香港城市大學
City University of Hong Kong

Department of Computer Science

BSCCS Final Year Project Report
2020-2021

20CS051

Mobile-based Logistics System
with Smart Routing and Tracking Services

(Volume 1 of 1)

Student Name : **WU, Kai On**

Student No. : **55216397**

Programme Code : **BSCEGU4**

For Official Use Only

Supervisor : **Dr HANCKE, Gerhard Petrus**

1st Reader : **Dr WONG, Tsui Fong Helena**

2nd Reader : **Dr YU, Yuen Tak**

Student Final Year Project Declaration

I have read the project guidelines and I understand the meaning of academic dishonesty, in particular plagiarism and collusion. I hereby declare that the work I submitted for my final year project, entitled:

Mobile-based Logistics System with Smart Routing and Tracking Services

does not involve academic dishonesty. I give permission for my final year project work to be electronically scanned and if found to involve academic dishonesty, I am aware of the consequences as stated in the Project Guidelines.

Student Name: WU, Kai On Signature: 

Student ID: 55216397 Date: April 12, 2021

Abstract

Nowadays, online shopping becomes a popular trend for modern people. People go shopping online and receive delivery items at a desirable location. To facilitate this kind of commercial activity, a strong and mature logistics system is inevitable. So, the items can be located easily and delivered quickly, customers can thus enjoy better shopping experiences.

By allocating the root cause, the major difficulty for a small scale delivery firm is that they sometimes find it hard to schedule optimized delivery plans for drivers. They also have difficulty justifying the operating cost of their business. To enhance the delivery process efficiency, making use of the mobile application is a great solution for tracking the delivery record by use GPS, drivers can easily view the pick-up and delivery items on the map. By communicating with the business server, drivers can exchange data and perform their work schedule accordingly. Business can also make use of the application to manage and assign the delivery jobs for the drivers, further optimizing the performance of business management.

This project aims to provide a state-or-art system that helps small scale logistics company in regular operation and boost their productivity. Thus, gain a better reputation and increase their competitiveness in the market.

Acknowledgement

I would like to express my very sincere gratitude to my supervisor, Dr Hancke, Gerhard Petrus. I really appreciate his guidance and efforts throughout the year. He has provided me with many constructive and inspiring advice to help me follow the right track and improve my project.

I would also like to thank Mr Birkett, Timothy Michael from Chan Feng Men-Ling Chan Shuk-Lin Language Centre of the City University of Hong Kong, for his advice and teaching to help me deliver an academic style report.

I also appreciate the support and help provided by the CS department and staff for providing me with equipment and guidelines.

Table of Contents

1. Introduction ---- 9

1.1 Motivation & background information ---- 9

1.2 Project Aims & Objectives ---- 10

1.2.1 Problem Statement ---- 10

1.2.2 Project Objectives & Scope ---- 11

2. Literature Review ---- 12

2.1 Review on current logistics and delivery systems ---- 12

2.1.1 Bill-tracking by SF Express ---- 12

2.1.2 Geopointe by Ascent Cloud ---- 13

2.1.3 Onfleet ---- 14

2.1.4 Conclusions ---- 15

2.2 Review on scheduling and routing problems ---- 16

2.2.1 Travelling Salesman Problem ---- 16

2.2.1.1 Christofides - Serdyukov Algorithm ---- 16

2.2.2 Bin Packing Problem ---- 17

2.2.2.1 First-fit decreasing algorithm ---- 17

2.3 Mapping Solution ---- 18

2.3.1 Google Map API ---- 18

3. Major Technical Components ---- 19

3.1 Main application structure ---- 19

3.2 Live Tracking ---- 19

3.3 QR Code components ---- 19

3.4 E-signature ---- 19

3.5 Cost calculation ---- 20

3.6 Items Check-In ---- 20

3.7 Smart Jobs Assignment ---- 20

3.8 Optimal Routes Planning ---- 20

3.9 Users and Drivers Management ---- 21

3.10 Technical challenges ---- 21

4. Expected results & Deliverables ---- 22

4.1 Application Testing ---- 22
4.2 Expected Deliverables ---- 22
5. System Design ---- 23
5.1 Review of technologies in use ---- 23
5.1.1 Database ---- 23
5.1.2 Related Package Library ---- 23
5.2 Development Environment and Language ---- 24
5.2.1 Mobile Application Development Platform ---- 24
5.2.2 Android Development Version ---- 25
5.3 System Logical Flow ---- 25
5.3.1 User System Logical Flow ---- 26
5.3.2 Driver System Logical Flow ---- 27
5.3.3 Admin System Logical Flow ---- 27
5.4 Use Case Diagram ---- 28
5.4.1 Actor Description ---- 28
5.4.2 Use Case Description ---- 29
5.5 Entity Relationship Diagram ---- 37
5.6 Class Diagram ---- 38
5.6.1 Modelling Object Class ---- 39
5.6.2 Requests Controller ---- 39
5.6.3 Application Screens ---- 39
5.6.4 Functionality Class ---- 39
5.7 Workflow Diagram ---- 41
6. Methodology and Implementation ---- 42
6.1 Database Structure ---- 42
6.2 User Authentication ---- 45
6.3 Items Check-in ---- 46
6.4 Bill History and Stock Management ---- 46
6.5 Live Tracking Mechanism (with ETA) ---- 47
6.6 Optimal Route Planning Algorithm ---- 47
6.7 Driver Location ---- 49
6.8 Driver Dispatch ---- 50

6.9 Weight Sorting and Smart Jobs Assignment Algorithm ---- 50

7. Project Results ---- 52

7.1 Login Page and Registration ---- 52

7.2 User Interface and Functionality ---- 54

7.2.1 User Dashboard ---- 54

7.2.2 Items Check-In ---- 54

7.2.3 Create Bill ---- 56

7.2.4 Bill History ---- 57

7.2.5 Bill Tracking ---- 58

7.2.6 User Profile ---- 58

7.3 Driver Interface and Functionality ---- 59

7.3.1 Driver Dashboard ---- 59

7.3.2 Assigned Jobs List ---- 59

7.3.3 Register Items ---- 60

7.3.4 Process Items ---- 61

7.3.5 Driver Profile ---- 62

7.4 Admin Interface and Functionality ---- 63

7.4.1 Admin Dashboard ---- 63

7.4.2 Driver Dispatch ---- 63

7.4.3 Stock Management ---- 65

7.4.4 User Management ---- 65

7.4.5 Driver Management ---- 66

8. System Testing Plan ---- 67

8.1 Frontend-Backend Communication Testing ---- 67

8.2 Function Completeness ---- 67

8.3 Test Cases ---- 68

9. Conclusion ---- 77

9.1 Achievement ---- 77

9.1.1 Mutiple Client applications with Access Control ---- 77

9.1.2 Flexible Online Items Check-In Services ---- 77

9.1.3 Items Real-Time Tracking with ETA ---- 77

9.1.4 Items Process Security ---- 77

9.1.5 Smart Jobs Assignment ---------- 78
9.1.6 Optimal Routing and Routes Suggestion ---------- 78
9.1.7 Easy Proof of Delivery---------- 78
9.2 Expectations ---------- 78
10. Project Schedule----------79
10.1 Project Timeline and Milestones ---------- 79
10.2 Gantt Chart ----------79
11. References ----------80
12. . Monthly Logs----------81
12.1 October 2020 ---------- 81
12.2 November 2020---------- 81
12.3 December 2020---------- 81
12.4 January 2021 ---------- 82
12.5 February 2021---------- 82
12.6 March 2021 ----------82

1. Introduction

1.1 Motivation & background information

Recent years, the logistics industry is becoming more important nowadays. Online shopping is a new trend for the modern lifestyle. Especially during the pandemic outbreak in 2020, people are more relying on point-to-point delivery services to acquire what they have bought online, safely and easily. It is also believed that there will be growth in e-commerce and grocery home deliveries [1].

To facilitate the delivery process and satisfy customers requirements, a mature logistics system is inevitable. A well-designed system structure which provides real-time tracking, online items check-in, optimal delivery routes planning, delivery jobs assignment and proof of delivery services can be developed to maintain the quality of the logistics industry. Thus, companies can survive in a highly competitive market.

However, it is quite costly for small-scale delivery companies to implement this complex system. Installing GPS trackers on the vehicles for tracking the drivers' current locations, setting up a giant server for data communication, hiring staff to calculate the drivers delivery routes and also managing parcels, these functions all cost a lot for the small-scale companies.

Maintaining a mature and trustworthy logistics system may seem impossible for companies that have limited capital. However, as mobile devices are essential things in human lives nowadays. Almost everybody has got a smartphone in their pockets. Android devices which come with GPS and decent functions are available in the market at friendly prices. The popularization of mobile devices makes the logistics system more budget-friendly and possible for small businesses. Businesses can easily acquire the drivers' location, calculate

the profits, check-in and check-out parcels; customers can conveniently track their shipping items, check the status of the items, just using a mobile phone. This makes me want to build an app-based logistics system for small-scale businesses.

Logistics services are also very important to me as I do online shopping all the time. However, when I use applications provided by logistics companies such as SF Express or DHL, I discover that it is hard for me to know the exact time for my items to arrive at my home. Usually, I only get the estimated arrival day and time period, then I may need to wait at home all day long for the call of the driver when he gets close to my home. I reckon that it is possible to get a more precise estimated arrival time when making use of GPS in the drivers' mobile device. By using the GPS module in the app, drivers can also plan the best delivery route to save his time and fuels cost.

1.2 Project Aims & Objectives

1.2.1 Problem Statement

There are many unique features provided by the existing logistics management apps in the market. However, these features are all in separate apps. If the small-scale businesses want to acquire all these functions, it will be a heavy financial burden for them as they need to subscribe and purchase different plans and services between those apps or software. Also, it is a difficult task for them to maintain all of those services individually.

Furthermore, existing logistics administration systems are usually web-based where admins need to process their work at the workstation. This may cause locational barriers for a business operation.

Therefore, I reckon that it is possible to address these problems by building an all-in-one mobile-based application for logistics businesses, enhancing their business performances

with low cost and high flexibility.

1.2.2 Project Objectives & Scope

Mobile-based logistics system (MLS) targets to provide a budget and easy way for logistics business to enhance their productivity and a convenient way for clients to inquire their delivery items, it aims to provide an all-in-one solution for the company to manage delivery items, update their status, tracking drivers location, implement algorithms for best route planning and calculate the profits of each delivery task. It also provides easy proof of delivery, QR code interfaces for clients to review the items current status and estimated arrival time.

MLS will be built on three major components, including three separated mobile application interfaces in the frontend for user, driver and admin, handled by a web server to process the requests and data transactions in the backends. Also, there will be a database connecting the web server for data storage (See Figure 1.1). It is assumed that the above users can send requests using mobile devices and retrieve processed data from the database.

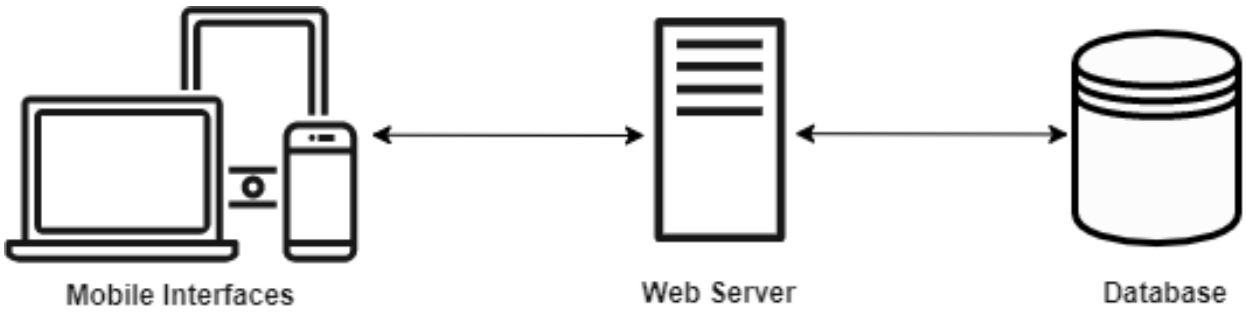


Figure 1.1 Basic illustration of MLS system design

2. Literature Review

2.1 Review on current logistics and delivery systems

2.1.1 Bill-tracking by SF Express [2]

Logistics company SF Express provides similar applications that are using this kind of technology such as web-based bill tracking. Customers can type in the bill number and track the delivery progress of the items to see item the current location and some bill information. The flow of the delivery and basic information are shown below (See Figure 2.1).

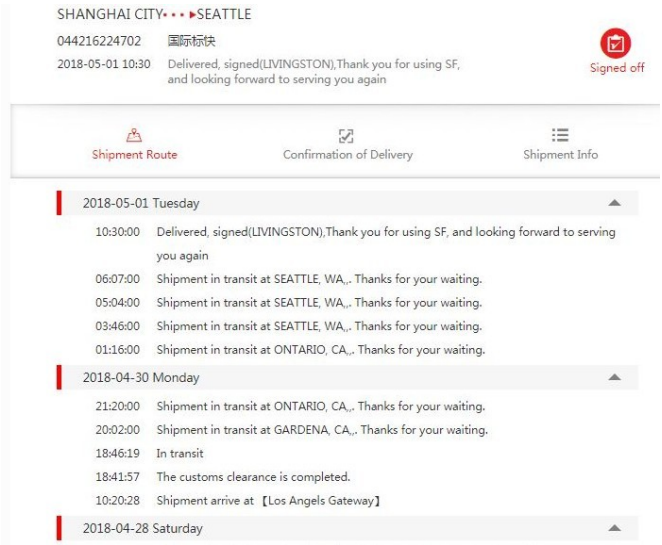


Figure 2.1 SF Express web-based bill tracking screenshot

SF Express also comes up with app-based bill tracking services providing the same kind of information with the web services. Synchronization has been done on both platforms to let customers inquire about their bill information anywhere. It also includes an e-signature function for proof of delivery and items check-in function (See Figure 2.2). However, it is hard to track the live location of the items as the information is updated a few hours later after the items have arrived and registered in the specific business checkpoints.

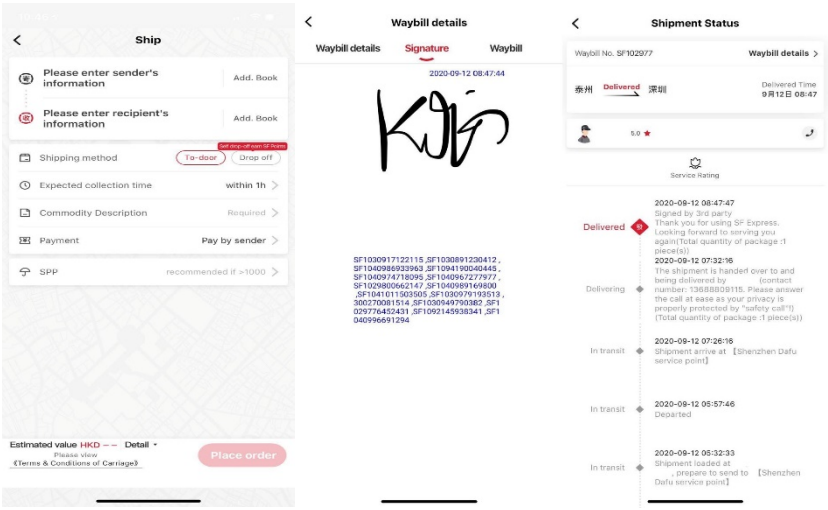


Figure 2.2 SF Express app-based check-in function, e-signature bill tracking screenshot

2.1.2 Geopointe by Ascent Cloud [3]

Geopointe is a geographic searching and analysis tool for business developed by Ascent Cloud company. Its web-based software can let users discover and analyze a specific area for business opportunities. There is a searching feature showing the target information, location and contact. It also provides a routing function that helps plan a suitable route to the target for users (See Figure 2.3). However, it does not include functions for items management and calculating the time of arrival.

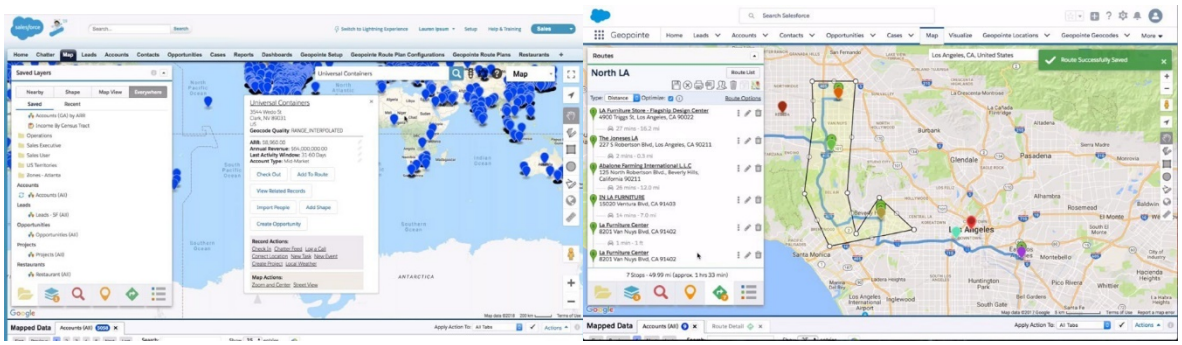


Figure 2.3 Target searching and route planning on Geopointe website

Geopointe also comes up with a mobile application that allows users to plan their routes conveniently. It follows the users' current locations and delivers business opportunities near them. Route planning can also be done using the current location provided by the mobile GPS module (See Figure 2.4).

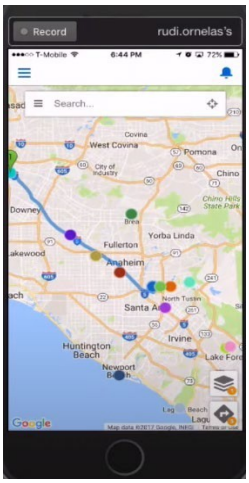


Figure 2.4 Target searching and route planning on Geopointe app

2.1.3 Onfleet [4]

Onfleet is a newly developed software that provides an advanced solution for logistics companies. On the website, companies can monitor drivers, on foot members and bikers who are doing the delivery (See Figure 2.5). It has live location tracking, route planning and dispatch management. This solution mainly focuses on increasing the efficiency of delivery and providing solid logistics management.

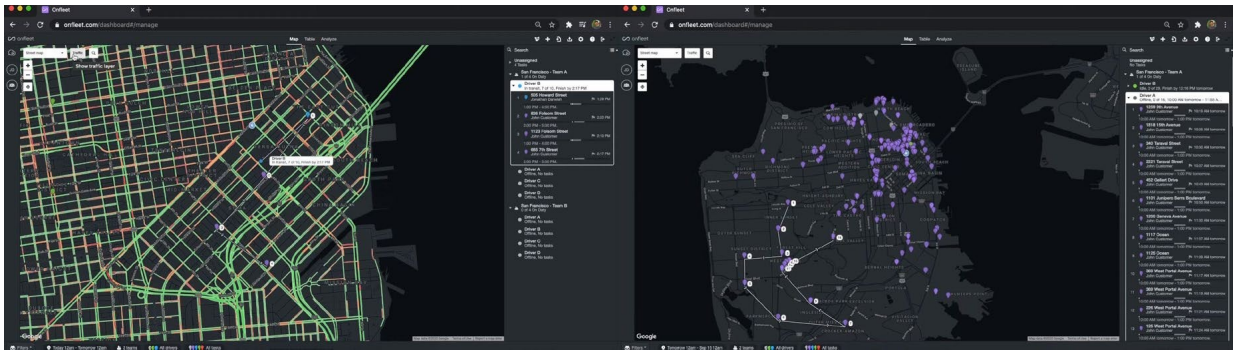


Figure 2.5 Route planning and location tracking on Onfleet website

Onfleet also provides an application version for facilitating delivery workers communication. Workers can contact the clients and dispatch centre according to the information provided on the app. Workers can also receive the details of every single task. Proof of delivery can also be done within the app (See Figure 2.6). However, it does not provide the function to check-in items online and calculate

the delivery profit of the tasks.

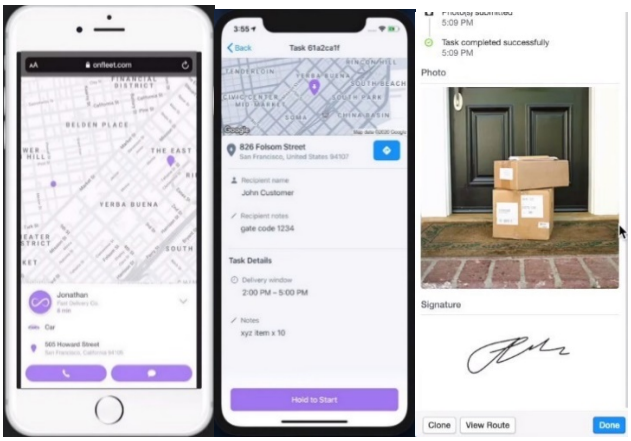


Figure 2.6 Communication function, task picking and proof of delivery on Onfleet app

2.1.4 Conclusions

From the research above, it shows that there are many similar systems in the market that provide logistics management functions for companies and delivery service functions for users. However, these functions are separated and some of them are not management-oriented. There are also limitations for tracking items live location, performing stock management, checking in items and deploying smart routes optimization.

After the comparison (See Table 2.1), I reckon that it is possible to build an all-in-one application that included all the features that mention above. So that it can provide logistics businesses with a productive, mature and efficient management solution and also an interactive and user-friendly logistics application users.

Features	Market Existing Products			Prosposd Product
	SF Express	Geopointe	Onfleet	Moblie-based Logistics System
App Interface	✔	✔	✔	✔
Items Check-in	✔			✔
Bill Tracking	✔		✔	✔
Live Location		✔	✔	✔
Proof of Delivery	✔		✔	✔
Route Planning		✔	✔	✔
Stocks Management				✔
Profit Calculation				✔
Bill History	✔			✔
Items QR Code				✔
Driver Identity Proof				✔
Smart Driver Dispatch			✔	✔
Users and Drivers Management				✔
Estimated Arrival Time				✔

Table 2.1 Comparison between Products

2.2 Review on scheduling and routing problems

2.2.1 Travelling Salesman Problem

Travelling Salesman Problem (TSP) is a well-known NP-hard combinatorial optimization problem in computing. It aims to acquire the optimal route solution of travelling to every given city once and returning to the starting point with the shortest distance [5]. TSP can be solved by the naive approach, brute-forcing algorithms on every combination. However, the overall computing time is not efficient and costly. It requires 0(n!) (n is the number of cities) running time for finding the optimal time. Although it is difficult to solve the TSP, there are heuristic and approximation algorithms suggested. Christofides and Serdyukov Algorithm is one of the approaches.

2.2.1.1 Christofides - Serdyukov Algorithm

Christofides - Serdyukov algorithm is an approach to find the approximate

solutions for the TSP, which was introduced by Nicos Christofides and Anatoliy I. Serdyukov in 1976 [6]. It is proved that the cost of the solution is within 3/2 of the optimum by using this algorithm.

The algorithm creates graph G = (V,w) be an instance of the TSP, where V is the number of vertices and w is the weight assigned to every edge of G. After that, G is reduced to a minimum spanning tree MST(G(V,w)), where all of the V are connected with the minimum total weight of w. Then, a minimum-weight perfect matching is deployed to all the odd-degree V in the subgraph MST. The nodes will have all even-degree after adding the perfect matching edges in MST. Then an Eulerian circuit is available on the graph which can visit all the edges once. Then shortcutting the circuit to be the output by skipping repeated vertices and make it into a Hamiltonian circuit [7].

However, the algorithm running time will increase with the increasing number of nodes. There are sometimes better solutions for the TSP in other algorithms. As Google API contains optimization methods, the above problem can be reviewed as references when the project is further developed.

2.2.2 Bin Packing Problem

In Bin Packing Problem, objects of different volumes are assigned to be packed into a set of bins or containers in the way that minimizes the usage of bins [8]. It is a NP-complete problem and it can be solved by the naive approach, brute force searching which tries to put the item into the bin and see if it is fit, then repeat the process. However, it is very slow and costly.

2.2.2.1 First-fit decreasing algorithm

First-fit decreasing algorithm is trying to sort the items in decreasing order by

using quick sort or merge sort in O(N log N) complexity [9]. Then the items will be put into the bins one by one correspondingly. It yields non-optimal solutions in the main but is helpful and efficient in common use. In this project, we will help company to calculate the cost and find the optimal the solutions which can be inspired by the algorithm above.

2.3 Mapping Solution

2.3.1 Google Map API

Google API services provide numerous APIs across different platforms such as iOS, Android and web (See Figure 2.7). Google Map APIs like Directions, Geocoding and Geolocation can allow users to plan their routes and coordinate their current position by converting and retrieving the latitude and longitude [10]. Interaction features, such as marking and information display controlling. Routing optimization is also available for multiple waypoints in Geocoding, which is useful in this project. Distance Matrix is also an important component which can help calculate the time and for those waypoints. In this project, Google Map APIs will be used in mapping and routing .

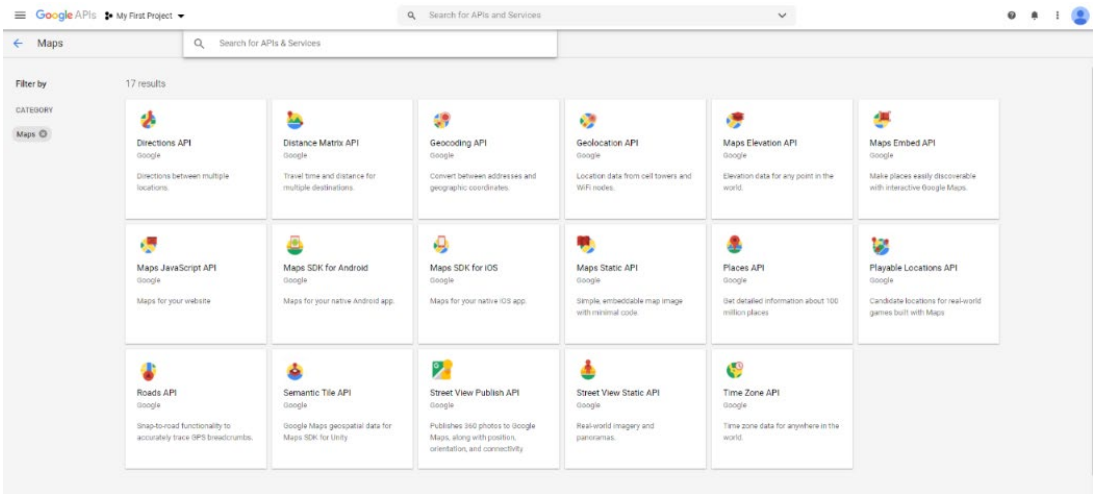


Figure 2.7 Google Map APIs components demo

3. Major Technical Components

3.1 Main application structure

In MLS, the application interfaces will be built using Flutter on Visual Studio Code. Google Firebase database will be also built to store the delivery information and users identities information. This can facilitate login function, bill tracking and check-in / check-out functions for the items. There will be three separate interfaces for user, driver and admin according to their login identities with access control.

3.2 Live Tracking

For live location tracking, Google Maps SDK for Android API and Flutter location package will be used in order to retrieve the GPS module information from the phone and display it on the map. The received location coordinates from driver GPS will be automatic updated to the database for users to tracking their delivery items that on the driver's vehicle.

3.3 QR Code components

QR Code function will also be deployed to foster the bill tracking process and delivery security. A key-value attribute will be created for every item entity. Then, it will generate a QR Code using Flutter qr_flutter package and display on the screen. Users and drivers can then scan the QR Code to read the key value to match the key-value attribute in the database to retrieve the target record for bill tracking and items delivery. For delivery security, there will be a QR Code generated for the drivers which contain their identity and information. Users can scan it before receiving the items to secure the delivery process.

3.4 E-signature

For easy proof of delivery, Flutter CustomPainter package will be implemented to create a sketch interface for users to sign their name when they receive the items.

Then the signatures will be uploaded to Firebase storage as images for records.

3.5 Cost calculation

For the delivery cost calculation, Google Distance Matrix API will be implemented to find out the actual distance in the map for a delivery task and multiply with the fuel costs input by admin. Admin can adjust the fees and it will become the delivery fees for the users. Admin can also check the total profit expected to earn by all the bills.

3.6 Items Check-In

Delivery items of users can be registered within the user interface. There are several technical components will be implemented to facilitate the check-in process. Google Map API will be used to display the map. Then location autocomplete API will help the user to input the accurate location. Google Distance Matrix API will help construct the fastest route between pick up point and destination. Lastly, the distance will be shown on screen and calculation the delivery fees.

3.7 Smart Jobs Assignment

Smart jobs assignment will be available on the driver dispatching screen. Admin can either assign the jobs to a driver one by one or use the smart assignment function to bulk assign the jobs to a driver. An algorithm will be designed to calculate the weight of the items and compare the driver's vehicle capacity. Exceptions will be handled if the capacity is exceeded or the bill status is not ready for assigning.

3.8 Optimal Routes Planning

Another component related to map coordinates is the algorithm for route planning. For that, Google Map Direction API, Google Distance Matrix will be implemented to calculate the distance. An algorithm will be designed to calculate the next

nearest destination regarding the current location for the driver to deliver the jobs list assigned to him. Multiple points of location can also be handle using this function.

3.9 Users and Drivers Management

Users and drivers management will be available in the admin interface, Admin can check the list of the users, drivers with detailed information. Admin can call and email the users and check the drivers' jobs schedule.

3.10 Technical challenges

Three application interfaces and a database system are needed to be designed and built, and it has to link to the webserver to process the requests done on the mobile devices. It is challenging to develop multiple functions and connect them to three application interfaces. There will be a lot of attributes to handle for the bills, users and drivers information. Besides, bill status includes "Pending", "Assigned", "Processing", "In Stock" and "Delivered", it is complicated to design and handle exceptions for each of the status.

It is also challenging to design suitable and optimal algorithms to calculate the best routes, to periodically update the coordinate information for real-time tracking and for smart assignment to automatically assign jobs to the drivers according to the items' weight and driver status. Also, mapping and tracking functioning together are difficult to construct. Lastly, access control is needed to separate the interfaces between users, drivers and admins.

4. Expected results & Deliverables

4.1 Application Testing

The application will be tested for every function. On each of the user, driver and admin accounts. Different use cases including extreme cases will be designed to the system to test out the results. I expect those applications to have no error and be compatible with different plugins and APIs.

4.2 Expected Deliverables

A mobile app which is connected to a database for the logistics system is expected to produce. This application can provide services to users for easier and efficient bill tracking and online check-in. It also provides them with a more accurate ETA of the delivery. For the drivers, they can easily check out the delivery items details and process easy proof of delivery. On the company side, it can manage the stocks, calculate the optimal travelling routes for each delivery and assign jobs to drivers efficiently to boost the productivity of logistics.

This will be a mobile-based application for user, driver and admin. Thus, a convenient, user-friendly and phenomenal application will be delivered which will also break the geographic and time boundary, it will be very efficient for logistics companies, especially during an epidemic.

5. System Design

5.1 Review of technologies in use

5.1.1 Database

Many popular databases in the market can provide stable and trusty services. MongoDB is one of the popular databases, which is a document-oriented type with a dynamic data structure. It uses BSON files for storing and with fast data handling. Firebase is another multi-functional platform provided by Google. There are functions for Authentication, Realtime Database, Cloud Storage, Hosting and Function.

In this project, Realtime Database from Firebase will be used to store and process data. It is a NoSQL document and collection database for mobile and web app development, providing offline data access for iOS, Android and web SDKs. The most important feature is realtime data synchronization as it makes use of socket connection to continuously send data directly from the database side, which is different from handling traditional Http requests first. Realtime location tracking is a major feature in this project. It requires fast and responsive data retrieval. Therefore, realtime database with synchronization is very usable in this case. Lastly, Authentication is also a significant feature that will be used for access control and distinguish login identities.

5.1.2 Related Package Library

Flutter app is built on Dart language and there are useful packages provide on pub.dev. Importing packages into this project can make development for different kinds of functions easier. The table below shows the corresponding packages for some major functions in this project (See Table 5.1).

Function	Package	Version
get_userLocation	geolocator	5.1.5
get_devicePermission	permission	0.1.7
scan_qrcode	barcode_scan	2.0.1
gen_qrcode	qr_flutter	3.2.0
auto_completeLocation	flutter_google_places	0.2.3
get_time	intl	0.16.1
display_map	google_maps_flutter	0.5.11+1
firebase_services	firebase_database	4.1.0
login / logout	Firebase_auth	0.18.1+2
upload_signature	Firebase_storage	4.0.0

Table 5.1 Function - package relationship

5.2 Development Environment and Language

5.2.1 Mobile Application Development Platform

According to business analytics company IDC, the expected market share for smartphone operation system will be dominated by iOS and Android, where Android is occupied 85% of total market share in 2020 [11] (See Figure 5.1). Therefore, Android oriented application will be developed to target the majority of smartphone users and logistics companies.

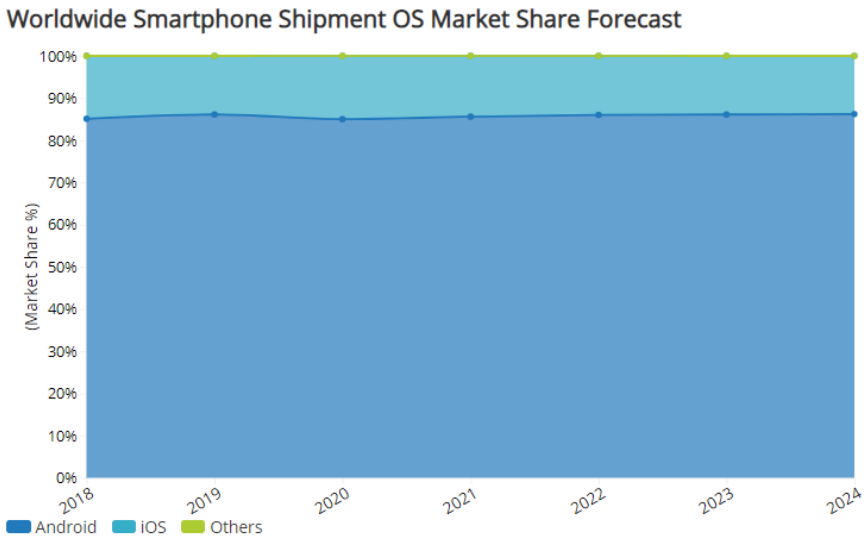


Figure 5.1 Market share forecast between iOS and Android

5.2.2 Android Development Version

From the information provided by Android Studio, it shows that most of the Android users are using Android 9 (See Table 5.2). In order to accept more users, the application will be developed under Android 9 “Pie” version.

Android Platform Version (API Level)	Distribution (as of April 10, 2020)
Android 4.0 “Ice Cream Sandwich” (15)	0.2%
Android 4.1 “Jelly Bean” (16)	0.6%
Android 4.2 “Jelly Bean” (17)	0.8%
Android 4.3 “Jelly Bean” (18)	0.3%
Android 4.4 “KitKat” (19)	4%
Android 5.0 “Lollipop” (21)	1.8%
Android 5.1 “Lollipop” (22)	7.4%
Android 6.0 “Marshmallow” (23)	11.2%
Android 7.0 “Nougat” (24)	7.5%
Android 7.1 “Nougat” (25)	5.4%
Android 8.0 “Oreo” (26)	7.3%
Android 8.1 “Oreo” (27)	14%
Android 9 “Pie” (28)	31.3%
Android 10 (29)	8.2%

Table 5.2 Android versions distribution

5.3 System Logical Flow

The logistics process will be divided into three components. First, users can login into the app to check-in their items for immediate or later delivery. Users should input the correct pickup location, destination, receiver information, delivery time, item details and contact information. The bill status will update to “Pending” once the bill is created. Users can also check the bill history and track the item’s live location on the app.

Then, the administration centre will receive the bill for delivery. Driver dispatching and smart jobs assignments will be done according to the items weight and the

vehicle capacity. The bill status will update to “Assigned” and the jobs requests will be sent to the selected driver. Besides, admin can also manage the stocks, users and drivers using the app.

Finally, driver should get online and register the item on the app when it is loaded on the vehicle or ready to be picked up so that the users can track the item live location. The driver app provides optimal routing for the jobs list assigned by the admin. Driver can perform the delivery service according to the route provided. Once the driver arrives at the destination, driver should provide the identity QR Code for the receiver to scan to secure the delivery. Then, driver should scan the QR Code generated from the user app to retrieve the bill number. A signature of the receiver will be used for the proof of delivery. The item status will update to “In Stock” for pick-up jobs or “Delivered” for delivering jobs. The signature will be uploaded to the database for references.

5.3.1 User System Logical Flow

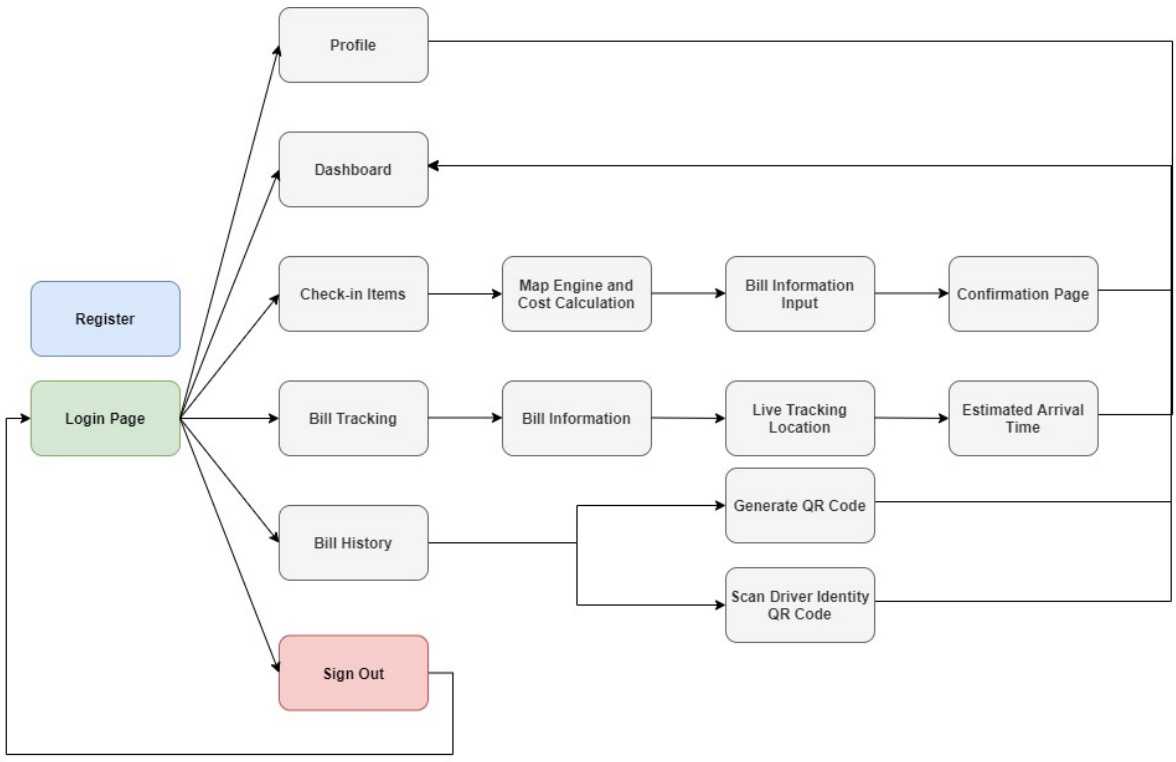


Figure 5.1 User System Logical Flow

5.3.2 Driver System Logical Flow

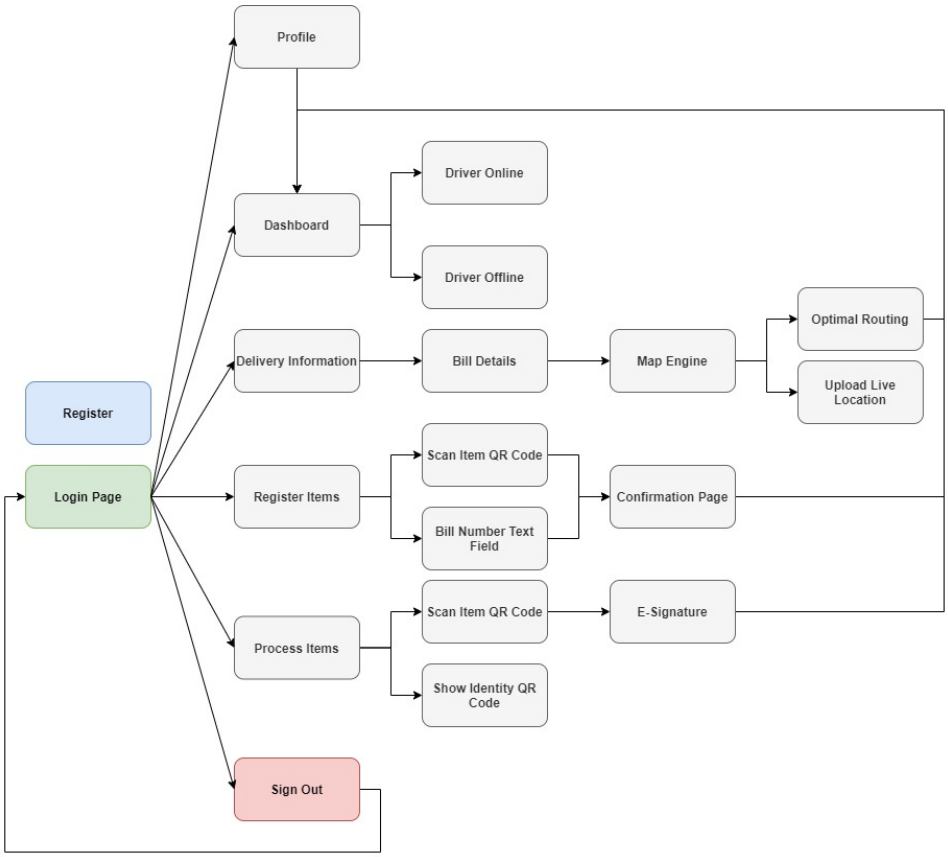


Figure 5.2 Driver System Logical Flow

5.3.3 Admin System Logical Flow

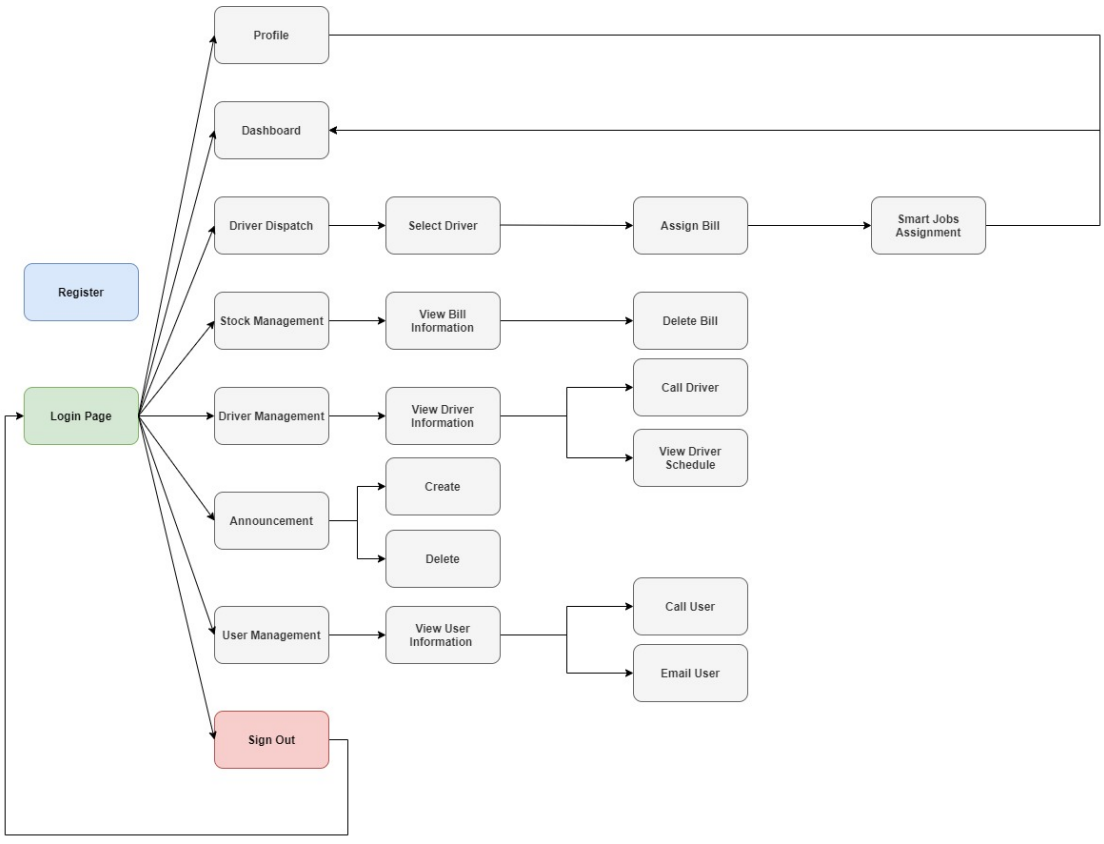


Figure 5.3 Admin System Logical Flow

5.4 Use Case Diagram

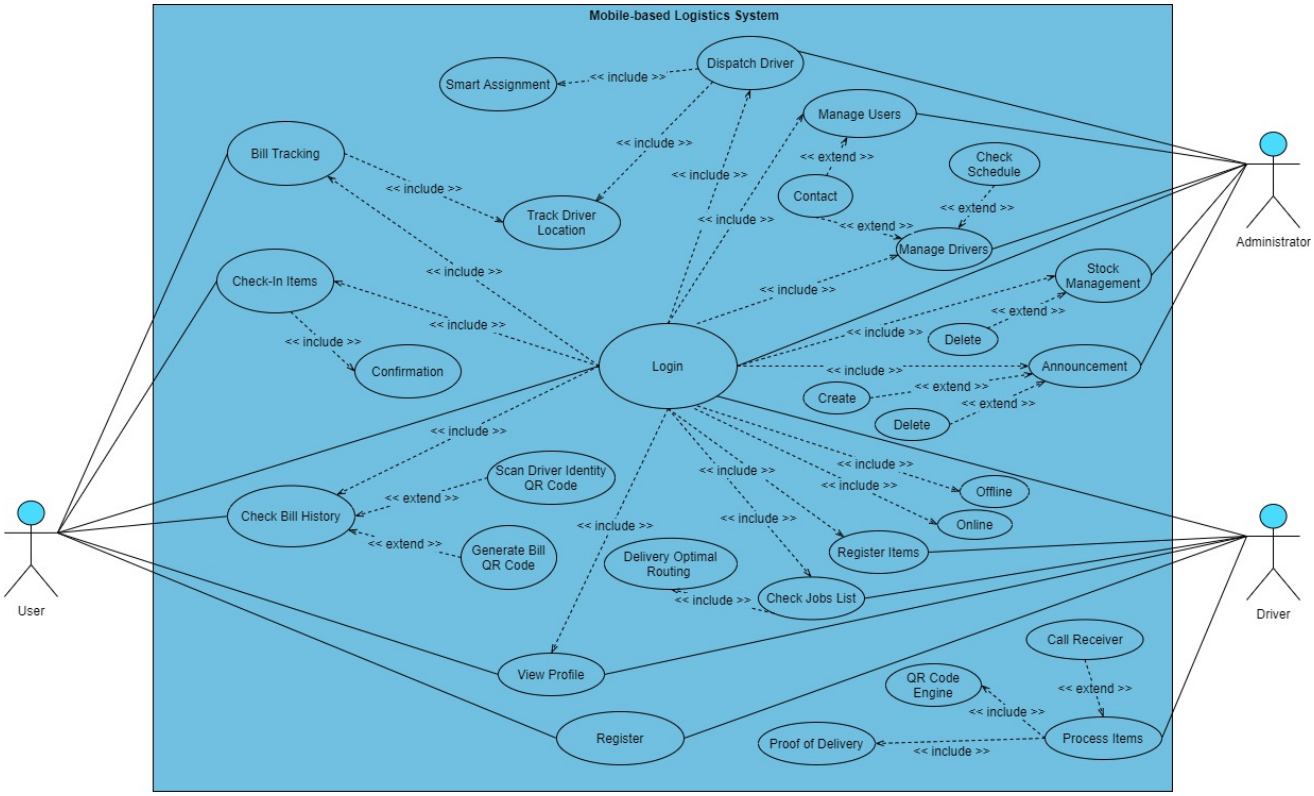


Figure 5.4 Use Case Diagram

5.4.1 Actor Description

Actor	**Description**
Administrator	Administrator operates the logistics functions, manage order and stock, assign driver schedule and dispatch the drivers.
Driver	Driver works with the mobile application to check schedule, delivery items after viewing the item details and help confirm the delivery.
User	User makes use of the mobile application to check-in items, track bills and check the bill history.

5.4.2 Use Case Description

ID	UC-01
Name	Register
Actor (s)	User
Description	Allow users to register an account for the mobile-based logistics system.
Typical Course of Events	**Step** / **Actor Action** / **System Response** 1 / User clicks the Register button in the Login Page. / 2 / User inputs all the information and submit. / 3 / / System inserts the new user information to the database and return success register message to user.
Alternate Courses	Step 3. If the user input missing or incorrect information, the register process will fail and ask user to register again.
Pre-conditions	N/A
Post-conditions	User can have an account with email and password to login the application.
Assumptions	User has no account before he/she register an account.

ID	UC-02
Name	Login
Actor (s)	User, Driver, Administrator
Description	Allow them to login and start to user the system.
Typical Course of Events	**Step** / **Actor Action** / **System Response** 1 / User inputs their email and password and click login button. / 2 / / System returns the Dashboard page.
Alternate Courses	Step 2. If there is empty input for either email or password, there will be error messages. Step 2. If the either email or password is incorrect, login will be fail and ask user to input again.
Pre-conditions	N/A
Post-conditions	N/A
Assumptions	User has an account for the logistics system.

<table>
<tr><td>ID</td><td>UC-03</td></tr>
<tr><td>Name</td><td>Check-in Items</td></tr>
<tr><td>Actor (s)</td><td>User</td></tr>
<tr><td>Description</td><td>Allow user to check-in items for delivery.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>User inputs the items information, sender information, receiver information, pickup location and destination location.</td><td></td></tr>
<tr><td>2</td><td></td><td>System inserts bill details to the database, bill status updates to “Pending”</td></tr>
<tr><td>3</td><td></td><td>System returns success message to user.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Step 2. If there is is missing required textfields, check-in will be fail and ask user to input again.
Step 2. If the input value is invalid or incorrect, check-in will be fail and ask user to input again.</td></tr>
<tr><td>Pre-conditions</td><td>N/A</td></tr>
<tr><td>Post-conditions</td><td>User can check-in items and view the order in bill history section.</td></tr>
<tr><td>Assumptions</td><td>User has an account for the logistics system and logged in.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-04</td></tr>
<tr><td>Name</td><td>Bill Tracking</td></tr>
<tr><td>Actor (s)</td><td>User</td></tr>
<tr><td>Description</td><td>Allow user live-track the ordered bill location.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>User clicks the Bill Tracking function in the sidemenu and input the bill number.</td><td></td></tr>
<tr><td>2</td><td></td><td>System gains the uid of the driver and the bill number from the database.</td></tr>
<tr><td>3</td><td></td><td>System shows the corresponding bill details and location.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Step 1. If the bill number is incorrect, no matching bill will be shown.</td></tr>
<tr><td>Pre-conditions</td><td>N/A</td></tr>
<tr><td>Post-conditions</td><td>N/A</td></tr>
<tr><td>Assumptions</td><td>Logged in account is a user account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-05</td></tr>
<tr><td>Name</td><td>Check Bill History</td></tr>
<tr><td>Actor (s)</td><td>User</td></tr>
<tr><td>Description</td><td>Allow user to check the past bill records.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>User clicks the Bill History button in the sidemenu.</td><td></td></tr>
<tr><td>2</td><td></td><td>System gains the uid of the user from the database.</td></tr>
<tr><td>3</td><td></td><td>System shows the list of past bill records to the user.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Step 3. If there is no records, no records message will be shown.</td></tr>
<tr><td>Pre-conditions</td><td>N/A</td></tr>
<tr><td>Post-conditions</td><td>N/A</td></tr>
<tr><td>Assumptions</td><td>Logged in account is a user account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-06</td></tr>
<tr><td>Name</td><td>View Profile</td></tr>
<tr><td>Actor (s)</td><td>User, Driver</td></tr>
<tr><td>Description</td><td>Allow user, driver to view their register information.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>User clicks the icon of their profile.</td><td></td></tr>
<tr><td>2</td><td></td><td>System gains the uid of user from the database.</td></tr>
<tr><td>3</td><td></td><td>System shows the register details of the user.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>N/A</td></tr>
<tr><td>Pre-conditions</td><td>N/A</td></tr>
<tr><td>Post-conditions</td><td>N/A</td></tr>
<tr><td>Assumptions</td><td>Logged in account is a user or driver account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-07</td></tr>
<tr><td>Name</td><td>Track Driver Location</td></tr>
<tr><td>Actor (s)</td><td>User, Administrator</td></tr>
<tr><td>Description</td><td>View the current location of the driver in the map engine.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>User clicks the track location button in the bill details page or admin clicks dispatch driver.</td><td></td></tr>
<tr><td>2</td><td></td><td>System gains the driver’s location from the database.</td></tr>
<tr><td>3</td><td></td><td>System passes the location coordinates to map engine and show a driver marker in the map engine.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Step 3. If the driver is offlined, it will not be displayed on the map.</td></tr>
<tr><td>Pre-conditions</td><td>User click the track location button in the bill details page or Admin clicks dispatch dirvers.</td></tr>
<tr><td>Post-conditions</td><td>User and Admin can track the driver current location in the map engine.</td></tr>
<tr><td>Assumptions</td><td>Logged in account is a user or admin account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-08</td></tr>
<tr><td>Name</td><td>Stock Management</td></tr>
<tr><td>Actor (s)</td><td>Administrator</td></tr>
<tr><td>Description</td><td>Allow administrator to view all bill records.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>Admin clicks the stock management button in admin interface.</td><td></td></tr>
<tr><td>2</td><td></td><td>System shows the bill details of the bill.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Step 2. If there is no records, no records message will be shown.</td></tr>
<tr><td>Pre-conditions</td><td>User click the stock management button in admin interface.</td></tr>
<tr><td>Post-conditions</td><td>N/A</td></tr>
<tr><td>Assumptions</td><td>Logged in account is an admin account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-09</td></tr>
<tr><td>Name</td><td>Manage Users/Drivers</td></tr>
<tr><td>Actor (s)</td><td>Administrator</td></tr>
<tr><td>Description</td><td>Allow administrator to manage the registered users and drivers.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>Admin clicks the manage users/drivers button in admin interface.</td><td></td></tr>
<tr><td>2</td><td></td><td>System shows the registered users/drivers.</td></tr>
<tr><td>3</td><td>Admin clicks call or send email to the users/drivers.</td><td></td></tr>
<tr><td>4</td><td></td><td>System directs the interface to phone’s dialing or email engine.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>N/A</td></tr>
<tr><td>Pre-conditions</td><td>User click the manage users/drivers button in admin interface.</td></tr>
<tr><td>Post-conditions</td><td>The user/driver is called or emailed after receiving the requests.</td></tr>
<tr><td>Assumptions</td><td>Logged in account is an admin account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-10</td></tr>
<tr><td>Name</td><td>Dispatch Driver</td></tr>
<tr><td>Actor (s)</td><td>Administrator</td></tr>
<tr><td>Description</td><td>Allow admin to dispatch driver.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>Admin clicks the dispatch driver button in the admin interface.</td><td></td></tr>
<tr><td>2</td><td></td><td>System retrieve the online drivers from the database and show their current location.</td></tr>
<tr><td>3</td><td>Admin selects the driver that want to get dispatched.</td><td></td></tr>
<tr><td></td><td></td><td>System return the pick-up jobs list or delivery jobs list according to the Admin selection.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>N/A</td></tr>
<tr><td>Pre-conditions</td><td>N/A</td></tr>
<tr><td>Post-conditions</td><td>Admin can assign the jobs to drivers according the jobs list.</td></tr>
<tr><td>Assumptions</td><td>Logged in account is an admin account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-11</td></tr>
<tr><td>Name</td><td>Smart Assignment</td></tr>
<tr><td>Actor (s)</td><td>Administrator</td></tr>
<tr><td>Description</td><td>Allow admin to assign jobs to the driver.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>User clicks Smart Assign button on the jobs list screen.</td><td></td></tr>
<tr><td>2</td><td></td><td>System calculates and sort the bills according to the designed algorthim.</td></tr>
<tr><td>3</td><td></td><td>System updates the bill status to “Assigned” and inserts the bill to the selected driver’s jobs list.</td></tr>
<tr><td>4</td><td></td><td>System shows success messages.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Step 2. If the status of the bill is already “Assigned”, “Processing” or “Delivered”, error messages will be shown.
Step 2. If the item’s weight exceeds the vehicle capacity, error messages will be shown.</td></tr>
<tr><td>Pre-conditions</td><td>Finished UC-10 click the smart assignment function</td></tr>
<tr><td>Post-conditions</td><td>The bill status is updated and jobs list is updated for the driver in the database.</td></tr>
<tr><td>Assumptions</td><td>Logged in account is admin account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-12</td></tr>
<tr><td>Name</td><td>Check Schedule</td></tr>
<tr><td>Actor (s)</td><td>Administrator</td></tr>
<tr><td>Description</td><td>Allow admin to check the driver working schedule.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>Admin selects the check schedule function in the driver list.</td><td></td></tr>
<tr><td>2</td><td></td><td>System gets the working schedule by uid of the driver and return the list of job.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Finished UC-09</td></tr>
<tr><td>Pre-conditions</td><td>Administrator has assigned job schedule to driver.</td></tr>
<tr><td>Post-conditions</td><td>N/A</td></tr>
<tr><td>Assumptions</td><td>Logged in account is an admin account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-13</td></tr>
<tr><td>Name</td><td>Process Items</td></tr>
<tr><td>Actor (s)</td><td>Driver</td></tr>
<tr><td>Description</td><td>Allow driver to confirm successful delivery.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>Driver clicks the process item function in the app.</td><td></td></tr>
<tr><td>2</td><td></td><td>System shows the interface for scanning the bill QR Code.</td></tr>
<tr><td>3</td><td>Driver scans bill QR Code from the user’s phone.</td><td></td></tr>
<tr><td>4</td><td></td><td>System shows the corresponding bill number.</td></tr>
<tr><td>5</td><td>Driver confirms to deliver/pickup the item.</td><td></td></tr>
<tr><td>6</td><td>Driver finishs the process of signature for the proof of delivery.</td><td></td></tr>
<tr><td>7</td><td></td><td>System updates the bill status to “Delivered” for delivery jobs or “In Stock” for pickup jobs and inserts the signature into the database.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Step 3. If the bill QR Code is invalid, error message will show up.</td></tr>
<tr><td>Pre-conditions</td><td>Driver click the process item function.</td></tr>
<tr><td>Post-conditions</td><td>The bill status is updated in the database and the signature will be uploaded as reference.</td></tr>
<tr><td>Assumptions</td><td>Logged in account is driver account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-14</td></tr>
<tr><td>Name</td><td>Delivery Optimal Routing</td></tr>
<tr><td>Actor (s)</td><td>Driver</td></tr>
<tr><td>Description</td><td>Allow driver to get the optimal route for travelling all the destinations.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>Driver clicks the optimal routing function.</td><td></td></tr>
<tr><td>2</td><td></td><td>System retrieves the jobs list from database and shows all the destinations on the map</td></tr>
<tr><td>3</td><td>Driver clicks start button for optimal routing.</td><td></td></tr>
<tr><td>4</td><td></td><td>System gets the driver current location from GPS and calculates the optimal route.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Step 1. The map will not be loaded up if the driver turned off GPS.
Step 2. No destinations will be shown if no jobs have been assigned.</td></tr>
<tr><td>Pre-conditions</td><td>Driver has been assigned with jobs.</td></tr>
<tr><td>Post-conditions</td><td>The routes will show up on the screen for driver to follow.</td></tr>
<tr><td>Assumptions</td><td>Logged in account is driver account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-15</td></tr>
<tr><td>Name</td><td>Register Items</td></tr>
<tr><td>Actor (s)</td><td>Driver</td></tr>
<tr><td>Description</td><td>Allow driver to register the items which are ready to pickup or deliver.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>Driver clicks the register item function in the app</td><td></td></tr>
<tr><td>2</td><td></td><td>System shows the interface for input bill details</td></tr>
<tr><td>3</td><td>Driver scans QR Code or input the bill number</td><td></td></tr>
<tr><td>4</td><td></td><td>System shows the corresponding bill information.</td></tr>
<tr><td>5</td><td>User confirms to register the item.</td><td></td></tr>
<tr><td>6</td><td></td><td>System updates the bill status to “Processing” for user to track.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Step 3: Error messages will show if the bill number or QR Code is invalid.</td></tr>
<tr><td>Pre-conditions</td><td>Admin finished UC-11 and driver click the delivery item function.</td></tr>
<tr><td>Post-conditions</td><td>The bill status is updated to “Processing” and user can now in the database</td></tr>
<tr><td>Assumptions</td><td>Logged in account is driver account.</td></tr>
</table>

<table>
<tr><td>ID</td><td>UC-16</td></tr>
<tr><td>Name</td><td>Announcement</td></tr>
<tr><td>Actor (s)</td><td>Administrator</td></tr>
<tr><td>Description</td><td>Allow admin to make accouncement.</td></tr>
<tr><td>Typical Course of Events</td><td>
<table>
<tr><th>Step</th><th>Actor Action</th><th>System Response</th></tr>
<tr><td>1</td><td>Admin inputs the accouncement on the dashboard.</td><td></td></tr>
<tr><td>2</td><td></td><td>System inserts accouncement to the database and shows success message.</td></tr>
</table>
</td></tr>
<tr><td>Alternate Courses</td><td>Step 1. Admin can delete accouncement and systems will remove the selected accouncement.</td></tr>
<tr><td>Pre-conditions</td><td>Admin inputs the accouncement.</td></tr>
<tr><td>Post-conditions</td><td>The accouncement will be seen on user, driver and admin dashboard.</td></tr>
<tr><td>Assumptions</td><td>Logged in account is admin account.</td></tr>
</table>

5.5 Entity Relationship Diagram

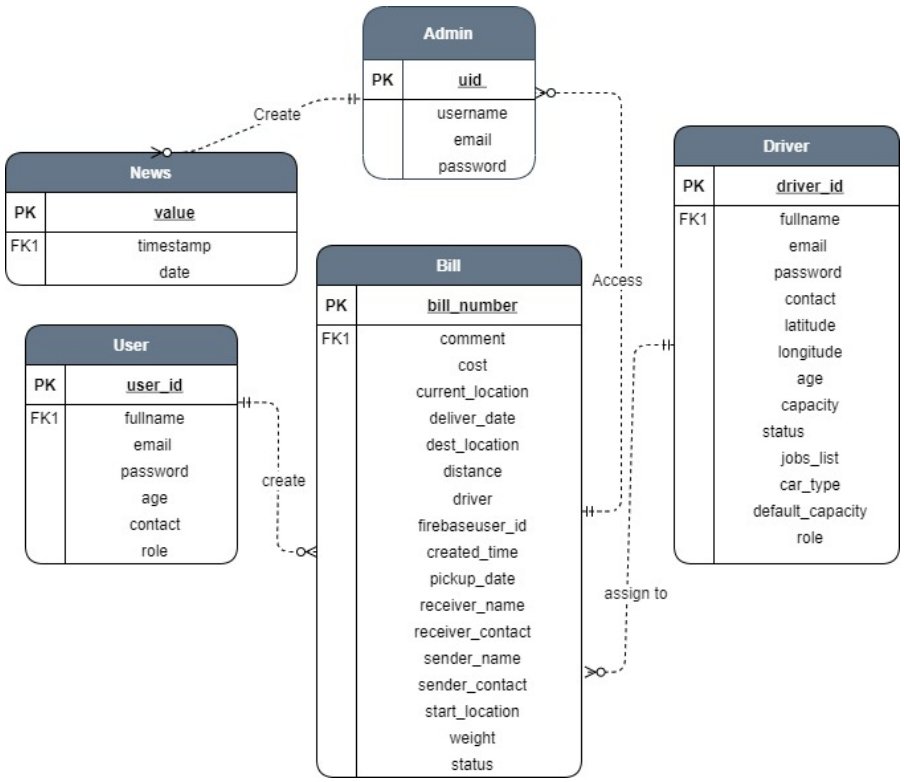


Figure 5.5 Entity Relationship Diagram

5.6 Class Diagram

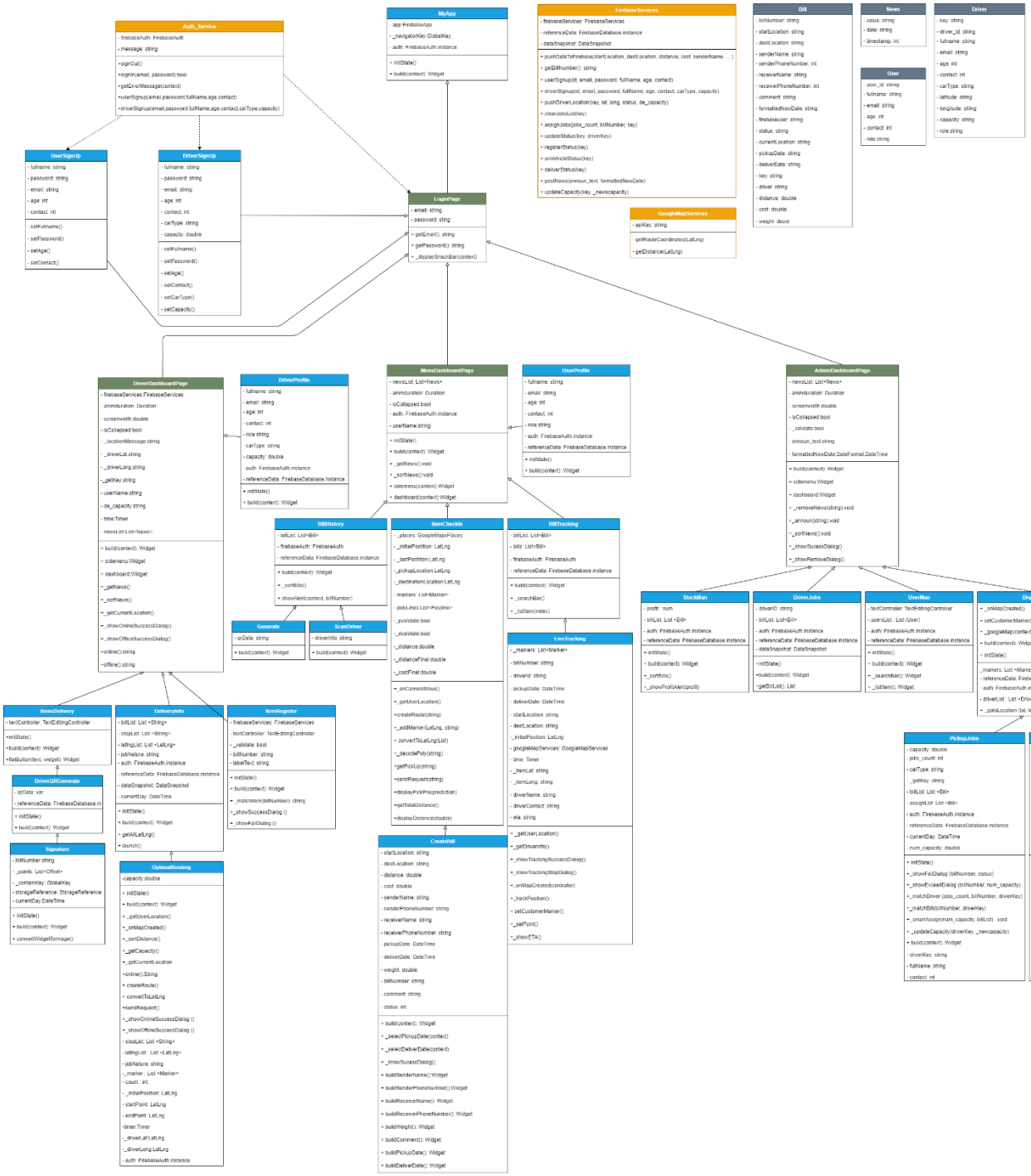


Figure 5.6 Class Diagram

5.6.1 Modelling Object Class

There are 4 modelling object classes indicated in grey color (See Figure 5.6).

Class	Description
News	Object class for admin announcements.
User	Object class for users.
Driver	Object class for drivers.
Bills	Object class for created bills.

5.6.2 Requests Controller

There are 3 requests controller classes indicated in orange color (See Figure 5.6).

Class	Description
FirebaseServices	For handling firebase requests and communicate with database.
GoogleMapServices	For handling mapping requests with Google API.
Auth_Services	For handling authentication requests such as sign up and sign in.

5.6.3 Application Screens

There are 4 application screen classes to handle access control with different login identities which indicated in green color (See Figure 5.6).

Class	Description
LoginPage	Application screen for login page.
MenuDashboardPage	Application screen for user interface.
DriverDashboardPage	Application screen for driver interface.
AdminDashboardPage	Application screen for admin interface.

5.6.4 Functionality Class

The rest of classes are functionality classes which handle different tasks and functions for the applications indicated in blue color. (See Figure 5.6). The following are some of the major classes.

Class	Description
MyApp	Act as main.dart to handle the application launch.
DriverSignUp	Handle driver sign up function.

UserSignUp	Handle user sign up function.
DriverProfile	Retrieve driver information data for profile.
UserProfile	Retrieve user information data for profile.
ItemsDelivery	Retrieve driver jobs list.
Signature	Handle proof of delivery function.
OptimalRouting	Handle optimal routing function.
ItemRegister	Handle driver item registration.
BillHistory	Retrieve user billing history.
ItemCheckIn	Handle user item online check-in.
CreateBill	For user to fill in billing information.
LiveTracking	Handle live tracking function.
StockMan	Retrieve and for admin to handle bill history.
DriverJobs	Retrieve and for admin to handle driver schedule.
UserMan	Retrieve and for admin to handle users.
Dispatch	Handle smart assignment function.

5.7 Workflow Diagram

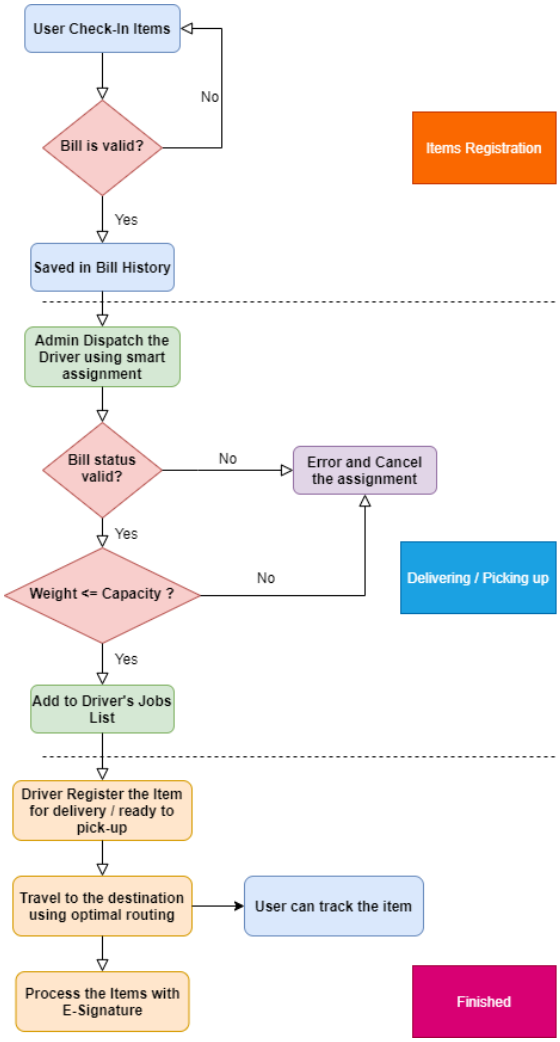


Figure 5.7 Workflow Diagram

Figure 5.7 illustrates the flow of the logistics services operation. Users need to create the bill first then the admin is responsible to assign the jobs to the driver. Then the driver should register the assigned items for delivery or pick-up. Then arrive at the destination for processing the bills.

6. Methodology and Implementation

6.1 Database Structure

Firebase realtime database will be implemented in this project for storing the data and user authentication information. There are currently three entities, ‘Bills’, ‘News’ and ‘Users’. For ‘Bills’ entities, each key value will have 18 attributes, storing the information of the bill. (See Figure 6.1)

Attributes	Description
String billNumber	Generated bill number
String comment	Additional bill information
double cost	Delivery fees of the bill
String currentLocation	Bill current location in form of latitude and longitude
DateTime deliverDate	Bill deliver date
String destLocation	Bill destination location
double distance	Distance between destination and pick up location
String driver	DiverID
String firebaseuser	Bill creator UserID
String FormattedNowDate	Bill created date and time
DateTime pickupDate	Bill pick up date
String receiverName	Receiver name
String receiverPhoneNumber	Receiver phone number
String senderName	Sender name
String senderPhoneNumber	Sender phone number
String startLocation	Bill pick up location
String status	Bill status, Pending/Assigned/Delivered
double weight	Bill wight in kg

Figure 6.1 Firebase Bills attributes

For 'Users' entities, 6 attributes will be assigned for users and 13 attributes will be assigned for drivers, storing the registered user profiles.

- For User (See Figure 6.2)

Attributes	Description
int age	User age
String contact	User contact
String email	Registered email
String fullName	User full name
String id	User firebase authentication id
String role	User role, User/Driver

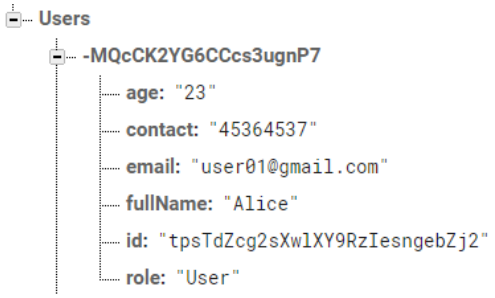


Figure 6.2 Firebase User attributes

➢ For Driver (See Figure 6.3)

Attributes	Description
int age	Driver age
double capacity	Vehicle current capacity in tons
String carType	Vehicle type
String contact	Driver contact
double default_Capacity	Vehicle default capacity in tons
String email	Registered email
String fullName	Driver full name
String id	Driver firebase authentication id
List jobs	Assigned jobs in billNumber
double latitude	Driver latitude
double longitude	Driver longitude
String role	User role, User/Driver
String status	Driver status

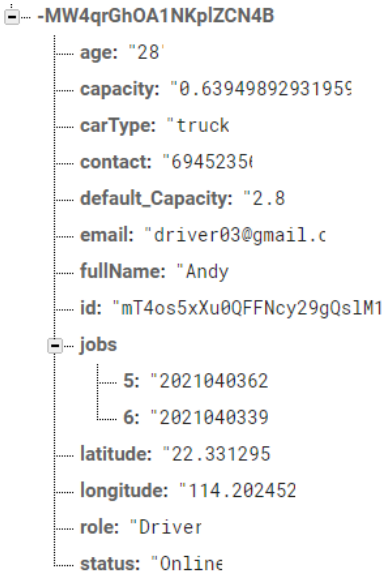


Figure 6.3 Firebase Driver attributes

➢ For News (See Figure 6.4)

For ‘News’ entities, 3 attributes will be assigned for storing the admin announcements.

Attributes	Description
DateTime date	Created date and time
int timestamp	Timestamp for sorting the news
String value	Announcement information

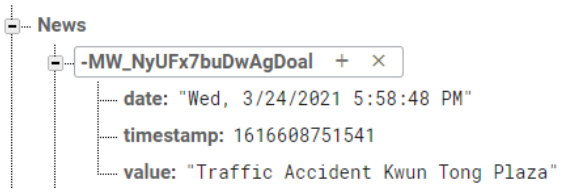


Figure 6.4 Firebase News attributes

6.2 User Authentication

User authentication will be done using firebase authentication services. Functions for both users and drivers to sign up have been created. New user needs to fill in the TextForm and pass the inputted String values to Firebase to get the authentication to sign up an account. The user information will then store into the database for the user profile. Whenever a user signs in, the application will retrieve the authentication list and determine if the user is verified to sign in and return the user, driver or admin interface according the user role stored in database. (See Figure 6.5).

Figure 6.5 Firebase authentication email list

6.3 Items Check-in

Only user interface provides the items check-in service. User needs to input the location for the item pick-up and delivery. Auto-complete function for the location is implemented to let user input location easier. Then the route between pick up and delivery location will be generated using the latitude and longitude with Google Distance Matrix API. The route which contains a list of latitude and longitude points generated by the API will be decoded using decoding function and polyline, which is the route will be drawn using the decoded list. Delivery fees will be also calculated by making use of the distance value. After that, users are required to fill in a TextForm to register the bill with detailed information which will be fetched to the database. Bill number which is the combination of the created day and a random number is generated by a function once the bill is created.

Pseudo code for generate bill number

```
String _getBillNumber() {
     final f = new DateFormat('yyyyMMdd').format(DateTime.now()).toString();
     Random rnd;
     int min = 1;
     int max = 1000;
     rnd = new Random();
     int r = min + rnd.nextInt(max - min);
     String billNumber = '$f$r';
     return billNumber;
   }
```

6.4 Bill History and Stock Management

Bill History and Stock Management which are the services for user and admin respectively. Functions have been designed for the application to retrieve the data from the database to display the bill on the screen. UserID will be used to determine the identity of the user and only bill history for that particular owner can see his/her own bills.

For admin stock management, admin can see all the bills that have been created via the application. There is also a function for admin to delete bills.

The method for retrieving the database bill data is to create a list type object and store all the bills data into the list once the data is retrieved from the database. Then the list will be shown on the screen using Flutter Card Widget.

6.5 Live Tracking Mechanism (with ETA)

Functions have been designed for the live tracking function. The location of the driver will be updated to the database periodically after the starting of the delivery services. Only bills with "Processing" status can be live tracked by the users. User can enter the bill number to retrieve the coordinates of the driver which will be also updated periodically on the user application to perform live update tracking services. The estimated time of arrival is calculated on a distance basis. Switch cases have been designed, when the coordinate is getting closer to the destination, the ETA will be also updated automatically to provide an accurate arrival time for the delivery.

6.6 Optimal Route Planning Algorithm

Algorithm have been designed to perform the optimal routing function for the drivers. The jobs list that assigned to the driver will contain the destination name and also their coordinates. The algorithm will first retrieve the current location of the driver, then calculate the next nearest destination, then the target will be updated and looping will be performed to calculate the followed nearest destination until all jobs are calculated. The following is an example:

<table>
<tr><td>Driver ID</td><td>Driver03</td></tr>
<tr><td>Destinations</td><td>D1. Tiu Keng Leng Station, Tiu Keng Wan, Hong Kong - 22.3039051/114.2522843
D2. IKEA Causeway Bay Store, Gloucester Road, Causeway Bay, Hong Kong - 22.2811761/114.1864329
D3. One Island East, Westlands Road, Quarry Bay, Hong Kong - 22.2862479/114.2135458
D4. Pacific Place, Queensway, Admiralty, Hong Kong - 22.277152/114.1648871</td></tr>
<tr><td>Driver Coordinates</td><td>22.3314414/ 114.2025193</td></tr>
<tr><td>Calculation</td><td>Step 1 :<table><tr><td>Current location to D1</td><td>12.2km</td></tr><tr><td>Current location to D2</td><td>10.9km</td></tr><tr><td>Current location to D3</td><td>14.0km</td></tr><tr><td>Current location to D4</td><td>9.7km</td></tr></table>Step 2 :<table><tr><td>D4 to D1</td><td>15.8km</td></tr><tr><td>D4 to D2</td><td>3.1km</td></tr><tr><td>D4 to D3</td><td>7.6km</td></tr></table>Step 3 :<table><tr><td>D2 to D1</td><td>13.6km</td></tr><tr><td>D2 to D3</td><td>5.4km</td></tr></table>Step 4 :<table><tr><td>D3 to D1</td><td>11.9km</td></tr></table></td></tr>
<tr><td>Optimal Path</td><td>Current Location -> D4 -> D2 -> D3 -> D1</td></tr>
</table>

The above example illustrates that the algorithm first gets the user location, then calculate the next nearest destination for delivery, then compare the other destinations and lastly build the optimal route for travelling all the destinations. The optimal path will vary upon the change of the driver location (i.e. if the driver is near D1 in the first place, the optimal path will start at D1).

Pseudo code for optimal routing

```
void _optimalRouting(List jobsList){
LatLng startPoint = _getUserLocation().latlng;
int _compareDistance = googleMapServices.getDistance(startPoint, jobsList[0].latlng);
for (int i = 0 ; I < jobsList.length ; i++){
int _tempDistance = googleMapServices.getDistance(startPoint, jobsList[i].latlng);
        if (_tempDistance <= _compareDistance){
                setState((){
                        endpoint = jobsList[i].latlng;
                        drawPolyline();
                        jobsList.remove[i];
                        });
                }
        }
}
```

6.7 Driver Location

Driver location will be uploaded to the database which consists of the latitude and longitude of that driver. When the driver presses the button to upload the location, the system will determine that driver is online and ready to assign jobs. The latitude and longitude positions will be shown on the map with the integration of Flutter Geolocator and Google Map APIs. When driver wants to get offline, the latitude and longitude value will set to null to avoid exposing the driver location on map.

6.8 Driver Dispatch

Driver dispatch will be done within the admin interface. All online drivers' latitude and longitude value will be retrieved and display on the map. Meanwhile, there will be a function to retrieve the driver information from the database and show on the button of the screen. Admin can select which driver for today pick-up or delivery jobs. After selecting the driver, there is a sorting function to sort the pick-up and delivery jobs according to the weights and only bills of today will be available for dispatch, which reduces the complexity and boosts the dispatching process. When assigning the bill to the selected driver, the bill number will be added to the driver's jobs list in the database for reference and optimal routing.

6.9 Weight Sorting and Smart Jobs Assignment Algorithm

Algorithm is designed for smart jobs assignment on admin application. It bases on the modified First-fit decreasing algorithm for the Bin Packing Problem to fit in the cases of this project. The algorithm will first check the status of the bill, only "Pending" and "In Stock" bills can be assigned. Then it will sort the bills in ascending order according to the weights by bubble sort algorithm. Then it will calculate the difference between the capacity and the weights, only bill that can fit in the driver's vehicle can be assigned, otherwise, error messages will be displayed.

Here is an example:

Driver ID	Driver03 (Capacity 5.5 tons = 5500kg)
Total Jobs	B1. 482.4kg (Status: Pending) B2. 9999.9kg (Status: Pending) B3. 28.6kg (Status: Assigned) B4. 76.9kg (Status: Pending)
Calculation	1. Sorting the Bill: B3, B4, B1, B2 2. Check the status: B3 has assigned - **FAIL** 3. Bill List available: B4, B1, B2 4. Assign B4 (Capacity = 5500-76.9 = 5423.1 kg) - **SUCCESS** 5. Assign B1 (Capacity = 5423.1-482.4 = 4940.7kg) - **SUCCESS** 6. Assign B2 (Capacity = 4940.7-9999.9 = -5059.2kg) - **FAIL**
Jobs Assigned	B4, B1
Jobs Fail to Assign	B3, B2

Pseudo code for smart jobs assignment

```
void _smartAssign(double num_capacity, List billList) {
for (var i = 0; i < billList.length; i++) {
        billList.sort((a, b) => a.weight.compareTo(b.weight));
}
for (var i = 0; i < billList.length; i++) {
if (num_capacity - billList[i].weight > 0 && billList[i].status == 'In Stock' || 'Pending') {
          _assignDriver();
          _updateCapacity();
      }else{
        _showFailDialog();
      }else if (num_capacity - billList[i].weight < 0){
        _showExceedDialog();
       }
    }
}
```

7. Project Results

7.1 Login Page and Registration

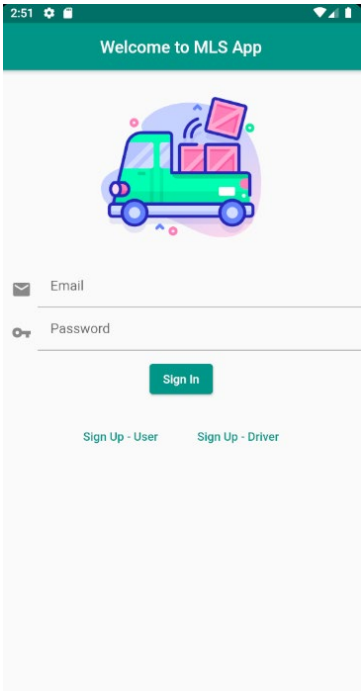


Figure 7.1.1 Login Page

Figure 7.1.1 is the login page for all users. Registered users are required to input their email and password to login the system. User, Driver or Admin interface will display according to the user role once the user logged in. For new user with no account, there will be sign up button for user and driver with can redirect to the registration page. Meanwhile, the admin account will not be open for registration.

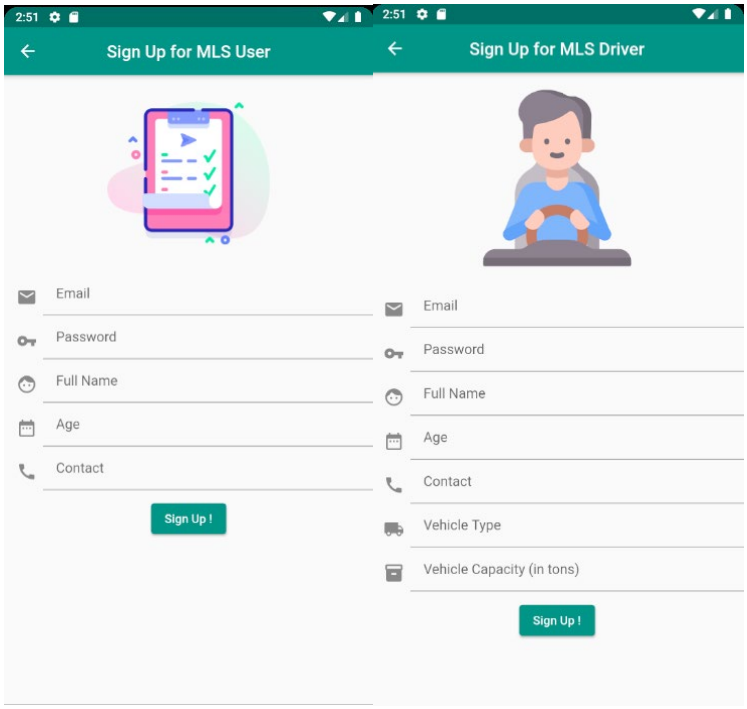


Figure 7.1.2 Sign up Page

Figure 7.1.2 shows the signup page for the regular user and driver. Information is required to be filled in to sign up an account. There will be an exception thrown when the inputted data is invalid or empty. Once the sign up is successful, the user profile will be stored in the database and will redirect to the dashboard interface.

7.2 User Interface and Functionality

7.2.1 User Dashboard

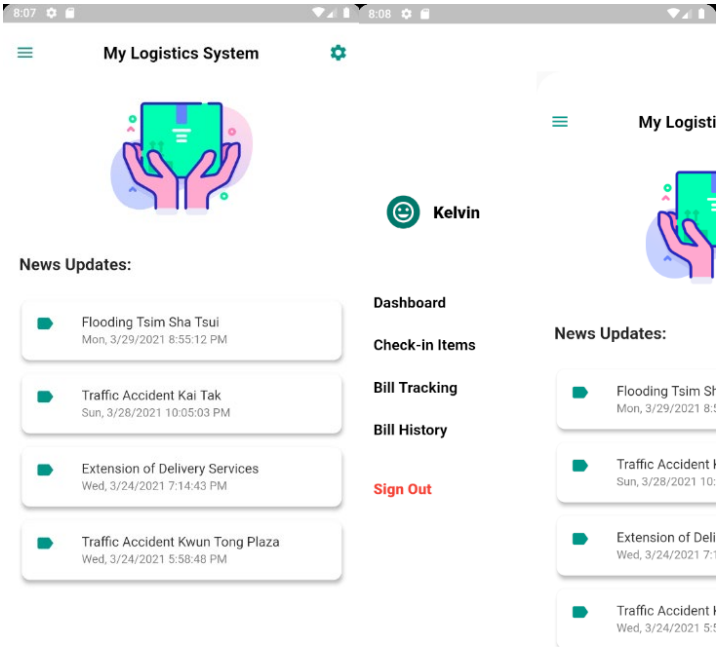


Figure 7.2.1 User Dashboard Interface

Figure 7.2.1 is the dashboard for user accounts. Admins announcements will be display on the screen. Pressing the icon on the top left, the side menu will be shown for users to access different functions, includes check-in items, bill tracking, bill history and sign out.

7.2.2 Items Check-In

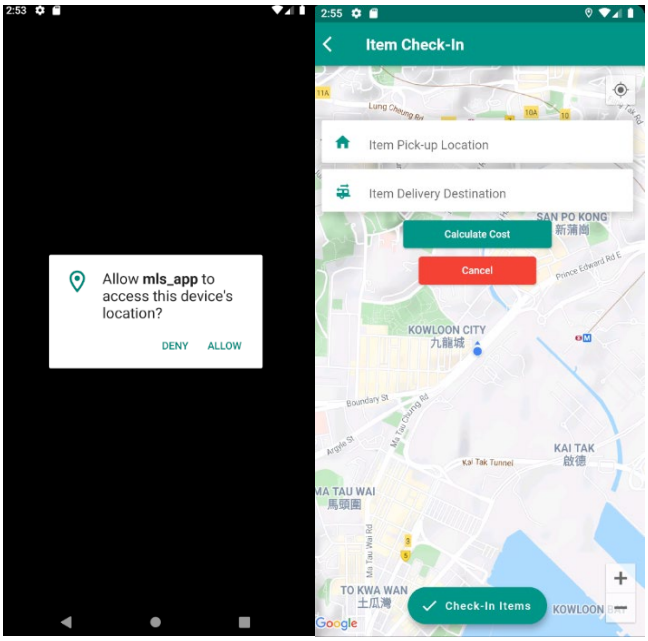


Figure 7.2.2 Items Check-in Page

Figure 7.2.2 is the items check-in page for the user. Users are required to allow the application to access the device's location before check-in the delivery items. After that, a map will be shown on the screen with the initial user location. There are two input text fields for pick-up location and delivery destination. When tapping the text field, region-based location auto-complete function will be called and data will be fetched from Google Places API to help users to input an accurate location (See Figure 7.2.3).

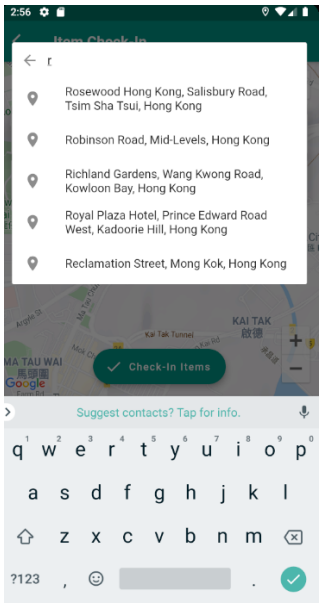


Figure 7.2.3 Location Autocomplete

After filling in the pick-up and delivery destination locations, the markers will be shown on the map for indicating the inputted location. A route will also be generated by designed functions to show the fastest way to travel between two points. 'Calculate Cost' shows the distance and delivery fees for this bill. Delivery fees will be a fixed cost set by admin multiply the distance. Information will be shown on the bottom of the screen. User can also press 'Cancel' to cancel this check-in (See Figure 7.2.4). If everything is ready, users can press 'Check-In Items' to create the bill. It will redirect to the Create Bill Page.

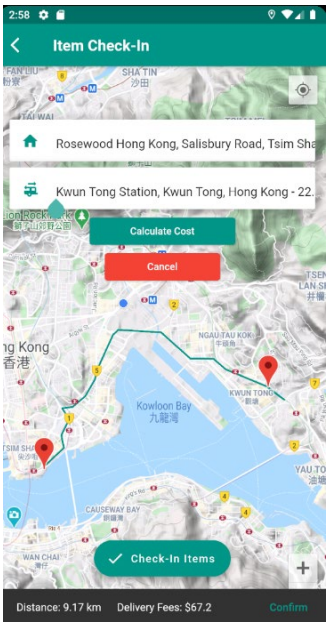


Figure 7.2.4 Distance and Delivery Fees

7.2.3 Create Bill

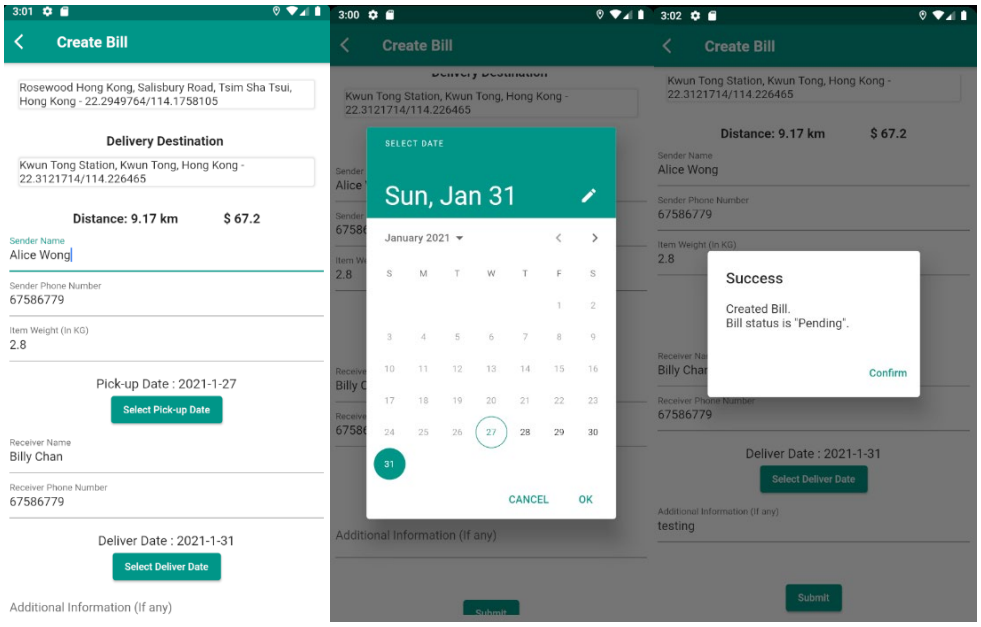


Figure 7.2.5 Create Bill Page

Figure 7.2.5 shows the Create Bill Page, further information is required to create a bill. Include sender name, sender phone number, item weight, pick-up date, receiver name, receiver phone, delivery date and additional information. Datepicker with designed restriction will be provided to input the date, in which the pick-up date must not before the current day and delivery date must be later than the pick-up date. There will be an exception thrown when the inputted data is

invalid or empty. Then users can press the ‘Submit’ button to submit the bill which stores in the database.

7.2.4 Bill History

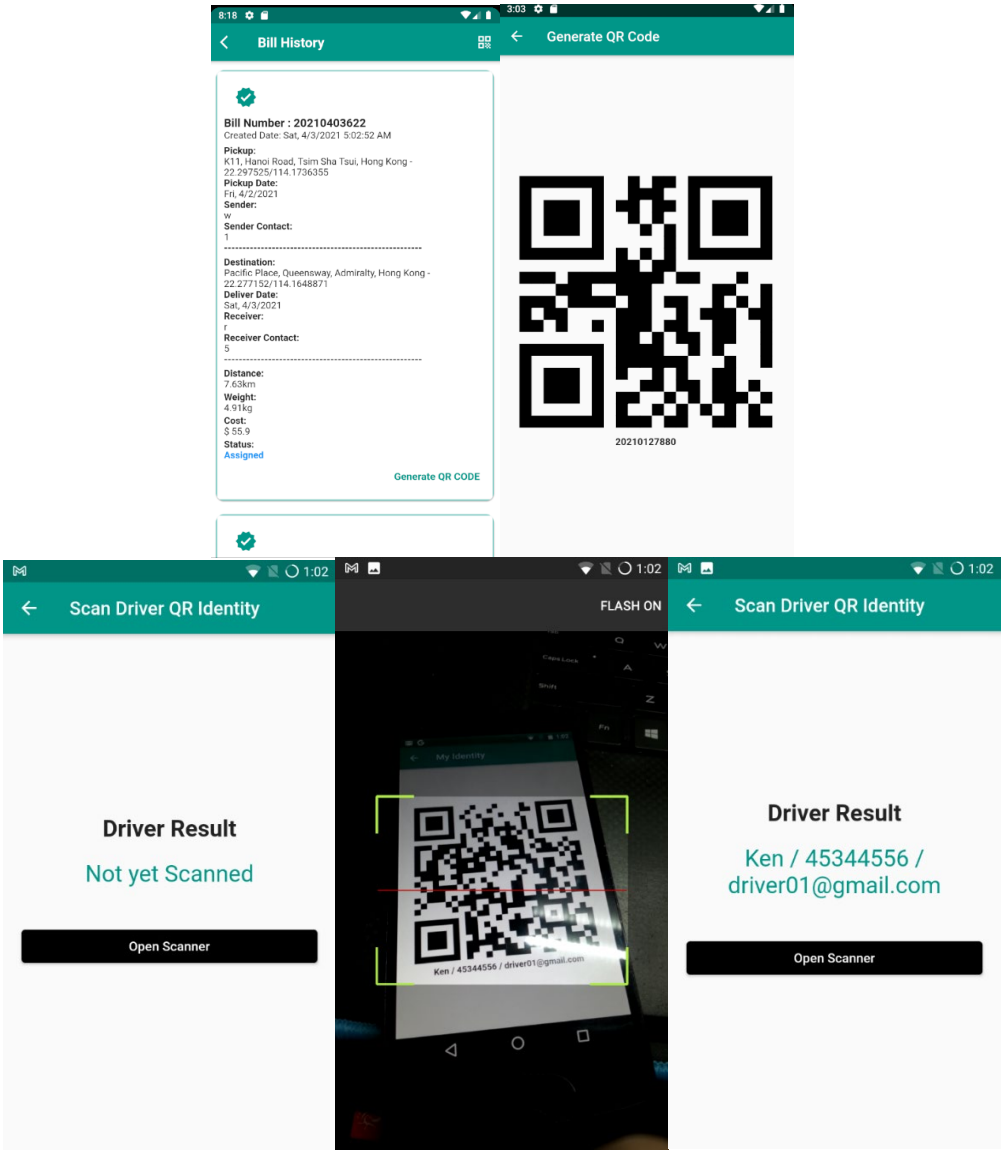


Figure 7.2.6 Bill History and Driver QR Code Scanning

Figure 7.2.6 shows the Bill History page for the user in which the bills that the users created will be displayed on the screen. The information will be listed out clearly and there is a ‘Generate QR CODE’ button for generating the unique QR Code for each bill according to the bill number. The QR Code icon on the top right corner is for users to scan the driver identity QR Code.

7.2.5 Bill Tracking

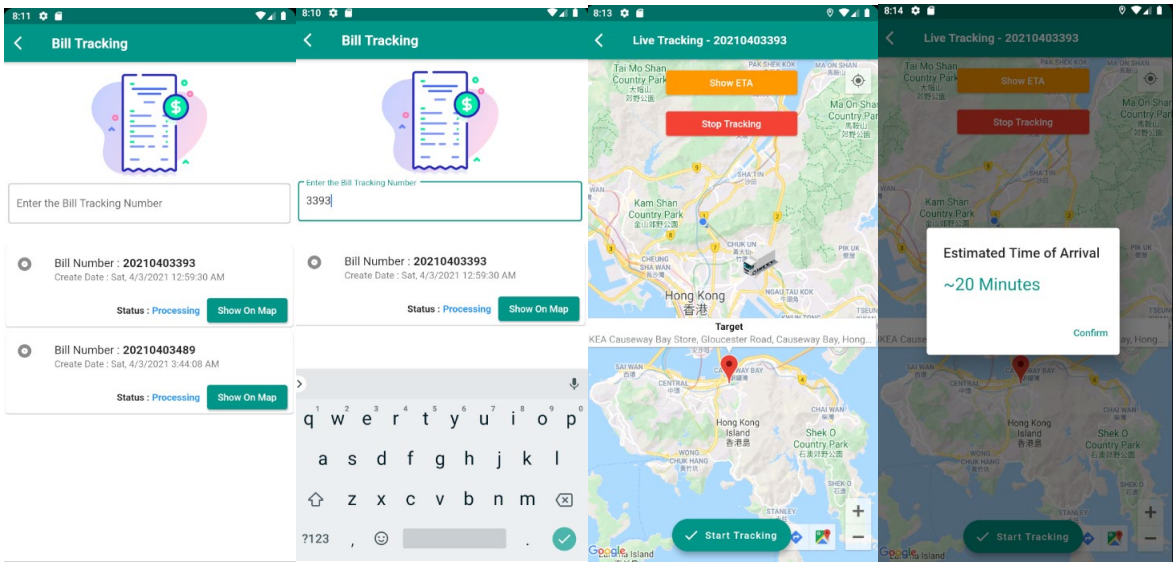


Figure 7.2.7 Bill Tracking

Figure 7.2.7 shows the Bill Tracking page. User can input the bill number to retrieve the bill that in the status of "Processing". The 'Show On Map' button brings the screen to the map which will track the driver's live location (the truck icon) after clicking the 'Start Tracking' button. The map also indicates the target destination and the ETA which will automatically be updated when the driver approaching the destination. "Stop Tracking" will stop the tracking function for user to reduce network data usages.

7.2.6 User Profile

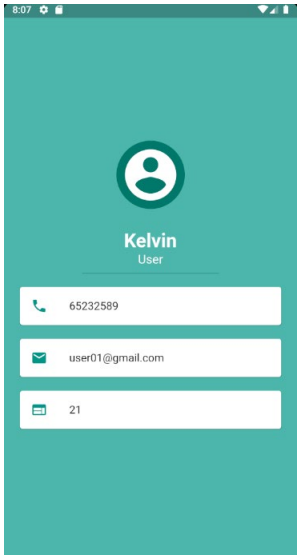


Figure 7.2.8 User Profile

Figure 7.2.8 shows the User Profile where it indicates registration details of current login user.

7.3 Driver Interface and Functionality

7.3.1 Driver Dashboard

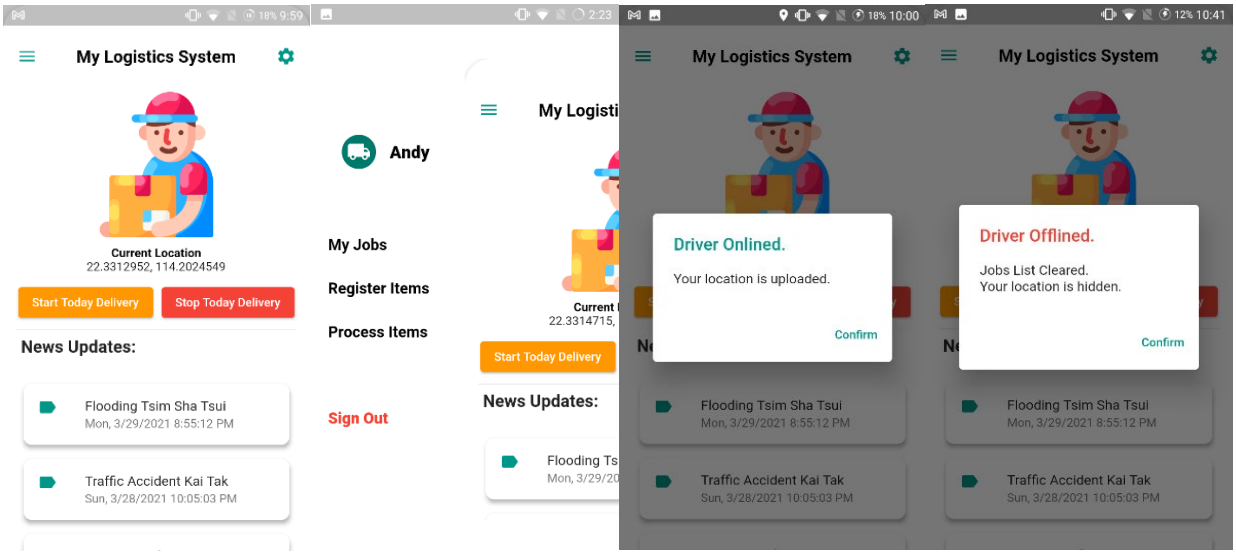


Figure 7.3.1 Driver Dashboard Interface

Figure 7.3.1 shows the Driver Dashboard where driver can see his current location and the admin announcements. The driver has to press 'Start Today Delivery' to fetch the location latitude and longitude to the database to get online. After finishing all the assigned jobs, driver can press 'Stop Today Delivery' to hide his location and clear the jobs list to get offline.

7.3.2 Assigned Jobs List

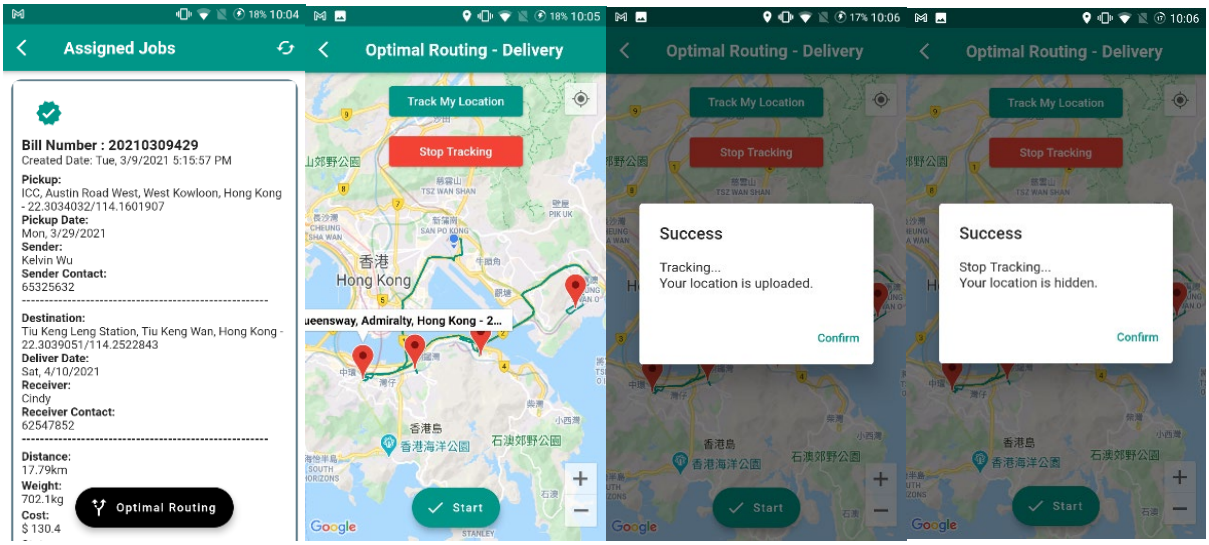


Figure 7.3.2 Assigned Jobs List and Optimal Routing

Figure 7.3.2 shows the assigned jobs list for the driver. By pressing 'Optimal Routing' button, the optimal routing function will be called where all the destaintion will be shown on the map.

By clicking ‘Start’ button, the optimal delivery route will be displayed for driver to follow. ‘Track My Location’ and ‘Stop Tracking’ will let the driver to upload and hide his current location.

7.3.3 Register Items

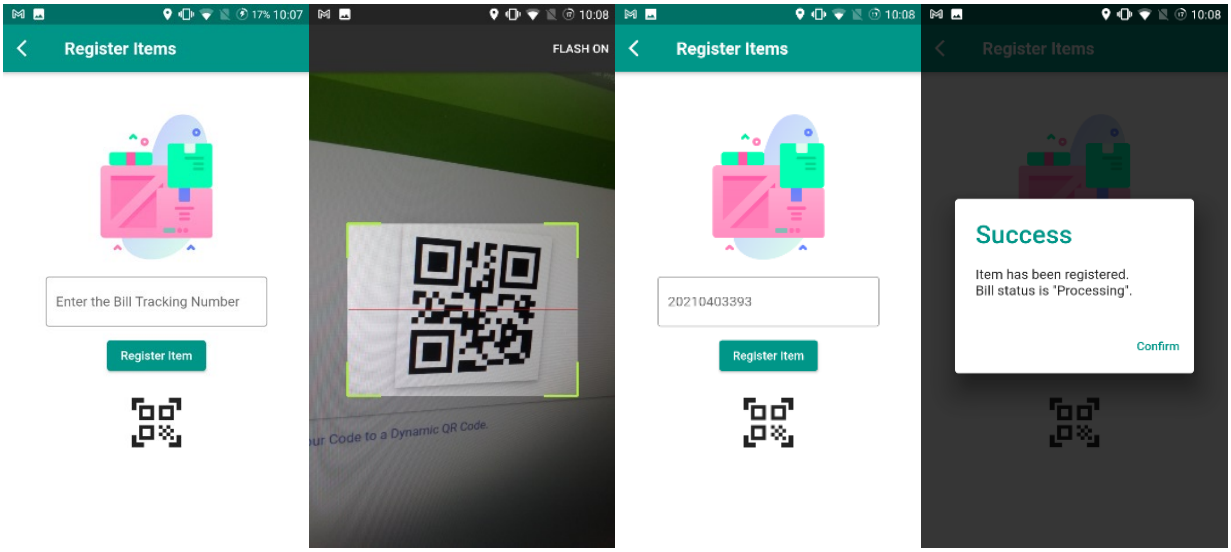


Figure 7.3.3 Items Registration

Figure 7.3.3 shows the items registration page. Driver can either input the bill number or scan the QR Code to register the items. This function is for drivers to register the items when they are loaded on the vehicle or ready to pick up. Bill status will update to “Processing” for users to track its location.

7.3.4 Process Items

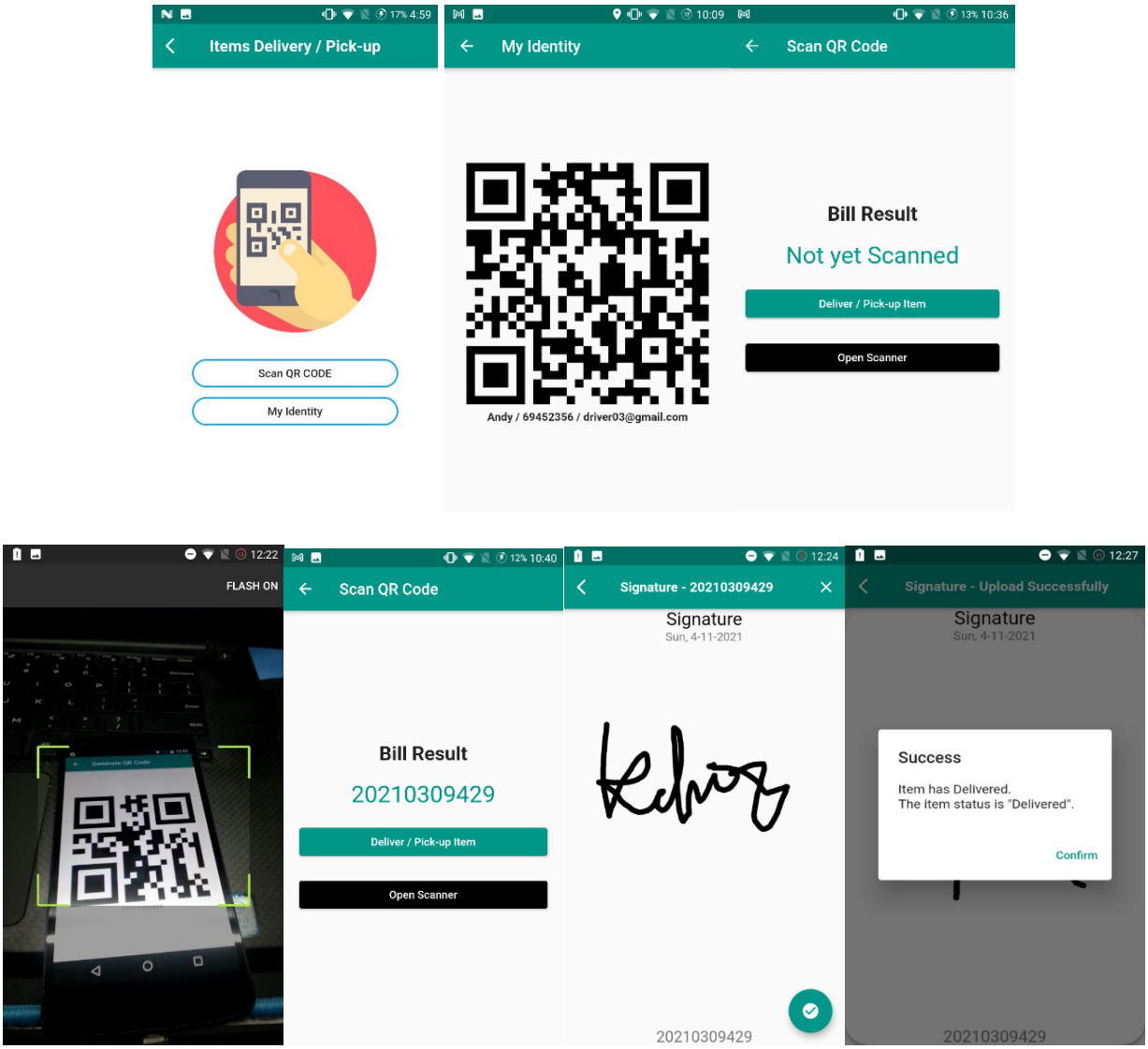


Figure 7.3.4 Process Items

Figure 7.3.4 shows the Process Items Page. Driver can show the QR Code identity for users to confirm the driver identity and secure the process. Then driver can scan the QR Code from the user application ‘Bill History’ page to retrieve the bill number. E-Signature will be provided on screen to perform proof of delivery. Then the item status will be updated to “In Stock” for pick-up jobs or “Delivered” for delivery jobs.

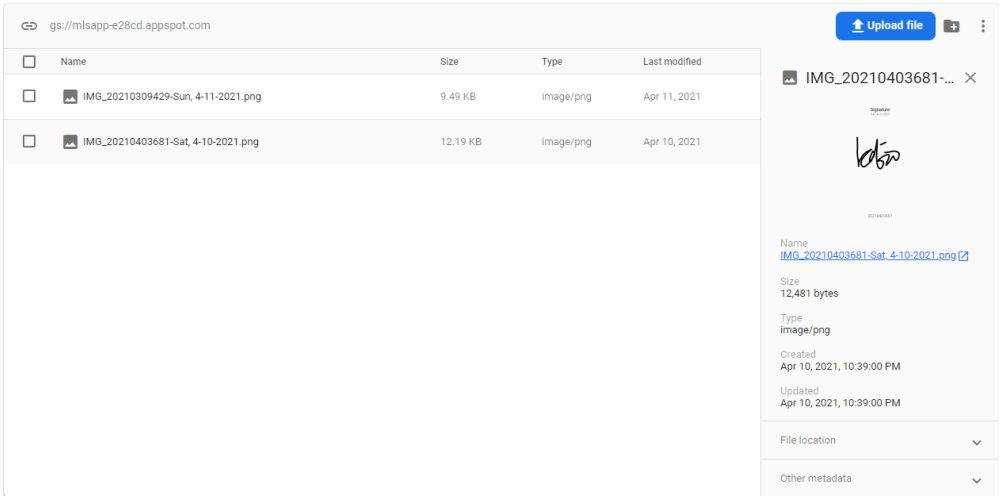


Figure 7.3.5 Signature Storage

After getting the proof of delivery, the E-Signature will be stored as .png in the Firebase for references. (See Figure 7.3.5).

7.3.5 Driver Profile

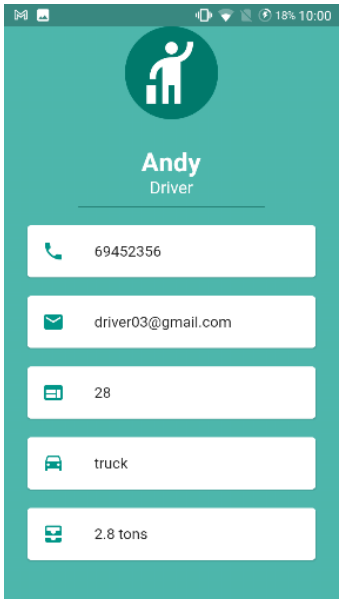


Figure 7.3.5 Driver Profile

Figure 7.3.5 shows the User Profile where it indicates registration details of current login user.

7.4 Admin Interface and Functionality

7.4.1 Admin Dashboard

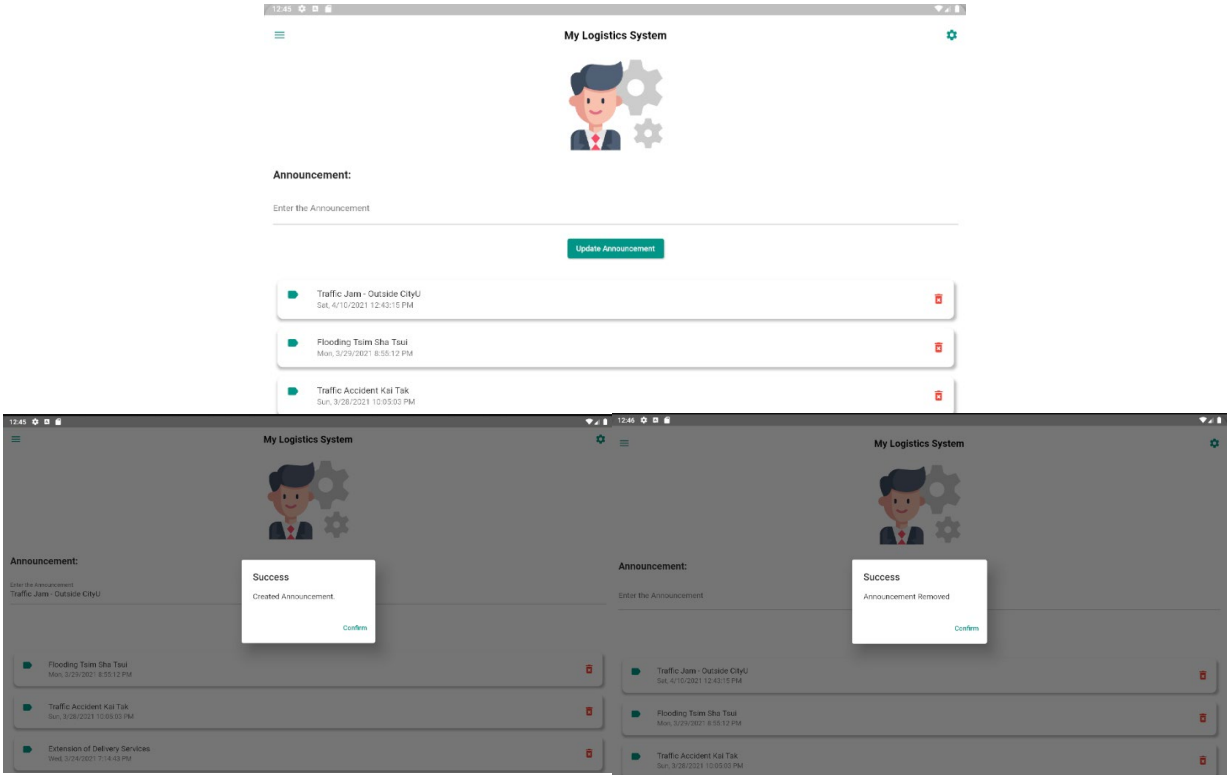


Figure 7.4.1 Admin Dashboard Interface

Admin interface is recommended to use tablets to access. Figure 7.4.1 shows the Admin Dashboard. Admin can make and remove announcements on the dashboard page.

7.4.2 Driver Dispatch

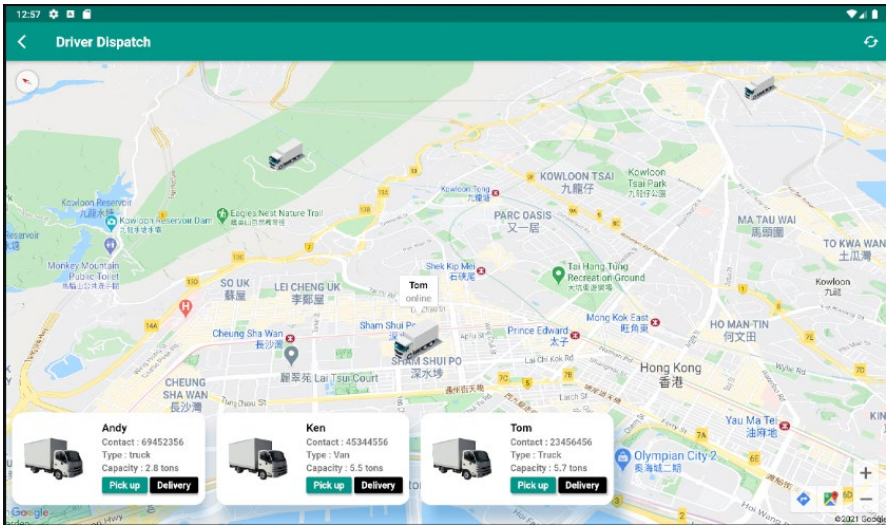


Figure 7.4.2 Driver Dispatch Interface

Figure 7.4.2 shows the Driver Dispatch Interface. The online driver will be shown on the screen with brief information. When pressing the widget of the driver on the bottom screen, the map

will animate to the corresponding driver. There are ‘Pick up’ and ‘Delivery’ button on the widget of the driver which will direct to the Assign Jobs Page for pick-up and delivery for the selected driver accordingly.

After pressing the pickup button, it will direct to the Assign Jobs Page (See Figure 7.4.4) which will only retrieve the Bill that the pick-up date is the current today. On each bill, there is a button ‘Assign’ that assign the job to the selected driver on the Dispatch Driver Interface. The status will be updated to ‘Assigned’ and added to the driver’s job list. The mechanism works the same on the Assign Jobs Page for delivery (See Figure 7.4.5). The ‘Smart Assign’ button on the bottom allows admin to call the smart jobs assignment function to bulk assign the jobs to a driver with weight calculated and status updates.

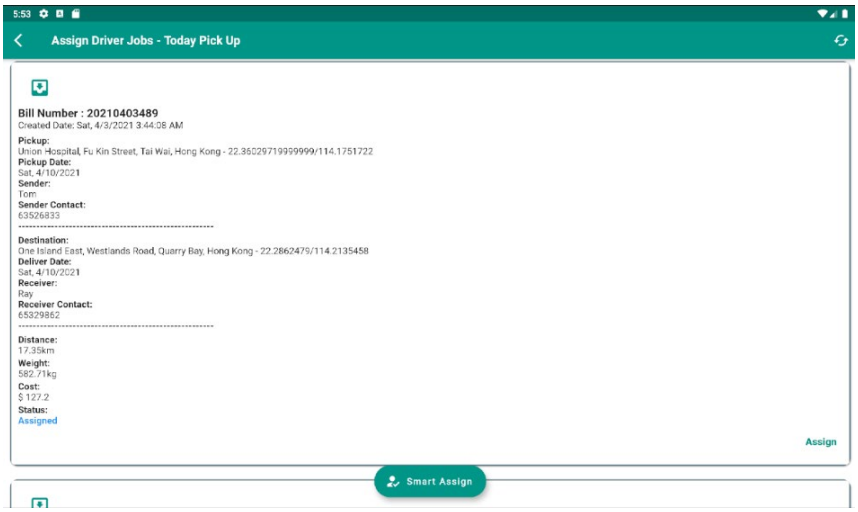


Figure 7.4.4 Assign Driver Jobs (Pick up)

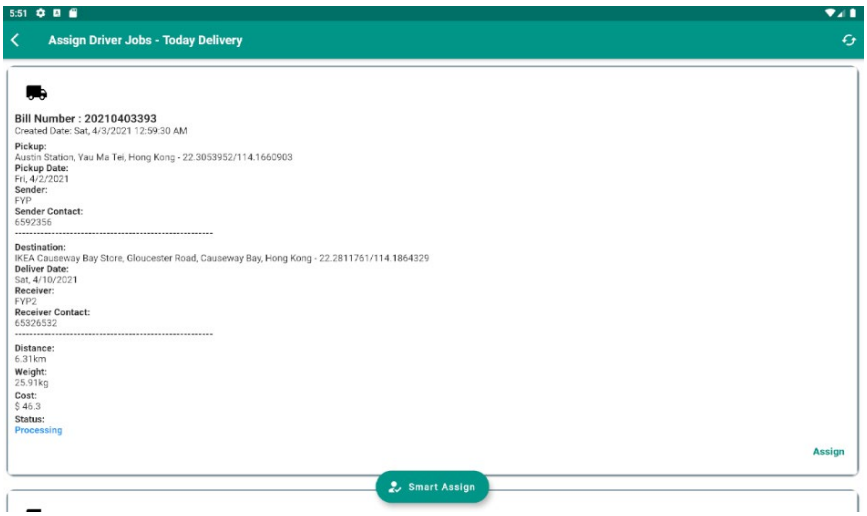


Figure 7.4.5 Assign Driver Jobs (Delivery)

7.4.3 Stock Management

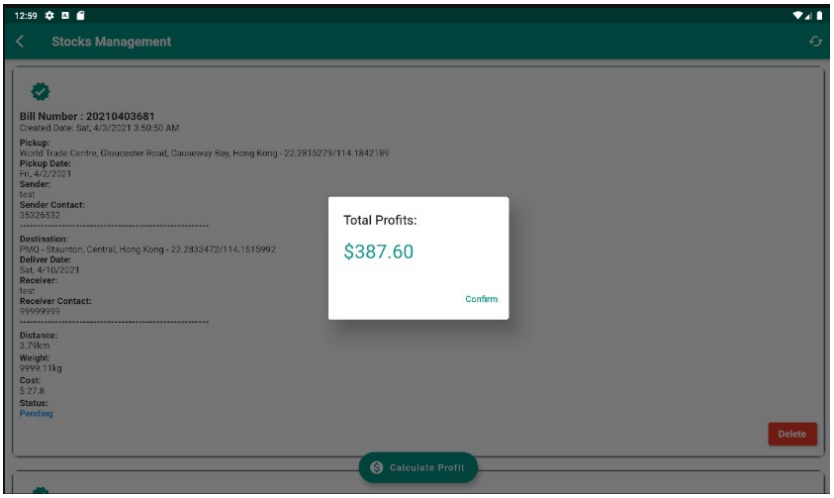


Figure 7.4.6 Stock Management Interface

Figure 7.4.6 shows the Stock Management Interface. It displays all the bill that have been created by users. There will be a 'Delete' button for admin to delete the bill and 'Calculate Profit' for admin to calculate the total profit bring by the all the delivery.

7.4.4 User Management

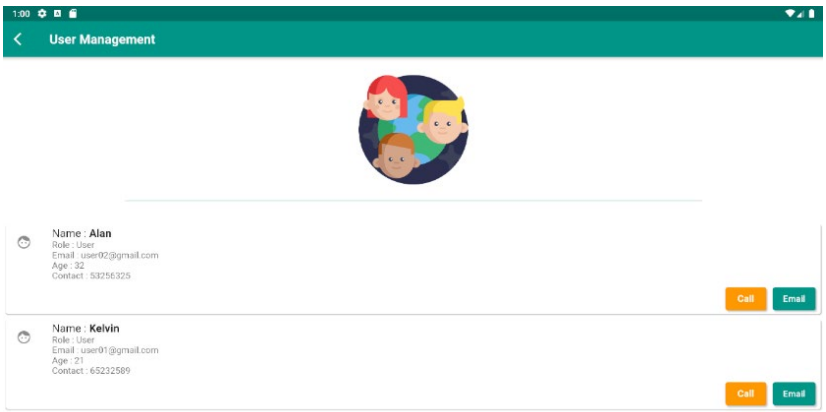


Figure 7.4.7 User Management Interface

Figure 7.4.7 shows the User Management Interface. Admin can view all the registered users and their details. Admin can make phone calls and email to communicate with the user. These functions will launch the Dialing and Email applications of the devices.

7.4.5 Driver Management

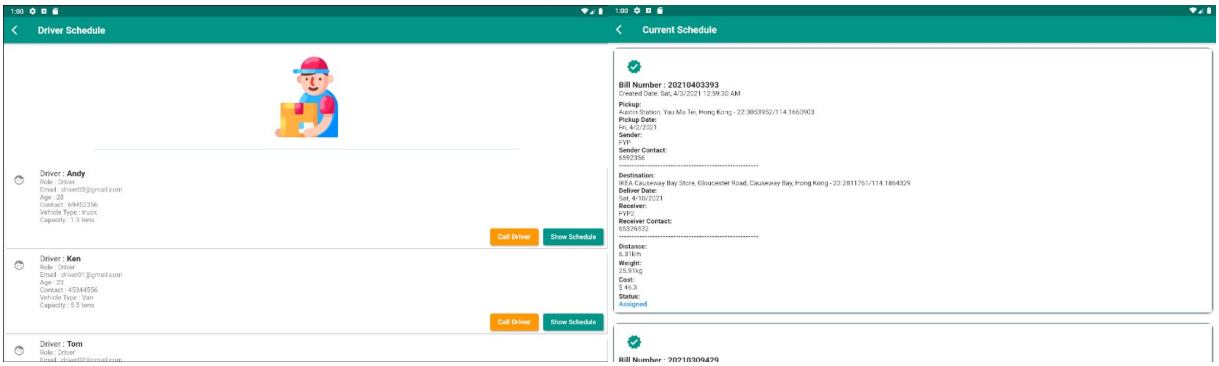


Figure 7.4.8 Driver Management Interface

Figure 7.4.8 shows the Driver Management Interface. Admin can view all the registered drivers and their details. Admin can make phone calls and check their schedule. The calling function will launch the Dialing application and the check schedule will bring to the schedule checking page.

8. System Testing Plan

There will be two major parts for the system to be developed. First is the frontend mobile application with three interfaces and second part is the backend database and APIs connection. For that, the testing plan will be separated into two parts. Frontend-backed communication test and function completeness test.

8.1 Frontend-Backend Communication Testing

White box testing will be conducted in this part of testing. Different parameters and inputs will be put to each unit, functions and then determine the expected results. Input values from the mobile application will be pass to the database. Then the changes from the database data will be capture to determine the connectivity. After that, there will be tests on functions like bill tracking, and items delivery to test the backend data retrieval.

8.2 Function Completeness

Black box testing will be conducted in this part of testing. Test cases will be designed from the users' perspectives to examine the actual operation of the mobile application. Different test cases will be designed for each of the functions.

1. Passing empty, correct and incorrect values to the system and submit.
2. Loss of Internet connection.
3. Back to phone’s home screen and re-open the app when operating functions.

Comparisons between actual values and expected values will be conducted and there will be catch methods to handle each of the error cases if discovered in this test. Apart from that, the tests for routing methods will also be done to examine if the path distances are correctly measured on Google Map. Comparisons will be done with other map engines such as Bing Map and Apple Map.

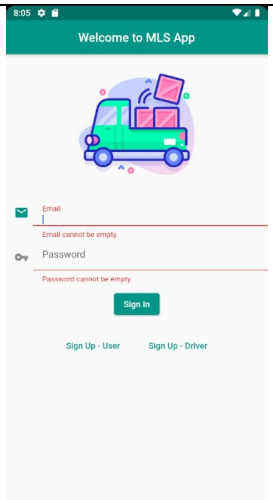
8.3 Test Cases

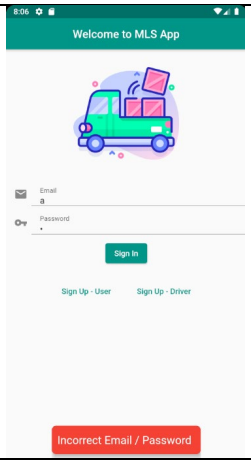
Test Case ID	T1
Description	Login with correct email and password for user identity.
Input Values	Email: user01@gmail.com Password: user01
Expected Result	Successful login and redirect to user interface.
Actual Result	Loaded for 2 seconds, logined and directed to user screen.
Pass?	Yes

Test Case ID	T2
Description	Login with correct email and password for admin identity.
Input Values	Email: admin123@gmail.com Password: admin123
Expected Result	Successful login and redirect to admin interface.
Actual Result	Loaded for 2 seconds, logined and directed to admin screen.
Pass?	Yes

Test Case ID	T3
Description	Login with correct email and password for driver identity.
Input Values	Email: driver01@gmail.com Password: driver01
Expected Result	Successful login and redirect to driver interface.
Actual Result	Loaded for 2 seconds, logined and directed to driver screen.
Pass?	Yes

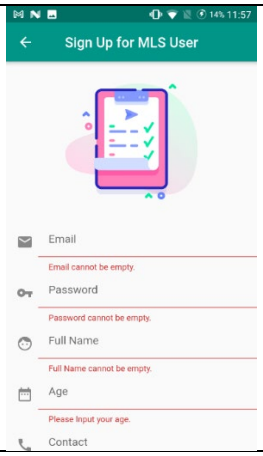
Test Case ID	T4
Description	Login with empty email and password.
Input Values	Email: null Password: null
Expected Result	Login Fail and show error messages.

Actual Result	
Pass?	Yes

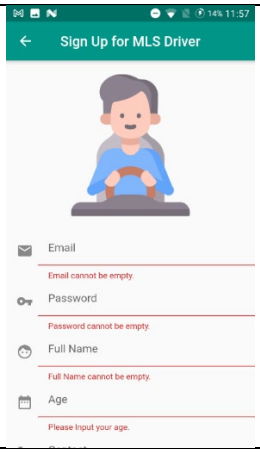
Test Case ID	T5
Description	Login with incorrect email and password.
Input Values	Email: a Password: a
Expected Result	Login Fail and show error messages.
Actual Result	
Pass?	Yes

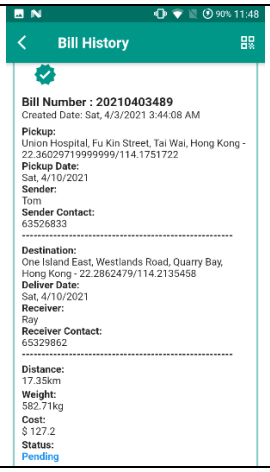
Test Case ID	T6
Description	Register as user with correct inputs.
Input Values	Email: user02@gmail.com Password: user02 Full Name: John Age: 21 Contact: 60505958
Expected Result	Successful registered and data stores to Firebase authentication and database.
Actual Result	User registered, data stored to Firebases authentication and created new user role database entity. Redirected to user interface dashboard.
Pass?	Yes

Test Case ID	T7
Description	Register as driver with correct inputs.
Input Values	Email: driver02@gmail.com Password: driver02 Full Name: Tim Age: 48 Contact: 62534568 Vehicle Type: Truck Vehicle Capacity: 5.5 tons
Expected Result	Successful registered and data stores to Firebase authentication and database.
Actual Result	Driver registered, data stored to Firebases authentication and created new driver role database entity. Redirected to driver interface dashboard.
Pass?	Yes

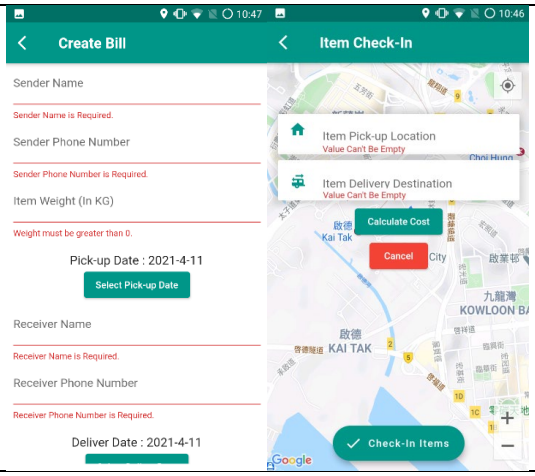
Test Case ID	T8
Description	Register as user with missing inputs.
Input Values	Email: null Password: null Full Name: null Age: null Contact: null
Expected Result	Fail registeration and show error messages.
Actual Result	
Pass?	Yes

Test Case ID	T9
Description	Register as driver with missing inputs.
Input Values	Email: null Password: null Full Name: null Age: null Contact: null Vehicle Type: null Vehicle Capacity: null
Expected Result	Fail registeration and show error messages.

Actual Result	
Pass?	Yes

Test Case ID	T10
Description	[User] Check-in item will correct inputs and same day delivery
Input Values	Pick-up Location: "Union Hospital, Fu Kin Street, Tai Wai, Hong Kong - 22.36029719999999/114.1751722" Destination: "One Island East, Westlands Road, Quarry Bay, Hong Kong - 22.2862479/114.2135458" Pick-up Date: Sat 4/10/2021 Sender Name: Tom Sender Contact: 63526833 Deliver Date: Sat 4/10/2021 Receiver Name: Ray Receiver Contact: 65329862 Weight: 582.71 kg
Expected Result	Calculated the cost, distance and suggested route. Successful checked-in, data stored to Firebases and created new bill entity.
Actual Result	
Pass?	Yes

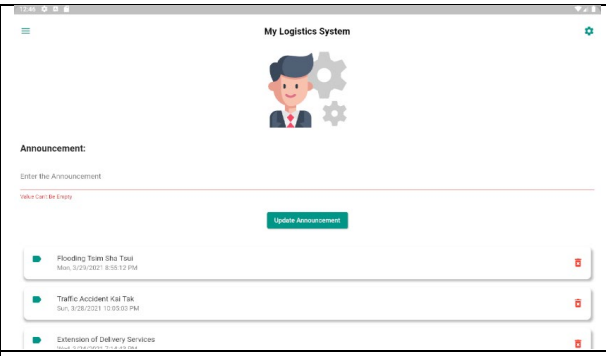
Test Case ID	T11
Description	[User] Check-in item will empty inputs.
Input Values	Pick-up Location: null Destination: null Pick-up Date: null Sender Name: null

	Sender Contact: null Deliver Date: null Receiver Name: null Receiver Contact: null Weight: null
Expected Result	Check-in fail and show error messages
Actual Result	
Pass?	Yes

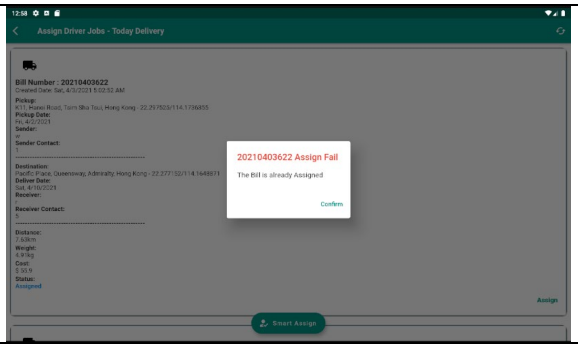
Test Case ID	T12
Description	[User] Check Bill History.
Input Values	N/A
Expected Result	Retrieved the Bills created by that user and show on screen.
Actual Result	The related bill history showed on screen.
Pass?	Yes

Test Case ID	T13
Description	[User] User Track Bill live location.
Input Values	N/A
Expected Result	Retrieved the “Processing” Bills and show the driver’s location on screen with realtime update.
Actual Result	The related bill tracking and driver’s location showed on screen.
Pass?	Yes

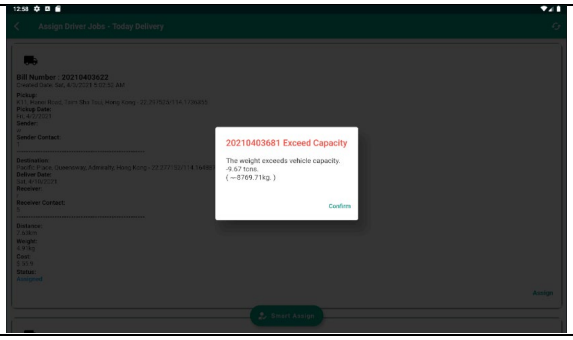
Test Case ID	T14
Description	[Admin] Admin inputs empty announcement.
Input Values	null
Expected Result	Announcement fail to create, show error messages.

Actual Result	
Pass?	Yes

Test Case ID	T15
Description	[Admin] Admin inputs valid announcement.
Input Values	Announcement: Traffic Jam - Kwun Tong
Expected Result	The announcement stores to Firebase database and show on user, admin and driver dashboard.
Actual Result	The related announcement is stored and shows on the correct dashboards.
Pass?	Yes

Test Case ID	T16
Description	[Admin] Admin performs smart jobs assignment but the bills status is invalid.
Input Values	N/A
Expected Result	Bill assignment fails and show the related error messages.
Actual Result	
Pass?	Yes

Test Case ID	T17
Description	[Admin] Admin performs smart jobs assignment but the bills is overweighted.
Input Values	N/A
Expected Result	Bill assignment fails and show the related error messages.

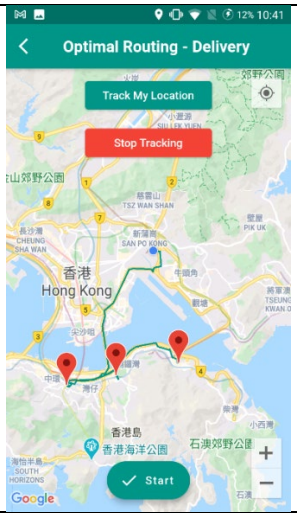
Actual Result	
Pass?	Yes

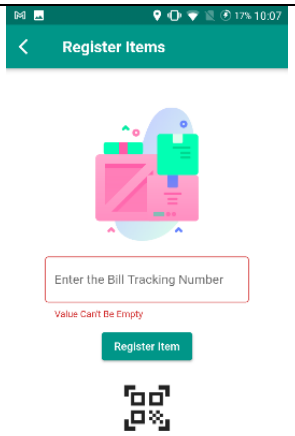
Test Case ID	T18
Description	[Admin] Admin calls a user.
Input Values	N/A
Expected Result	The device launches the dialing app with the user contact number.
Actual Result	Dialing app is launched with the selected contact number.
Pass?	Yes

Test Case ID	T19
Description	[Admin] Admin emails a user.
Input Values	N/A
Expected Result	The device launches the email app with the user email.
Actual Result	Email app is launched with the selected email.
Pass?	Yes

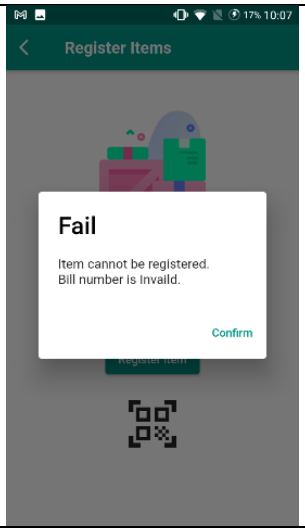
Test Case ID	T20
Description	[Admin] Admin checks a driver schedule.
Input Values	N/A
Expected Result	It directs to the selected driver's jobs list.
Actual Result	The related jobs list is shown with information.
Pass?	Yes

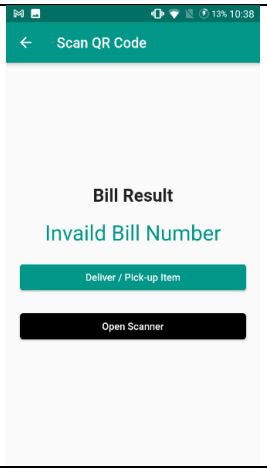
Test Case ID	T21
Description	[Driver] Driver performs optimal routing after finishing a job.
Input Values	N/A
Expected Result	The finished job destination remove from the list and re-route a new optimal route.

Actual Result	
Pass?	Yes

Test Case ID	T22
Description	[Driver] Driver enters empty bill number for item register.
Input Values	null
Expected Result	Item register fails and show the error messages.
Actual Result	
Pass?	Yes

Test Case ID	T23
Description	[Driver] Driver enters/ scan invalid bill number for item register.
Input Values	Bill Number : 22C$@23
Expected Result	Item register fails and show the error messages.

Actual Result	
Pass?	Yes

Test Case ID	T23
Description	[Driver] Driver scan an invalid bill number for processing the item.
Input Values	Bill Number : 22C$@23
Expected Result	Item process fails and show the error messages.
Actual Result	
Pass?	Yes

9. Conclusion

In this section, the achievement and expectation of this Mobile-based Logistics System will be discussed.

9.1 Achievement

Mobile-based Logistics System has achieved these project objectives: (1) provide an all-in-one mobile-based application for logistics businesses, (2) enhancing logistics performances with optimal routing and smart assignment, (3) reduce the operation cost for small-scaled businesses, (4) boost businesses productivity and flexibility, (5) provide additional features for users and businesses to improve user experiences.

The system provides features including:

9.1.1 Mutiple Client applications with Access Control

This system contains an application but with three interfaces which are well-designed for different account access of users, administrator and drivers.

9.1.2 Flexible Online Items Check-In Services

Users can now check-in items for logistics services online. This can minimize the locational barriers and time barriers also lower the operation cost for businesses to provide check-in services.

9.1.3 Items Real-Time Tracking with ETA

The system will retrieve real-time coordinates information from drivers. Users and administrator can hence track and monitor the order location and status. The ETA services also provide the accurate time for users to reference for the arrival of pick-up or delivery driver.

9.1.4 Items Process Security

QR Code scanning is required for both users to identify the driver and also drivers to identify the items. This double-sided procedure will surely security the logistics

process.

9.1.5 Smart Jobs Assignment

The created bills will be assigned to drivers automatically on the administrator application. The system will help calculate the weight and bill status to assign the item to the suitable drivers. This reduces the labour cost for businesses.

9.1.6 Optimal Routing and Routes Suggestion

Optimal Routing and routes suggestion will be available to the drivers. The system can provide suitable routes for the delivery according to the driver's location. This helps drivers to save time and fuel cost.

9.1.7 Easy Proof of Delivery

The system also provides E-Signature for the proof of delivery. This can be adopted to replace the traditional hard-writing signature. It can also be retrieved in the database easily. This can increase the efficiency of the logistics services and favour the user.

9.2 Expectations

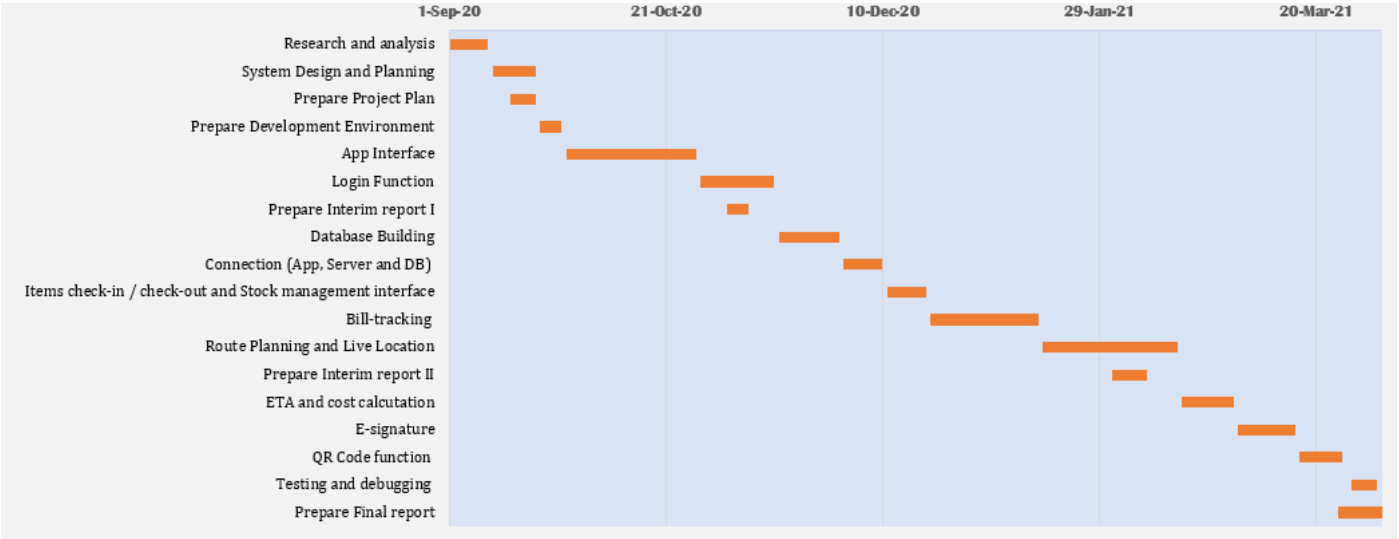
Mobile-based Logistics System provides an all-in-one solution for logistics businesses. Besides, this logistics management solution can also deploy in transportation, security, tourism, IoT and public services. Hope this project can bring inspiration for the growing sharing economy which may need functions for tracking and assignment for sharing property.

10. Project Schedule

10.1 Project Timeline and Milestones

Features	Start Date	End Date	Duration
Research and analysis	1-Sep-20	10-Sep-20	9
System Design and Planning	11-Sep-20	21-Sep-20	10
Prepare Project Plan	15-Sep-20	21-Sep-20	6
Prepare Development Environment	22-Sep-20	27-Sep-20	5
App Interface	28-Sep-20	28-Oct-20	30
Login Function	29-Oct-20	15-Nov-20	17
Prepare Interim report I	4-Nov-20	9-Nov-20	5
Database Building	16-Nov-20	30-Nov-20	14
Connection (App, Server and DB)	1-Dec-20	10-Dec-20	9
Items check-in / check-out and Stock management interface	11-Dec-20	20-Dec-20	9
Bill-tracking	21-Dec-20	15-Jan-21	25
Route Planning and Live Location	16-Jan-21	16-Feb-21	31
Prepare Interim report II	1-Feb-21	9-Feb-21	8
ETA and cost calcutation	17-Feb-21	1-Mar-21	12
E-signature	2-Mar-21	15-Mar-21	13
QR Code function	16-Mar-21	26-Mar-21	10
Testing and debugging	28-Mar-21	3-Apr-21	6
Prepare Final report	25-Mar-21	4-Apr-21	10

10.2 Gantt Chart



11. References

[1] Rising opportunities in industrials and logistics after Covid-19. (n.d.). Retrieved October 22, 2020, from https://www.aberdeenstandard.com/en/insights-thinking-aloud/article-page/the-winner-takes-it-all-industrials-and-logistics-after-covid19/

[2] SF Express - Chinese multinational delivery services and logistics company. Retrieved November 04, 2020, from https://www.sf-express.com/us/en/

[3] Geopointe – native AppExchange application by Ascentcloud. Retrieved November 04, 2020, from https://www.geopointe.com/

[4] Onfleet - delivery management system. Retrieved October 21, 2020, from https://onfleet.com/

[5] Applegate, David L. (2006) "*The Traveling Salesman Problem: A Computational Study*.", Princeton University Press, pp. 1–5

[6] Goodrich, Michael T .; Tamassia, Roberto. (2015) "*18.1.2 The Christofides Approximation Algorithm*", Algorithm Design and Applications, Wiley, pp. 513–514

[7] Christofids, Nicos. (1976) "*Worst-case analysis of a new heuristic for the travelling salesman problem.*" No. RR-388. Carnegie Mellon University Pittsburgh Pa Management Sciences Research Group

[8] Ataei, Masoud. (2015) "*A Branch-and-Price Algorithm for Bin Packing Problem.*", York University

[9] Ausiello, Giorgio. (2003) "Complexity and Approximation.", Springer, pp. 84–85

[10] Svennerberg, Gabriel (2010) "*Beginning Google Maps API.*" , Apress

[11] Smartphone Market Share - OS. (2020, September 14). Retrieved November 04, 2020, from https://www.idc.com/promo/smartphone-market-share/os

12. . Monthly Logs

12.1 October 2020

1. At the beginning of October (1 Oct to 5 Oct), a comparison is made between react native and flutter. Owing to the long learning path of react native, the development language is changed to flutter for this project.

2. From 6 Oct to 8 Oct, the development environment for Flutter is constructed.

3. From 9 Oct to 20 Oct, basic user interfaces for the app is created for user, driver and admin accounts. Including sidebar, functions page route and text fields.

4. From 15 Oct to 20 Oct, Google Map is imported for the user interface for items check-in.

5. From 17 Oct to 23 Oct, Google Firebase is created and used to classify the login identities.

6. From 23 Oct to 31 Oct, QR Code function is studied and will be implemented next month.

Project plan and draft for the interim report I, are also written in October.

The target for next month is to develop functions for user interface and create database attributes for the app.

12.2 November 2020

1. At the beginning of November, a few adjustments are made on the project.

2. From 1 Nov to 7 Nov, Intermin Report I is prepared for submission.

3. From 8 Nov to 15 Nov, the login function is being accelerated.

4. From 15 Nov to 20 Nov, Firebase database is developed with attributes for users, drivers, admin and items.

6. From 21 Nov to 30 Nov, the connection between the database and the Google Map for retrieving the coordinates is being investigated and studied.

The target for next month is to develop functions for online items check-in and real-time live location tracking.

12.3 December 2020

1. At the beginning of December, a few adjustments are made on the project, including debugging and environment updates.

2. From 1 Dec to 5 Dec, Android emulator has problems in connecting Firebases authentication APIs and has been fixed using downgraded Android version.

3. From 6 Dec to 15 Dec, text autocomplete for searching for 'PickUp' and 'Destination' has been done on Google map.

4. From 16 Dec to 18 Dec, routing can be shown on screen including the distance and cost.

6. From 19 Dec to 25 Dec, items check-in form has been created and designed.

7. From 26 Dec to 31 Dec, created items bill can be uploaded to Firebase.

The target for next month is to develop functions Bill History retrieve, adding the ticket number and status, also will start developing admin functions for stock management.

12.4 January 2021

1. From 1 Jan to 10 Jan, Bill History function is finished including generating the bill number, getting the bill status and generating the QR Code.

2. From 11 Jan to 20 Jan, admin stock management is finished include delete function for the bill, calculate the total profit and retrieving the bill from the database into a list form.

3. From 21 Jan to 25 Jan, the driver and user profile are created in the database for registration.

4. From 26 Jan to 28 Jan, Dispatch Driver Interface is created which can retrieve driver location and display on the screen.

5. From 29 Jan to 31 Jan, Assign Jobs function to drivers is developed.

Interim report II, are also written in January.

The target for next month is to develop functions for optimal path and driver's location realtime tracking.

12.5 February 2021

1. From 1 Feb to 11 Feb, the Optimal Path Routing function is completed.

2. From 12 Feb to 18 Feb, debug for the dispatch drivers function is proceed.

3. From 19 Feb to 25 Feb, the driver tracking and bill tracking function are completed.

4. From 26 Feb to 28 Feb, the ETA function is completed and the drivers can also register items to their vehicle.

The target for next month is to develop functions for QRCode scanning and E-signature.

Several minor functions will be underdeveloped and project debugging will proceed.

12.6 March 2021

1. From 1 March to 10 March, the QR Code and E-signature functions are completed.

2. From 11 March to 17 March, the generation of driver identity QR Code is completed.

3. From 18 March to 21 March, the management of users and drivers interfaces for admin is completed.

4. From 22 March to 25 March, the announcement function is completed.

5. From 26 March to 28 March, user profiles for users and drivers are completed.

6. In late March, Smart Jobs Assignment Function for drivers is completed.

7. Several functions are changed and advanced, included QR Code for driver items register and driver offline option.

Project debugging, testing, Final Report, demonstration and presentation will be expected to complete in April.