the freshly generated hashed value and the hashed value in database, the identity of the user can be confirmed. Upon successful login, a JWT token will be generated and sent back to user. JWT is an encrypted string that is used for Token-based Authentication in which a user only needs to log in for once to get a JWT and mobile application can present it to the web server for each request [8]. The authentication method using JWT is suitable for mobile application because it only requires the user to log in once, easy to implement, and are reasonably safe [8].

In summary, the web server adopts a modern user authentication approach which makes use of SSL, hashing, and JWT to protect the passwords of users.

Network Communication method

The communication method for the mobile application, web server, matching engine, and database will be REST. REST is the most common communication style of web services in which one computer uses a stateless network protocol like HTTP to send data to another computer. Another popular communication method is SOAP, which is more complicated and uses not only HTTP but also SMTP and FTP. The differences between REST and SOAP are shown in Table 1. It is illustrated that REST is much easier to implement, has better performance, involves fewer technologies and thus has lower learning overhead [9].

#	SOAP	REST
1	A XML-based message protocol	An architectural style protocol
2	Uses WSDL for communication between consumer and provider	Uses XML or JSON to send and receive data
3	Invokes services by calling RPC method	Simply calls services via URL path
4	Does not return human readable result	Result is readable which is just plain XML or JSON
5	Transfer is over HTTP. Also uses other protocols such as SMTP, FTP, etc.	Transfer is over HTTP only
6	JavaScript can call SOAP, but it is difficult to implement	Easy to call from JavaScript
7	Performance is not great compared to REST	Performance is much better compared to SOAP - less CPU intensive, leaner code etc.

Table 2 - Comparison between SOAP and REST communication method [9]

For the app to app communications such as drivers chat with riders, we have used another communication method called socket.io. Socket.io support real-time communication which gives a less overhead compared to HTTP connection.

2.3 Testing Approach

This subsection describes the test tools and testing methods used in software testing for this project.

Currently, no formal testing has been conducted for any code base and only exploration tests have been made casually. To ensure the quality and correctness of the whole system, unit test, integration, and user acceptance test will be conducted for the mobile application, the web server, and the matching engine in the immediate future. In the unit test, each feature will be tested, and the purpose is to guarantee a particular feature has been completed and is bug-free. On the other hand, for the integration test, a story which is a workflow consists of multiple features, will be tested to ensure those features can work together. Lastly, the user acceptance test will be conducted with potential users to evaluate the user experiences by collecting comments from the users and investigating the reasons of dissatisfaction of the users.

Regarding the testing tools, the mobile application will be tested manually using Expo, which is a mobile application build tool for React Native framework. In contrast, we will use a tool called Insomnia to test the web server and matching engine. Insomnia provide a graphical interface for the users to manually sending requests to the server with different parameters so we don't need to make test scripts for the server testing.

To conclude, three kinds of testing will be conducted for the mobile application, the web server, and the matching engine, they are unit test, integration test, and user acceptance test. Also, the test tools used will be Expo and Insomnia.

3. Evaluation

In this section, the current progress is reported, the limitations and difficulties and the future plan are discussed.

3.1 Current Status and Results

In estimation, more than 80% of the features have been completed for the mobile application and the web server. On the other hand, no progress has been made for the matching engine yet, as we have planned to work on the algorithm part at the second semester.

For the App, a login and register system, a map view and a message system have been made.

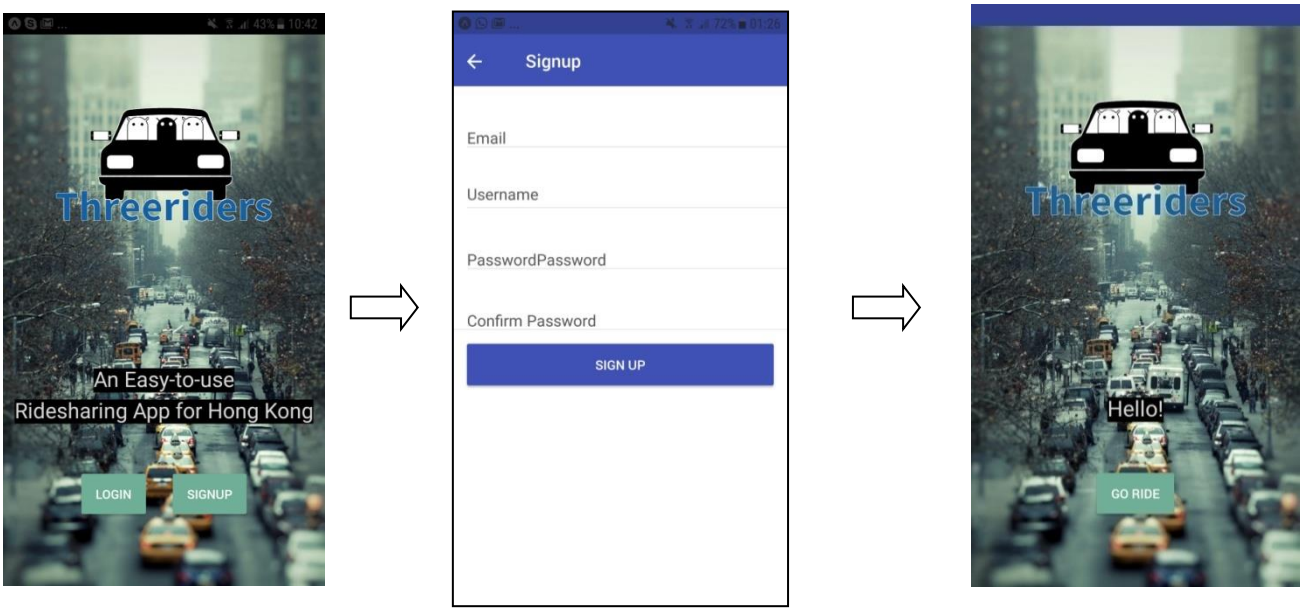

Fig.3: The signup workflow

Figure 3 shows the signup process of the user. The user first presses the signup button

on the first page and then fills in the information on the signup page. A welcome page will be shown if all the inputs are correct (no blank fields and the passwords inputted are the same).

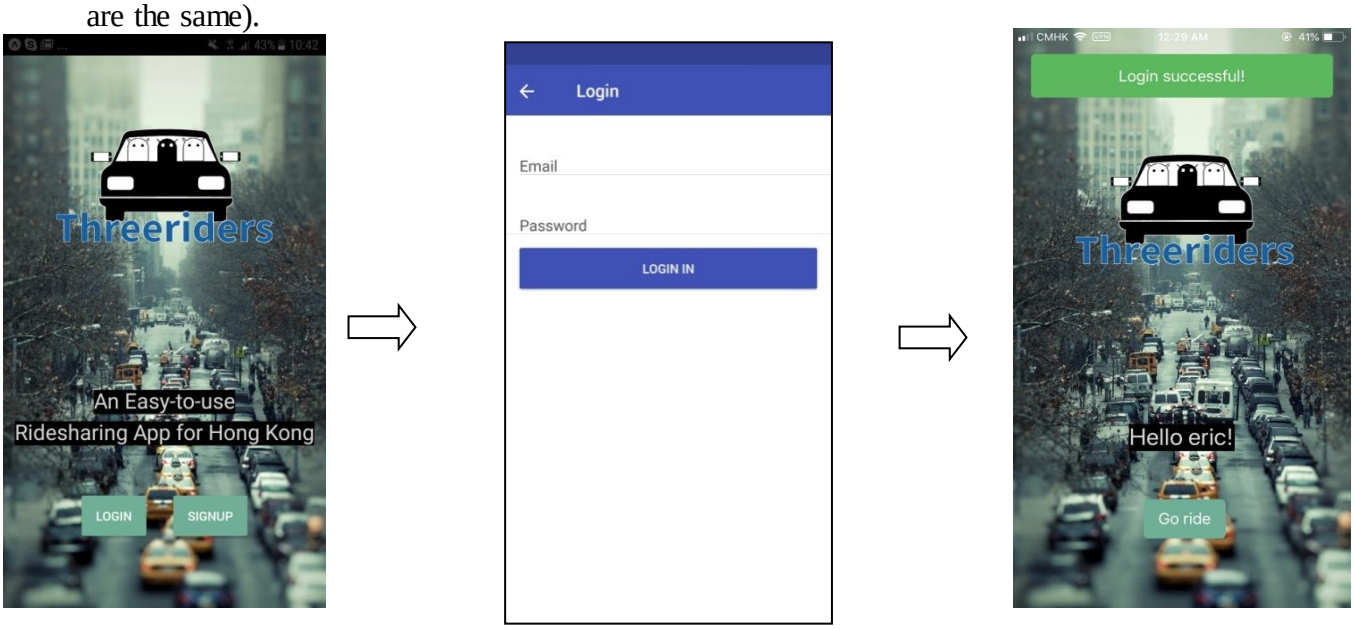

Fig.4: The login workflow

Figure 4 shows the login process of the user. The user first presses the login button and enters the login information. If the login is successful, a successful message and the username of the user will be displayed in the welcome page.

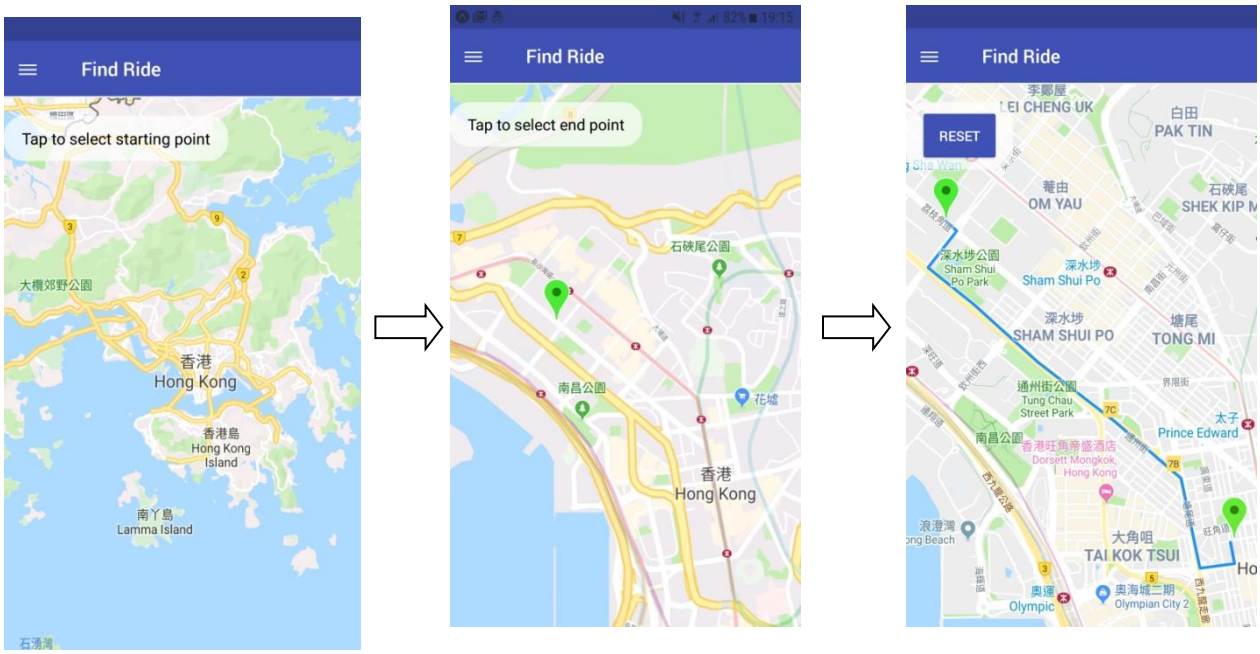

Fig.5: The path workflow

Figure 5 shows the map view of the app. When the user taps the screen, a pin will be dropped and a path will be displayed if there are two pins. The user can reset the pins dropped by pressing the reset button.

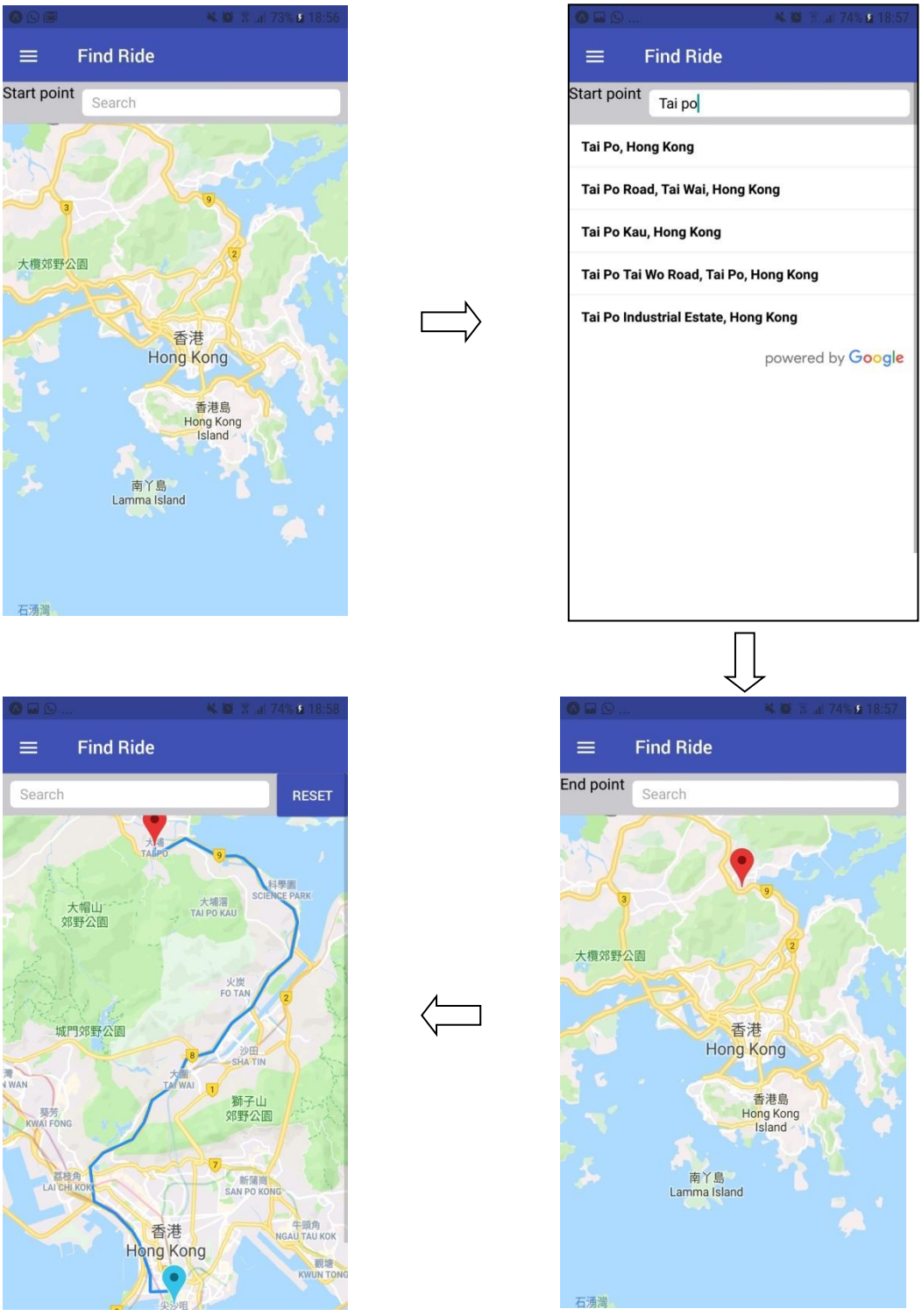

Fig.6: The place finding bar workflow

Figure 6 also shows the map view of the app. When the user taps the search bar and

enter the start point, a pin will be dropped to the place and a path will be displayed the user enter the end point also. The user can reset the pins dropped by pressing the reset button.

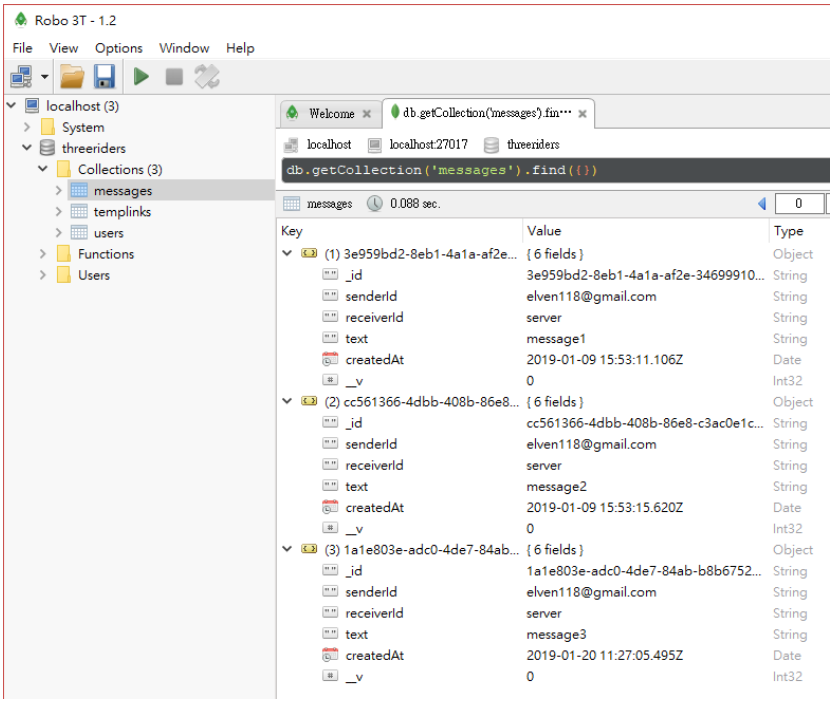

Fig.7: The message system frontend

Fig.8: The message system backend

Figure 7 shows the message page of the user. The message function is using socket.io to set up. When the user sends out the message, the server will receive the message and save to database. In figure 8, “message3” is newly sent to the server. Also, the message that saved before will be reloaded after the user navigates to the message page such as “message1” and “message2”. The message function will introduce more features after setting up the matching engine. For example, after matching the driver and the passengers, the server automatically send the message to the user to notify him.

Apart from the functions of the app, a backend server with Mongodb and Redis database has been set up and hosted on Google cloud.

Other than the functions mentioned above (user management function and message system), receive riders requests and update driver locations function have been done also.

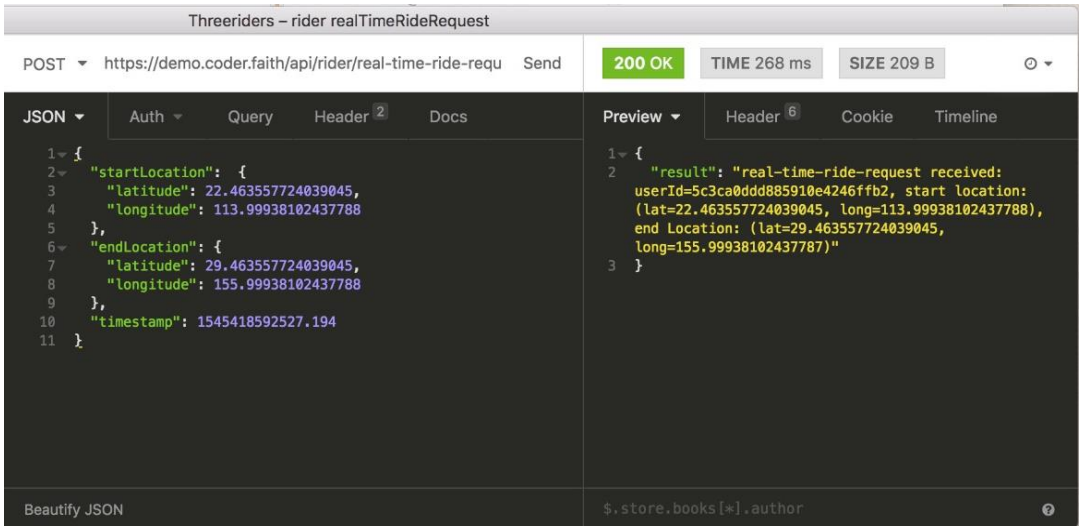

Fig.9: The backend receives rider request

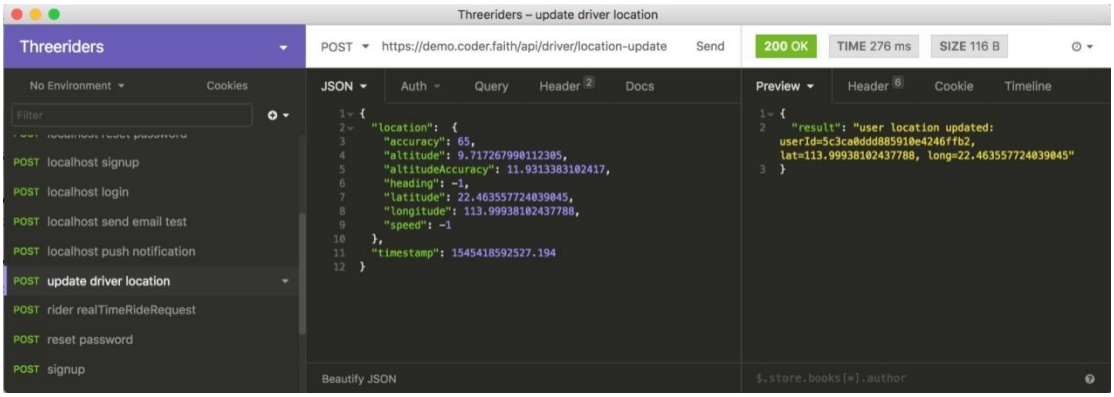

Fig.10: The backend receives driver location update

Figure 9 and 10 shows that after receiving the rider request and driver location

update, it will return a JSON object that will be used by matching engine later.

The overall progress of this project is slightly slower than expected, attributed to the underestimation of the learning overhead of mobile application development. In the original schedule, by this time, we are starting to read the papers suggested by our supervisor. According to the current status, a new schedule has been generated and shown in appendix 1.

3.2 Limitations and Difficulties

Below are the challenges that we encountered in the first three months of development.

Testing of IOS platform

Another problem encountered is only one of the team members has an IOS device. This may bring problems in the testing phase as the compatibility of other IOS devices is unknown. Also, the testing of IOS app can only be done by one person. It may slow down the development speed.

A possible solution is to use IOS simulator for the testing. The simulator can be run in the Mac Computers and it can simulate different device with different IOS versions

and hence the compatibility of other IOS devices can be tested.

Testing of backend

After the implementation of the login and register system of the backend, we didn't find a built-in way to test the backend. We can only use the python to make test scripts to send requests to the server to see the result if we want to test our new code. It is time consuming as we need to make scripts for the testing and this will slow down the development.

We have already found a solution, which is using an application called Insomnia. Insomnia allows us to send requests directly to the backend without additional implementation and changing of code. The time used for testing the backend is reduced.

3.3 Future Works

The UI of the ride matching

We have finished the backend code for the ride matching but we haven't finished the front end yet. The works are creating a button to start a ride, serve rides and end rides.

Fare calculation

Apart from the ride matching UI, fare calculation is also an important part of the project. We have decided the calculation of the fare is based on the distance traveled and the tunnel fee. The waiting time will not be considered for simplicity. The fare calculation will be implemented after the completion of the ride matching UI.

Ride matching algorithm part

The ride matching algorithm is the most important part of the project. They are the study of academic papers and the implementation and comparison of the algorithms.

Rules to compare algorithm

We only have a rough idea of what is a good matching algorithm for now. Clear rules need to be made to find which matching algorithm we implemented has the best performance.

4. Conclusion

This project presents a free-of-charge, full-featured, safe, and easy-to-use ride-sharing digital platform for smartphone users with the mission of reducing the traffic congestion by utilizing the capacity of private cars in Hong Kong.

In methodology section, the abstract level design of the whole system, the details of implementation of the current development works, and the testing approach were explained. Furthermore, the current progress and the limitations and difficulties encountered were discussed in the evaluation section.

The future plan of this project is to finish the app and the backend (except the matching engine) as soon as possible. After the whole digital platform is bug-free, we will focus on the algorithm implementation and analysis.

References

[1] Hong Kong Transport Advisory Committee. (2014). Report on Study of Road Traffic Congestion in Hong Kong [Online]. Available: http://www.thb.gov.hk/eng/boards/transport/land/Full_Eng_C_cover.pdf. [Accessed Oct. 16, 2018].

[2] Stefansdotter, A., Danielsson, C., Nielsen, C. K., & Sunesen, E. R. (2015). Economic benefits of peer-to peer transport services [Online]. Available: https://www.copenhageneconomics.com/dyn/resources/Publication/publicationPDF/0/320/1441009386/economics-benefits-of-peer-to-peer-transport-services.pdf. [Accessed Oct. 16, 2018].

[3] A. Belz, E. Healey, and K. Hudgins. (2016). Car Sharing: A Feasibility Study in Hong Kong [Online]. Available: https://web.wpi.edu/Pubs/E-project/Available/E-project-030716-041007/unrestricted/Car_Sharing-_A_Feasibility_Study_in_Hong_Kong.pdf. [Accessed Oct. 17, 2018].

[4] Federal Highway Administration. (2015). Initial Stage Reference Search: Real-time ridesharing [Online]. Available: https://www.fhwa.dot.gov/publications/research/ear/15069/15069.pdf. [Accessed Oct. 17, 2018].

[5] C. Dutta, and C. Sholley. (2018). Online Matching in a Ride-Sharing Platform [Online]. Available: https://arxiv.org/pdf/1806.10327.pdf. [Accessed Oct. 17, 2018].

[6] Z. Y. Huang, N. Kang, Z. H. G. Tang, X. W. Wu, Y. H. Zhang, and X. Zhu. (2018). How to Match when All Vertices Arrive Online [Online]. Available: https://arxiv.org/pdf/1802.03905.pdf. [Accessed Oct. 13, 2018].

[7] Tarek Salah. (2013, Nov 18). Redis vs. MongoDB Performance [Online]. Available: https://badrit.com/blog/2013/11/18/redis-vs-mongodb-performance#.W8_Jd2gzaUl [Accessed Oct.24,2018]

[8] K. Lathiya. (2018, Feb 22). Node Js JWT Authentication Tutorial From Scratch [Online]. Available: https://appdividend.com/2018/02/07/node-js-jwt-authentication-tutorial-scratch/. [Accessed Oct. 22, 2018].

[9] Main differences between SOAP and RESTful web services in java. (2017). Stack Overflow. [Online]. Available: https://stackoverflow.com/questions/2131965/main-differences-between-soap-and-restful-web-services-in-java. [Accessed Oct. 21, 2018].

Appendix 1

Updated Project Schedule

Deadline	Tasks	
Inception		
30/09/2018	Research on ride-sharing related papers	✓
30/09/2018	Research on programming languages / frameworks / technologies available for building the ride-sharing system	✓
30/09/2018	FYP Website	✓
30/09/2018	Detailed project plan	✓
Elaboration		
31/12/2018	Study of Mobile App Development Framework	✓
31/12/2018	Web server's login, register, reset password features	✓
31/12/2018	Database setup	✓
7/1/2019	Mobile Application's login, register, reset password features	✓
31/12/2019	Matching engine with a simplified algorithm	✓
31/12/2019	Detailed interim report	✓
Construction		

14/04/2019	Matching engine with cutting-edge algorithms	
14/04/2019	Mobile Application with full features	
14/04/2019	Web server with full features	
14/04/2019	Testing and performance evaluation of the system	
14/04/2019	Final report	

(Note: finished tasks are marked with ✓ symbol)