



Department of Computer Science and Engineering  
The Chinese University of Hong Kong



# Indoor Guidance Application

FYP Report Fall 2015

Supervised By  
**Prof. Michael R. Lyu**

Written By:  
**CHOI Mei Shan** 1155045904  
**WONG Tsz Kin** 1155038146

Date:  
2<sup>nd</sup> December, 2015

This is a Blank Page.

## **Abstract**

We are going to design and implement an indoor navigation system for driver used in car park. There are a lot of indoor navigation systems nowadays, but most of them are focus on the behavior of walking such as exhibition, guidance in supermarket. Since there is no navigation application is aimed for driving in car-park. So, we would like to develop a car-park guidance system for driver to solve some common problem will occur on a drivers, such as “driver don’t know where is the available parking space for he to park.” Or some drivers even don’t remember the location where he parked his car. In this project, we have studied a trendy technology — Beacon technology, and have done some experience on how should we deploy the Beacon device in the car park so that it will have the best performance. Finally, we make use of Beacon (including iBeacon and Eddystone) to achieve guidance service for driver in car park.

## Contents

Abstract .....	3
Chapter 1 - Introduction .....	7
Overview .....	7
Motivation .....	7
Objective .....	8
Assumed Users .....	8
Project Scope .....	8
Chapter 2 - Technology Overview .....	9
Indoor Navigation System - INS .....	9
Beacon .....	9
Near Field Communication - NFC .....	11
Chapter 3 - Design Overview .....	12
3.1 Why do we use Beacon for Indoor Positioning? .....	13
3.2 Why User App asks server to retrieve information but not stored locally (E.G. Local database)? .....	13
3.3 Will Beacon signal cover all area of car park? If no, where to deploy Beacon? .....	13
3.4 What functions provided by Car-park Guidance System (CGS) for End-User? .....	14
3.5 How could you help the driver to solve the problem? .....	15
3.6 What functions will be provided by Car-park Guidance System (CGS) for Car-park Administrator? .....	15
3.7 How could you help the administrative person to make decision? .....	15
3.8 How to achieve indoor guidance using Beacon? .....	16
3.9 Project will use Eddystone or iBeacon? .....	20
Chapter 4 - Feasibility Study .....	21
Assumption 1 — Beacons are detectable by smart phone within a moving vehicle .....	22
Experience on Time Delay Error .....	22

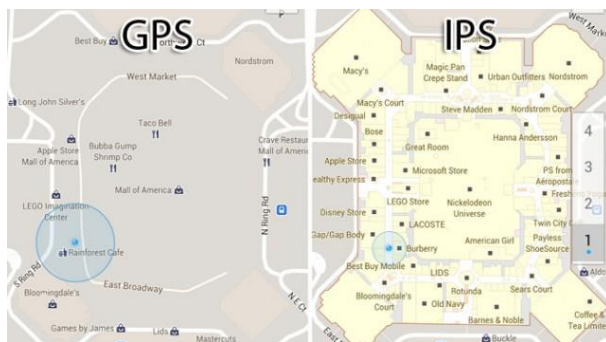
Assumption 2 — Beacons signals will not be interfered by WiFi signals .....	31
Chapter 5 - Design Specification.....	32
5.1 Content Management System (Back-end).....	33
5.3 Web’s User Interface (Draft version in the summer 2015) .....	33
5.4 Web’s User Interface (2 <sup>nd</sup> version).....	37
5.5 Web’s Security Issues.....	44
5.6 Application Programming Interface (API) Specification .....	48
5.7 Routing Table .....	57
5.8 User Android App (Front-end).....	64
5.9 User Interface Design of Android App.....	65
5.10 Class Diagram of Android App.....	67
5.11 Functionality of User App .....	71
5.12 Database Specification .....	80
Chapter 6 - Project Implementation (First term) .....	83
Web Hosting for Content Management System (CMS).....	83
Database Server .....	84
Beacon installation in Lady Shaw Building Car Park .....	84
Testing the Guidance System .....	86
Chapter 7 - Goal for the Second Term .....	93
Dynamic Routing (dynamic route cost based on different situation).....	93
Data Analytic.....	93
Graph Database Engine .....	93
Prediction on user preference .....	94
Chapter 8 - Conclusion.....	95
Acknowledgement.....	96
Reference.....	96

Appendix .....	97
Appendix — MySQL database script.....	97

# Chapter 1 - Introduction

## Overview

As we know Global Position System – GPS has already become a mature technology for outdoor, but GPS cannot provide indoor position service. Actually, Indoor position system – IPS have already exist for a while but there are lots of limitations include accuracy, cost, etc.



In recent years, Internet of Things (IoT) becomes a new trend. It brings the Internet into real physical world. Even Google has followed the trend and created a project named Physical Web Project, which want to connect real world item to the internet through an URL tag. QR code is a reflection of Physical Web. Beacon technology can be considered as an improved version QR code which showed up in recent years. People are not focus a lot on indoor guidance until the beacon technologies was showed up. Beacon is cheap, can get high accuracy with enough beacons, and low power consumption. Developer willing to develop app with beacon such as exhibition, that's why indoor positioning technology become more mature.

## Motivation

Nowadays, there are a lots of indoor guidance applications focus on the behaviors of walking like exhibition indoor guidance. We found that the behaviors of walking are different than driving. INS which concerns the behaviors of walking cannot apply on some driving scenario such as parking.

In the walking scenario, users can walk around with dynamic path, but in the driving scenario, drivers can only drive with a specific path. Also, there are some limitations of driving scenario such as traffic congestion, barricades. These are the reasons that developers are not willing to write an INS for carpark.

We wonder that an carpark indoor guidance can be a new trend of beacon usage, because there are lots of carparks in the world, drivers may not know details of all carparks like floorplan or the parking spaces' status.

Sometime, drivers may take time to search available parking spaces inside a large carpark, also it waste lots of time when they find out the carpark is full after they arrived. Therefore we will design an indoor guidance app for drivers to solve some common problems in carpark.

## **Objective**

In our project, we are going to study indoor guidance system with beacon and BLE.

- 1) To provide a completed indoor carpark guidance service,
- 2) To provide further services for the drivers not only focus on carpark.
- 3) To analyze user experience and suggest user a prefer parking space.
- 4) To generate the statistic report to carpark administrator.

## **Assumed Users**

There are 2 kinds of users will use our system. The primary user is driver, the end-user of our system. Drivers are expected to use our Android application to park their car. Our Android Application will provide indoor navigation and guidance to the primary user.

The secondary user is administrator of car parks. He is responsible to update the information of our system through the content management system (CMS). Our CMS will provide a platform to secondary user to update the information of the guidance system, and also will provide some statistic information to secondary user.

## **Project Scope**

Our scope of project is to develop a system which can provide indoor guidance function to driver in any car parks. Secondary user must input enough information of car parks (such as physical location, floor plan, Beacon information...etc) into our system, so our system can analyze the given information to provide indoor guidance service. Therefore, the scope of this project is not to cover all car parks in CUHK, but the car park already inputted into our system. The action of inputting car park information into our system is out of our project scope. But for demonstration, we will input the information of Lady Shaw Building car park into our system.

Second, the project scope mainly focuses on the indoor guidance. Obtaining the real-time information of the car park (such as total number of available parking space, the location of available parking space) is out of our project scope. We will assume system can obtain the real-time information of car park through the third-party API.



## Chapter 2 - Technology Overview

During summer 2015, we have studied some technology related to indoor navigation system. Here is the result of our finding.

### Indoor Navigation System - INS

INS provide more user experience for indoor which can locate people or object inside the building by using beacons, Wi-Fi, NFC etc. Also, INS reduce the GPS problem because GPS signals are hard to be useful in indoor environment. Although there are many INS app in the market, there is no standard for INS.

### Beacon

Beacons work on Bluetooth Low Energy – BLE which transmit a signal up to a certain distance, ranging from 15cm (~6 in) to 70m (~230 ft). Beacons provide a virtual region, when we scan the specific region of beacon then we can say that we are within that region

#### - **iBeacon**

An implementation of BLE technology which announced by Apple in 2013 for iOS 7 or latest device with BLE.

UUID, major and minor are used to identify iBeacon.

Field	Size	
UUID	16 bytes	Differentiate related beacon in same large group
Major	2 bytes	Differentiate a subset within the large group
Minor	2 bytes	Further differentiate of subset, identical of each beacon within the subset.

Here is an example that how UUID, major and minor work.

Store Location	Tai Po	Shan Tin
UUID	A8C5DB1E-6785-1A25-778B-5E25DA57BC82	
Major	1	2
Minor	Books	20
	CDs	30
	Stationery	40

In this case, UUID present an identification of a company, major present the region of the store and minor present types of product that are sold in the store.

## - Eddystone

An implementation of BLE technology which announced by Google in 2015 for both Android 4.3 and iOS 7 or latest device with BLE.

Field	Size	
Eddystone-UID	16 bytes	Name space (10 bytes) like UUID, Instance (6 bytes) like Major and Minor.
Eddystone-URL	2 bytes	Contain URL, the size is depend on the length of that URL.
Eddystone-TLMv	2 bytes	Contain telemetry packet include battery voltage, beacon temperature, etc.

### Eddystone-UID

Eddystone-UID contains an identifier like iBeacon's UUID, major and minor. UID divide in to two parts: Name space and instance.

Namespace's purpose is similar to iBeacon's UUID, but UID's namespace can be customized. According to the Eddystone specification which recommends taking the first 10 bytes of an SHA-1 hash of our domain name. "Another method is to simply remove the three middle parts of a Version 4 UUID"<sup>1</sup>

8B0CA750-E7A7-4E14-BD99-095477CB3E77 becomes 8B0CA750095477CB3E77 .

Instance is used to further identify unique beacon similar to iBeacon major and minor.

### Eddystone-URL

Eddystone-URL contain URL, propose of Eddystone-URL just like Physical Web, the URL will be directly opened by standing near by the specific beacon. Also, URL could be dynamic web application or regular website.

### Eddystone-TLMv

Eddystone-TLMv contain telemetry packet, this packet contain the information of the beacon such as battery voltage, temperature, and counts of broadcast packets. Based on these information we can monitor beacon health, fix it or replace it as soon as possible.

---

<sup>1</sup> Sources: <http://developer.estimote.com/eddystone/>;  
<https://github.com/google/eddystone/blob/master/protocol-specification.md>

Compare with Eddystone and iBeacon

	<b>Eddystone</b>	<b>iBeacon</b>
<b>Official Support Device</b>	Android & iOS	iOS
<b>Protocol</b>	Open protocol	Close
<b>Packet type</b>	Eddystone-UID Eddystone-URL Eddystone-TLMv	UUID Major Minor

### **Near Field Communication - NFC**

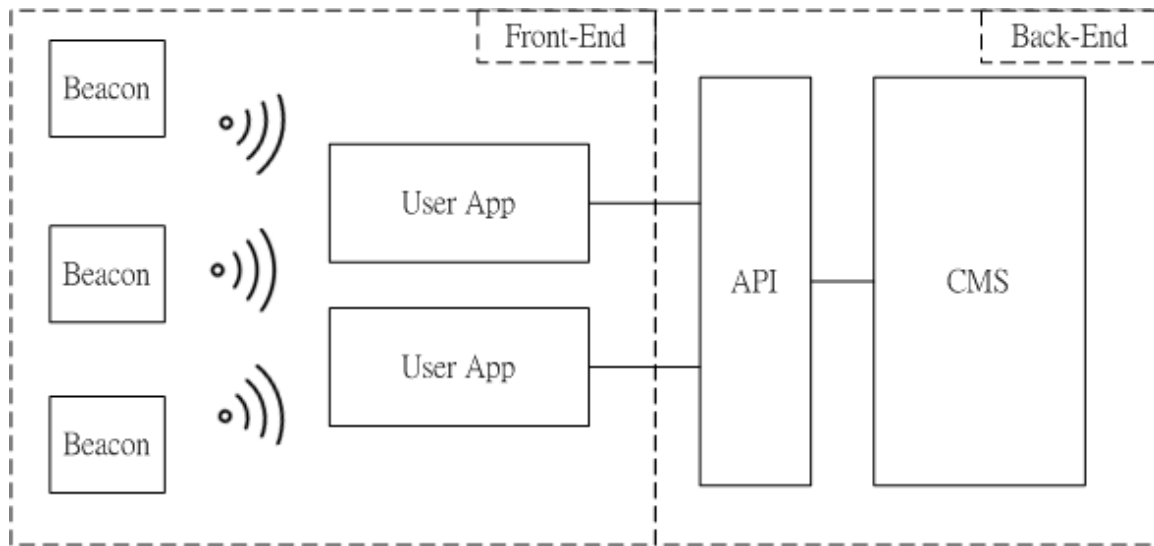
NFC is a short range communication technology, the communication occur at 10cm or less, it would be perfect range. The transmission frequency for data through NFC is 13.56 MHz and data rates either 106, 212 or 424 Kb/s. The operation modes of NFC include Peer-to-Peer mode, Read/write mode, Card emulation.

Peer-to-Peer mode: Exchange data between two active NFC devices, commonly occur by smart phone.

Read/write mode: It is a one way data transmission, the active device connects to another devices to read information or write data to NFC tags.

Card emulation: Store private information such as credit card, identity card, passport into the NFC devices act as a smart card. When smart card tapping in a NFC reader, we can get the information from the card.

## Chapter 3 - Design Overview



Car-park Guidance System (CGS) can be divided into front-end and back-end. In the part of front-end, it contains Beacon and User App. For back-end, it contains Application Programming Interface (API) and Content Management System (CMS). API acts as a communication gateway between User App and Content Management System (CMS). First, let's talk about the basic operation flow of our designed system.

### *Beacon*

Beacon broadcasts Bluetooth signal which contains unique information of itself frequently, so the coverage area will be filled with beacon's signal, the area is called Beacon Region.

### *User App*

When user enters into the Region, User App can detect the Beacon's signal and retrieve information advertised by Beacon. So, we know the physical location of end-user base on which Beacons did User App detected.

### *API*

User App retrieves more information about that Beacon from CMS through API. The information includes but not limit to Beacon's physical location, number of available parking space, the fastest way to go to the parking space, etc...

After User App get enough information from server, for example:

- i) Physical location of driver (E.G. LSB's Car-park G/F Region A)
- ii) Physical location of Available parking space (E.G. LSB's Car-park 1/F Region F)
- iii) The Shortest Path to go to Region F from Region A.

User App can guide driver to the destination with the above information.

### **3.1 Why do we use Beacon for Indoor Positioning?**

After we studied several common wireless communication ways, we found that Beacon is the most suitable one. Beacon technology has the following benefits that are great for our situation.

#### *1. Rely on Bluetooth Low Energy (BLE)*

The main advantage of BLE is low power consumption, a Beacon device can work 24 hours 7 days over a year with a button cell battery, but WiFi cannot. Moreover, the BLE device size is smaller, which is much easier to install in car-park.

#### *2. High supported range*

The range of BLE theoretically support up to 50 meters, whereas NFC only support up to 0.2 meters. NFC is not suitable for our situation, because it is not possible that driver will place his phone less than 10 cm to the NFC tag in car-park during driving.

#### *3. Cross-platform and support wide range of devices*

BLE is supported by different platforms including Android, IOS, and Windows... etc. In addition, according to Bluetooth SIG, they predict that “by 2018 more than 90 percent of Bluetooth-enabled smartphones will support Bluetooth Smart”<sup>2</sup> Therefore, we can say BLE or Beacon is a trend in future.

### **3.2 Why User App asks server to retrieve information but not stored locally (E.G. Local database)?**

We designed our User App as a light weight client. Client will not download all the things from server, but vice visa, it only downloads the resource it needed at that time. If every stored in the client side, then it requires very large memory to store the resource. On the other hands, since not all resources are useful for end-user, for example if driver is looking for guidance in Lady Shaw Building car park, then it is not necessary to retrieve the information about the Ho Sin Hang Engineering Building car park.

In our scenario, some information is changing in time (E.G. location of available parking space). So the information retrieved from server can ensure that it is the latest information, whereas, if the User App is based on the local database, the information maybe expired.

### **3.3 Will Beacon signal cover all area of car park? If no, where to deploy Beacon?**

From our point of view, it is not necessary to deploy Beacons to cover all area of car park.

---

<sup>2</sup> Source: "Mobile Telephony Market". Bluetooth Special Interest Group. Retrieved January 16, 2014 on <http://www.bluetooth.com/Pages/Mobile-Telephony-Market.aspx>  
Source: “[https://en.wikipedia.org/wiki/Bluetooth\\_low\\_energy#cite\\_note-5](https://en.wikipedia.org/wiki/Bluetooth_low_energy#cite_note-5)”

First, it requires many Beacons to achieve (See figure 3.1.1). Because each Beacon only covers about 4 to 5 meters diameter coverage (we've done an experience about the RSSI against Distance).

Second, what we need to know is whether the user has entered specific regions (For example, lift lobby, crossroads...etc.), but not the exact location in car park. So, to minimize the number of Beacon required, we only need to deploy Beacons in some special locations which including toilet, lift lobby, electrical vehicle charger (EVC), turning point and crossroad. Figure 3.1.2 has shown the example of Beacon deployment. Beacons are deployed at the turning point and crossroad only.

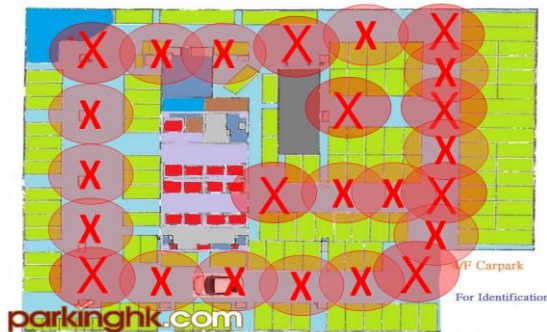


Figure 3.1.1 – Large amount of Beacons required to covering all area



Figure 3.1.2 – Beacons are only deployed on turning point and crossroad

### 3.4 What functions provided by Car-park Guidance System (CGS) for End-User?

For end-users (E.G. driver), they will mainly focus on the User App, which is the front-end of our system. The app will provide the following functions to driver:

- 1) Display the nearest available car-parks
- 2) When driver drive into either one of our supported car-parks, the car-park's floor plan will be showed on User App. And a marker appears on floor plan which indicates the physical location of driver.
- 3) Driver can then select the available parking space on floor plan, and app will guide him to drive to that parking space.
- 4) After the driver has parked his car, app will store the parking location, and guide driver to walk to the lift lobby or toilet.
- 5) After the driver has parked his car, app will ask user to set notification on phone to remind him how long he parked.
- 6) When driver return to car-park, app will guide him to walk to his car's location.
- 7) When driver go back to his car, app will display the parking information such as "How long he parked" and "Total fee he need to pay".
- 8) When driver go back to his car, app will guide driver drive to the exit of car-park.

### **3.5 How could you help the driver to solve the problem?**

Most of the drivers will have the same problem. Sometime they will ask “Where is the nearest car-park for him to park?” CUHK is an excellent example, because there is no sign, no app, nothing will tell driver the location of car-park in CUHK. How can you expect a non-CUHK staff / student know there is a car-park in Lady Shaw Building, Engineering Building...etc.? It’s doesn’t make sense. By using our app can help to solve this problem, drivers are able to list out all available car parks in CUHK.

The second question that drivers always ask is “Where did my car parked? Is it on 2<sup>nd</sup> floor?” By using our app, it will store and guide driver to go to the parking location. So, drivers no need to remember which floor or the exact location of his car.

The third question that people (not just driver) always ask is “Where is the nearest toilet?” By using our app, it will show the toilet location of floor plan and guide user to the there.

The last common question is “How long did I parked?” or even they don’t remember they has parked their car in car-park. By using our app, it will send notification to driver to remind them how long they parked. So, driver won’t forget their car.

### **3.6 What functions will be provided by Car-park Guidance System (CGS) for Car-park Administrator?**

For car-park administrator, they will mainly focus on the Content Management System (CMS), which is the back-end of our system. The CMS will provide the following functions to administrator:

- 1) Add / Delete Car-park information (including car-park’s floor plans, car-park’s physical location)
- 2) Add / Delete Beacon in car park.
- 3) Add / Delete Route between Beacons
- 4) Test the API (to ensure the API is normally functioning)
- 5) Check the Log history to see who are using the CGS and what did he done.
- 6) To view some statistic report (E.G. how many active user? Which car-park is the most famous? The demand of electrical vehicle charger (EVC) in car-park )

### **3.7 How could you help the administrative person to make decision?**

From the statistic report, administrators know which car-park has higher demand of electrical vehicle charger (EVC), then administrator can put the car-park has higher priority to install new EVC in car-park. Vice visa, if administrator found that the EVC in car-park has zero utilization, administrator can decide to move the EVC to other car-park which has higher demand.

### 3.8 How to achieve indoor guidance using Beacon?

As a network engineer, I found that we can borrow the idea of network routing into our scenario. In telecommunication, an IP packet sent from source host to destination host invoke network routing. First, the packet is forward to network gateway (normally it's a router), and the gateway will look up the routing table to determine the packet should be forwarded to which neighbor router, or called *next hop routing* (Figure 3.1.3).

Let me briefly explain the *next hop routing* showed in Figure 3.1.3.

- Step ① : Every IP packet contains two important fields, *source* (where are packet from), *destination* (where are the packet to). Here we want to send an IP packet from A to C.
- Step ② : Router A searches an entry with *source* equals to A and *destination* equals to C in its routing table. If the entry exists, then forward the IP packet to *next hop* through Port 1.
- Step ③ : Router B gets the IP packet and searches an entry with *source* equals to A and *destination* equals to C in its routing table. If the entry exists, then forward the IP packet to *next hop* through Port 2.
- Step ④, ⑤ : Router C gets the IP packet. Since the packet reached the *destination*, so Router C will not forward the IP packet to any other places. The routing process is done.

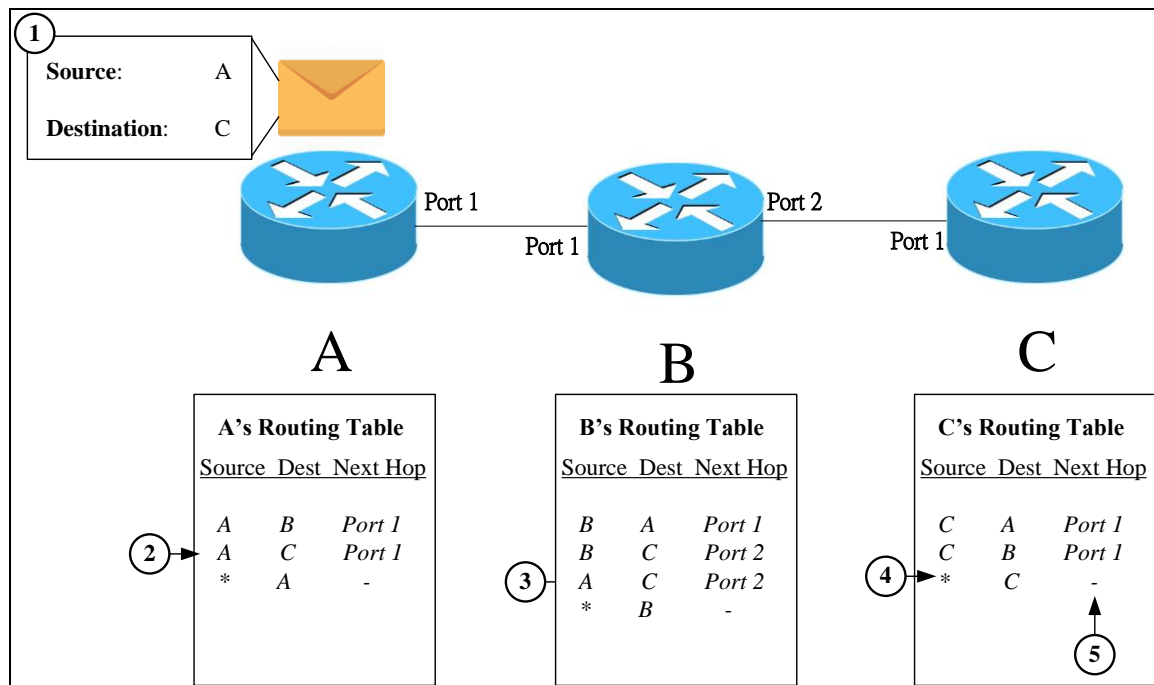


Figure 3.1.3 – Example of Next Hop Routing



Indoor guidance problem is a kind of routing problem or the shortest path problem. We can transfer the idea to be the indoor guidance algorithm. If we treat the *IP packet* as vehicle, *router* as Beacon, *next hop* as absolute direction (North, South, East and West), then the whole idea can be used in indoor guidance. Let's see an example (Figure 3.1.4).

- Step ① : Driver want to go to the destination C from A. User App detects the Beacon A, and asks server how to go to Beacon C given that driver is in the region of Beacon A.
- Step ② : Server searches an entry with source equals to A and destination equals to C in routing table. If the entry exists, then replies the *absolute direction* (North) to User App. Then, driver can follow the direction to drive to Beacon B.
- Step ③ : User App detects the Beacon B, and asks server how to go to Beacon C. Server searches an entry with source equals to B and destination equals to C in routing table. If the entry exists, then replies the *absolute direction* (East) to User App.
- Step ④ : Driver can follow the direction to drive to Beacon C. The guidance process is done.

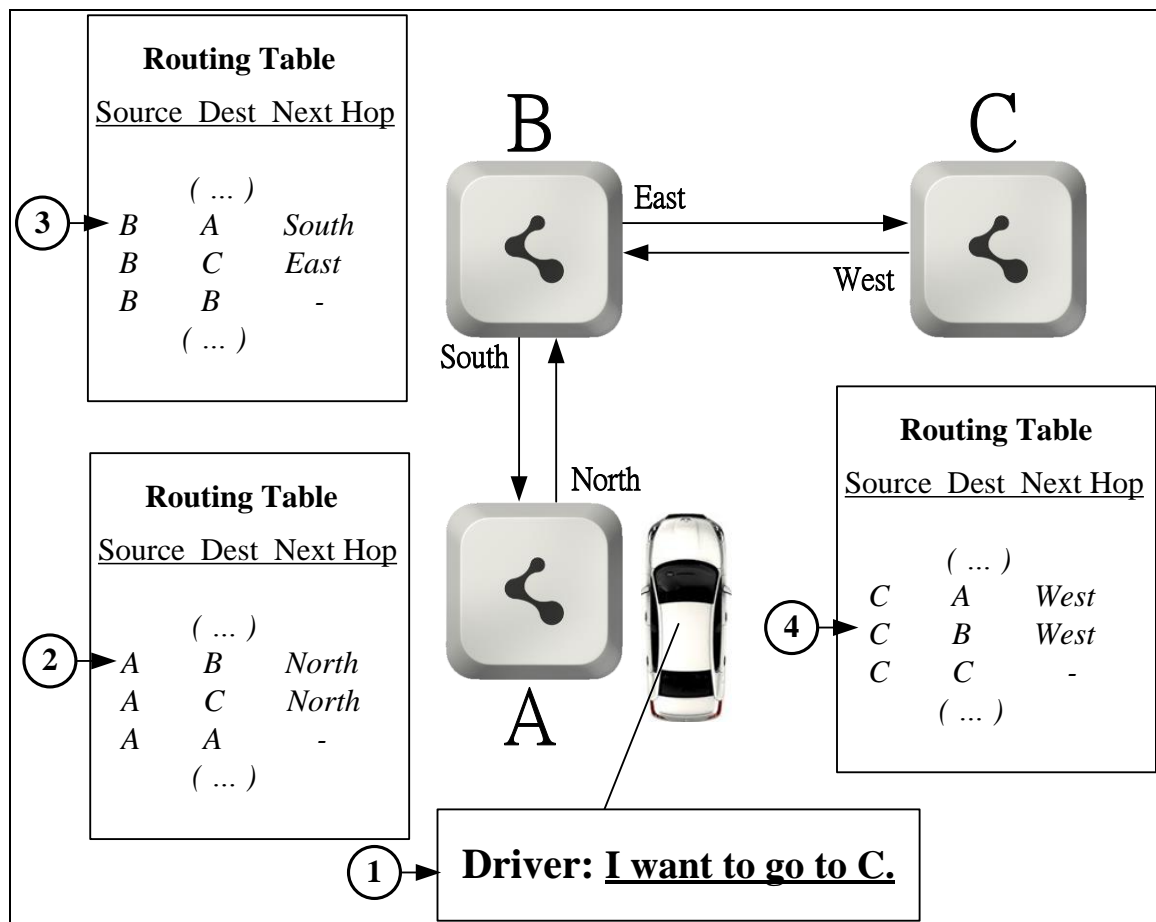


Figure 3.1.4 – Indoor guidance based on idea of next hop routing

The communication between User App and Server will be like this:

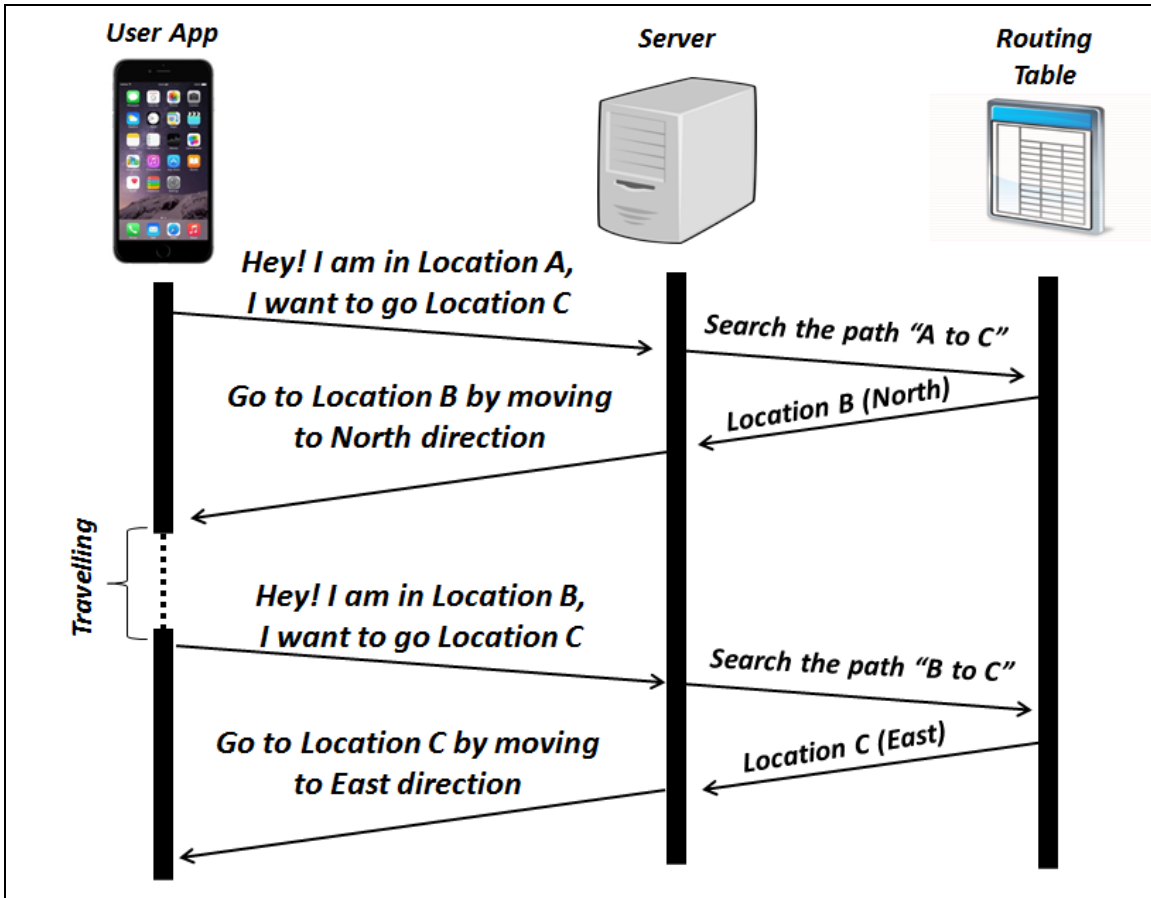


Figure 3.1.5 – Sequence Diagram of communication between User App and Server

Here, you may ask the question “What exactly is the routing table in your guidance system?”

In fact, routing table is just a table stored in database which contains some important fields like Source, Destination, and Next Hop Direction. Each entry in routing table is an answer to the question “I am at {SOURCE}, what is the fastest way to go to {DESTINATION}?”

Here is a routing table for a simple directed graph (Figure 3.1.6).

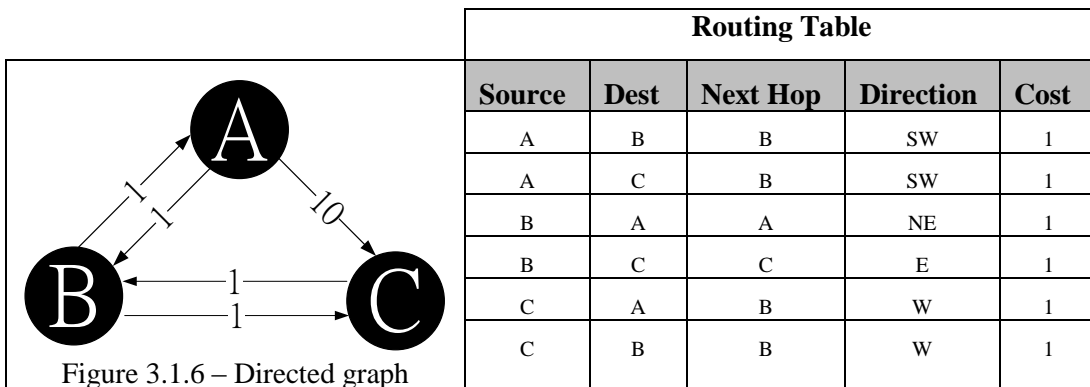


Figure 3.1.6 – Directed graph

If we are looking for the shortest path from *vertex i* to *vertex j*, we just need to look up the routing table recursively. For example, let's find the shortest path from A to C.

- i. Initially, we don't know the shortest path, so the set of shortest path only contains the source.

$$\text{Shortest Path} = \{ A \} \quad \text{Shortest Path Cost} = 0$$

- ii. We look for an entry that source is equal to A, and destination is equal to C.

$$\text{Shortest Path} = \{ A, \mathbf{B} \}$$

$$\text{Shortest Path Cost} = 0 + 1 = 1$$

Routing Table				
Source	Dest	Next Hop	Direction	Cost
A	B	B	SW	1
A	C	<b>B</b>	SW	1
B	A	A	NE	1
B	C	C	E	1
C	A	B	W	1
C	B	B	W	1

- iii. We look for an entry that source is equal to B, and destination is equal to C.

$$\text{Shortest Path} = \{ A, B, \mathbf{C} \}$$

$$\text{Shortest Path Cost} = 1 + 1 = 2$$

Routing Table				
Source	Dest	Next Hop	Direction	Cost
A	B	B	SW	1
A	C	B	SW	1
B	A	A	NE	1
B	C	<b>C</b>	E	1
C	A	B	W	1
C	B	B	W	1

- iv. Next Hop is equal to Destination, stop the recursive look up.

The complexity of recursive look up is  $O(n)$ , where  $n$  = the number entry in routing table.

And the upper-bound of  $n$  = all vertexes is connected to all other vertexes

$$= \text{Number of Vertex} \times (\text{Number of Vertex} - 1)$$

Thus, the complexity is  $O(n) = O(m^2)$

The complexity of finding shortest path is growth exponentially, which is quite slow. Therefore, we need to do indexing on the routing table in SQL server. It probably reduces the complexity from  $O(n)$  to  $O(\log n)^3$ . At the result,  $O(\log n) = O(\log m^2) = O(2 \log m)$ .

<sup>3</sup> Original statement: "With an index a SELECT is probably  $O(\log(n))$  (although it would depend on the algorithm used for indexing)", Source: <http://stackoverflow.com/questions/1347552/what-is-the-big-o-for-sql-select>

### **3.9 Project will use Eddystone or iBeacon?**

Eddystone is newly introduced by Google, Inc in July, 2015. It's can be considered as an improved version of iBeacon which owned by Apple, Inc. We've studied their standard, specification in the summer 2015. Here is the major different between Apple iBeacon and Google Eddystone.

iBeacon will broadcast their unique identifiers (UUID, major, minor) in every second. Eddystone also do the same things, but it has 2 more frame types that enable it broadcasting other information such as URL and telemetry information including battery voltage, battery life..etc.

Moreover, IOS does not allow developer to scan unknown Beacons (the UUID, major minor is not known) and the number of Beacons can be scanned at a time is limited to 20.

Both Eddystone and iBeacon is supported by both famous mobile platforms (IOS and Android). However, iBeacon in IOS has better performance since it can be scanned in background by kernel (it is in operating system level), which means the result of iBeacon scanning can be shared to multiple apps. Android cannot do this, if there are 5 apps want to get the results of Beacon scanning, then it required to scan 5 times.

Based on the comparison between iBeacon and Eddystone, we found that Eddystone is much better than iBeacon because it has fewer limitation, and more function is supported. However, Eddystone is a new thing that require some time to become more people use. By considering the backward compatibility, we decided to include both iBeacon and Eddystone in our project.

## Chapter 4 - Feasibility Study

The design of Car-park Guidance System (CGS) has 2 assumptions so it can have the highest performance.

First, the Beacons installed in car park must be detectable by smart phone within a moving vehicle. Otherwise, Car-park Guidance System cannot determine the current position of driver. So, we would like to study whether vehicle (E.G. the wind shield) will block the Beacon signal. If yes, how much signal strength (dBm) will be deduced?

Second, Wi-Fi is now everywhere in the world, the radio spectrums used by Wi-Fi are 2.4GHz and 5GHz, whereas, Beacon uses the same radio spectrum — 2.4 GHz. As far as we know, two similar frequency signals will interfere to each other. In telecommunication, this phenomenon called Intersymbol Interference (ISI). We would like to study the interference problem of Beacon and Wi-Fi if exist.

To summary, here are the 2 basic assumptions:

1. Beacons are detectable by smart phone within a moving vehicle.
2. Beacons signals will not be interfered by Wi-Fi signals.

Based on these 2 assumptions, we have done a feasibility study to see whether these 2 assumptions are reasonable and achievable.

### Assumption 1 — Beacons are detectable by smart phone within a moving vehicle

At the very beginning, we asked a question “Can smart phone detect Bluetooth Low Energy (BLE) signal within the car?” As we know, car shell is made of steel which will block and reflect electro-magnetic (EM) wave. So, we were afraid that if smart phone placed inside the car, then it cannot detect the signal sent from beacons outside the car.

First things first, we would like to know whether Beacons are detectable by smart phone within vehicle. If the answer is no, we need to give up our design. If yes, we move on to study time delay error for a moving vehicle to receive signal from a stationary device (E.G. Since the vehicle is moving, should I deploy the Beacon a little bit earlier to migrate the time delay error?). It’s important for us to determine the threshold value of RSSI, where and how to deploy the beacons which will be discussed in the later part.

### Experience on Time Delay Error

#### 1.1 Experience

In order to obtain the above data, we have conducted an experience in September 2015 — Time Delay Error Experience. In this experience, we first install a beacon on the ceiling of Lady Shaw Building (CUHK) car park. Then we placed a Beacon signal analyzer inside the car which is an Android application written by us for RSSI measurement and data logging purpose. Outside the car, there will be a guy to keep tracking the car’s position (E.G. at which check point), and the checkpoints arrival time. The overall setup of the experience is showed in the figure 4.1.1.

#### 1.2 Objective

The objective of the experience is to measure the RSSI of Beacon’s signal against displacement of a moving vehicle.

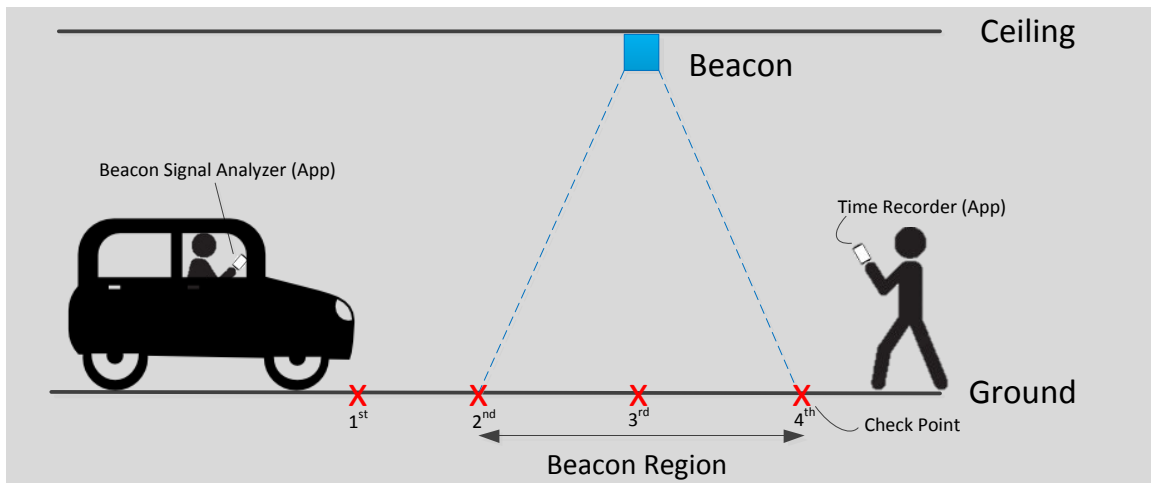
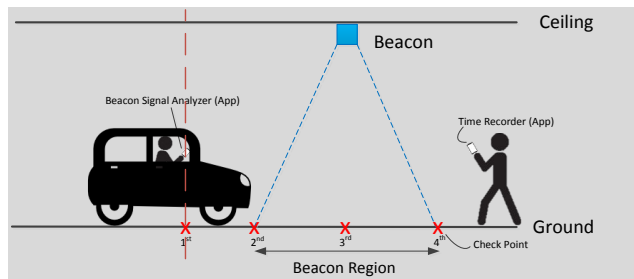


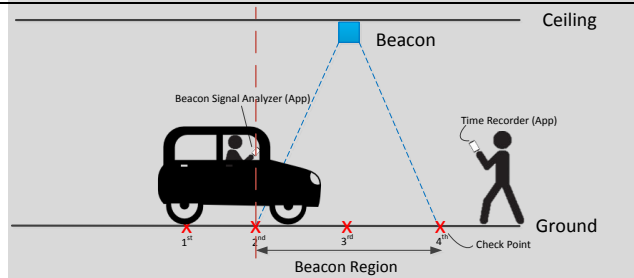
Figure 4.1.1 – The Setup of Time Delay Error Experience

### 1.3 Expected Result

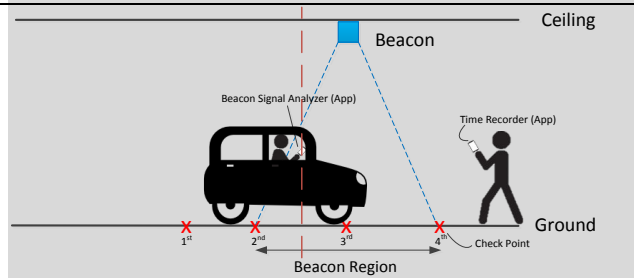
i. Before the car entering into the Beacon Region (Beacon's coverage area), the Beacon signal analyzer should detect no Beacon signal.



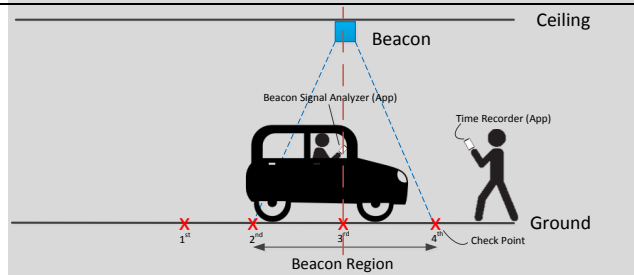
ii. After the car just entered to the Beacon Region (the 2<sup>nd</sup> checkpoint), Beacon signal analyzer will detect Beacon signal, but it's a weak signal (RSSI is low).



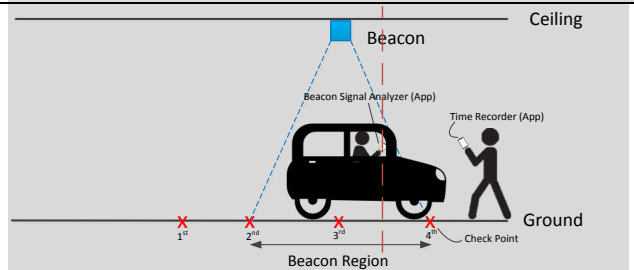
iii. Along with the car moving toward to the 3<sup>rd</sup> checkpoint (directly below the Beacon), the RSSI will become higher and higher.



iv. When the car is at 3<sup>rd</sup> checkpoint, the distance between Beacon and the car is closest, so the RSSI will be highest.



v. The car is moving continuously, along with the car moving toward to the 4<sup>th</sup> checkpoint, the RSSI will become lower and lower until it leaves the Beacon Region.



### 1.4 Logging Tools

We have written 2 mini Android apps for this experience, they are Beacon Signal Analyzer and Time Recorder. Here is the introduction of their role in this experience.

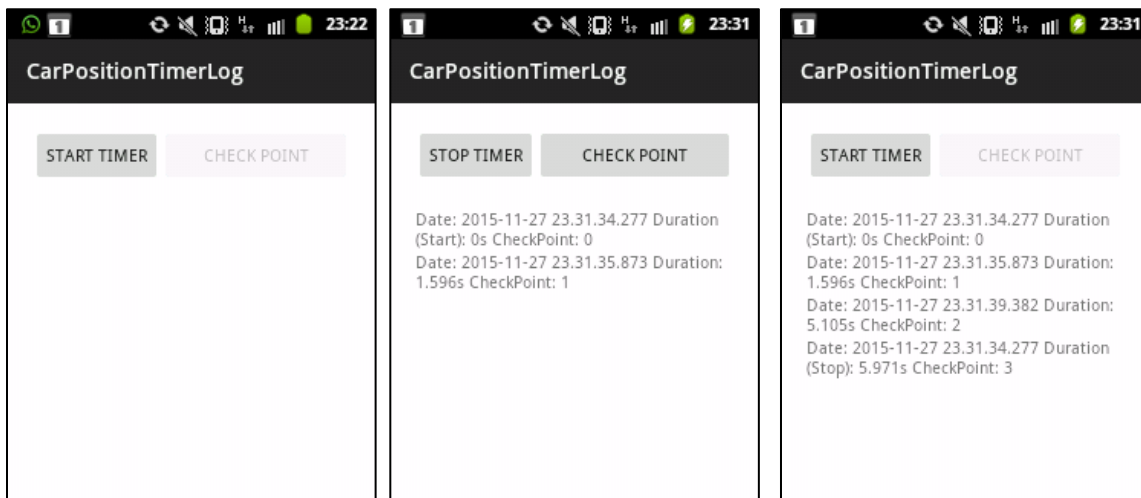
#### i. Beacon Signal Analyzer

Once this Android App started, it will scan the Beacon Signal for every 1s. It then decodes the signal, and obtains some useful information such as “Beacon Unique Identifier”, “Beacon Name”, “TX Power”, “RSSI”, and “Estimated Distance”. The above information will append to a log file with a timestamp for all detected Beacons in every 1s. Here is an **example** of output result:

Table 4.1.1 – Beacon Signal Analyzer’s output

(Timestamp)	(Beacon ID)	(RSSI)	(Estimated Distance)
2015-11-27 23:31:34:250	0x75547a41696f	-96 dBm	> 8 meters
2015-11-27 23:31:34:250	0x6c445a75696f	-50 dBm	< 1 meters
2015-11-27 23:31:35:250	0x75547a41696f	-76 dBm	~5 meters
2015-11-27 23:31:35:250	0x6c445a75696f	-64 dBm	~4 meters

#### ii. Time Recorder



Timer recorder is used to record “the car reached n<sup>th</sup> check point at what time”. Once the vehicle start moving, the guy standing outside the car will start the timer. When the car reaches the 1<sup>st</sup> check point, the guy will press “Check point” button to record the arrival time. After the car reaches the 2<sup>nd</sup> check point, the guy will press “Check point” button again to record the arrival time. These steps will keep repeating until the car reached the last check point in this experience, and then data will be stored in a log file. Here is an **example** output:



Table 4.1.2 - Time Recorder's output

(Timestamp)	(Car Position)	(Duration)
2015-11-27 23:31:34:000	Check Point 0	0s
2015-11-27 23:31:37:200	Check Point 1	3.2s
2015-11-27 23:31:41:300	Check Point 2	4.1s
2015-11-27 23:31:46:300	Check Point 3	5s

From the above 2 tables (Table 4.1.1 & Table 4.1.2), we can plot a “RSSI against car’s position graph” and see whether it match out expected result. Let’s see the real experience data.

### 1.5 Actual Result

We invited CUHK ViewLab Technical Manager — Edward Yau to be our driver in this experience. We have done the experience in 2<sup>nd</sup> floor of CUHK Lady Shaw Building’s car park. The checkpoints and Beacon is marked in the floor plan below (Figure 4.1.2).

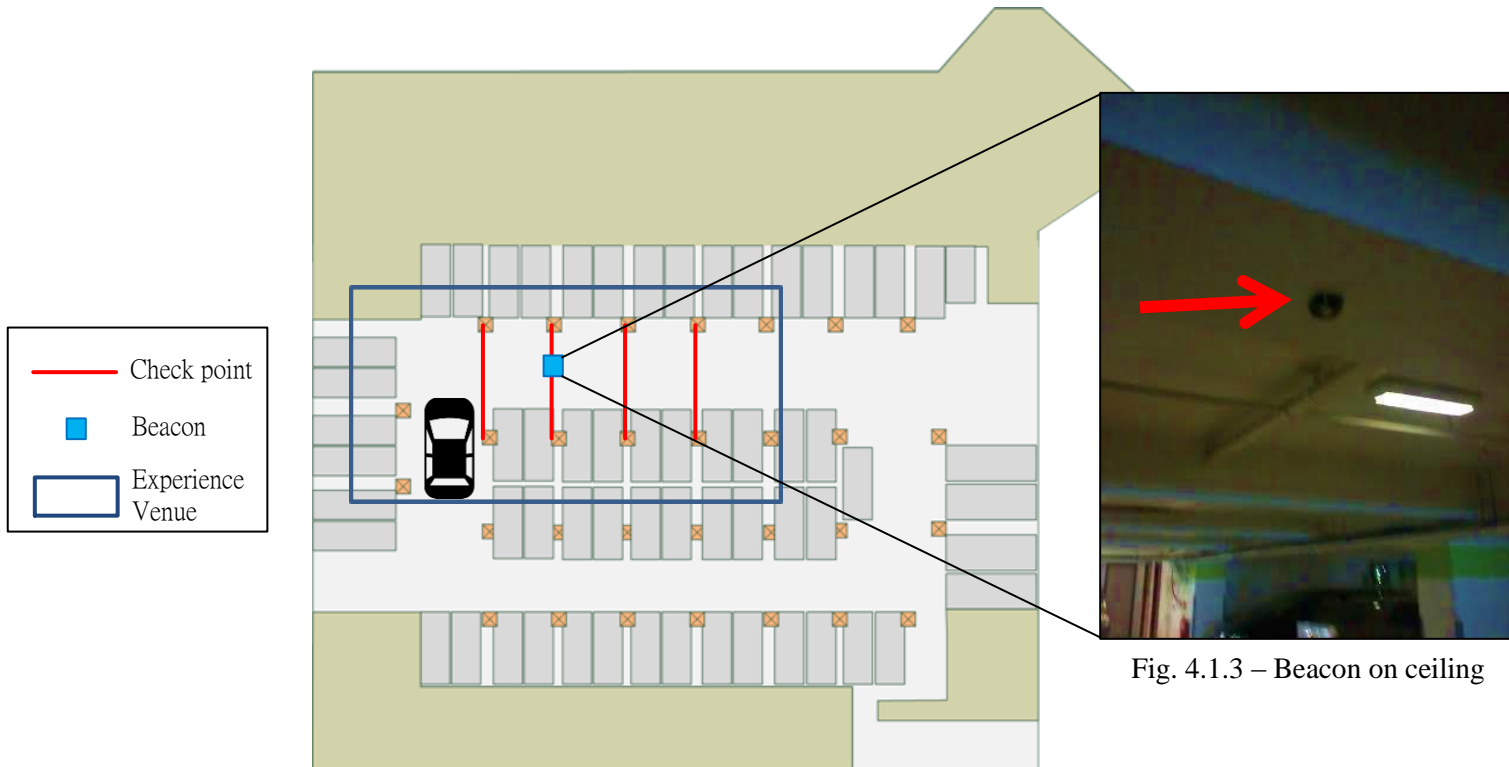


Fig. 4.1.3 – Beacon on ceiling

Figure 4.1.2 – Floor plan of 2<sup>nd</sup> floor of LSB car park

The Beacon is about 2.2 meters from ground, and Android smartphone is placed in car near to the steering wheel (Figure 4.1.4) which is about 1.2 meters from ground. The car moves from check

point 0 to check point 3. The total path length is 18.1 meters long. Beacon Signal Analyzer and Time Recorder start recording when the car moving from check point 0 to check point 3.

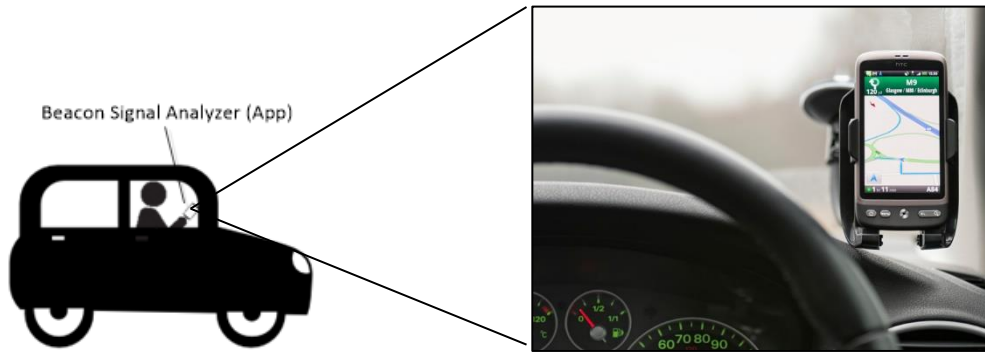


Figure 4.1.4 - Android smartphone is placed in car near to the steering wheel<sup>4</sup>

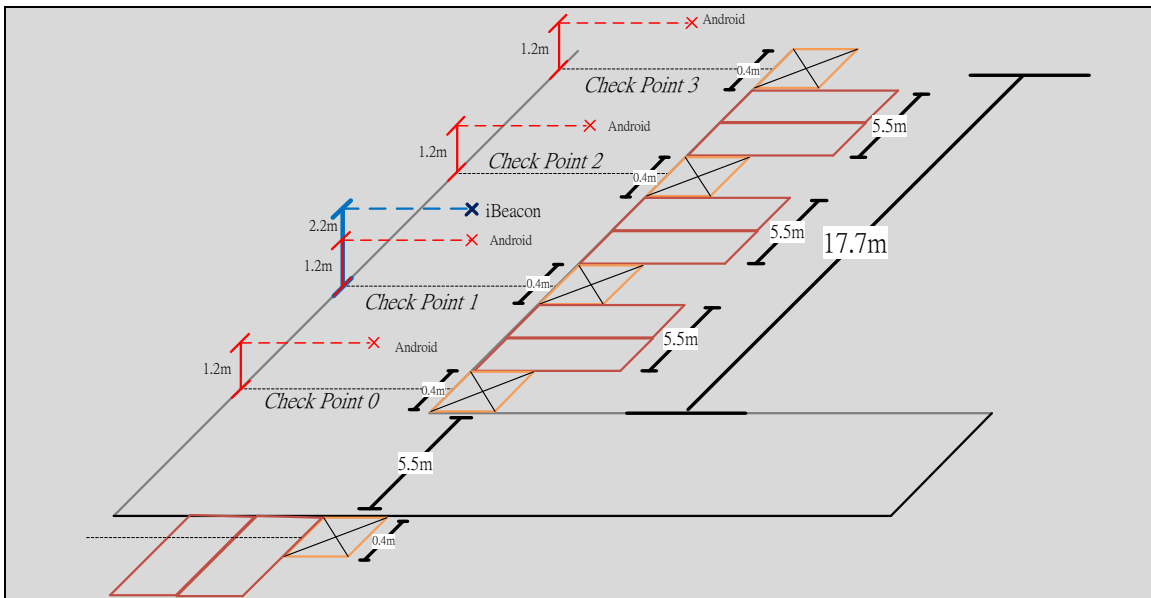


Figure 4.1.5 – Isometric View of Experience venue

<sup>4</sup> Image source: <http://www.androidcentral.com/ntsb-recommends-states-ban-all-cell-phone-use-cars>

The experience has repeated for 2 times, the detailed result is as below.

### *1<sup>st</sup> Trial - Beacon Signal Analyzer Log*

```
Date: 2015-09-25 16.38.05.074 Duration (Start): 0s null
Date: 2015-09-25 16.38.08.492 Duration: 3.418s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -95 Distance : 12.07 meters away.

Date: 2015-09-25 16.38.09.610 Duration: 4.536s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -91 Distance : 9.09 meters away.

Date: 2015-09-25 16.38.10.750 Duration: 5.676s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -75 Distance : 4.69 meters away.

Date: 2015-09-25 16.38.11.885 Duration: 6.811s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -79 Distance : 3.99 meters away.

Date: 2015-09-25 16.38.12.983 Duration: 7.909s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -87 Distance : 4.78 meters away.

Date: 2015-09-25 16.38.14.101 Duration: 9.027s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -92 Distance : 5.56 meters away.

Date: 2015-09-25 16.38.05.074 Duration (Stop): 14.469 Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-
8024-bc5b71e0893e major : 19857 minor : 60946 RSSI : -92 Distance : 5.56 meters away.
```

### *1<sup>st</sup> Trial – Time Recorder Log*

```
Date: 2015-09-25 16.38.06.569 Duration (Start): 0s CheckPoint: 0
Date: 2015-09-25 16.38.08.099 Duration: 1.530s CheckPoint: 1
Date: 2015-09-25 16.38.10.026 Duration: 3.457s CheckPoint: 2
Date: 2015-09-25 16.38.11.857 Duration: 5.288s CheckPoint: 3
Date: 2015-09-25 16.38.13.454 Duration: 6.885s CheckPoint: 4
Date: 2015-09-25 16.38.05.522 Duration (Stop): 9.973s CheckPoint: 5
```

### *2<sup>nd</sup> Trial – Beacon Signal Analyzer Log*

```
Date: 2015-09-25 16.38.39.794 Duration (Start): 0s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-
8024-bc5b71e0893e major : 19857 minor : 60946 RSSI : -96 Distance : 10.99 meters away.

Date: 2015-09-25 16.38.42.531 Duration: 2.737s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -93 Distance : 10.74 meters away.

Date: 2015-09-25 16.38.44.729 Duration: 4.935s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -84 Distance : 10.17 meters away.

Date: 2015-09-25 16.38.45.854 Duration: 6.06s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -71 Distance : 7.59 meters away.

Date: 2015-09-25 16.38.46.967 Duration: 7.173s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -84 Distance : 6.32 meters away.

Date: 2015-09-25 16.38.48.078 Duration: 8.284s Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-8024-
bc5b71e0893e major : 19857 minor : 60946 RSSI : -88 Distance : 5.92 meters away.

Date: 2015-09-25 16.38.39.794 Duration (Stop): 10.9 Detect Beacon : 1 uuid1 : f7826da6-4fa2-4e98-
8024-bc5b71e0893e major : 19857 minor : 60946 RSSI : -88 Distance : 5.92 meters away.
```

### *2<sup>nd</sup> Trial – Time Recorder Log*

```
Date: 2015-09-25 16.38.41.142 Duration (Start): 0s CheckPoint: 0
Date: 2015-09-25 16.38.43.584 Duration: 2.442s CheckPoint: 1
Date: 2015-09-25 16.38.45.494 Duration: 4.352s CheckPoint: 2
Date: 2015-09-25 16.38.47.090 Duration: 5.948s CheckPoint: 3
Date: 2015-09-25 16.38.48.684 Duration: 7.542s CheckPoint: 4
Date: 2015-09-25 16.38.38.375 Duration (Stop): 9.438s CheckPoint: 5
```

Here we summarized the data into a figures and graph which can be comparable with our expected result and the result of controlled experience.

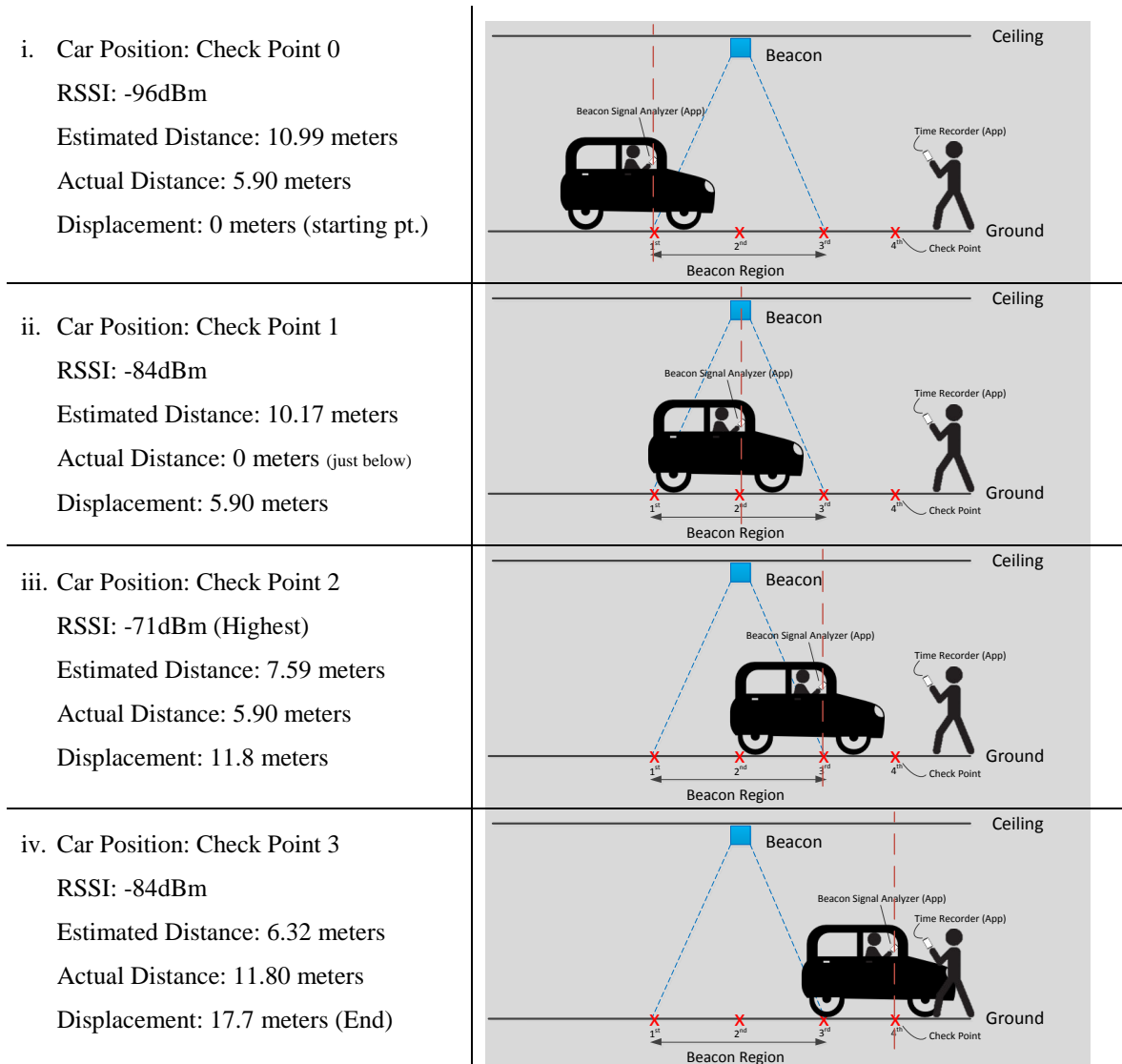


Table 4.1.3 – Experience Result

(Timestamp)	(Car Position)	(Displacement from Starting Point)	(RSSI)	(Estimated Distance from Beacon to Car)	(Actual Distance from Beacon to Car)
2015-09-25 16.38.41.142	Check Point 0	0 meter	-96 dBm	10.99 meters	5.90 meters
2015-09-25 16.38.43.584	Check Point 1 (Beacon is here)	5.90 meters	-84 dBm	5.9 meters	0 meters
2015-09-25 16.38.45.494	Check Point 2	11.80 meters	-71 dBm	7.59 meters	5.90 meters
2015-09-25 16.38.47.090	Check Point 3	17.7 meters	-84 dBm	6.32 meters	11.80 meters

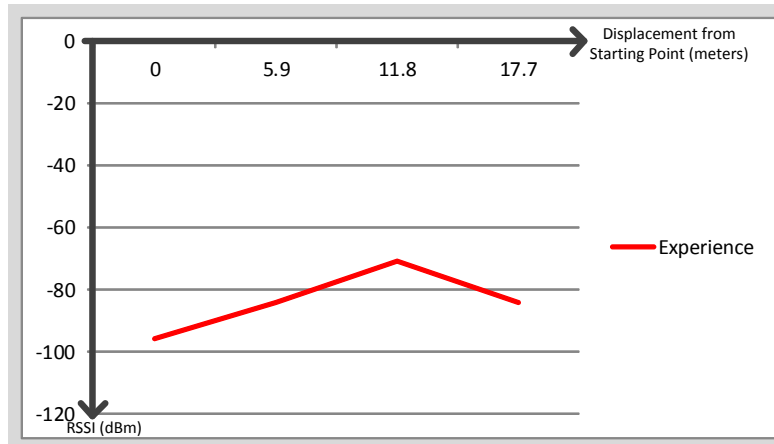


Figure 4.1.6 – RSSI against Displacement graph (Experience Result)

### 1.6 Controlled Experience

We also designed a controlled experience, where the smart phone is not moving continually, but vice visa, it will stop at each check point for at least 5 seconds. Also, the smart phone no longer inside the vehicle, but held by a guy instead. The result of controlled experience should exactly match with our expected result (in section 1.3), because there is no time delay error due to the car motion. By comparing with controlled experience’s result, we find out how big is the time delay error, and how should we deploy the Beacon to migrate the error. Here we summarized the data into a table 4.1.3 and figure 4.1.7.

(Timestamp)	(Car Position)	(Displacement from Starting Point)	(RSSI)	(Estimated Distance from Beacon to Car)	(Actual Distance from Beacon to Car)
2015-09-25 16.39.01.133	Check Point 0	0 meter	-80 dBm	7.27 meters	5.90 meters
2015-09-25 16.39.10.627	Check Point 1 (Beacon is here)	5.90 meters	-61 dBm	4.92 meters	0 meters
2015-09-25 16.39.14.044	Check Point 2	11.80 meters	-81 dBm	3.4 meters	5.90 meters
2015-09-25 16.39.16.250	Check Point 3	17.7 meters	-88 dBm	5.02 meters	11.80 meters

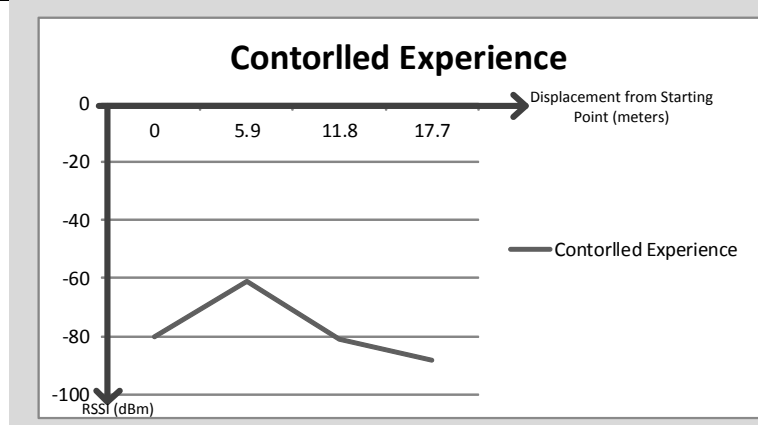


Figure 4.1.7 – RSSI against Displacement graph (Controlled Experience Result)

### 1.7 Conclusion to the experience

Let's combine the result of experience and controlled experience into same graph (figure 1.8). It's not difficult to see the Time Delay Error, and Signal Dissipation. Time Delay Error is 5.9 meters of RSSI, and signal dissipation due to vehicle shell is about 10 dBm.

Note that by equation,

$$Velocity = \frac{Displacement}{Time}$$

$$Velocity\ of\ vehicle = \frac{17.7\ meters}{16.250 - 1.133s} = 4.215km/Hour$$

In other words, if the vehicle moves with velocity 4.23 km/H, then we need to deploy our Beacon 5.9 meters earlier to original place to migrate the Time Delay Error.

We use RSSI threshold to determine whether the vehicle has reached the location or not, if the RSSI over the threshold value, then we can say the vehicle has reached the location. Based on the finding, signal dissipation due to vehicle's metal shell is about 10 dBm. So we need to change the RSSI threshold value to -75 dBm instead of -60 dBm.

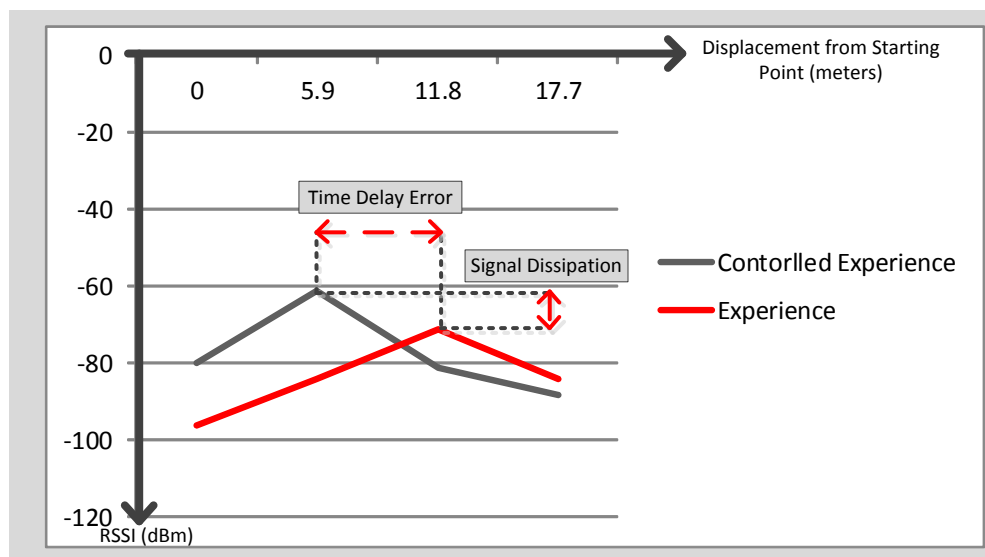


Figure 4.1.8 – RSSI against Displacement graph

## Assumption 2 — Beacons signals will not be interfered by WiFi signals

Both WiFi and BLE devices operate in 2.4 GHz license-free band, and broadcast their identifiers. We would like to study the frequency spectrum used by Beacon and WiFi, and see if there is any overlapping which causes Inter-Symbol Interference. Refer to a research paper — “*An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications*” by R. Faragher and R. Harle, University of Cambridge, UK. The paper includes the study on channel bandwidth of WiFi and Beacon. It stated that “The BLE advertisement channels are nominally labeled 37, 38, and 39 and are centered on 2402 MHz, 2426 MHz and 2480 MHz, respectively (see Figure 2)”<sup>5</sup>, these spectrums are called Frequency Hopping Spread Spectrum (FHSS) which minimize or even eliminate the channel overlapping problem.

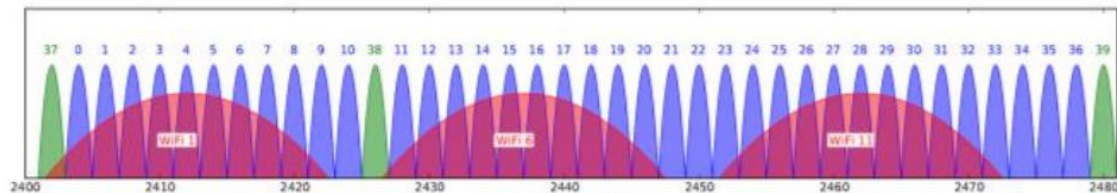


Figure 2 shows the 40 BLE channels within the 2.4 GHz band. The green channels are the advertising channels used by BLE beacons. Three WiFi channels are shown for comparison (red).

According to an article written by the manufacturer of Beacon, it also stated that “if WiFi is configured to use channel 1, 6, 11, then there is no interference problem between WiFi and Beacon”<sup>6</sup>

In telecommunication, “channel 1, 6, 11” is a very famous and common radio setting for WiFi 2.4 GHz spectrum, because it maximizes the spectrum utilization for 2.4GHz spectrum which allow 3 non-overlapping 22MHz channels exist at the same time. However, Beacon uses channel 37, 38, 39 to advertising their identifiers. So we can safely claim that Beacons signals will not be interfered by WiFi signals.

<sup>5</sup> “*An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications*” by R. Faragher, University of Cambridge, UK and R. Harle, University of Cambridge, UK.

<sup>6</sup> “*Will wireless interference and Wi-Fi impact beacons?*” by Estimote, manufacturer of Beacon (Available on: <https://community.estimote.com/hc/en-us/articles/200794267>)

## Chapter 5 - Design Specification

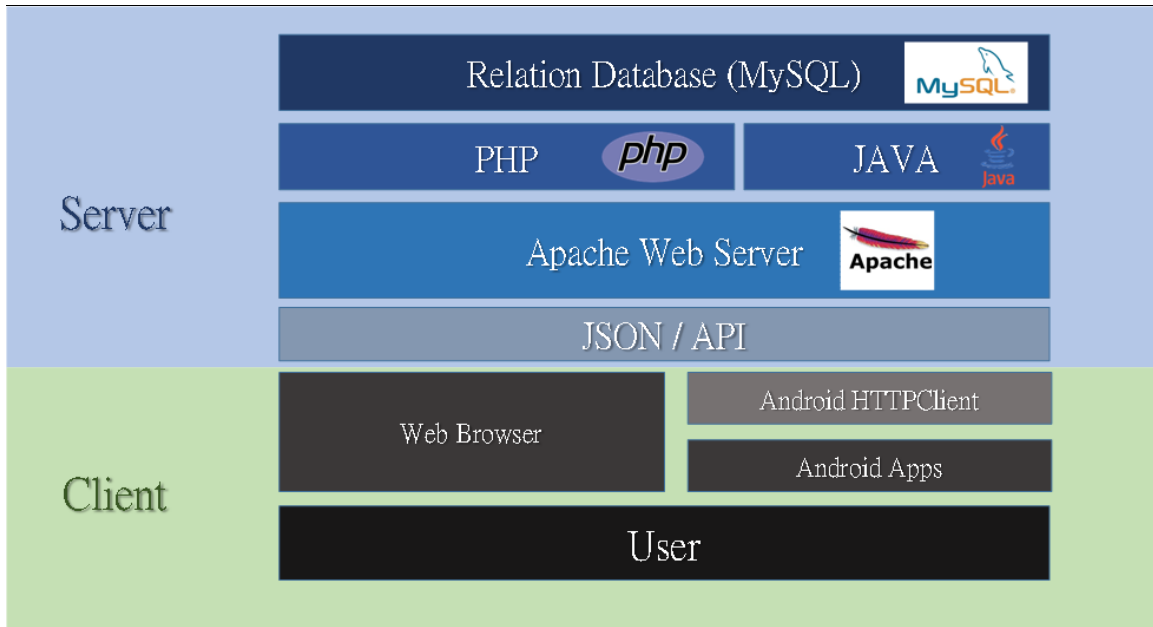


Figure 5.1.1 - Proposed Hierarchical Model View of Car-park Guidance System

Car-park Guidance System (CGS) can be divided into 2 parts, they are Android App (Front-end) and Web-based Content Management System (Back-end). The functionality of Android App and Content Management System (CMS) are different. For the security reason, App will not directly deal with Database Server, whereas App will through Application Programming Interface (API) provided by CMS to retrieve data from Database Server. The figure 5.1.1 has shown the proposed hierarchical model view of CGS where “Server” is the CMS of our system.



## 5.1 Content Management System (Back-end)

Content Management System (CMS) is a web based system which allows car park administrators to access to CMS on different platforms. CMS can be further divided into smaller parts:

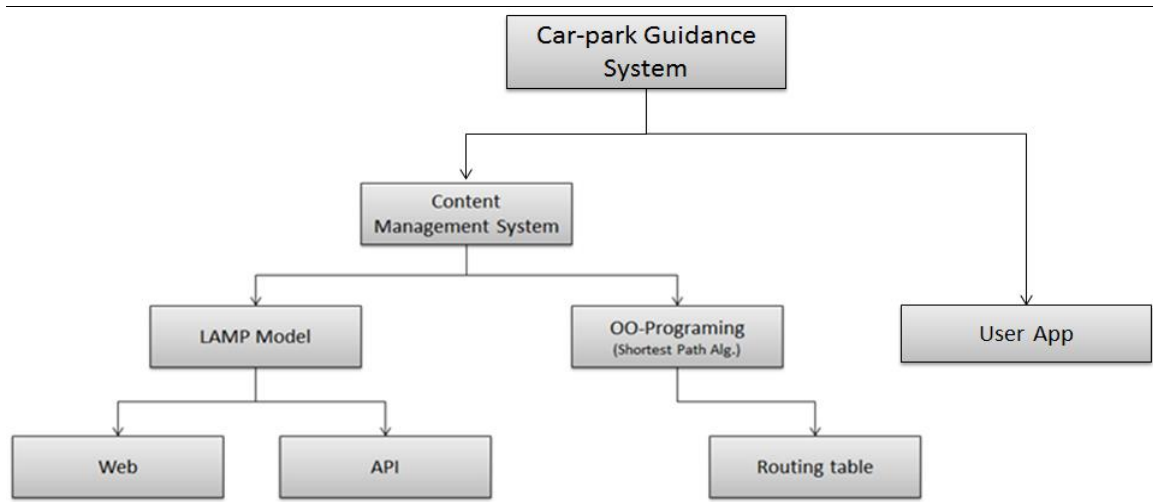


Figure 5.1.2 – Architecture of Content Management System

We are following LAMP to develop the web-based content management system. LAMP “is an archetypal model of web service solution stacks which suitable for building dynamic web sites and web applications”<sup>7</sup>. LAMP invokes 4 components, they are Linux kernel, web server, CGI scripting, and database. As showed in figure 5.1.1, we will use Linux as the operating system, Apache Web Server as web server component, MySQL as database component, PHP as CGI scripting component. That is, the true meaning of LAMP (Linux, Apache, MySQL, PHP).

From figure 5.1.2, you can see below the LAMP, there are Web and API. The first thing we will talk about is Web. Car-park administrator can manage their car-parks through the web interface. The web interface is not used by end-user (E.G. drivers). As mentioned in design overview. Car-park administrator is able to do the following through the web interface of our system.

1. Car-park Management
2. Car-park’s Beacon Management
3. Car-park’s Routing Management
4. User Account Management
5. View Log History
6. Test API’s status

## 5.3 Web’s User Interface (Draft version in the summer 2015)

We’ve started to build our Content Management System (CMS) from August, 2015. The first draft of the web site is showed in figure 5.1.3. Here we will use “CMS\_v1” to represent this draft version of CMS created in the summer 2015.

<sup>7</sup> Source: [https://en.wikipedia.org/wiki/LAMP\\_\(software\\_bundle\)](https://en.wikipedia.org/wiki/LAMP_(software_bundle))

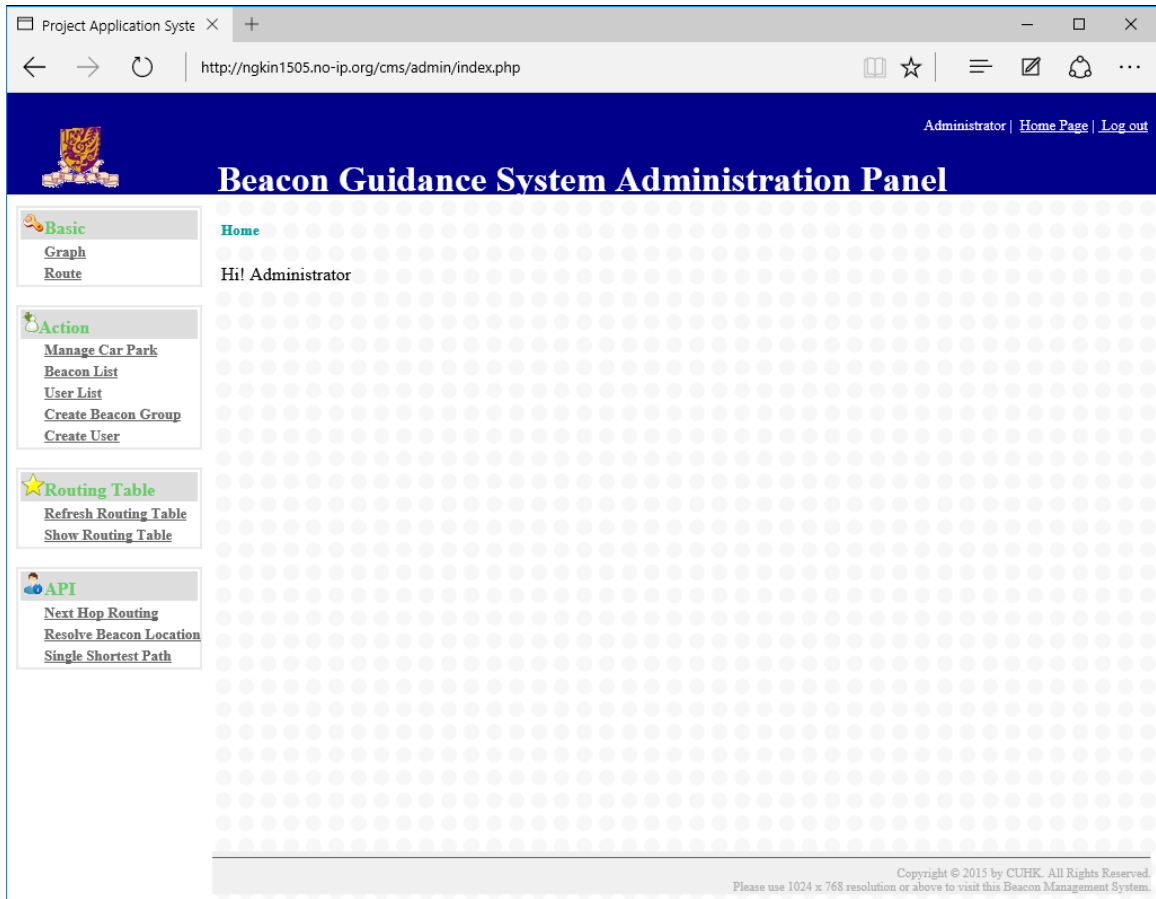


Figure 5.1.3 – Home Page of CMS\_v1’s web site

For the first version of CMS’s website, you can see from figure 5.1.3, the layout is clearly divided into 3 frames. They are top frame, left frame, and main frame. The optimal resolution of CMS\_v1 is 1024px \* 768px. As you can see, there is a notice — “*Please use 1024 x 768 resolutions or above to visit this Beacon Management System*” at the bottom in figure 1.3. Top frame is 1024px \* 90px, left frame is 200px \* 768px, and main frame is 824px \* 768px. You can find the information about the resolution of CMS\_1’s web in figure 5.1.4 and figure 5.1.5.

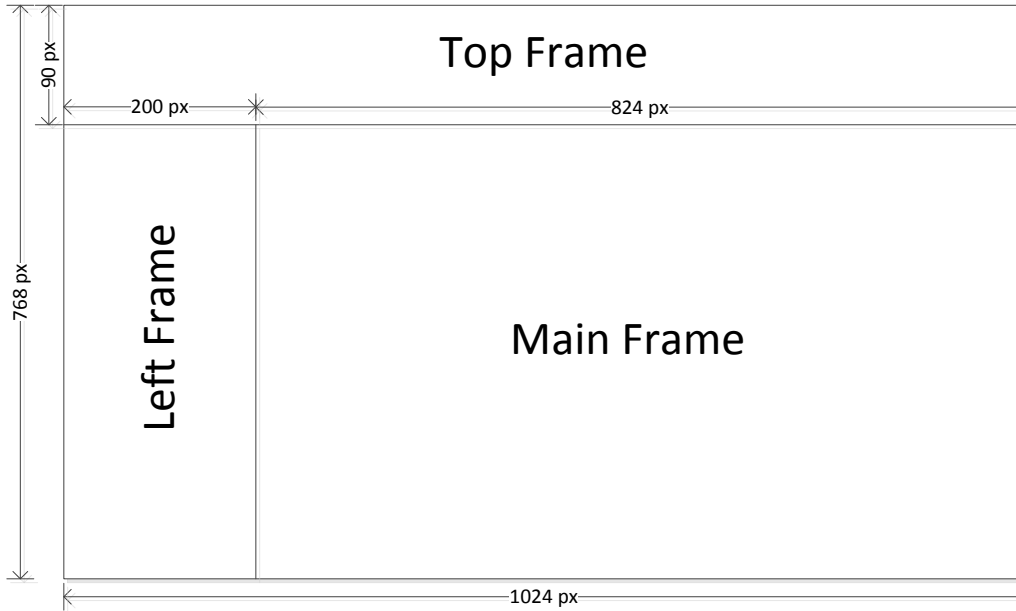


Figure 5.1.4 – Layout of CMS\_v1's web site

```

9 <frameset rows="90,*" cols="1080">
10 <frame src="admin_topFrame.php" name="topFrame" frameborder="0" scrolling="auto" noresize="noresize" title="topFrame" />
11 <frameset rows="*" cols="200,930">
12 <frame src="admin_leftFrame.html" name="leftFrame" frameborder="0" scrolling="auto" noresize="noresize" title="leftFrame" />
13 <frame src="admin_mainFrame.php" name="mainFrame" frameborder="0" scrolling="auto" noresize="noresize" title="leftFrame" />
14 </frameset>
15 <noframes><body></body>
16 </noframes></frameset>

```

Figure 5.1.5 – The HTML code of CMS\_v1's frame layout

The main purpose of each frame is:

- *Top Frame:* Display the banner of CMS
- *Left Frame:* Show the menu of CMS, can change the content of *Main Frame*
- *Main Frame:* Show the content to user.

From the *Left Frame*, user can go to other pages, and the page will be displayed on *Main Frame*. Here is the list of pages that user can find in *Left Frame*:

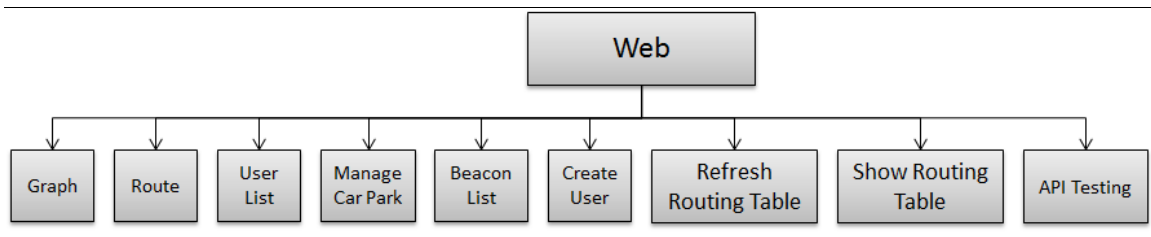


Figure 1.6 – Site Map of CMS\_v1's web site

The first version of our CMS web site is a static website. The meaning of static website is the content of web page cannot be changed without reloading the page. For example, figure 1.6 has

shown the list of Beacons' information. If you want to sort or filter the data within the HTML table, then you need to reload the whole page with special request (E.G. reload the page with PHP arguments such as *index.php?action=sort*).

Although the web site is not dynamic, but it used CSS and JavaScript to make it look better. Let's look at an example in figure 1.8, when user is filling a HTML form, if user has inputted some invalid input, there will be a warning icon (✘) next to the input field immediately. Vice versa, if user has inputted some data which fulfill the requirement, it shows a "tick" icon (✔) next to the input field.

The screenshot shows a web browser window with the URL `appsrv.cse.cuhk.edu.hk/~tkwong4/cms/admin/index.php`. The page title is "Beacon Guidance System Administration Panel". The breadcrumb trail is "Home > Action > Create Beacon". The main content is a "Beacon List" table with 13 rows. Each row contains columns for BID, Type, UUID/EID, Major, Minor, BName, BDesc, CPID, and Action. The "Action" column contains a "Remove" link for each entry.

BID	Type	UUID/EID	Major	Minor	BName	BDesc	CPID	Action
1	B	f7826da6-4fa2-4e98-8024-bc5b71e0893e	44703	47796	C4zL (Entrance)	Real Beacon Lady Shaw Building	11	Remove
2	B	f7826da6-4fa2-4e98-8024-bc5b71e0893e	19857	60946	4xPj	Real Beacon Lady Shaw Building	11	Remove
3	B	f7826da6-4fa2-4e98-8024-bc5b71e0893e	5161	41406	kNVT(Charger)	Real Beacon Lady Shaw Building	11	Remove
4	E	0x75547a41696f	0	0	uTzA	Real Beacon Lady Shaw Building	11	Remove
5	B	ffffffff-ffff-ffff-ffffffff	9999	0	Fake Beacon 0 (Toilet)	Fake Beacon in Lady Shaw Building	11	Remove
6	E	0x6c445a75696f	0	0	IDZu	Real Beacon Lady Shaw Building	11	Remove
7	E	0x6b584b4f696f	0	0	kXKO	Real Beacon Lady Shaw Building	11	Remove
8	E	0x4f6f7834696f	0	0	Oox4	Real Beacon in Lady Shaw Building	11	Remove
9	B	ffffffff-ffff-ffff-ffffffff	9999	9999	Fake Beacon 1	Fake Beacon in Lady Shaw Building	11	Remove
10	E	0x45543459696f	0	0	ET4Y	Real Beacon Lady Shaw Building	11	Remove
11	B	ffffffff-ffff-ffff-ffffffff	9999	2	Fake Beacon 2	Fake Beacon in Lady Shaw Building	11	Remove
12	E	0x324c727a696f	0	0	2Lrz(Lift)	Real Beacon Lady Shaw Building	11	Remove
13	B	ffffffff-ffff-ffff-ffffffff	9999	3	Fake Beacon 3	Fake Beacon in Lady Shaw Building	11	Remove

Figure 1.7 – CMS\_v1 Web, List of Beacons' information

The screenshot shows a "Create User" form with two input fields. The "Login Name" field contains the text "Administrator" and has a green checkmark icon to its right, indicating valid input. Below the field is the text "#Not more than 20 Characters.". The "Login Password" field contains several dots and has a red X icon to its right, indicating invalid input. Below the field is the text "#Not more than 8 Characters:".

Figure 1.8 – Example of valid input and invalid input in HTML form

The first version of CMS is finished on September 2015. As a draft version of CMS, the user interface is quite ugly. Especially, it is a static website which is not interactive to user. However, we were not focusing on the user interface of CMS at that time. Instead, we were concerning the functionality of CMS only. After the November 2015, the overall functionality become stable, so

we decided to re-design the web of CMS to be a dynamic website. That's the second version of CMS (figure 1.9).

#### 5.4 Web's User Interface (2<sup>nd</sup> version)

The functionality of 2<sup>nd</sup> version CMS is exactly as same as the 1<sup>st</sup> version we discussed above. The only different is we re-designed the 1<sup>st</sup> version of web to be dynamic website and it now look better.

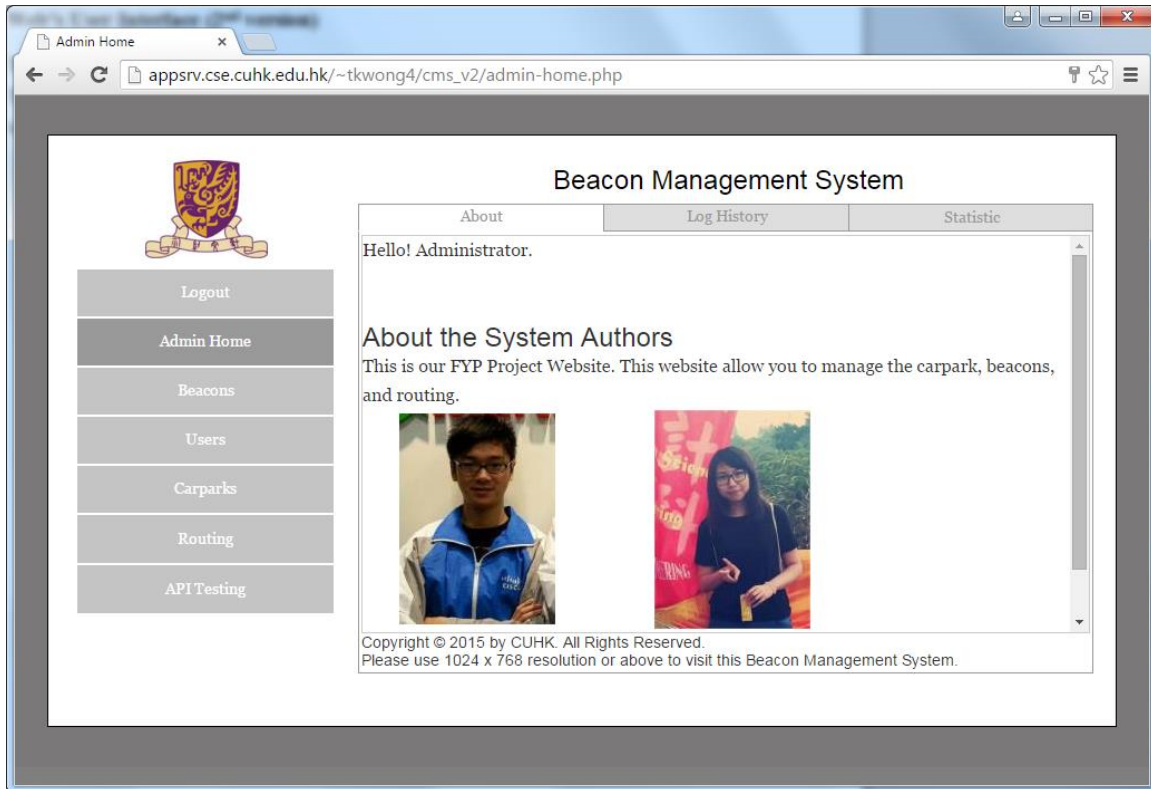


Figure 1.9 – The home page of 2<sup>nd</sup> of CMS

Some characteristics are similar to the first version, for example, the optimal resolution is still 1024 x 768. And the menu is still on the left hand side. The different is, we removed the top frame, and the main frame replaced by a tabbed panel. Figure 1.10 has showed the user interface layout of 2<sup>nd</sup> version CMS.

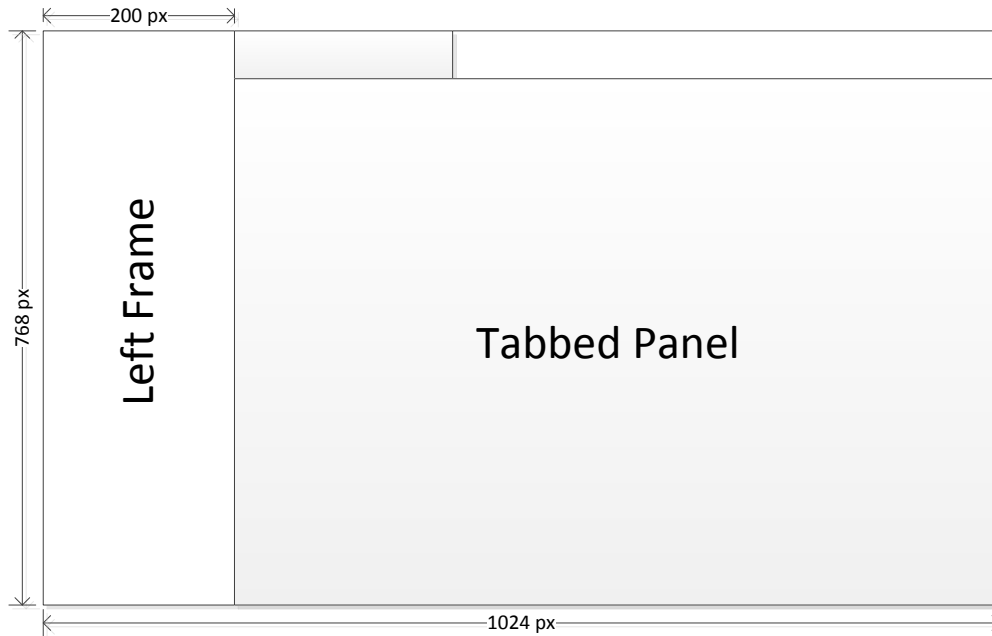


Figure 1.10 – Layout of CMS’s web site

In the following parts, we will discuss the user interface of managing car parks, beacons, and routing between beacons.

### 1. Managing Car Park

- i. Add a new Car Park to our system.

<p style="text-align: center;"><b>Beacon Management System</b></p> <p style="text-align: center;">Carpark List    Add Carpark    Add Floorplan</p> <p style="background-color: #4F81BD; color: white; padding: 2px;">Add New Carpark</p> <p><b>Name</b>    <input type="text" value="Carpark Name (E.G. Lok Fu Carpark)"/></p> <p><b>Detail</b>    <input type="text" value="Description of the carpark"/></p> <p><b>LB Lat</b>    <input type="text" value="Left-Bottom Latitude"/></p> <p><b>LB Long</b>    <input type="text" value="Left-Bottom Longitude"/></p> <p><b>RT Lat</b>    <input type="text" value="Right-Top Latitude"/></p>	<p>First, administrator need to enter the information about the Car park to CMS. For example, the name and physical location of car park.</p>
--	---

Figure 1.11(a) – Adding new Car Park

Figure 1.11(b) – Adding new Car Park Floor

After the car park is added, administrator can upload the floorplan for each floor of car park to CMS in this page.

## 2. Managing Beacon

- i. Add a new Beacon to our system.

Figure 1.12 (a) – Adding new Beacon

First, administrator need to enter the information about the Beacon to be added to CMS. For example, if it is an iBeacon, then it required to enter UUID, major, minor.

Figure 1.12 (b) – Adding new Beacon

Second, administrator need to select where to deploy the Beacon. The information including:

- a) Car Park
- b) Car Park Floor
- c) Latitude and Longitude

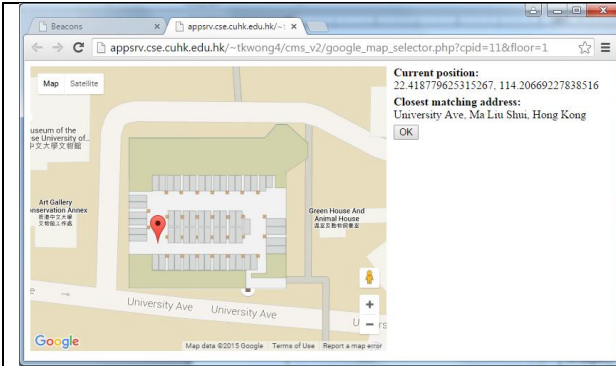


Figure 1.12 (c) – Map Selector

However, if we require administrator to input latitude and longitude is very inconvenient. So, we provided a *Map Selector* which allows administrator select a point on map.

On the *Map Selector*, the floor plan of car park is overlaid on *Google Map*. After the administrator clicked on *Google Map*, a marker will show up and display the correspondence latitude and longitude on right hand side.

Click ok to confirm the location. And the latitude and longitude will automatically input in the input field. Click “Add” button and the beacon will be added into CMS.

Figure 1.12 (d) – Adding new Beacon

ii. View or Delete a Beacon from CMS

BID	Type	UUID/EID	Major	Minor	BName	BDesc	CPID	Act
1	B	f7826da6-4fa2-4e98-8024-bc7b71e0893e	44703	47796	C4zL (Entrance)	Real Beacon Lady Shaw Building	11	<a href="#">Remove</a>
2	B	f7826da6-4fa2-4e98-8024-bc7b71e0893e	19857	60946	4xPj	Real Beacon Lady Shaw Building	11	<a href="#">Remove</a>
3	B	f7826da6-4fa2-4e98-8024-bc7b71e0893e	5161	41406	kNVT(Charger)	Real Beacon Lady Shaw Building	11	<a href="#">Remove</a>

Figure 1.13 (a) – View all Beacons in CMS

Administrator can view all Beacons in the CMS from this page. Because of JQuery, administrator can simply click on the one of the column to sort the table.

The table also supports paging, which means administrator can select how many entries will show up in a page (E.G. 10 entries per page is selected in Fig 1.13a).



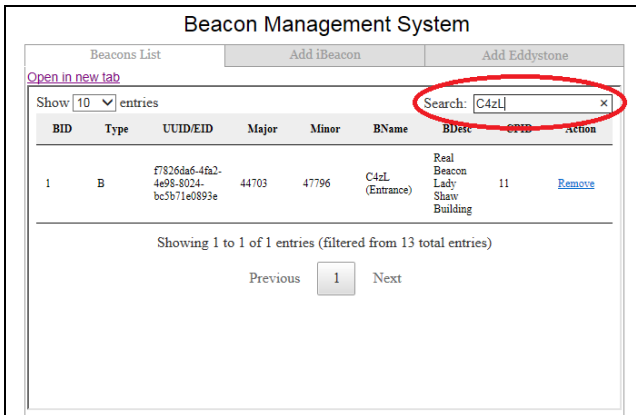


Figure 1.13 (b) – Searching a Beacon from list

If Administrator wants to check a particular Beacon, administrator can use the search function in the circled area.

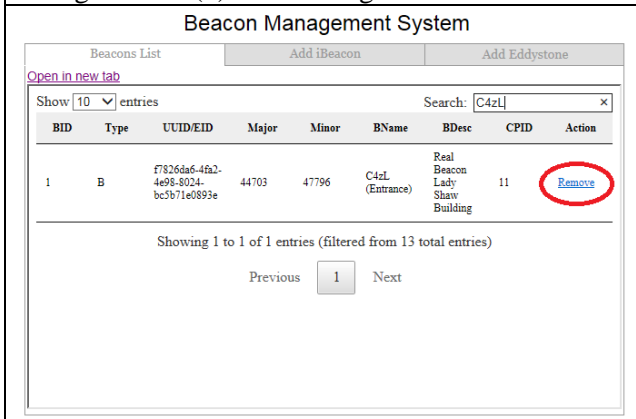


Figure 1.13 (c) – Removing a Beacon from list

If Administrator wants to delete the particular Beacon, administrator can simply click on the “Remove” hyperlink in the circled area. Then the Beacon and all edges of graph related to that Beacon will be deleted.

### 3. Managing Routing between Beacons

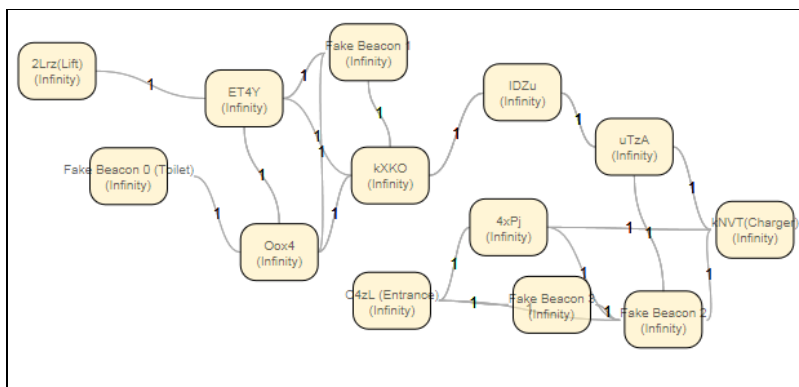


Figure 1.14 – Example of Topology of Beacon Network

In the design overview, we have discussed the role of *Routing Table* in our system. Here we will show how an administrator can modify the content of *Routing Table*. In fact, we don't allow administrator change the *routing table* directly, but change the *graph* (or called “topology of Beacon network” — Figure 1.14) instead which will be discussed in later part.

i. Adding or Deleting an edge to graph (topology of Beacon network)

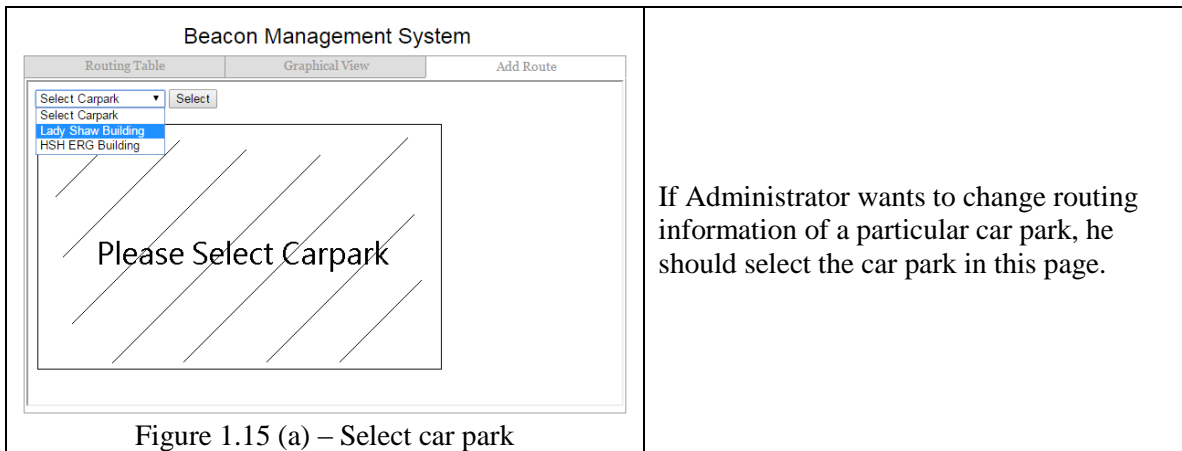


Figure 1.15 (a) – Select car park

If Administrator wants to change routing information of a particular car park, he should select the car park in this page.

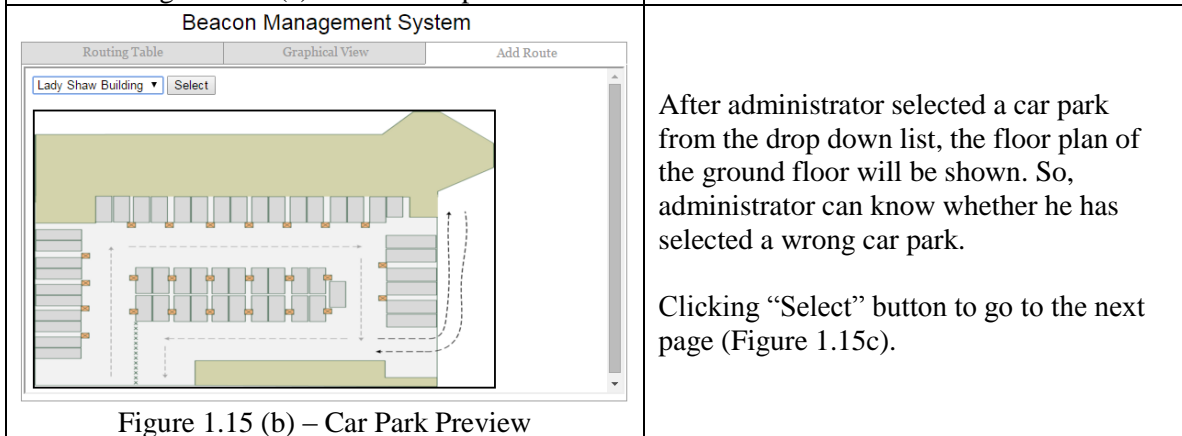


Figure 1.15 (b) – Car Park Preview

After administrator selected a car park from the drop down list, the floor plan of the ground floor will be shown. So, administrator can know whether he has selected a wrong car park.

Clicking “Select” button to go to the next page (Figure 1.15c).

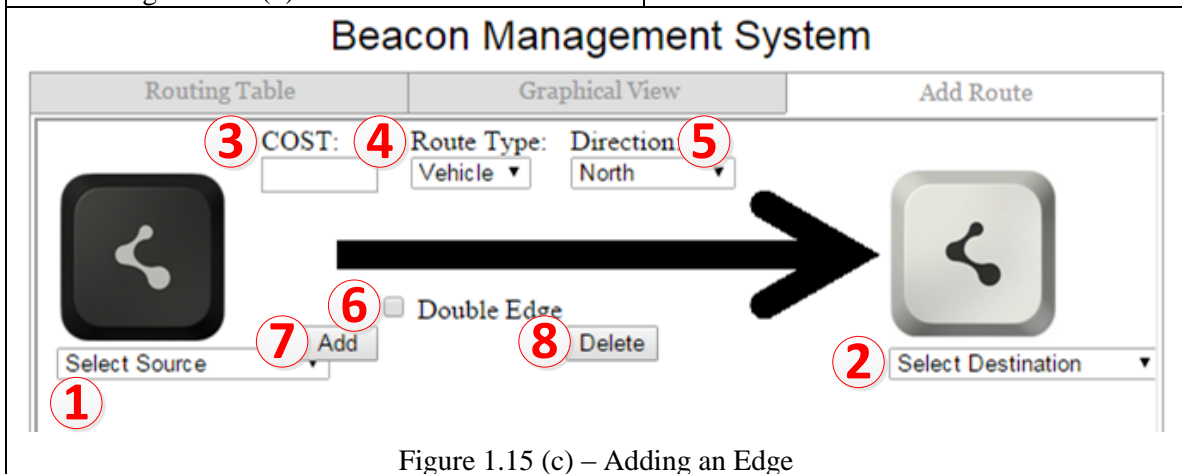


Figure 1.15 (c) – Adding an Edge

Description of each HTML input fields:

- ① : A drop down list to select the source’s Beacon
- ② : A drop down list to select the destination’s Beacon
- ③ : Cost is the cost of edge between the source’s Beacon and destination’s Beacon. Cost will be used to calculate shortest path which will be discussed in the later part.

- ④ : There are two graphs for each car park, one is for driving, and one is for walking. Since the behavior for driving and walking is different, so it is necessary to have 2 graphs. In here, selecting route type means selecting which graph to add this edge.
- ⑤ : The absolute direction pointing to Destination Beacon from Source Beacon.

### Beacon Management System

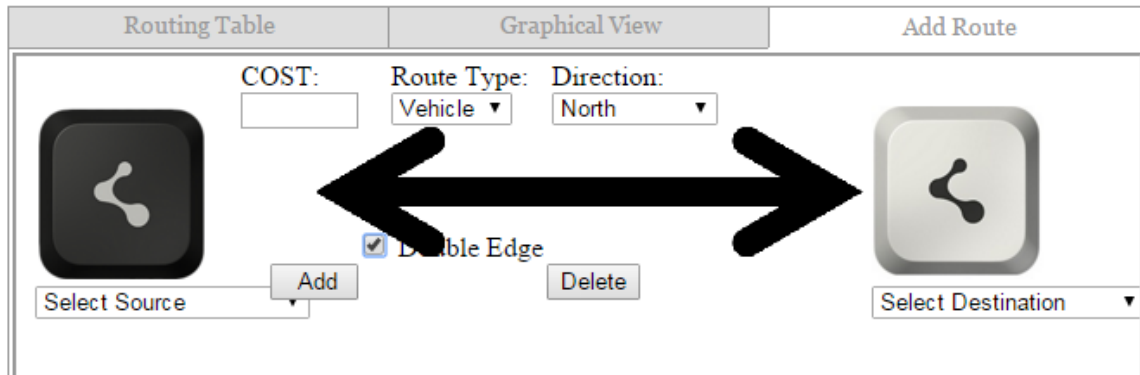
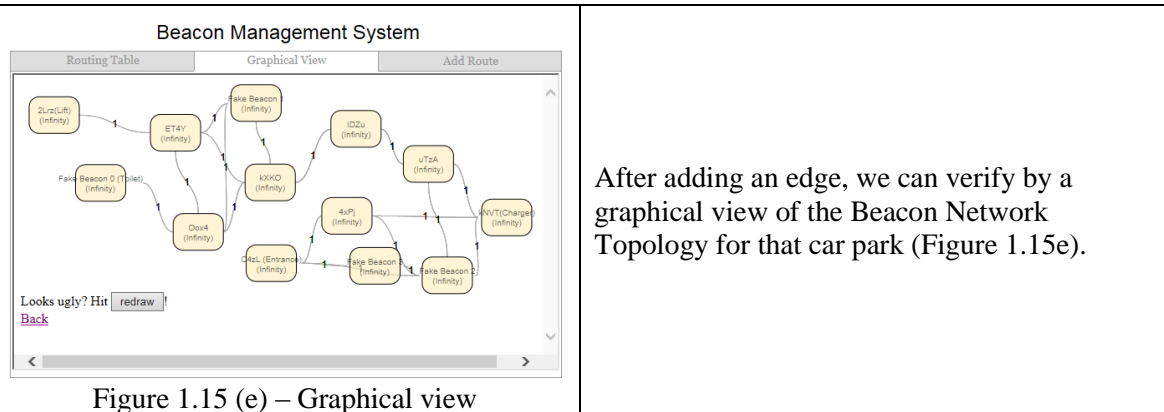


Figure 1.15 (d) – Adding an Undirected Edge

- ⑥ : If the “Double edge” is checked (Figure 1.15d), then system will help to add a revert edge to graph. For example, if the original edge is “A to B in North direction”, then the revert edge will be “B to A in South direction”.
- ⑦ : After ① to ⑥ has inputted, then click the “Add” button to add the edge.
- ⑧ : After ①, ② & ④ has inputted, then click the “Delete” button to delete the existing edge.



After adding an edge, we can verify by a graphical view of the Beacon Network Topology for that car park (Figure 1.15e).

Figure 1.15 (e) – Graphical view

Up to here, we have gone through our UI design of the CMS’s website. Now let’s talk about the potential security issue of CMS.

## 5.5 Web's Security Issues

### 1. Authorization

User can use CMS' web site to add / delete information on the server, therefore the user must be authorized. Car park administrator needs to log on to the system with the highest privilege level's account. There are 3 privilege levels:

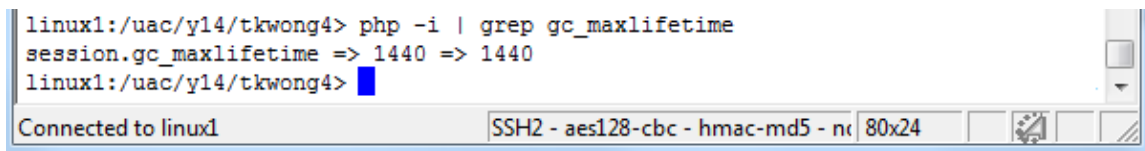
User Privilege Level	Description
15	Administrative User; Able to read / write any information on the CMS.
10	General User; Only able to access to the API provided by the CMS.
0	Unauthorized / Black Listed User; Not allow to access to neither API nor CMS website.

After enter car park administrator entered a correct user name and password on the login page (Figure 1.18), CMS will create a session to store information of user such as UID (User ID), User Name, Login ID, and Privilege. Figure 1.16 has showed the code of creating a session by PHP.

```
session_start(); //!↓
$_SESSION['UID'] = $row['UID'];↓
$_SESSION['Name'] = $row['Name'];↓
$_SESSION['loginID'] = $row['loginID'];↓
$_SESSION['Privilege'] = $row['Privilege']; //set session↓
```

Figure 1.16 – Code of creating a session by PHP

After the session is created, car park administrator no needs to log-on again until the session expired. The session will be expired in 1440 seconds (24 minutes) as showed in Figure 1.17.



```
linux1:/uac/y14/tkwong4> php -i | grep gc_maxlifetime
session.gc_maxlifetime => 1440 => 1440
linux1:/uac/y14/tkwong4>
```

The screenshot shows a terminal window with a blue prompt. The command 'php -i | grep gc\_maxlifetime' is entered, and the output is 'session.gc\_maxlifetime => 1440 => 1440'. The terminal title bar indicates 'Connected to linux1' and 'SSH2 - aes128-cbc - hmac-md5 - nx 80x24'.

Figure 1.17 – Expire Time of session in PHP

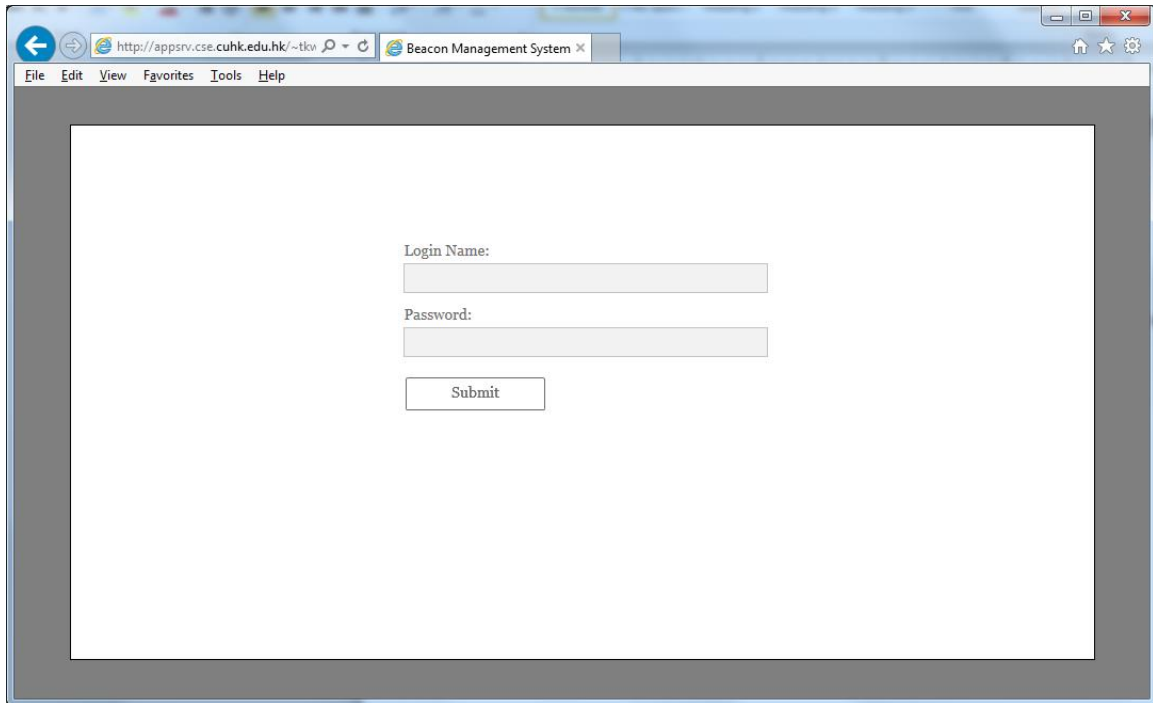


Figure 1.18 – Login Page of CMS

## 2. Preventing Unauthorized Access

The second security issue we encounter is how can we prevent the unauthorized access? For example, if someone who know the full URL of User Management page (E.G. [http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms\\_v2/users.php](http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms_v2/users.php)), then he can go create a new user to grant access to himself.

In order to prevent this happen, we have to check the PHP session on every page, if there is no session or the session has already expired, CMS will redirect him to the login page. We created a simple PHP header showed in Figure 1.19 called *Login Redirection*. The purpose of this header is to check the PHP session, if there the session does not exist, it will forward user to the login page.

```
<?php↓
    // start of Login_Redirection.php↓
    session_start();↓
    if(! (isset($_SESSION['UID']))) { header('location:index.php'); }↓
    // End of Login_Redirection.php↓
?>←
```

Figure 1.19 – Login Redirection.php

We only needs to include the *Login Redirection* header in every PHP files in CMS, the unauthorized access issue will be solved. The PHP file include statement is showed in the figure 1.20.

```
<!DOCTYPE html>↓
<html>↓
<head>↓
<?php include 'Login_Redirection.php'; //redirect users to login page if they were no logged in. ?> ↓
<?php include './Connection/conn.php'; //Connect to MYSQL server ?>↓
<?php include 'SQL_Injection.php'; //Provided a method to process string to prevent sql injection. ?>↓
<?php include './log/logFunction.php'; ?>↓
```

Figure 1.20 – Include statement for preventing unauthorized access

### 3. SQL Injection Attack

SQL injection attack is injection of SQL statements into form input. If server does not check the input from user, the injected SQL statements may be executed. The effect of SQL injection attack is disastrous, it may cause database data leakage, database data being modified, or even the worst case is the whole database being erased.

Here is an example of SQL Injection, if we enter `1' or '1' = '1` as login ID, and anything as password (Figure 1.21). After we click submit, it will show successful login.

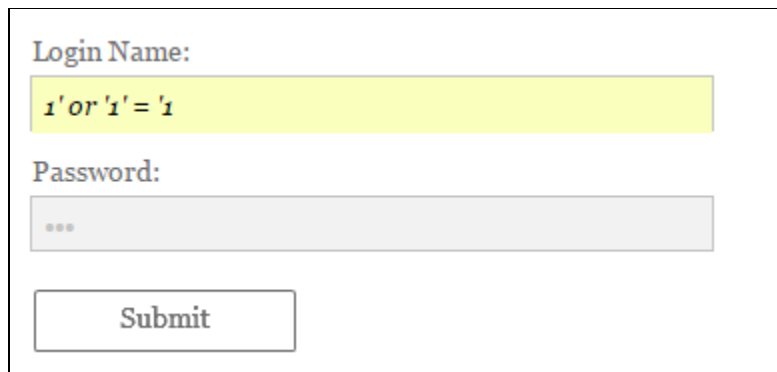


Figure 1.21 – SQL Injection Attack

It's because the original SQL statement (Figure 1.22) injected with an always true condition, so the condition checking in the original statement become nothing. In this case, if we enter `1' or '1' = '1` as login ID, the SQL will become:

```
SELECT * FROM lyu1502_user WHERE loginID = '1' or '1' = '1' and loginPswd = 'haha'
```

As same as:

```
SELECT * FROM lyu1502_user
```

That's the mechanism of SQL injection attack. To prevent, we need to check every input from user. If the user input contains some special characters, we reject the request. The code for checking user input is showed in Figure 1.23. After we adopted the checking on login page, the code is showed in Figure 1.24.

```

        if(isset($_POST['name'])){ ↓
            $id=$_POST['name'];↓
            $pw=$_POST['pw'];↓
            $sql="select * from lyu1502_user where loginID = '$id' and loginPswd = '$pw'";↓
            //find the user role↓
            $result=mysqli_query($conn,$sql) or die('MySQL query error');↓
            $row = mysqli_fetch_array($result);↓

```

Figure 1.22 – Before the SQL injection attack is protected

```

<?php↓
    ↓
↓
function cleanQuery($string)↓
{↓
↓
    if(get_magic_quotes_gpc()) // prevents duplicate backslashes↓
    {↓
        $string = stripslashes($string);↓
    }↓
    if (phpversion() >= '4.3.0')↓
    {↓
        $hostname_conn = "appsrddb.cse.cuhk.edu.hk";↓
        $database_conn = "viewtech";↓
        $username_conn = "viewtech";↓
        $password_conn = " ";↓
        $conn = mysqli_connect($hostname_conn, $username_conn, $password_conn, $database_conn)
    ↓
        $string = mysqli_real_escape_string($conn,$string);↓
    }↓
    else↓
    {↓
        $string = mysqli_escape_string($conn,$string);↓
    }↓
    return $string;↓
}↓
?>↓

```

Figure 1.23 – PHP Function to verify and prevent SQL Injection

```

include 'SQL_Injection.php'; //Provided a method to process string to prevent sql injection.↓
if(isset($_POST['name'])){ ↓
    $id=cleanQuery($_POST['name']);↓
    $pw=cleanQuery($_POST['pw']);↓
    $sql="select * from lyu1502_user where loginID = '$id' and loginPswd = '$pw'";↓
    //find the user role↓
    $result=mysqli_query($conn,$sql) or die('MySQL query error');↓
    $row = mysqli_fetch_array($result);↓

```

Figure 1.24 – After the SQL injection attack is protected

## 5.6 Application Programming Interface (API) Specification

API will handle the HTTP POST request from User App, and reply with JavaScript Object Notation (JSON) as shown in figure 2.1. The main reason to use JSON instead of XML is the former occupy fewer data size, so User App uses fewer time to retrieve the data.

Here is an example to compare the size of XML and JSON which are representing the same data.

JSON:

```
"users" : {  "loginName" : "Admin",
             "loginPswd" : "admin",
             "uid" : 1,
             "privilege" : 15 }
```

XML:

```
<user>
  <loginName>Admin</loginName>
  <loginPswd>admin</loginPswd>
  <uid>1</uid>
  <privilege>15</privilege>
</user>
```

Both JSON and XML in the above are representing the same data. However, the file size of XML is 108 Bytes, but the file of JSON is 72 Bytes. Clearly, the file size of JSON expression is 33% smaller than XML. Smaller file size can reduce the time required to retrieve the data especially for some people still using 3G or 2.75G cellular network which only support up to 2Mbps data rate. That's why we will use JSON instead of XML.

The detailed API specification will be shown below.

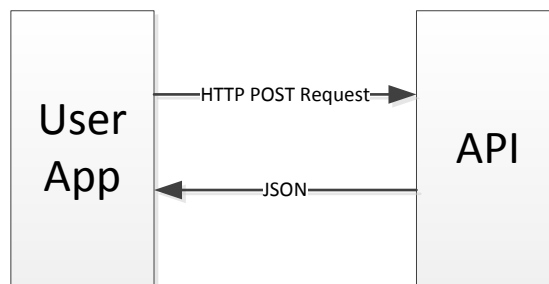


Figure 2.1 – API handle HTTP POST Request, reply with JSON data



API Name	Description	HTTP Request and Response		
Beacon Name	Resolve iBeacon/Eddystone Name to UUID/Major/Minor or EID respectively.	URL:	http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/api/jsonBeaconName.php	
		Method:	POST	
		Data Params:	<b>Required:</b> bname=[String (max. length:36)] <i>Example: bname=Viewlab01</i>  <b>Optional:</b> ( None )	
		URL Params:	( None )	
		Success Response:	Content: <pre>           {             "success_tag" : "success",             "0" : [{               "UUID" : "63bb198a-3b7c-11e5-9bdd-0021859773f6",               "Major" : "1101",               "Minor" : "2",               "CPID" : "0",               "floor" : 1,               "X" : 1.0,               "Y" : 0.5             }           ],             "1" : [{               "EID" : "0x0987654444",               "CPID" : "5",               "floor" : 1,               "X" : 1.0,               "Y" : 1.0             }           ]         }           </pre>	
		Error Response:	Content: { "success_tag" : "fail" }	
		Sample Call:	N/A	
		Note:	<b>1 Beacon Name can be linked to more than one beacon.</b> 1 UUID/Major/Minor can only link to 1 beacon name. 1 EID can only link to 1 beacon name.	

API Name	Description	HTTP Request and Response		
Beacon Name	Resolve UUID/Major/Minor or EID to Beacon Name.	URL:	http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/api/jsonBeaconName.php	
		Method:	POST	
		URL Params:	<b>Required:</b> uuid=[String (max. length:36)] <i>Example: uuid=0x987654444</i> <i>Example: uuid=63bb198a-3b7c-11e5-9bdd-0021859773f6</i>	<b>Optional:</b> major=[Small Integer] <i>Example: major=1101</i> minor=[Small Integer] <i>Example: minor=2</i> device_id =[String]
		Data Params:	( None )	
		Success Response:	Content: <pre> {   "success_tag" : "success",   "BName" : "ViewLab02",   "BDesc" : "N/A",   "CPID" : "0"   "floor" : 1,   "X" : 1.0,   "Y" : 0.5 } </pre>	
		Error Response:	Content: <pre> {   "success_tag" : "fail" } </pre>	
		Sample Call:	(N/A)	
		Note:	When you want to resolve an Eddystone ID, you only need to use URL param "uuid" as EID. When you want to resolve an iBeacon ID, you need to use all the URL params (uuid, major, minor).	

API Name	Description	HTTP Request and Response	
Next Hop	Get the next beacon information according to the shortest path.	URL:	http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/api/jsonGetNextHop.php
		Method:	POST
		Data Params:	<p><b>Option 1 (Source is Eddystone, Destination is Eddystone)</b>  <b>Required:</b>  source_uuid=[String (max. length:36)]      <i>Example: source_uuid =0x987654444</i>  dest_uuid=[String (max. length:36)]      <i>Example: dest_uuid =0x987655555</i>  type=[String (max. length:3)]      <i>Example: type=V (V = Vehicle, P=People)</i></p> <p><b>Option 2 (Source is Eddystone, Destination is iBeacon)</b>  <b>Required:</b>  source_uuid=[String (max. length:36)]      <i>Example: source_uuid =0x987654444</i>  dest_uuid=[String (max. length:36)]      <i>Example: dest_uuid =63bb198a-3b7c-11e5-9bdd-0021859773f6</i>  dest_major=[Small Integer]      <i>Example: major=1101</i>  dest_minor=[Small Integer]      <i>Example: minor=2</i>  type=[String (max. length:3)]      <i>Example: type=V (V = Vehicle, P=People)</i></p> <p><b>Option 3 (Source is iBeacon, Destination is Eddystone)</b>  <b>Required:</b>  source_uuid=[String (max. length:36)]      <i>Example: dest_uuid =63bb198a-3b7c-11e5-9bdd-0021859773f6</i>  source_major=[Small Integer]      <i>Example: major=1101</i>  source_minor=[Small Integer]      <i>Example: minor=3</i>  dest_uuid=[String (max. length:36)]      <i>Example: source_uuid =0x987654444</i>  type=[String (max. length:3)]      <i>Example: type=V (V = Vehicle, P=People)</i></p> <p><b>Option 4 (Source is iBeacon, Destination is iBeacon)</b>  <b>Required:</b>  source_uuid=[String (max. length:36)]      <i>Example: dest_uuid =63bb198a-3b7c-11e5-9bdd-0021859773f6</i>  source_major=[Small Integer]      <i>Example: major=1101</i>  source_minor=[Small Integer]      <i>Example: minor=2</i>  dest_uuid=[String (max. length:36)]      <i>Example: dest_uuid =63bb198a-3b7c-11e5-9bdd-0021859773f6</i>  dest_major=[Small Integer]      <i>Example: major=1101</i>  dest_minor=[Small Integer]      <i>Example: minor=3</i>  type=[String (max. length:3)]      <i>Example: type=V (V = Vehicle, P=People)</i></p> <p><b>Option 5 (Don't Care the type of beacon, but BID is needed)</b>  <b>Required:</b>  source=[ Small Integer]      <i>Example: source=25</i>  dest=[Small Integer]      <i>Example: dest=26</i>  type=[String (max. length:3)]      <i>Example: type=V (V = Vehicle, P=People)</i></p>
		URL Params:	( None )

		<p>Success Response:</p>	<p>Example Content: {  "success_tag" : "success",  "NextHop" : "15", <i>Remark: This field contains the next hop's BID</i>  "Bname" : "Dijkstra Test 06",  &gt;Type" : "B", <i>Remark: If next hop is an iBeacon, Type filed is always "B".</i>  "UUID" : "ffffffff-ffff-ffff-ffff-ffffffffffffff",  "Major" : "0",  "Minor" : "5",  "EID" : NULL, <i>Remark: If next hop is an iBeacon, EID field will be NULL.</i>  "direction": "SE",  "RouteCost": "66"  }</p> <p>Example Content: {  "success_tag" : "success",  "NextHop" : "26",  "Bname" : "Dijkstra Test 09",  &gt;Type" : "E", <i>Remark: If next hop is an Eddystone, Type filed is always "E".</i>  "UUID" : "", <i>Remark: If next hop is an Eddystone, UUID field will be empty.</i>  "Major" : "0", <i>Remark: If next hop is an Eddystone, Major field will be 0.</i>  "Minor" : "0", <i>Remark: If next hop is an Eddystone, Minor field will be 0.</i>  "EID" : "0x0987651235",  "direction": "N",  "RouteCost": "10"  }</p>
		<p>Error Response:</p>	<p>Content: {  "success_tag" : "fail" <i>Remark: If source and destination are same, or next hop is not find (no path to that destination), API will reply with this.</i>  }</p>
		<p>Sample Call:</p>	<p>(N/A)</p>
		<p>Note:</p>	<p>If source and destination are same, API will generate an error response.</p>

API Name	Description	HTTP Request and Response		
Logging	Send Log data to server for analytics purpose	URL:	http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/api/jsonAddLog.php	
		Method:	POST	
		URL Params:	( None )	
		Data Params:	<p><b>Required:</b></p> tag=[String (max. length:32)] <i>Example: tag=CLIENT_LOCATION_REGISTER</i> message=[String (max. length:255)] <i>Example: message="Client is just entered into a known beacon area (UUID:AAAA, Major:BB, Minor:CC)"</i>  detail=[String (max. length:512)] <i>Example: detail="Client is come from pervious beacon (Lift Lobby, UUID:1111, Maojr:22, Minor:33), \n and just entered into a known beacon area (UUID:AAAA, Major:BB, Minor:CC)"</i>	
		Success Response:	<p><b>Optional:</b></p> keyword1=[String (max. length:255)] <i>Example: keyword1=AAAA</i> keyword2=[String (max. length:255)] <i>Example: keyword2=BB</i> keyword3=[String (max. length:255)] <i>Example: keyword3=CC</i> keyword4=[String (max. length:255)] <i>Example: keyword4=</i> keyword5=[String (max. length:255)] <i>Example: keyword5=</i>	
		Error Response:	<p>Content:</p> <pre>{   "success_tag" : "fail" }</pre>	
		Sample Call:	( N/A )	
		Note:	The fields of keyword1 to keyword5 are not mandatory to have data. Keyword fields are used to store data (like datetime, number, name...) which are for analytic purpose.	

API Name	Description	HTTP Request and Response	
GetFloorplan	Get the Image's URL of specific floorplan(s) and the beacons' location on particular floor.	URL:	http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/api/jsonGetFloorplan.php
		Method:	POST
		URL Params:	( None )
		Data Params:	<b>Required:</b> cpid=[Unsigned Integer] <i>Example: cpid=1</i> <b>Optional:</b> floor=[Integer] <i>Example: floor=-2</i>
		Success Response:	Content (with "floor" optional field): <pre>{   "success_tag": "success", "CPID": 1, "floor": -2,   "image": "http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/files/floorplan/1_-2.jpg"   "count": 1,   "0": [{     "BName": "Beacon Name",     "X": "123.3",     "Y": "321.1" }]} </pre> Content (without "floor" optional field): <pre>{   "success_tag": "success", "CPID": 1,   "count": "2",   "0": [{     "floor": "1",     "image": "http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/files/floorplan/1_1.jpg"     "count": "2",     "0": [{       "BName": "Beacon Name 1",       "X": "111.1",       "Y": "222.2" }],     "1": [{       "BName": "Beacon Name 2",       "X": "333.1",       "Y": "222.2" }]}   ]},   "1": [{     "floor": "2",     "image": "http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/files/floorplan/1_2.jpg"     "count": 0   ]}] </pre>
		Error Response:	Content: { "success_tag": "fail" }
		Sample Call:	( N/A )
Note:	( N/A )		

API	Description	HTTP Request and Response	
Get Car Park Info	Get the Name and description of car park.	URL:	http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/api/jsonGetCarparkInfo.php
		Method:	POST
		URL Params:	( None )
		Data Params:	<b>Required:</b> cpid=[Unsigned Integer] <i>Example: cpid=1</i>
		Success Response:	Content : { "success_tag" : "success", "Cname" : "Test Car Park", "Cdesc" : "This is a car park for testing.....", "Latitude_LB" : "-99.9", "Longitude_LB" : "66.6", "Latitude_RT" : "-102.2", "Longitude_RT" : "33.3" }
		Error Response:	Content: { "success_tag" : "fail" }
		Sample Call:	( N/A )
Note:	( N/A )		

API	Description	HTTP Request and Response	
Authentication	Perform Authentication	URL:	http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/api/jsonAuthen.php
		Method:	POST
		URL Params:	( None )
		Data Params:	<b>Required:</b> name=[String (max. length = 16) ] <i>Example: name=test</i> pswd=[String (max. length = 16) ] <i>Example: name=test</i>
		Success Response:	Content : { "success_tag" : "success", "name" : "Test User's Name"   }
		Error Response:	Content: { "success_tag" : "fail" }
		Sample Call:	( N/A )
Note:	If either login name or password is incorrect, API will return fail in <i>success_tag</i> .		

API Name	Description	HTTP Request and Response		
Authentication	Perform Authentication	URL:	http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms/api/jsonAuthen.php	
		Method:	POST	
		URL Params:	( None )	
		Data Params:	<b>Required:</b> name=[String (max. length = 16) ] <i>Example: name=test</i> pswd=[String (max. length = 16) ] <i>Example: name=test</i>	
		Success Response:	Content : { "success_tag" : "success", "name" : "Test User's Name" }	
		Error Response:	Content: { "success_tag" : "fail" }	
		Sample Call:	( N/A )	
		Note:	If either login name or password is incorrect, API will return fail in <i>success_tag</i> .	



## 5.7 Routing Table

In the design overview, we have discussed how to use routing table to achieve guidance function. Here we will talk about how we construct the routing table by using graph data structure.

First, Beacons are deployed in car park. After car-park administrator has added all the routes for all Beacons through CMS's web site, the whole picture will become a graph. We can treat Beacons as vertexes, and routes as edges (Figure 3.1). So, we have turned a routing problem into a graph problem.

Second, we turn the graph into routing table using graph data structure and shortest path algorithm. Since it invokes data structure programming, we decide not to use PHP to calculate shortest path because PHP is not an object-oriented program. Instead, we used JAVA to calculate the shortest path. We choose to use JAVA because JAVA is a cross-platform object-oriented programming language.

We will write a Java program using Java Database Connectivity (JDBC) to retrieve graph information (vertexes, edges, and cost) from MySQL database server of CMS. Once the program gets all the information of graph, it uses open source graph library — JGraphT to build up the graph and calculate the shortest path. Finally, it stores the shortest paths back to the database, which is the so called *Routing Table*.

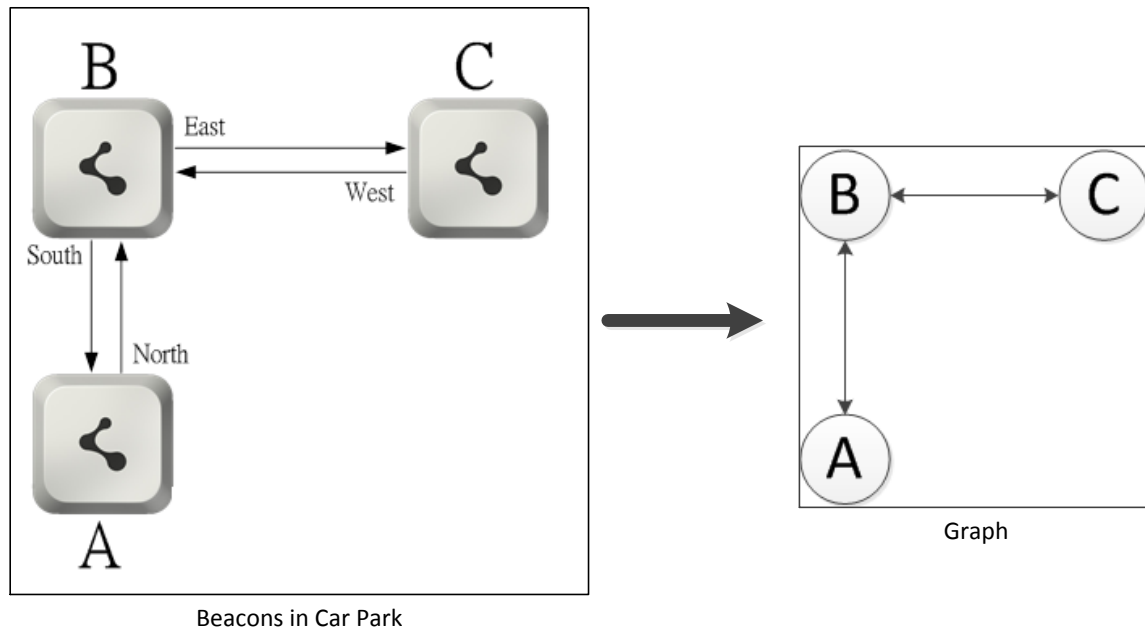


Figure 3.1 – Treat Beacons as vertexes, and routes as edges

## JGraphT

JGraphT is an open source graph library for Java written by Barak Naveh. The library includes mathematical graph-theory objects and algorithms<sup>8</sup>. JGraphT is quite easy to use, no matter creating new graph, adding new vertex or edge, it just invoke single line of Java code.

Example on creating a new simple directed weighted graph:

```
WeightedGraph<String, DefaultWeightedEdge> g = new SimpleDirectedWeightedGraph<String, DefaultWeightedEdge>(DefaultWeightedEdge.class);↓
```

Example on adding new vertexes to the graph *g* created as above.

```
g.addVertex("Vertex_Name");↓  
g.addVertex("Vertex_Name 2");↓
```

Example on adding an edge between “*Vertex\_Name*” and “*Vertex\_Name 2*”.

```
g.addEdge("Vertex_Name", "Vertex_Name2");↓
```

Example on modifying the weight or the cost of the edge, for example cost = 10.

```
g.setEdgeWeight(g.addEdge("Vertex_Name", "Vertex_Name2"), 10);\
```

---

<sup>8</sup> Original Statement: “JGraphT is a free Java graph library that provides mathematical graph-theory objects and algorithms.” Source: <http://jgrapht.org/>

## Initial Approach to build Routing Table

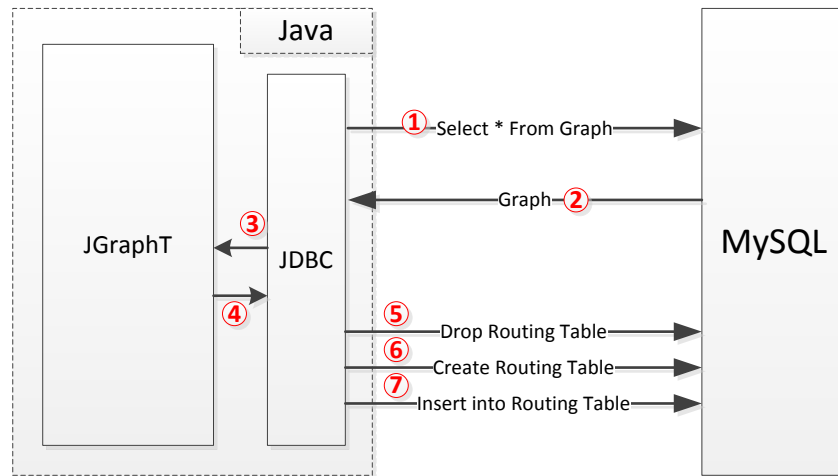


Figure 3.2 – Initial approach to build Routing table

- ① : The Java program will send an SELECT query to MySQL to obtain the Graph table.
- ② : MySQL received the query and reply with the content of table named Graph.
- ③ : JDBC passes the data to JGraphT to build a graph using the information retrieved from MySQL. (Figure 3.3)
- ④ : JGraphT calculate the Shortest Path for all *Beacon i* to *Beacon j*. (Figure 3.4)
- ⑤ : Drop the old and updated routing table.
- ⑥ : Create an empty routing table.
- ⑦ : Fill in the routing table with the calculated shortest path (Figure 3.5).

The process will be repeated on server for every 5 seconds. So, the routing table will be updated every 5 second. But this approach has a hidden problem, which is occur in Step 5 to Step 7.

When more and more Beacons are used in our CMS, the graph will also become larger. To calculate the shortest path for all *Beacon i* to *Beacon j* will take some time, which also affect that it takes longer time to finish the routing (note that the table is now empty because it has been dropped). The problem is, during that period of time, User App cannot use the indoor guidance function since the routing table is not ready yet.

```

//Get vertexes from MySQL↓
selectString = "SELECT * FROM lyu1502_beacon";↓
rs = stmt.executeQuery(selectString);↓
↓
rs.beforeFirst();↓
//Build an new Graph↓
WeightedGraph<String, DefaultWeightedEdge> g = new SimpleDirectedWeightedGraph<String, ↓
DefaultWeightedEdge>(DefaultWeightedEdge.class);↓

while(rs.next())↓
{↓
    //add Vertex to the Graph↓
    g.addVertex(rs.getObject("BID") + "");↓
    listHop.add(rs.getObject("BID") + "");↓
}↓
//Get edges from MySQL↓
selectString = "SELECT * FROM lyu1502_graph WHERE routetype LIKE '%V%'";↓
rs = stmt.executeQuery(selectString);↓
rs.beforeFirst();↓
while(rs.next())↓
{↓
    //add Edge to the Graph↓
    g.setEdgeWeight(g.addEdge(rs.getObject("Source") + "",rs.getObject("Destination") + ""),↓
Integer.parseInt(rs.getObject("Cost")+"")+Integer.parseInt(rs.getObject("DynamicCost")+""));↓
}↓

```

Figure 3.3 – JAVA Code on building a graph using the information retrieve from MySQL

```

for (int i = 0; i < listHop.size(); i++) {↓
    String sourceHop = listHop.get(i);↓
    for (int j = 0; j < listHop.size(); j++) { ↓
        if (i != j) {↓
            path = DijkstraShortestPath.findPathBetween(g, sourceHop, listHop.get(j)+"");↓
            ↓
            // ...↓
        }
    }
}

```

Figure 3.4 – JAVA code on calculating the shortest path for all *Beacons i* to *Beacon j*

```

if (path != null && !((path.toString()).equals("null"))) {↓
    ↓
    ↓
    // get Next Hop Beacon ID↓
    String nextHopID = g.getEdgeTarget(path.get(0));↓
    ↓
    ↓
    // get Next Hop Direction↓
    selectString = "SELECT * FROM lyu1502_graph WHERE source = " + sourceHop + " and destination = " + nextHopID;↓
    rs = stmt.executeQuery(selectString);↓
    rs.beforeFirst();↓
    rs.next();↓
    String nextHopDirection = rs.getObject("Direction")+"";↓
    ↓
    // calculate total path cost↓
    int totalCost = 0;↓
    for (int x = 0; x < path.size(); x++) {↓
        totalCost += g.getEdgeWeight(path.get(x));↓
    }↓
    ↓
    insertString = "INSERT INTO lyu1502_carroutingtable VALUE (" + RTID + ", " + sourceHop + ", " + ↓
listHop.get(j) + ", " + nextHopID + ", " + "↓
+ nextHopDirection + ", " + path.size() + ", " + ↓
+ totalCost + " , 'V', " + revisionNumber + " " );↓
↓
}

```

Figure 3.5 – JAVA code on inserting the shortest path to routing table

## Improved Approach to build Routing Table

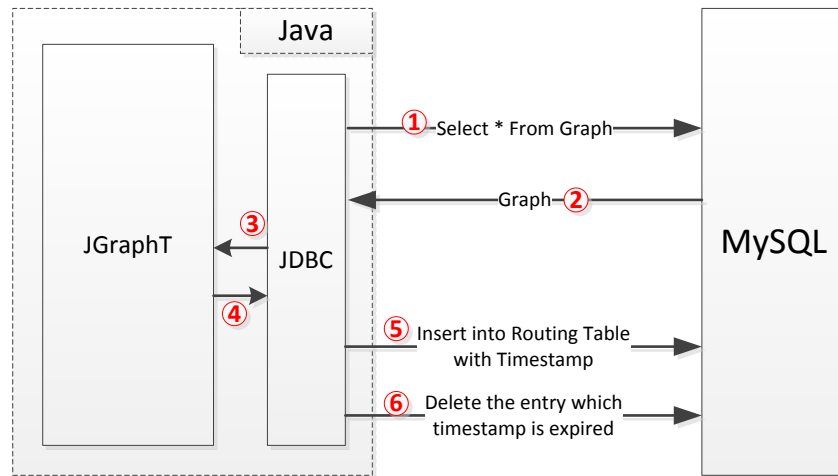
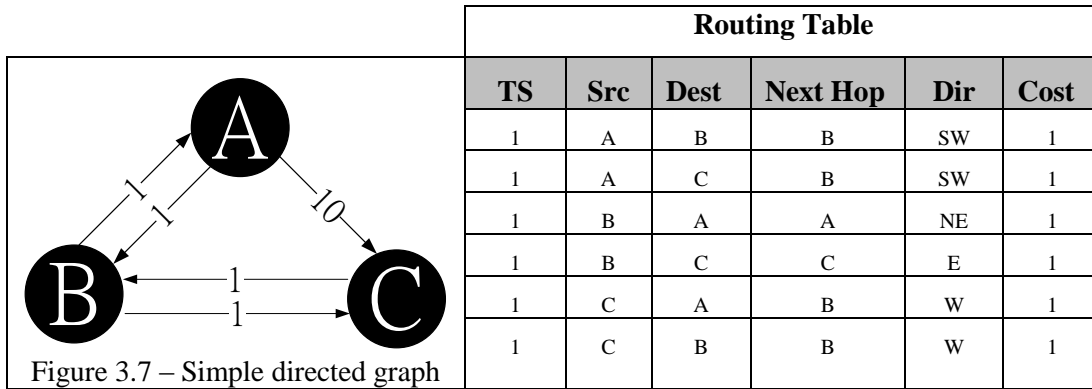


Figure 3.6 – Improved Approach to build routing table

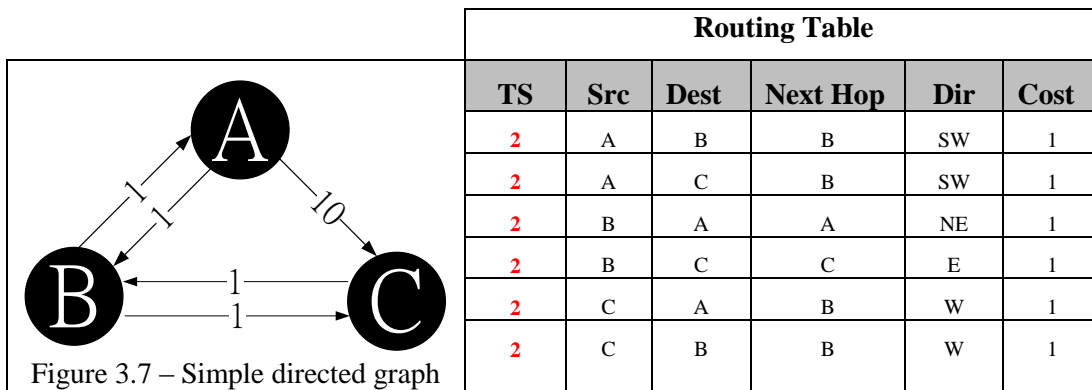
In order to solve the (*routing table not yet ready*) problem, we no longer drop the table every time. Instead, we update the each entry in routing table with a timestamp. If the timestamp is expired, then the entry will be deleted from routing table.

- ① : The Java program will send an SELECT query to MySQL to obtain the Graph table.
- ② : MySQL received the query and reply with the content of table named Graph.
- ③ : JDBC passes the data to JGraphT to build a graph using the information retrieved from MySQL. (Figure 3.3)
- ④ : JGraphT calculate the Shortest Path for all *Beacon i* to *Beacon j*. (Figure 3.4)
- ⑤ : Insert the shortest path into Routing Table with timestamp.
- ⑥ : Delete the entry for which timestamp is expired. (Figure 3.9)

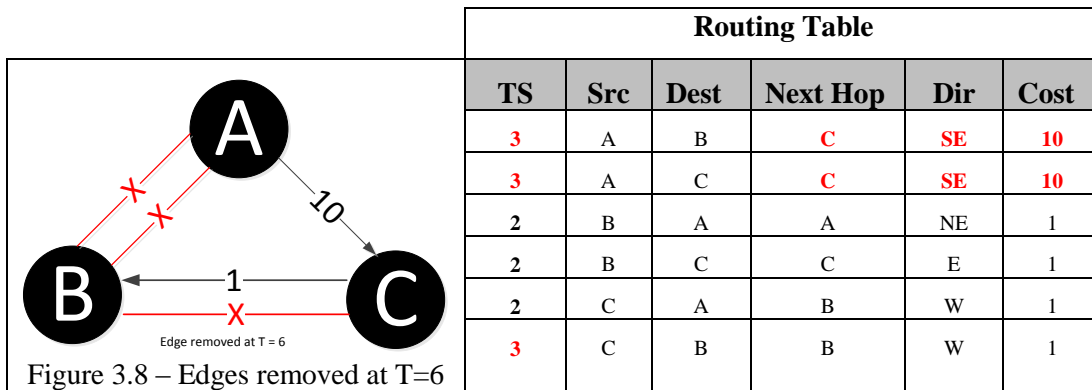
So, the routing table will look like this initially (t = 0s):



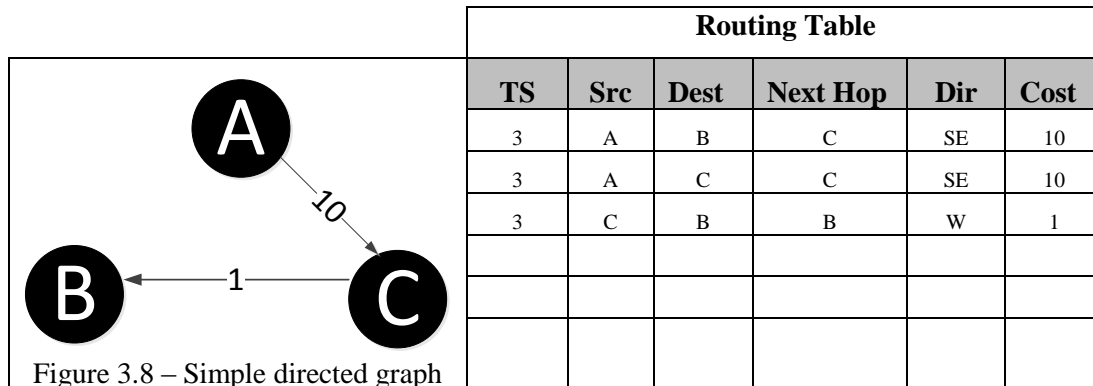
After 5 seconds (t=5s), the routing table refreshed by program, the TS field (timestamp) increase by one:



The edge from B to C is removed at t=6s, and the routing table will be updated on t=10s:



Finally (t=10s), all entry with TS < 3 in routing table will be removed, so the routing table becomes:



Routing table is still available even when it is updating by the Java program. The last routing information will be kept until all the new routing information is ready. So, the routing table is always available for User App, which is quite different than our initial approach.

```

for (int j = 0; j < listHop.size(); j++) { ↓
    if (i != j) { ↓
        path = DijkstraShortestPath.findPathBetween(g, sourceHop, listHop.get(j)+""); ↓
        ↓
        if (path != null && !((path.toString()).equals("null"))) { ↓
            ↓
            // ↓
            // ( codes skipped) ↓
            // ↓
            // Add an entry to routing table ↓
            insertString = "INSERT INTO lyu1502_carroutingtable VALUE (" + RTID + ", " + sourceHop + ", ↓
                " + listHop.get(j) + ", " + nextHopID + ", " + nextHopDirection + ", " + path.size() + ", " + totalCost + ", 'P', " + revisionNumber + ")";
            try { ↓
                stmt.executeUpdate(insertString); ↓
            } catch (SQLException sqle) { ↓
                // Assume that this is the record duplication problem ↓
                // so, use UPDATE instead ↓
                updateString = "UPDATE lyu1502_carroutingtable SET NextHop = " + nextHopID + ", NextHopDirection = " + ↓
                    nextHopDirection + ", HopCount = " + path.size() + ", RouteCost = " + totalCost + ↓
                    " , RevisionNumber = " + revisionNumber + " WHERE source = " + sourceHop + " and destination = " + ↓
                    listHop.get(j) + " and routetype = 'P'"; ↓
                stmt.executeUpdate(updateString); ↓
            } ↓
            // Delete old revision number records ↓
            String deleteString = "DELETE FROM lyu1502_carroutingtable WHERE revisionnumber < " + revisionNumber; ↓
            stmt.executeUpdate(deleteString); ↓
        }
    }
}
  
```

Figure 3.9 – JAVA code of improved approach for building Routing Table

## 5.8 User Android App (Front-end)

Mobile Application – Android

### SDK version

Our app supported from Android 4.3 (API level 18) to Android 5.0 (API level 22), because Bluetooth low energy (BLE) supported devices with Android 4.3 and later.

### AndroidManifest.xml

According to google API Guides, permission is a restriction which protects important data and code on the device. “Those data or code could be misused to distort or damage the user experience”<sup>9</sup>, so when users install app, they will get list of permissions.

The permission has a unique label which represents the action that is restricted. In our app, we have use kinds of permission as below.

- **Beacon**

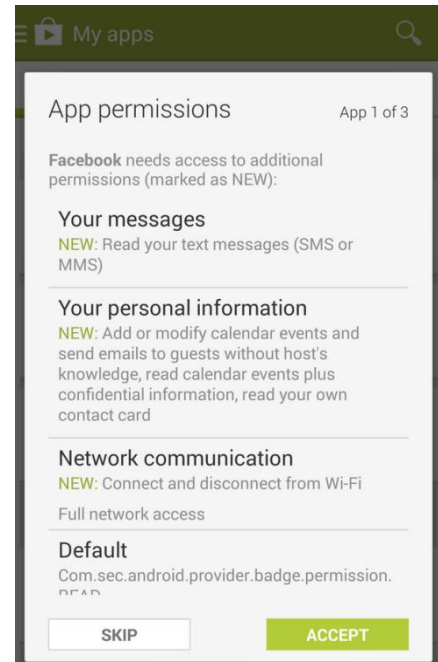
In order to use features of Bluetooth,  
"android.permission.BLUETOOTH"  
"android.permission.BLUETOOTH\_ADMIN"

- **Google map**

In order to use Google Maps Android API,  
"android.permission.INTERNET"  
"android.permission.WRITE\_EXTERNAL\_STORAGE"  
"android.permission.ACCESS\_COARSE\_LOCATION"  
"android.permission.ACCESS\_FINE\_LOCATION"  
"com.google.android.providers.gsf.permission.READ\_GSERVICES"

- **Commination with Server (Call API)**

In order to use Http Client,  
"android.permission.INTERNET"  
"android.permission.ACCESS\_NETWORK\_STATE"



<sup>9</sup> Source: <http://developer.android.com/develop/index.html>



## Library

Currently, google have Proximity Beacon API which focus on register and manage beacons such as monitor the health of beacons, assign or update associated data, and Nearby API which used by application interact with beacon, but it need to spend lots of time and handle amount of works to do something like ranging and monitoring by using these two google API. Cause of the time efficiency, we finally chose to use three party library - Android Beacon Library. This library not only allows user to detect AltBeacon standard, but also allows user to detect different type of beacon such as iBeacon, Eddystone, etc.



Here are some benefits by using Android Beacon Library<sup>10</sup>:

- 1) Can get a ranging update from one or more beacons at a frequency of approximately 1Hz.
- 2) Can get notifications when one or more beacons appear or disappear
- 3) Allow Android 5.0 and later devices to send beacon transmissions in both foreground and background.

## **5.9 User Interface Design of Android App** UI design

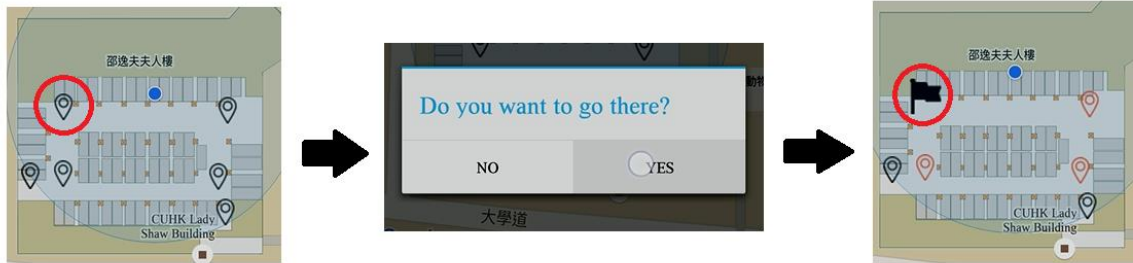
In UI design, we have concerned a lot about the convenience of a driver. Actually, drivers may not have other attention when they driving their car, so we decide to put everything within one page and minimize the number of buttons.

- 1) This is a Drop down list which contain all carpark name. User can choose which carpark they want to go.
- 2) These two buttons are used for selecting floor level, one with the plus icon is +1 floor level, other one with minus icon is -1 floor level.

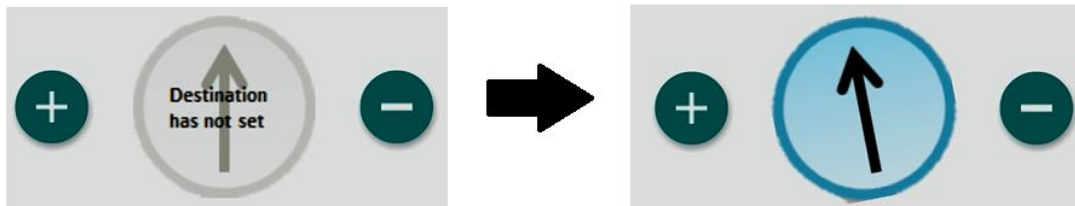


<sup>10</sup> Source: <https://altbeacon.github.io/android-beacon-library/index.html>

- 3) These are markers in google map which allowed user to choose as destination. The icon of marker will change after user confirm go there.



- 4) This is a guidance pointer. It will start rotating after user chooses the destination, the arrow keep pointing to next hop. It represents the relative direction (E.G. forward, backward, left, right) based on the absolute direction (E.G. North, South, West, East)



- 5) Action bar



Button on the action bar



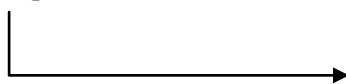
Icon 1) it is used to search user's car which parked in carpark.



Icon 2) it is used to refresh the whole page to prevent API called unexpectedly.



Icon 3) it is used to show information about the app and developers.



About Us



This is our FYP Project APP. This app will guide you to park your car inside the carpark.

Wong Tsz Kin  
(CENG Year3)

Choi Mei Shan  
(CENG Year 3)

[tkwong@cse.cuhk.edu.hk](mailto:tkwong@cse.cuhk.edu.hk) [mschoi4@cse.cuhk.edu.hk](mailto:mschoi4@cse.cuhk.edu.hk)

Copyright © 2015 by CUHK. All Rights Reserved.



## 5.10 Class Diagram of Android App

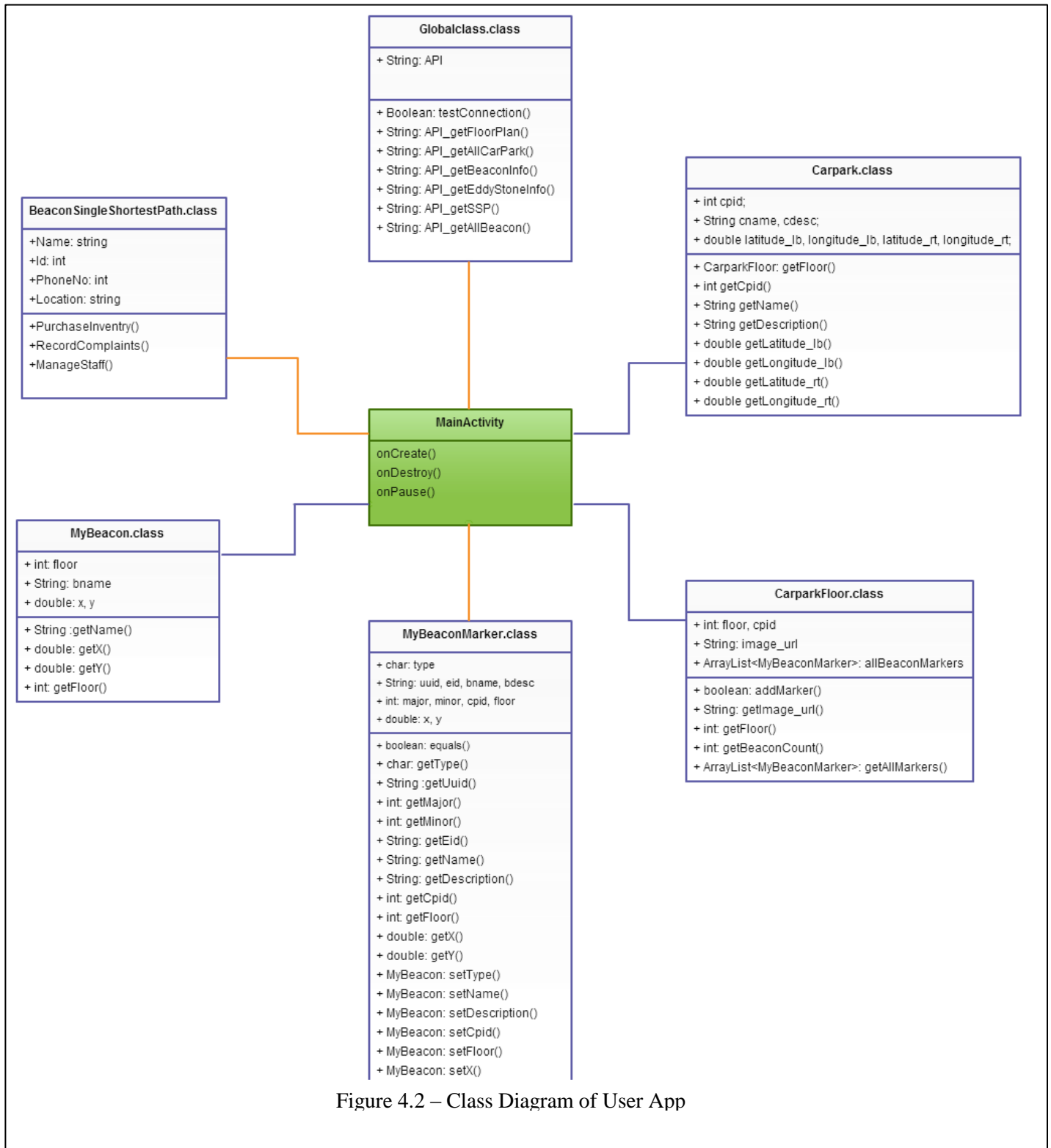


Figure 4.2 – Class Diagram of User App

We will show all our system classes. The classes are used in MainActivity:

### 1) BeaconSingleShortestPath

```
public class BeaconSingleShortestPath {  
  
    // add beacon to shortest path  
    public boolean addHop(MyBeacon newHop, String direction);  
  
    // get number of beacons  
    public int getSize();  
  
    // get beacon from shortest path  
    public MyBeacon getHop(int index);  
  
    // get direction of the next beacon  
    public String getNextHopDirection(MyBeacon source);  
  
    // determine the beacon is in shortest path or not  
    public boolean isWithinShortestPath(MyBeacon beacon);  
  
    // determine user is pass over the location in shortest path or not  
    public boolean isPassed(MyBeacon currentLocation, MyBeacon  
        checkLocation);  
  
}
```

### 2) Carpark

```
public class Carpark {  
  
    // get carpark id  
    public int getCpid();  
  
    // get carpark name  
    public String getName();  
  
    // get carpark latitude (left-bottom)  
    public double getLatitude_lb();  
  
    // get carpark longitude (left-bottom)  
    public double getLongitude_lb();  
  
    // get carpark latitude (right-top)  
    public double getLatitude_rt();  
  
    // get carpark longitude (right-top)  
    public double getLongitude_rt();  
  
}
```

### 3) CarparkFloor

```
public class CarparkFloor{
    // determine is the CarparkFloor add marker successfully
    public boolean addMarker(String bname, double x, double y,
                             int floor);

    // get floor plan image
    public String getImage_url();

    // get the level of floor
    public int getFloor();

    // get the number of beacon
    public int getBeaconCount();

    // get all marker information
    public ArrayList<MyBeaconMarker> getAllMarkers();
}
```

### 4) MyBeacon

```
public class MyBeacon {

    // Constructor for ibeacon
    public MyBeacon(char type, String uuid, int major, int minor, String bname, String
                    bdesc, int cpid, int floor, double x, double y);

    // Constructor for Eddystone
    public MyBeacon(char type, String eid, String bname, String bdesc, int cpid, int
                    floor, double x, double y);

    // determine the new beacon and beacon of MyBeacon is equal or not
    public boolean equals(MyBeacon beacon);

    // get beacon type //set beacon type
    public char getType(); public MyBeacon setType(char type);
    // get ibeacon uuid //set beacon name
    public String getUuid(); public MyBeacon setName(String bname);
    // get ibeacon major id //set beacon description
    public int getMajor(); public MyBeacon setDescription(String bdesc);
    // get ibeacon minor id //set carpark id of beacon
    public int getMinor(); public MyBeacon setCpid(int cpid);
    // get beacon eddystone id //set floor level of beacon
    public String getEid(); public MyBeacon setFloor(int floor);
    // get beacon name //set latitude of beacon
    public String getName(); public MyBeacon setX(double x);
    // get beacon description //set longitude of beacon
    public String getDescription(); public MyBeacon setY(double y);
    // get carpark id of beacon
    public int getCpid();
    // get carpark floor of beacon
    public int getFloor();
    // get latitude of beacon
    public double getX();
    // get longitude of beacon
    public double getY() {}
}
```

## 5) MyBeaconMarker

```
public class MyBeaconMarker {  
    // get marker name  
    public String getName();  
  
    // get latitude of marker  
    public double getX();  
  
    // get longitude of marker  
    public double getY();  
  
    // get the level of floor of marker  
    public int getFloor();  
}
```

## 5.11 Functionality of User App

### 1. Beacon Ranging

Setup and activate beacon scanner using the ALT Beacon SDK:

```
// Scan Beacon
beaconManager = BeaconManager.getInstanceForApplication(this);
beaconManager.getBeaconParsers().add(new BeaconParser().setBeaconLayout("m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24"));
beaconManager.getBeaconParsers().add(new BeaconParser().setBeaconLayout("s:0-1=feaa,m:2-2=00,p:3-3:-41,i:4-13,i:14-19"));
beaconManager.bind(this);
```

By setting up *BeaconParser()*, app can support different kinds of beacons.

For example:

```
"m:2-3=0215,i:4-19,i:20-21,i:22-23,p:24-24" //iBeacon
"s:0-1=feaa,m:2-2=00,p:3-3:-41,i:4-13,i:14-19" //Eddystone
```

Four prefixes are allowed in the string:

```
m - matching byte sequence for this beacon type to parse (exactly one required)
s - ServiceUuid for this beacon type to parse (optional, only for Gatt-based beacons)
i - identifier (at least one required, multiple allowed)
p - power calibration field (exactly one required)
d - data field (optional, multiple allowed)
```

Figure 4.3 – Meaning of prefix for setBeaconLayout()

In our app, we just used Ranging for scanning beacon continually, so that we can know where the user is.

```
@Override
public void onBeaconServiceConnect() {
    beaconManager.setRangeNotifier(new RangeNotifier() {
        @Override
        public void didRangeBeaconsInRegion(Collection<Beacon> beacons, Region region) {
            if (beacons.size() > 0) {
                //Log.i(TAG, "The first beacon I see is about "+beacons.iterator().next().getDistance()+" meters away.");
            }
        }
    });
}
```

Figure 4.4 – Beacon Ranging Listener provided by ALT Beacon Library

```

int closestRssi = -120;
int index = 0, closestIndex = -1;

for (Beacon beacon : beacons) {
    // **** find the closest one ****
    if (beacon.getRssi() > closestRssi) {
        closestRssi = beacon.getRssi();
        closestIndex = index + 1;
        closetBeacon = beacon;
    }
    index++;
}

```

Figure 4.5 – Find the closest Beacon

Figure 4.4 has showed the Beacon ranging listener. Inside *didRangeBeaconsInRegion()*, we do the following thing.

1) We can get a list of beacons that are scanned by the app, so base on the beacon list we can found the closest beacon (Figure 4.5).

After that, we need to check the signal strength value (RSSI) to determine whether the beacon is really near to user or it is far away from user. From the “*Time Delay Error experience*” discussed in feasibility study (Figure 4.6), we found that -75 dBm is the most suitable threshold to determine whether user is within the Beacon’s region. So, here we will reject all beacons that have RSSI lower than -75dBm. Vice visa, we will accept the Beacon having RSSI higher or equal to -75 dBm.

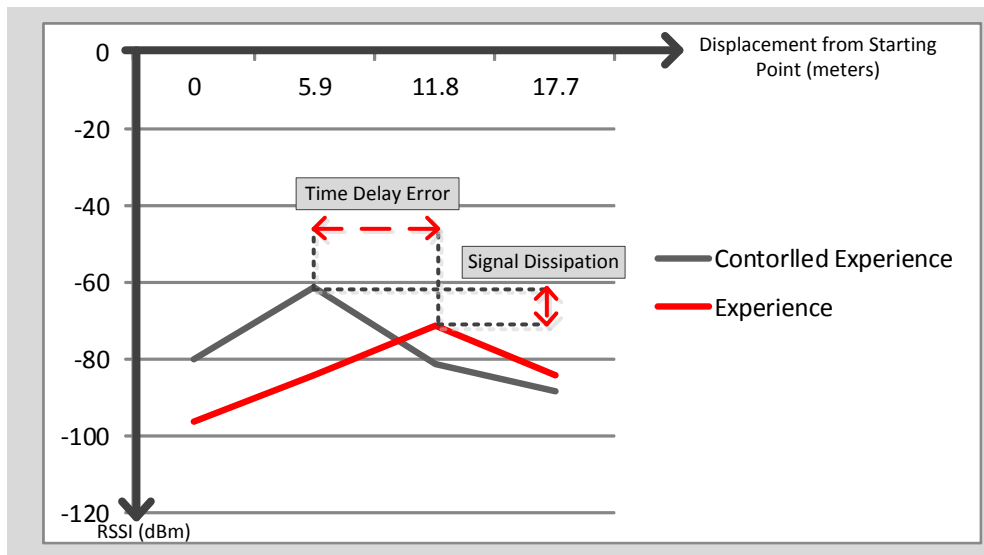


Figure 4.6 – RSSI against Displacement graph of Delay Time Error Exp.



## 2. Google Map API

### Ground Overlays

In order to set up an indoor map, we decide to overlay the indoor map on google map. It is very convenience that users can find their current location before go inside the carpark through GPS and they don't need to switch another app. Figure 4.7 and 4.8 has showed the effect of ground overlays.



Figure 4.7 - Before overlaid



Figure 4.8 - After overlaid

By using `addGroundOverlay()` provided by Google Map Library, we just need two point (left bottom and right top), then the indoor will be added successfully (Figure 4.9).

```
private void setUpMap() {  
    mMap.addGroundOverlay(new GroundOverlayOptions().  
        image(BitmapDescriptorFactory.fromBitmap(bm)).anchor(0, 1).positionFromBounds(bounds));  
}
```

Figure 4.9 – Android code used to overlay a bitmap on Google Map

### 3. Guidance Pointer

Guidance Pointer is an arrow that always pointing to the next Beacon Region no matter which direction user is facing to. As shown in figure 4.10, no matter how I place my smartphone, the guidance pointer is always pointing to the Beacon. It means that it is a relative direction (left, right) instead of absolute direction (N/W/S/E direction). User can follow the direction to get to the destination without thinking, just need to follow the arrow and go.

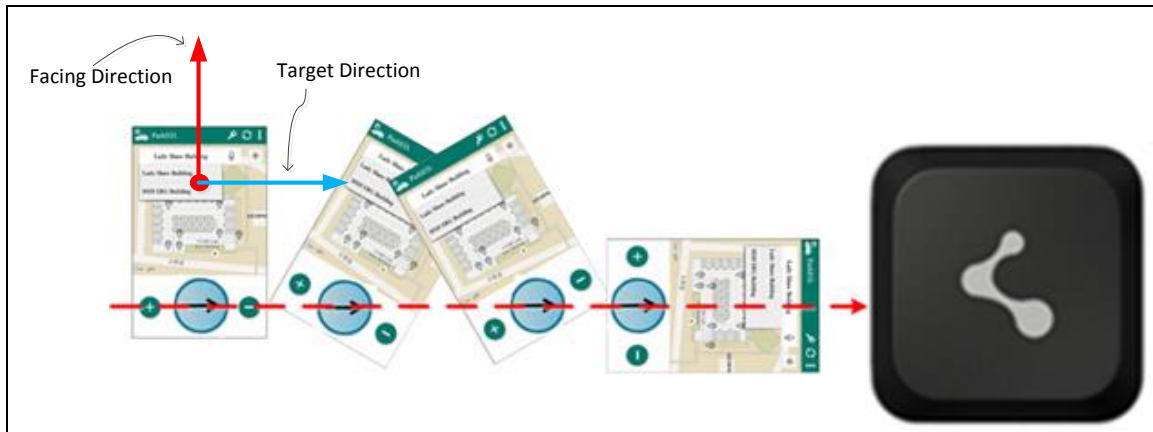


Figure 4.10 – Guidance Pointer is always pointing to next Beacon

In order to obtain the relative direction, we need 2 important things:

- i. The absolute direction pointing to next Beacon.
- ii. The absolute direction that user currently facing.

We can obtain the first information from CMS through the API. For example, Beacon A is in the north of Beacon B. For the second information, we required to mix use of the magnetic sensor and accelerometer sensor embedded in smartphone to become a compass, so that we know the absolute direction that user currently facing.

First, we need to initialize the sensors in *onCreate()* as shown in Figure 4.10.1.

```
private SensorManager sensorManager;  
//Compass  
sensorManager.registerListener(this, sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION), SensorManager.SENSOR_DELAY_GAME);  
sensorManager=(SensorManager) getSystemService(SENSOR_SERVICE);
```

Figure 4.10.1 – Sensor Initialization

After that, we can get the sensor reader ( *radius degree* ) in the event listener — *onSensorChanged()* (Figure 4.10.2)

```
@Override↓  
public void onSensorChanged(SensorEvent event)↓  
{↓  
    float degree=Math.round(event.values[0]);↓  
  
    // ...↓
```

Figure 4.10.2 – Sensor Reading

The degree can be converted into absolute direction as showed in figure 4.10.3:

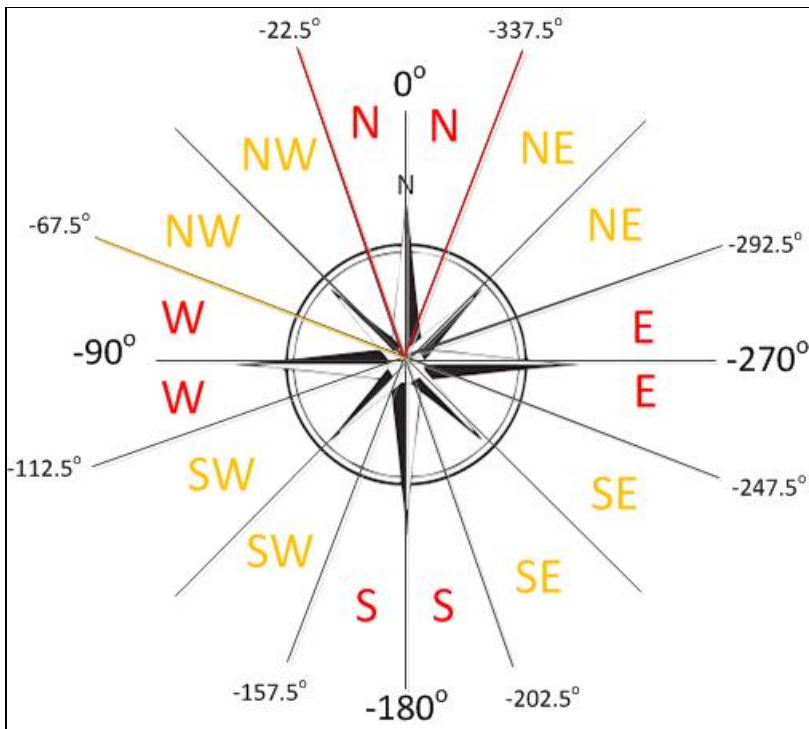


Figure 4.10.3 – Degree (Sensor Reading) to Absolution Direction conversion

At last, to convert two absolute directions to a relative direction is simple. The degree different between “The absolute direction pointing to next Beacon” and “The absolute direction that user currently facing” is equal to the relative degree as well as the relative direction.

$$\textit{relative degree} = \textit{current facing degree} - \textit{target degree}$$

## 4. Notifications

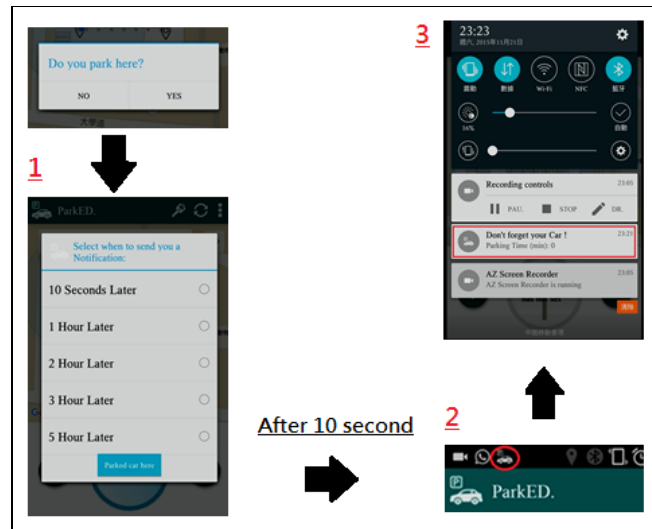


Figure 4.11 – whole process of getting a parking notification

Step 1) After user confirms parking, the parking time will record in the device.

```
carParkingLocation = lastKnownBeacon.getName();
carParkingDate = new Date();

// Record the info into text file
SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
Date curDate = new Date(System.currentTimeMillis());
String dateTimeString = formatter.format(curDate);

writeToFile(" " + carParkingLocation + "@" + dateTimeString);
```

Figure 4.12 – Storing the parking location and parking timestamp into a file

Step 2) Then app will show up an alert dialog to let user to choose parking time.

```
// Ask user to set notification
AlertDialog.Builder builderSingle = new AlertDialog.Builder(MainActivity.this);
builderSingle.setIcon(R.drawable.parking12);
builderSingle.setTitle("Select when to send you a Notification:");

final ArrayAdapter<String> arrayAdapter = new ArrayAdapter<>(
    MainActivity.this,
    android.R.layout.select_dialog_singlechoice);
arrayAdapter.add("10 Seconds Later");
arrayAdapter.add("1 Hour Later");
arrayAdapter.add("2 Hour Later");
arrayAdapter.add("3 Hour Later");
arrayAdapter.add("5 Hour Later");
```

Figure 4.13 – Alert dialog for user to select notification time

Step 3) After select the parking time, alarm will be set until it is time to activate *pendingIntent*, then notification will show up.

```
if (which == 0) {
    Intent intent = new Intent(MainActivity.this, CustomAlarm.class);
    intent.putExtra("parkTime", carParkingDate.getTime());
    PendingIntent pendingIntent = PendingIntent.getBroadcast(MainActivity.this, 0,
        intent, PendingIntent.FLAG_ONE_SHOT);
    am.set(AlarmManager.RTC_WAKEUP,
        System.currentTimeMillis() + (10 * 1000), pendingIntent);
}
```

Figure 4.14 – Alarm to activate Notification

Step 4) Notification will be set up in *CustomAlarm.class* which extends *BroadcastReceiver*.

```
public class CustomAlarm extends BroadcastReceiver {
    NotificationManager nm;

    @Override
    public void onReceive(Context context, Intent intent) {
        nm = (NotificationManager) context
            .getSystemService(Context.NOTIFICATION_SERVICE);
        long parkTime = intent.getExtras().getLong("parkTime");
        CharSequence from = "Don't forget your Car !";
        CharSequence message = "Parking Time (min): " + (new Date().getTime() - parkTime)/(1000*60);
        PendingIntent contentIntent = PendingIntent.getActivity(context, 0,
            new Intent(), 0);
        Notification notif = new Notification(R.drawable.parking12,
            "Parking Time(min): " + (new Date().getTime() - parkTime)/(1000*60), System.currentTim
        notif.setLatestEventInfo(context, from, message, contentIntent);
        nm.notify(1, notif);
    }
}
```

Figure 4.15 – A custom notification class which extends *BroadcastReceiver* class

## 5. Calling API

Since the API provided by CMS is accepting a HTTP POST request only, so here we will use HTTP Client to send POST to API to obtains data in JSON format. The detailed API specification is in the section *Application Programming Interface (API) Specification of Design Specification* in this report.

```
public static String API_getFloorPlan( String cpid) {
    List<NameValuePair> params = new ArrayList<>();
    params.add(new BasicNameValuePair("cpid", cpid));

    String returnStr;
    try {
        returnStr = getFloorPlan("jsonGetFloorplan.php", params);
    } catch (ClientProtocolException e) {
        return "#CP error: " + e.getMessage();
    } catch (IOException e) {
        return "#IO error: " + e.getMessage();
    } catch (Exception e) {
        return "#error: " + e.getMessage();
    }
    return returnStr;
}
```

Our Android app cannot work without communication to server. We use HttpClient to connect server which is very useful. By using HttpPost, we can get string of JSON from HttpResponse.

Figure 4.16 – Packing the HTTP POST Parameters

```
private static String getFloorPlan(String servletCall, List<NameValuePair> params) throws ClientProtocolException, IOException, Exception {
    HttpPost httpRequest = new HttpPost(API + servletCall);
    httpRequest.setEntity(new UrlEncodedFormEntity(params, HTTP.UTF_8));
    final HttpParams httppara = new BasicHttpParams();
    HttpConnectionParams.setConnectionTimeout(httppara, 10000);
    HttpResponse httpResponse = new DefaultHttpClient(httppara).execute(httpRequest);
    if (httpResponse.getStatusLine().getStatusCode() == 200) {
        String strResult = EntityUtils.toString(httpResponse.getEntity());
        return strResult;
    }
    return null;
}
```

Figure 4.17 – Using HTTP Client to send POST request to API of CMS

After we get the string of JSON, we use the function of *API\_GetFloorPlans.class* to parse the string to decode the information from JSON format string, and save the data into static variable.

Usage:

```
API_GetFloorPlans.parseJson(result);
```

Code:

```
public static void parseJson(String json) {
    JSONObject obj;
    try {
        obj = new JSONObject(json);
        success_tag = obj.getString("success_tag");
        if (success_tag.equals("success")) {
            floor_count = Integer.parseInt(obj.getString("count"));
            cpid = Integer.parseInt(obj.getString("CPID"));
            allFloors = new CarparkFloor[floor_count];

            for (int i = 0; i < floor_count; i++) {
                JSONObject tempFloor = new JSONObject(obj.getString(i + ""));
                allFloors[i] = new CarparkFloor(cpid, Integer.parseInt(tempFloor.getString("floor")), tempFloor.getString("image"));
                int beaconCount = Integer.parseInt(tempFloor.getString("count"));
                for (int j = 0; j < beaconCount; j++) {
                    JSONObject tempBeacon = new JSONObject(tempFloor.getString(j + ""));
                    allFloors[i].addMarker(tempBeacon.getString("BName"), Double.parseDouble(tempBeacon.getString("X")),
                        Double.parseDouble(tempBeacon.getString("Y")), Integer.parseInt(tempFloor.getString("floor")) );
                }
            }
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Figure 4.18 –Decoding the JSON String into static variable

## 5.12 Database Specification

After we discussed our design of Car-park Guidance System (CGS), we designed some tables that should exist in the database. Here is our proposed database, and the Relational diagram of our database design (Figure 5.1).

lyu1502_user	<p>UID: (User ID) Primary key for User. Maximum 16 characters. Can't be null.</p> <p>Name: (User Name) Maximum 16 characters. Cannot be null.</p> <p>Privilege: (User Privilege) Tiny Integer. Distinguish user's permission.</p> <p>loginID: (Login ID) Maximum 16 characters. Cannot be null</p> <p>loginPswd: (Login Password) Maximum 16 characters. Cannot be null.</p>
--------------	--

lyu1502_userprivilege	<p>Privilege: (User Privilege) Tiny Integer. Primary key for User Privilege.</p> <p>Pname: (Privilege Name) Maximum 16 characters. Cannot be null.</p> <p>Desc: (Privilege Description) Maximum 255 characters.</p>
-----------------------	---

lyu1502_beacon	<p>BID: (Beacon ID) Integer. Primary Key for Beacon. Can't be null.</p> <p>Type: (Beacon Type) Maximum 1 characters. B = iBeacon, E = Eddystone.</p> <p>UUID: (UUID of iBeacon) Maximum 36 characters.</p> <p>Major: (Major of iBeacon) Integer.</p> <p>Minor: (Minor of iBeacon) Integer.</p> <p>EID: (Ephemeral ID of Eddystone) Maximum 12 characters.</p> <p>BName: (Beacon Name/Location Name) Maximum 36 characters.</p> <p>BDesc: (Beacon Description) Maximum 255 characters.</p> <p>CPID: (Carpark ID) Integer. Belong to which Car Park.</p> <p>Floor: (Floor) Integer. Belong to which floor of car park.</p> <p>X: (Coordinate X) Float. Relative coordinate on floor plan. X = [0.0,1.0]</p> <p>Y: (Coordinate Y) Float. Relative coordinate of floor plan. Y = [0.0, 1.0]</p>
----------------	---



lyu1502_carpark	<p>CPID: (Carpark ID) Integer. Primary Key for Car Park. Can't be null.</p> <p>Cname: (Carpark Name) Maximum 64 characters. Can't be null.</p> <p>Cdesc: (Carpark Description) Maximum 64 characters.</p> <p>Latitude_LB: (Latitude of the most Left Bottom point) Double. Can't be null.</p> <p>Longitude_LB: (Longitude of the most Left Bottom point) Double. Not null.</p> <p>Latitude_RT: (Latitude of the most Right Top point) Double. Can't be null.</p> <p>Longitude_RT: (Longitude of the most Right Top point) Double. Not null.</p>
-----------------	---

lyu1502_carroutingtable	<p>RTID: (Route ID) Integer. Primary Key for Car Routing Table. Not null.</p> <p>Source: (Source's Beacon ID / Current Location) Integer.</p> <p>Destination: (Destination's Beacon ID) Integer.</p> <p>NextHop: (Next Hop's Beacon ID) Integer.</p> <p>NextHopDirection: (N/E/S/W/NE/NW/SE/SW) Max. 3 characters.</p> <p>HopCount: (Number of Beacons need to pass through) Integer.</p> <p>RouteCost: (Total Cost of Route) Integer.</p> <p>RouteType: (Route Type) V = Vehicle, P = People, VP = Both</p> <p>RevisionNumber: (Record Revision No.) Integer. (highest = latest)</p>
-------------------------	---

lyu1502_graph	<p>Source: (Source's Beacon ID / Current Location) Integer.</p> <p>Destination: (Neighbor's Beacon ID) Integer.</p> <p>Cost: (Edge Weight) Integer</p> <p>Direction: (N/E/S/W/NE/NW/SE/SW) Max. 3 characters.</p> <p>RouteType: (Route Type) V = Vehicle, P = People, VP = Both</p> <p>DynamicCost: (Cost Modifier) Total Cost = Cost + DynamicCost</p> <p>ExpireTimer: (Expire Timer) When timer expired, reset Dynamic Cost.</p>
---------------	--

lyu1502_log	<p>Time: (Timestamp) MySQL Timestamp. Can't be null.</p> <p>TAG: (Log Tag) Categorize the LOG Message. Max. 32 Char. Not null.</p> <p>Message: (Log Title/Summary) Max. 255 characters. Not null.</p> <p>Detail: (Detailed Information) Max. 512 characters. Not null.</p> <p>Keyword{ 1-5}: (Key Information for analytic purpose) Max. 255 chars.</p>
-------------	---

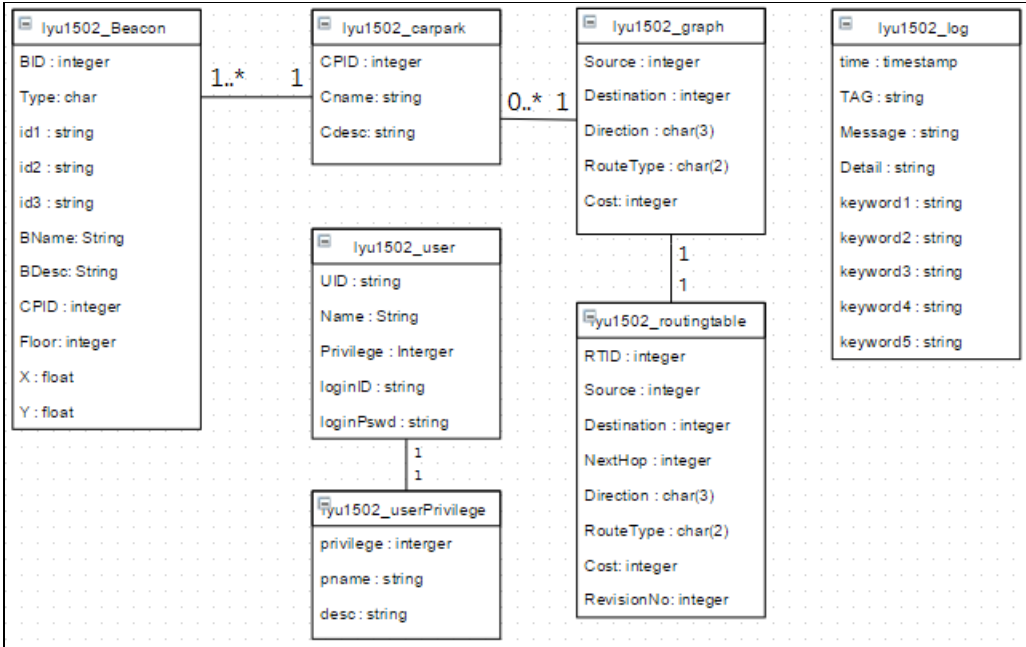


Figure 5.1 – Relational Diagram of database

## Chapter 6 - Project Implementation (First term)

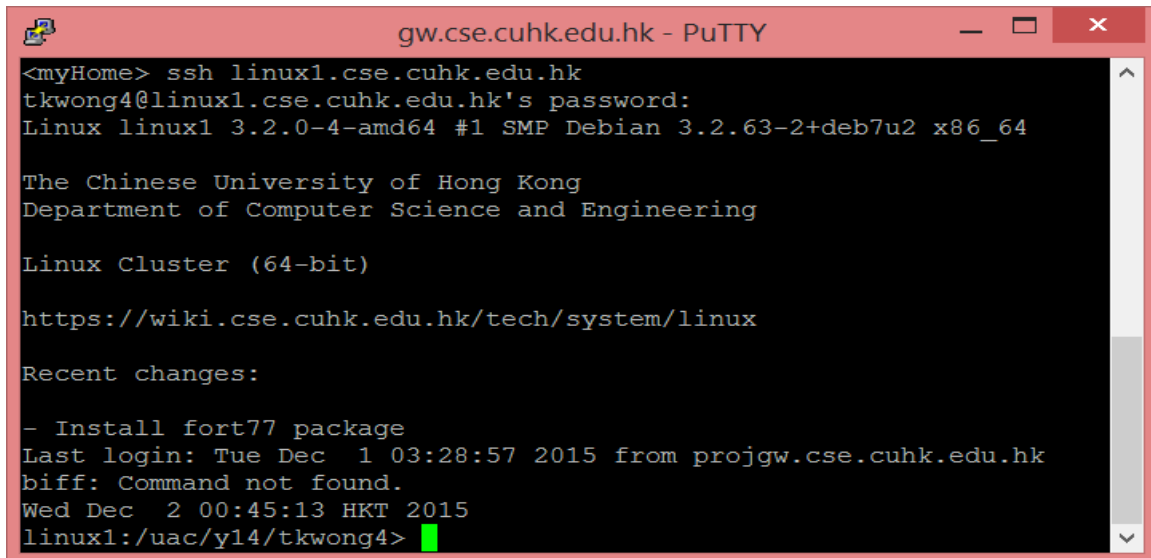
In this part, we will talk about how we implement our system into the Lady Shaw Building's car park (LSB car park). The content including how we deploy the beacons in LSB car park, how we bring our design into real world. First things first, we need to find web hosting service to host our Content Management System (CMS).

### Web Hosting for Content Management System (CMS)

CSE department provides web hosting service based on LAMP model for CSE student to use. Therefore, we can host the CMS on our personal page. Following the instructions given by technical support, we successfully host our CMS on CSE's application server.

Here are the steps for hosting:

1. Connect to *linux1.cse.cuhk.edu.hk* through SSH.



```
gw.cse.cuhk.edu.hk - PuTTY
<myHome> ssh linux1.cse.cuhk.edu.hk
tkwong4@linux1.cse.cuhk.edu.hk's password:
Linux linux1 3.2.0-4-amd64 #1 SMP Debian 3.2.63-2+deb7u2 x86_64

The Chinese University of Hong Kong
Department of Computer Science and Engineering

Linux Cluster (64-bit)

https://wiki.cse.cuhk.edu.hk/tech/system/linux

Recent changes:

- Install fort77 package
Last login: Tue Dec  1 03:28:57 2015 from projgw.cse.cuhk.edu.hk
biff: Command not found.
Wed Dec  2 00:45:13 HKT 2015
linux1:/uac/y14/tkwong4>
```

2. On the home directory, create a directory named */www* by using *mkdir* command.

*mkdir ~/www*

3. Upload the HTML, PHP files to the *~/www* directory.
4. Change the files permission to 755 to allow everyone to execute the PHP files, but not to modify the content. Using *chmod* with argument *-R* can change the files permission.

*chmod -R 755 ~/www*

5. The web site is already host by CSE department's application server. The URL is:

[http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms\\_v2/](http://appsrv.cse.cuhk.edu.hk/~tkwong4/cms_v2/)

## Database Server

CSE department provides MySQL database server for CSE student to use. Therefore, we can use it to be our database server. There is a limitation on accessing to database server, which is it only allows connection from *linux1.cse.cuhk.edu.hk*. Therefore, User App cannot directly connect to the database server, whereas it must use API to obtain data from CMS. Following the instructions given by technical support, we successfully host our database tables on CSE's database server.

Here are the steps for hosting:

1. Creating tables using MySQL queries on MySQL database server. For the detailed SQL queries, please see the appendix — MySQL database script.
2. In PHP, connect to *appsrvdb.cse.cuhk.edu.hk* using the given account name and password.
3. Done.

## Beacon installation in Lady Shaw Building Car Park

Since we don't have permission to install any device to LSB car park, so the installation is just temperately, and will be removed after the testing. For the temperately installation, we used plastic tape to stick Beacons on to the ceiling of LSB car park. Following the Beacon deployment plan in design overview, we placed our Beacons in the turning points and cross roads of LSB car park. Figure 1.1 has shown the deployment of Beacons in LSB car park.

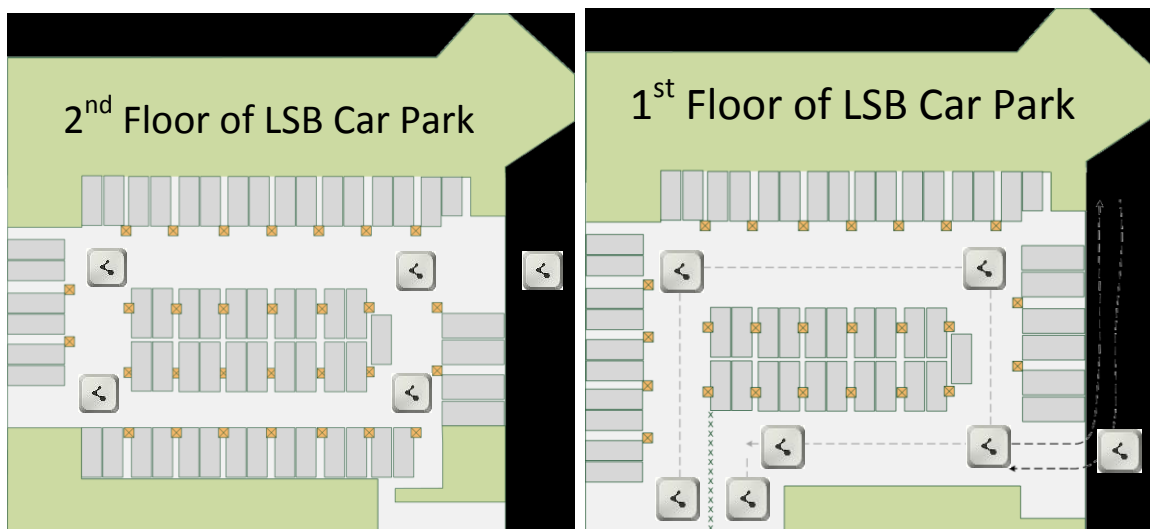
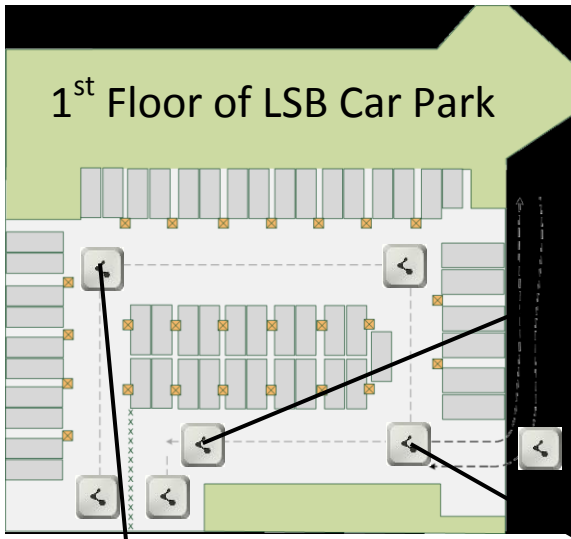


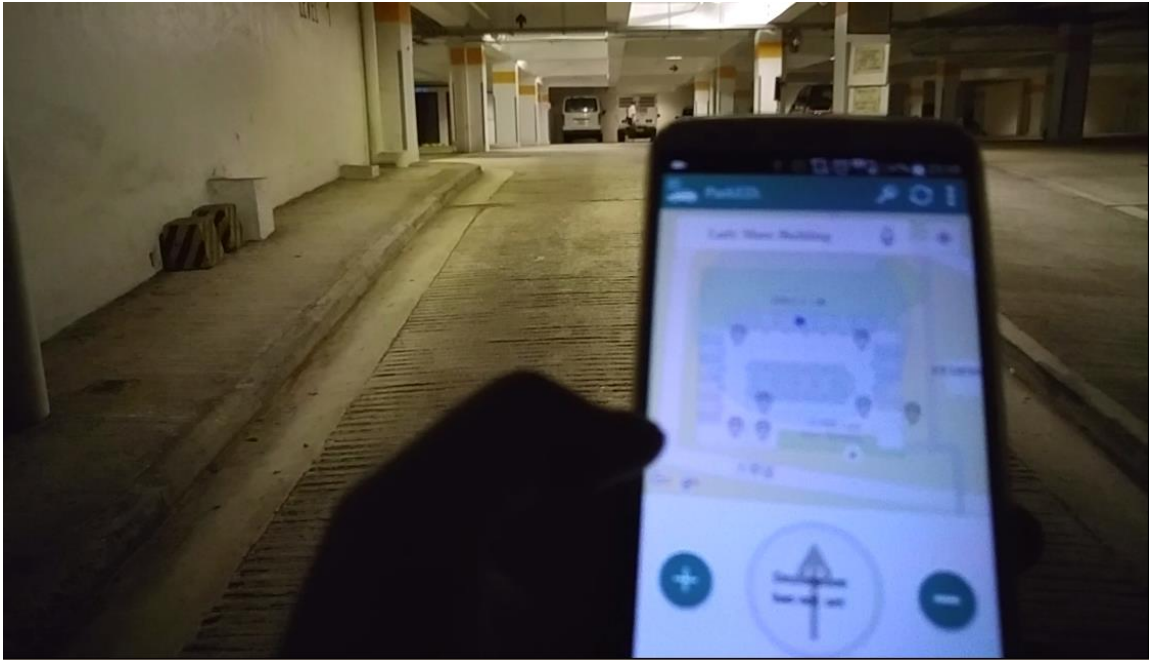
Figure 1.1 – Deployment of Beacons in LSB car park

Here are some photos of Beacons installed on ceiling of LSB car park.

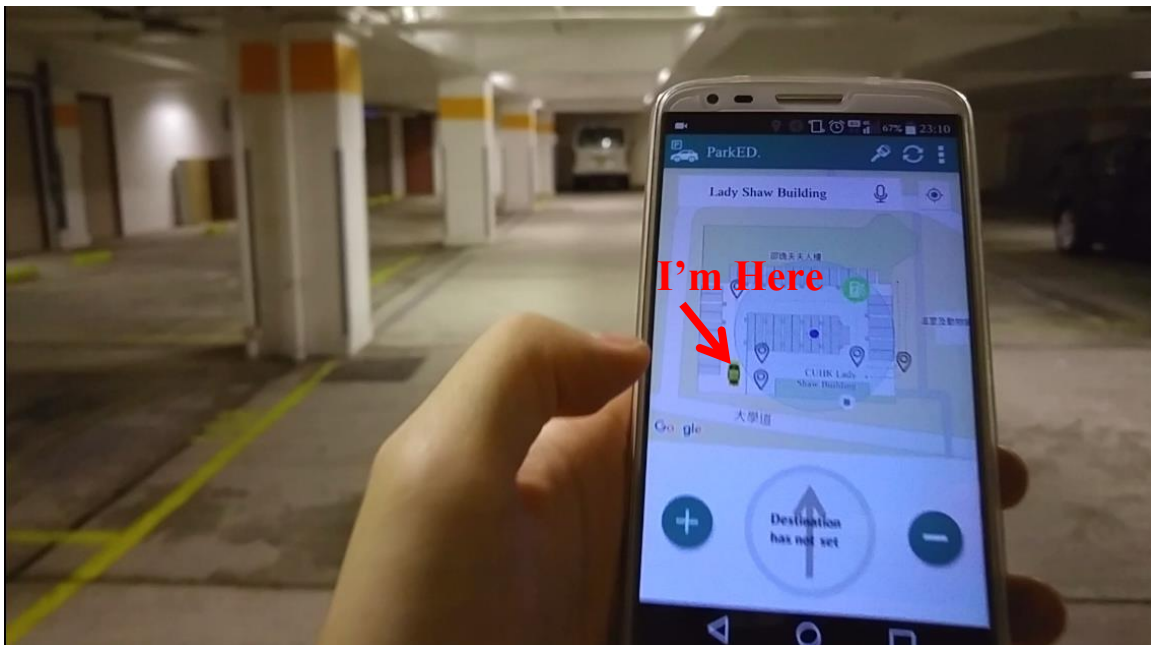


## Testing the Guidance System

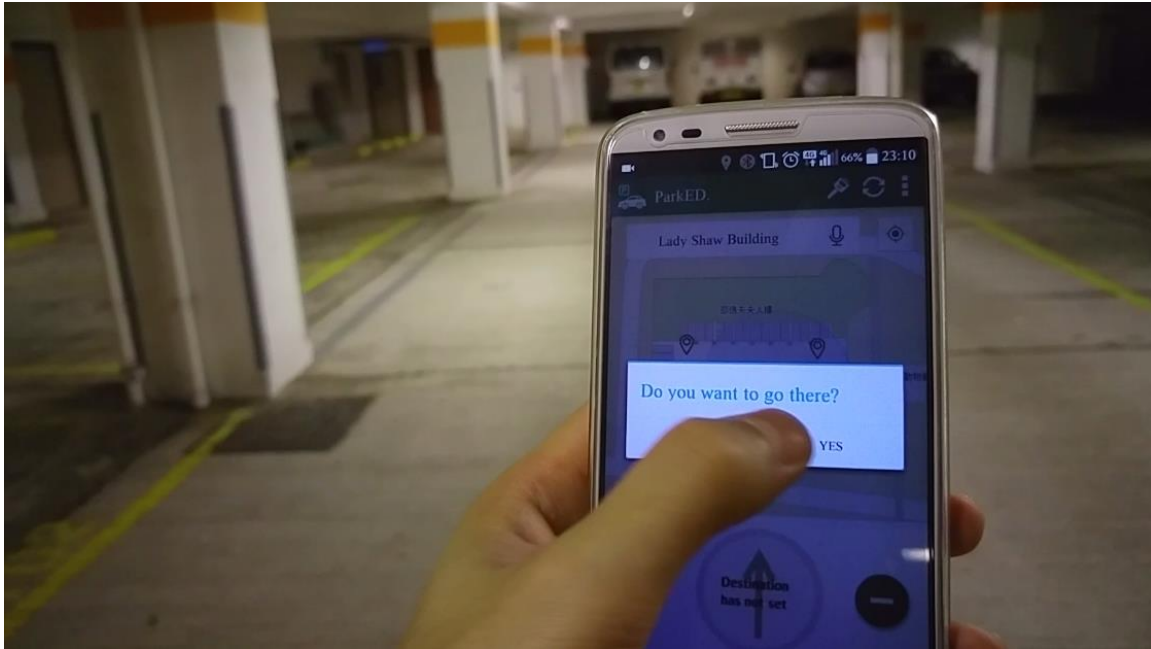
1. Starting from the entrance of LSB Car Park



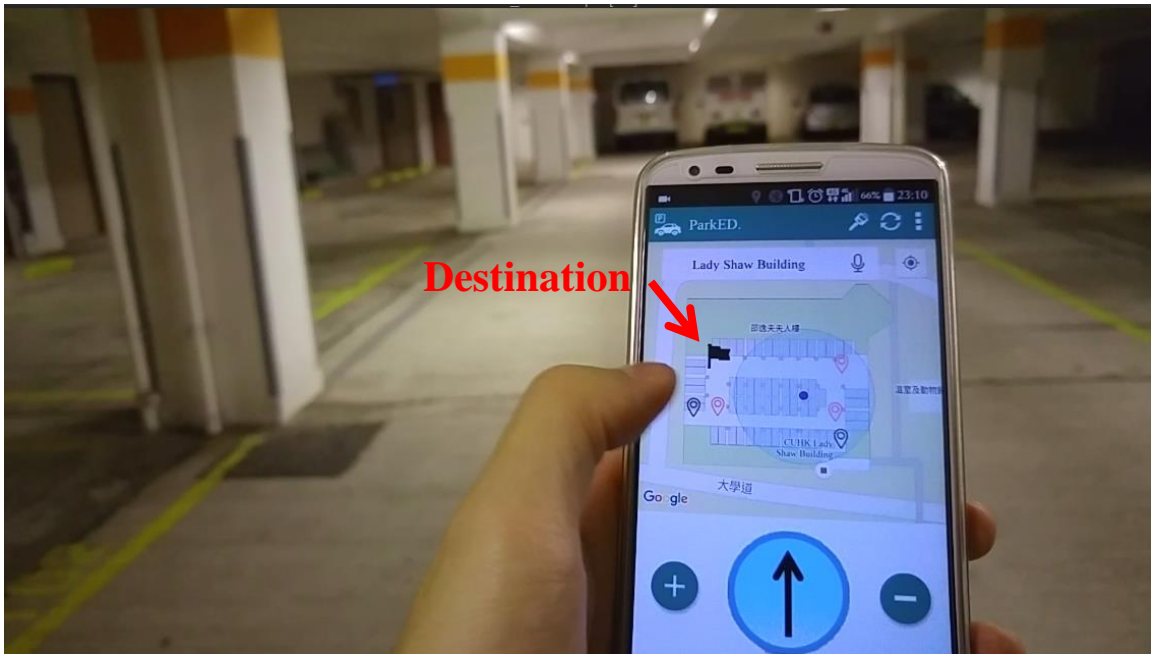
2. User App detects the Beacon signal which installed near to the entrance of LSB car park. And show “car icon” on the map to indicate my physical location on floor plan.



3. Select the parking space on 2<sup>nd</sup> floor, so the app will guide me to go there.



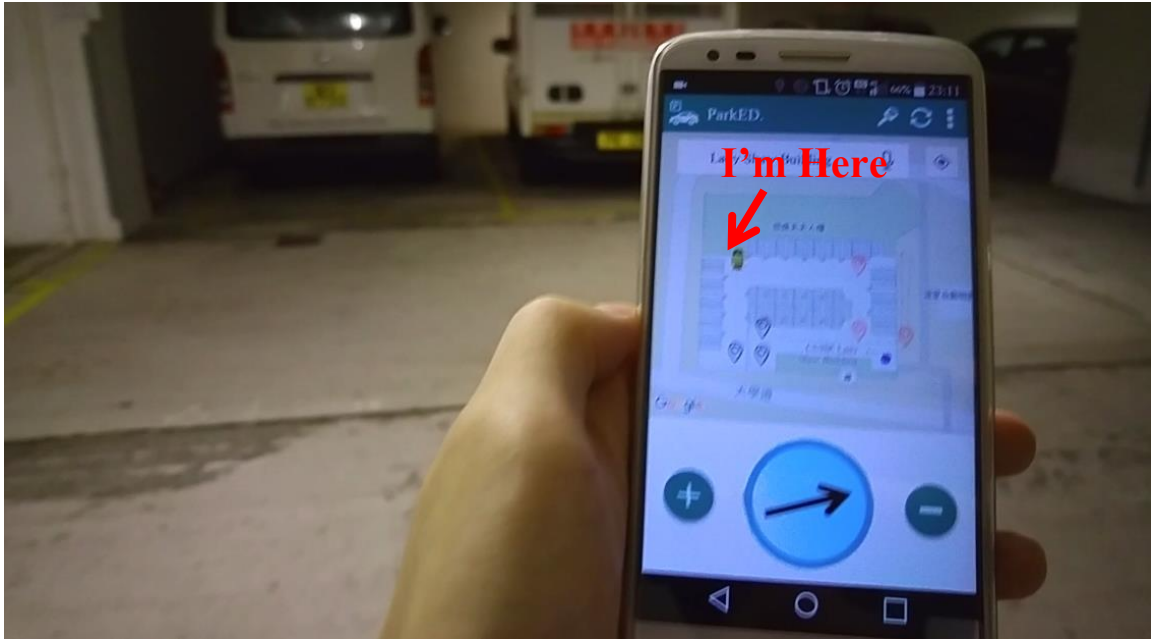
4. Guidance pointer is telling me to go forward.



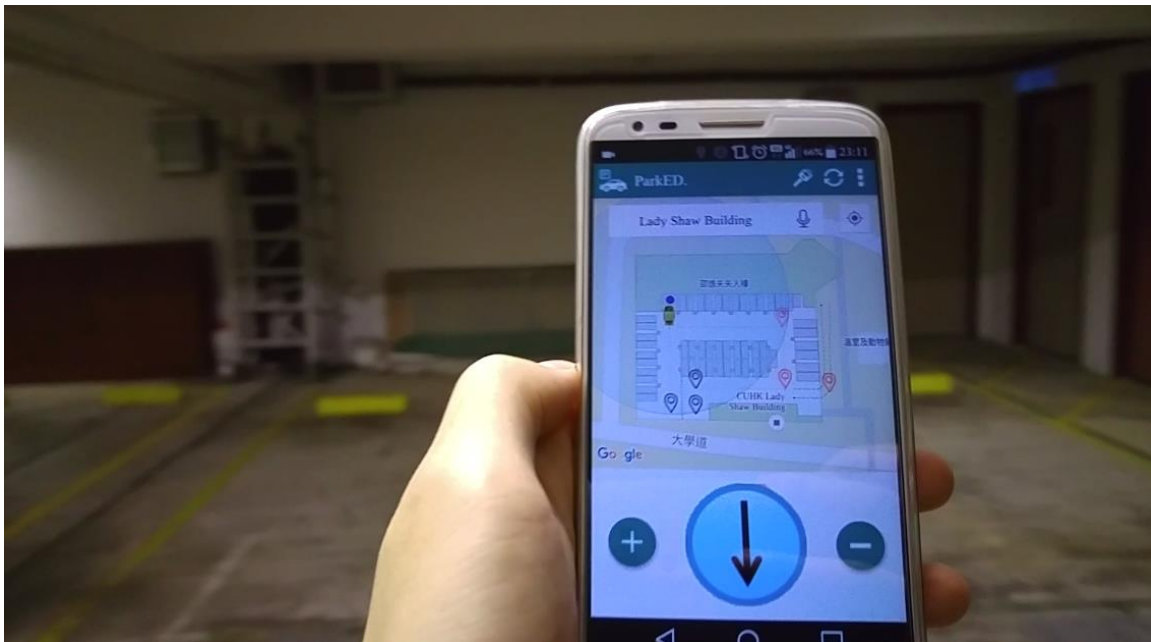
**Destination**

**Guidance Pointer**

5. The guidance pointer is telling me to turn Right

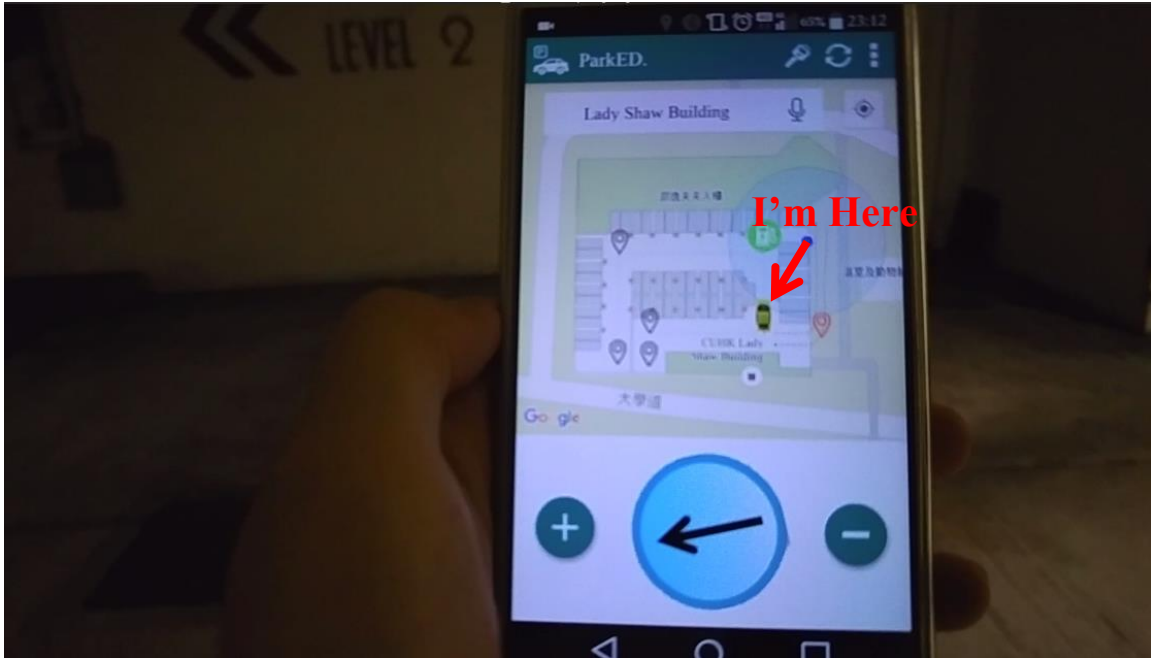


6. The guidance pointer telling me to turn around if I facing to another direction

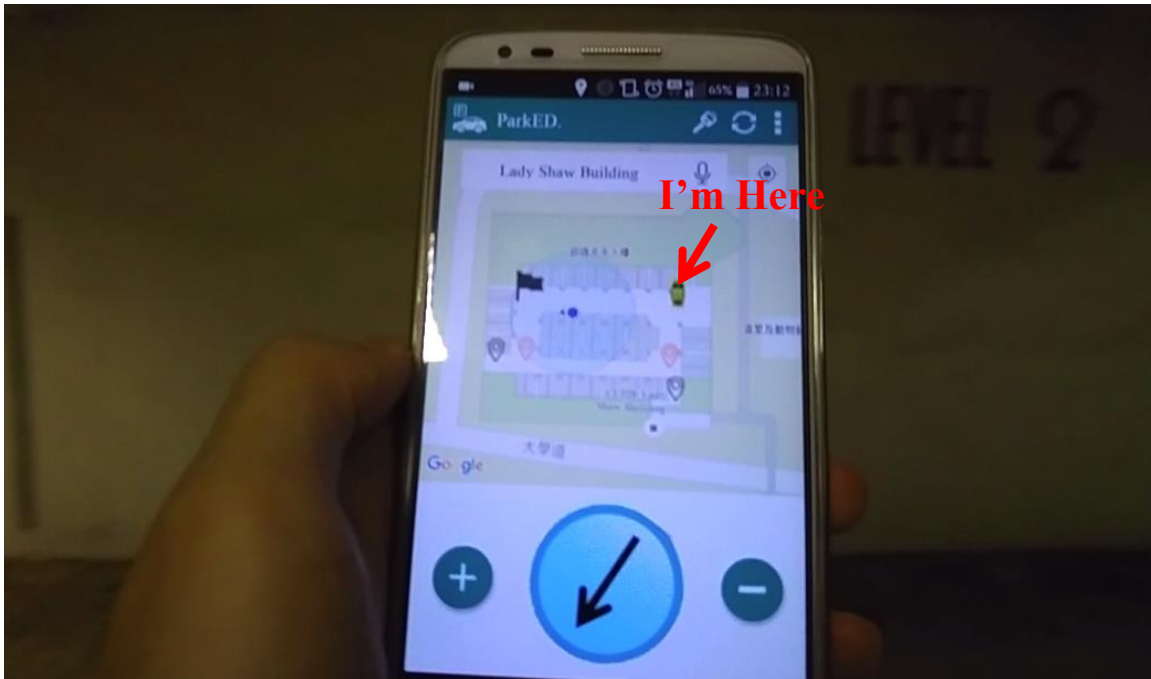




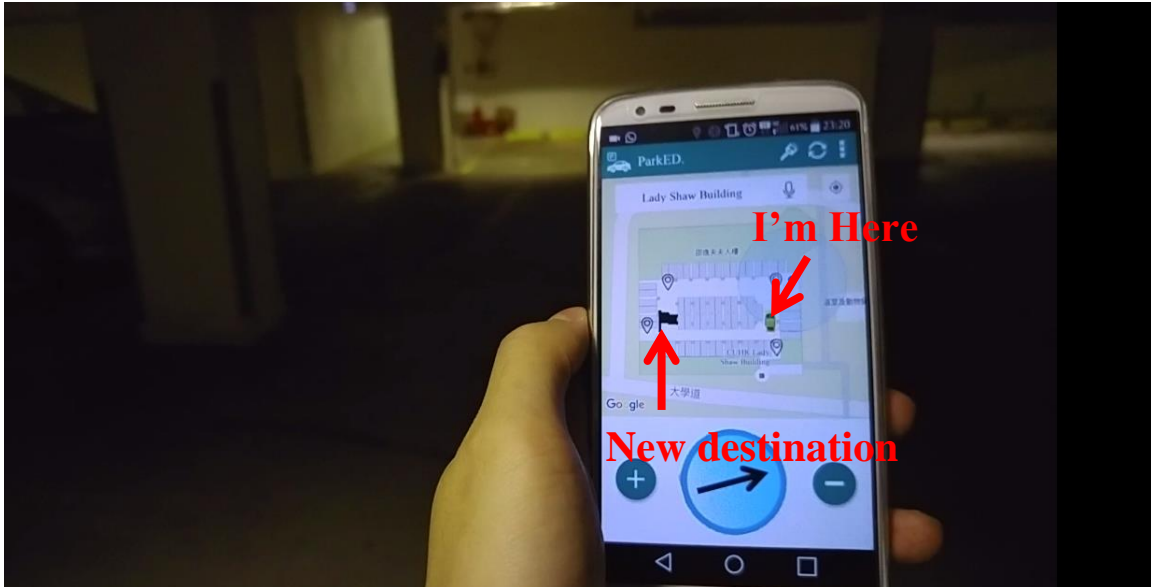
7. The guidance pointer is telling me to turn left to go to 2<sup>nd</sup> floor.



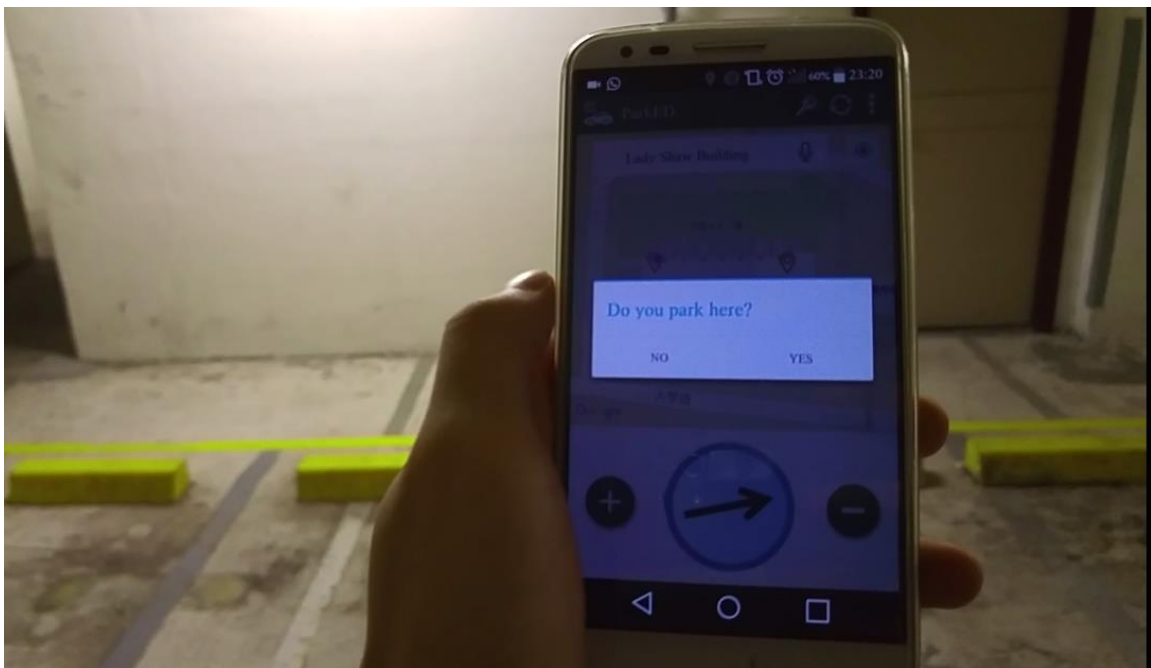
8. Come to the 2<sup>nd</sup> floor, the guidance point is still guiding me to the destination.



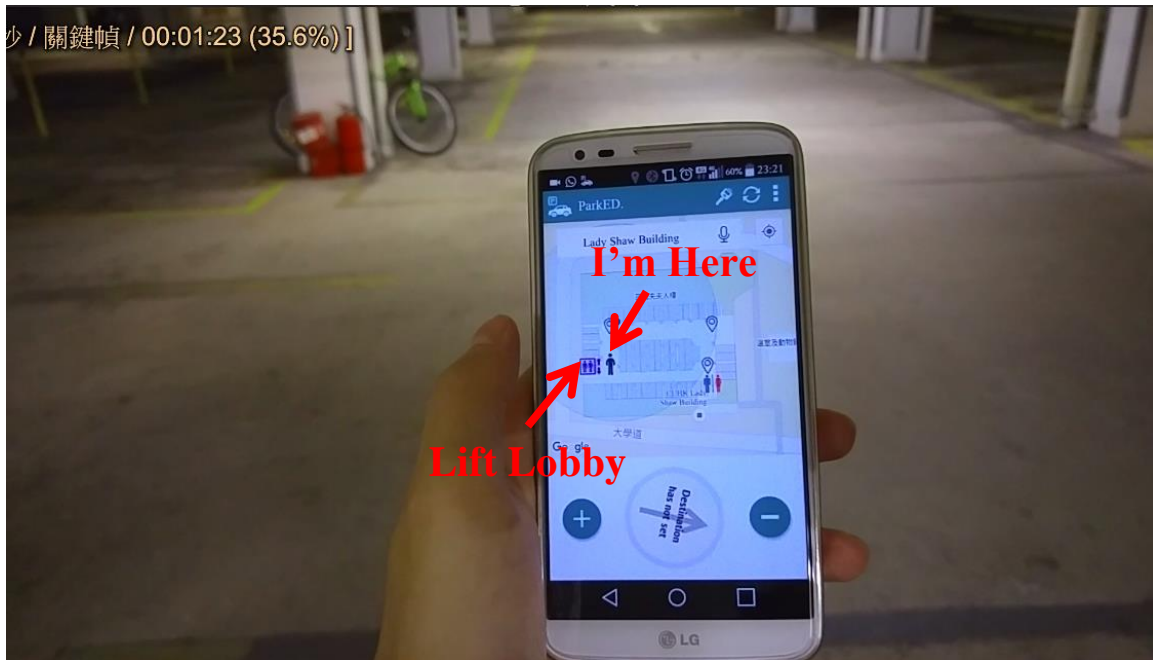
9. We can change the destination at any time.



10. Arrive the parking space. User App asks me if I want to park here.



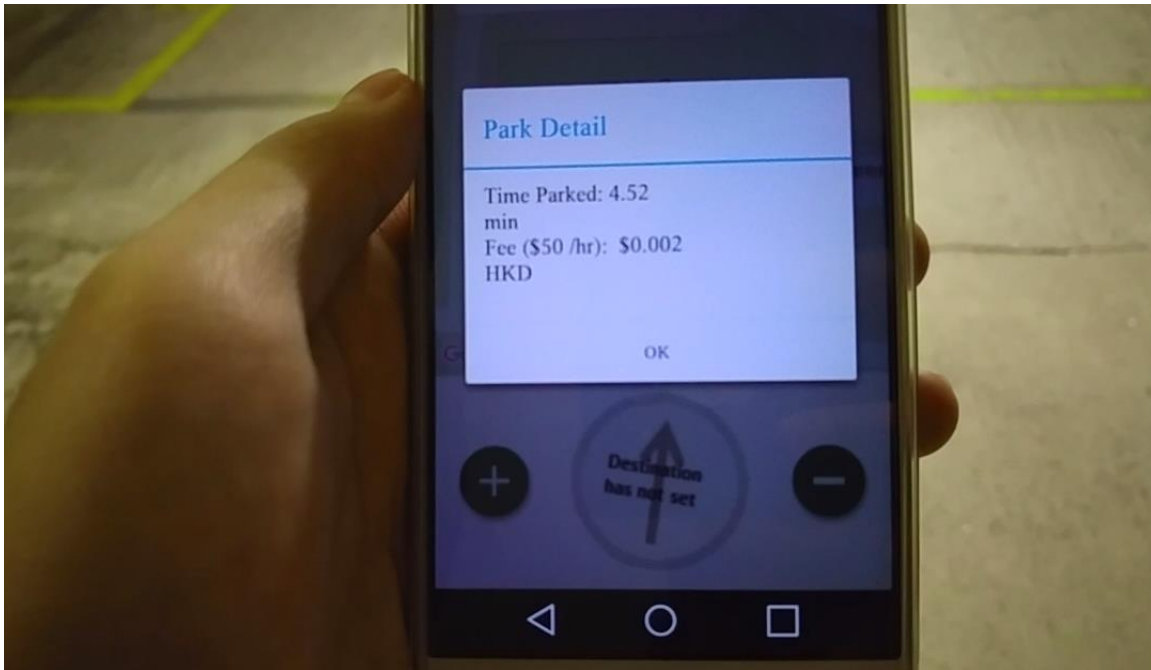
11. After the car is parked, the UI changes a little bit. First, it will show where my car is parked. Second, it will show where am I. Third, it will show the location lift lobby. Guidance Pointer is blurred, since guidance function is not in use.



12. After I return to the car park. I can ask for guidance to get back my car by clicking the “Key” button.



13. After I get back my car, some parking information is shown.



## **Chapter 7 - Goal for the Second Term**

In the first term, we have achieved indoor guidance function of our system. In the second term, we will focus on several things to improve our system, so that user can have better user experience. We planned the following milestones for the second term:

### **Dynamic Routing (dynamic route cost based on different situation)**

- According to different situation (E.G. dead lock prevention / congestion detection or prevention)
- App for car park administrator to enable / disable Beacon temperately to change the routing.
- A single route has multiple path cost for multiple group of users. (User with passcode can use shortcut, otherwise, use longer path)

### **Data Analytic**

- Logging history is already available in our system in 1<sup>st</sup> term. Every action will generate a log history to the log database. Now, the question is, can we analytic the logs and filter some useful information to generate statistic information (For example, heat-map of parking space usage in car-park, demand of eCharger in car parks...etc.).
- Base the analytic report, car park administrator can use it as a reference to make some administrative decision. (E.G. should car park admin buy new electronic vehicle charger (EVC)? Where to deploy the EVC?)

### **Graph Database Engine**

- Currently, our system is based on relational database. Calculating shortest path from a relational engine is not fast enough. In our design, the size of routing table growth exponentially with the growth of number of Beacon in system. It's not good for large scale handling. So, we may need to change to use graph database instead of relational database to make our algorithm easier.
- Titan, is a distributed graph database for large scale handling. We will migrate our graph algorithm from MySQL to Titan graph engine in the 2<sup>nd</sup> term.

### **Prediction on user preference**

- Predict user's preference on which parking space he want to park.
- For example, every time when user come to the car park is parked in the specific location, then our system will recognized it. When user come to the same car park next time, our app will automatic guide driver to that parking space.

## **Chapter 8 - Conclusion**

To conclude our project, our topic is Indoor Guidance Application using Beacon Technology. We designed an indoor navigation system that provides guidance for driver who is driving a vehicle. Nowadays, there are a lot of indoor navigation systems, but most of them focus on the behaviors of walking like exhibition indoor guidance. But there is no or selfdom mobile application will do the indoor navigation for driving. This is our motivation to develop an INS especially for drivers in car parks. Our project is aimed to provide a complete indoor car park guidance service, to provide further services for drivers not only focus on parking, to analyze user experience and suggest user a prefer parking space, and to generate the statistic report to car park administrative person.

We've started our project during the summer 2015, and up to now we have achieve 50% of our project objectives. We will keep improving what we have done in 1<sup>st</sup> semester and adding new element to our car-park guidance system. We've planned some millstones for the next semester, they are including but not limited to "Dynamic Routing", "User Experience analysis", "Graph Database Engine to replace MySQL", and "Statistic Report".

Time is so tiny to do many research tasks in 1<sup>st</sup> term such as doing experience for feasibility study, studying new Beacon technology (Eddystone), designing the system layout...etc. In 2<sup>nd</sup> term, we expect we will spend more time on implementation instead of doing research and designing the system's UI or architecture.

For us, this is our 1<sup>st</sup> time to develop is a complete system including both front-end and back-end. The knowledge required to design and implement front-end and back-end are totally different. For example, we required the knowledge on mobile application development to build a mobile app as our front-end system. We required the knowledge on LAMP (Linux, Apache, MySQL, and PHP) to build the content management system as the back-end of our system. So we've learnt much and have great improvement on technical skill. Beside the technical skill, we also learnt some soft skills such as how to develop a system by following the SDLC (System Development Life Cycle), time management on doing list of tasks under limited time.

## Acknowledgement

We would like to cherish this opportunity to express our thanks to our supervisor Prof. Michael Lyu for providing a lot of helpful advice. Also, Professor Michael Lyu is willing to spend time to keep tracking our progress by weekly meeting.

Moreover, we would like to thank Mr. Edward Yau, Hon Hei, technical manager of VIEW Lab. He provided a lot of idea to us to make our project become more attractive and impressive.

Besides, he also provides a lot of technical advice on developing a web-based content management system during the non-regular meeting in summer 2015.

## Reference

- [1] Estimote Developer Site: <http://developer.estimote.com/eddystone/>
- [1] Eddystone's Github Page: <https://github.com/google/eddystone/blob/master/protocol-specification.md>
- [2] "Mobile Telephony Market". Bluetooth Special Interest Group. Retrieved January 16, 2014 on <http://www.bluetooth.com/Pages/Mobile-Telephony-Market.aspx>
- [2] "Bluetooth Low Energy", Wikipedia : [https://en.wikipedia.org/wiki/Bluetooth\\_low\\_energy#cite\\_note-5](https://en.wikipedia.org/wiki/Bluetooth_low_energy#cite_note-5)
- [3] Stack Overflow Discussion Forum: <http://stackoverflow.com/questions/1347552/what-is-the-big-o-for-sql-select>
- [4] PHOTO SOURCE: <http://www.androidcentral.com/ntsb-recommends-states-ban-all-cell-phone-use-cars>
- [5] "*An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications*" by R. Faragher, University of Cambridge, UK and R. Harle, University of Cambridge, UK.
- [6] "Will wireless interference and Wi-Fi impact beacons?" by Estimote, manufactory of Beacon (Available on: <https://community.estimote.com/hc/en-us/articles/200794267>)
- [7] LAMP, Wikipedia: [https://en.wikipedia.org/wiki/LAMP\\_\(software\\_bundle\)](https://en.wikipedia.org/wiki/LAMP_(software_bundle))
- [8] "JGraphT: <http://jgrapht.org/>
- [9] Android Developer Web Site: <http://developer.android.com/develop/index.html>
- [10] ALT SDK web site: <https://altbeacon.github.io/android-beacon-library/index.html>



## Appendix

### Appendix — MySQL database script

```
-- phpMyAdmin SQL Dump
-- version 4.0.8
-- http://www.phpmyadmin.net
--
-- 主機: appsrvdb.cse.cuhk.edu.hk
-- 產生日期: 2015 年 11 月 16 日 15:20
-- 伺服器版本: 5.0.51a-24+lenny5
-- PHP 版本: 5.4.4-14+deb7u14

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS
*/;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;

CREATE TABLE IF NOT EXISTS `lyu1502_beacon` (
  `BID` int(10) unsigned NOT NULL,
  `Type` varchar(1) NOT NULL default 'B',
  `UUID` varchar(36) default NULL,
  `Major` int(10) unsigned default NULL,
```

```
`Minor` int(10) unsigned default NULL,  
`EID` varchar(24) default NULL,  
`BName` varchar(36) NOT NULL,  
`BDesc` varchar(255) default NULL,  
`CPID` int(10) unsigned NOT NULL,  
`floor` int(11) NOT NULL,  
`X` double NOT NULL,  
`Y` double NOT NULL,  
PRIMARY KEY (`BID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE IF NOT EXISTS `lyu1502_carpark` (
```

```
`CPID` int(11) NOT NULL,  
`Cname` varchar(64) NOT NULL,  
`Cdesc` varchar(64) default NULL,  
`Latitude_LB` double NOT NULL default '0',  
`Longitude_LB` double NOT NULL default '0',  
`Latitude_RT` double NOT NULL default '0',  
`Longitude_RT` double NOT NULL default '0',  
PRIMARY KEY (`CPID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE IF NOT EXISTS `lyu1502_carparkfloor` (
```

```
`CPID` int(10) unsigned NOT NULL,  
`Floor` int(10) NOT NULL,
```

```

`Image` varchar(255) NOT NULL,
PRIMARY KEY (`CPID`,`Floor`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `lyu1502_carroutingtable` (
  `RTID` int(10) unsigned NOT NULL,
  `Source` int(10) NOT NULL,
  `Destination` int(10) NOT NULL,
  `NextHop` int(10) NOT NULL,
  `NextHopDirection` varchar(3) NOT NULL,
  `HopCount` int(11) default NULL,
  `RouteCost` int(11) default NULL,
  `RouteType` varchar(15) NOT NULL default "",
  `RevisionNumber` int(10) unsigned NOT NULL default '0',
  PRIMARY KEY (`Source`,`Destination`,`RouteType`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `lyu1502_graph` (
  `Source` int(10) NOT NULL,
  `Destination` int(10) NOT NULL,
  `Cost` int(10) unsigned NOT NULL,
  `Direction` varchar(3) default NULL,
  `RouteType` varchar(15) NOT NULL default "",
  `DynamicCost` int(11) NOT NULL default '0' COMMENT 'Not permanent. Volatile
Property.',
  `ExpireTimer` datetime default '0000-00-00 00:00:00',

```

```

PRIMARY KEY (`Source`,`Destination`,`RouteType`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
--
CREATE TABLE IF NOT EXISTS `lyu1502_log` (
  `Time` timestamp NOT NULL default CURRENT_TIMESTAMP on update
CURRENT_TIMESTAMP,
  `Tag` varchar(32) NOT NULL,
  `Message` varchar(255) NOT NULL,
  `Detail` varchar(512) default NULL,
  `Keyword1` varchar(255) default NULL,
  `Keyword2` varchar(255) default NULL,
  `Keyword3` varchar(255) default NULL,
  `Keyword4` varchar(255) default NULL,
  `Keyword5` varchar(255) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `lyu1502_user` (
  `UID` int(11) default NULL,
  `Name` varchar(16) default NULL,
  `Privilege` tinyint(4) default NULL,
  `loginID` varchar(16) default NULL,
  `loginPswd` varchar(16) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `lyu1502_userprivilege` (
  `Privilege` tinyint(4) NOT NULL,

```

```
`Pname` varchar(16) NOT NULL,  
`Desc` varchar(255) NOT NULL,  
PRIMARY KEY (`Privilege`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;  
  
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;  
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;  
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```