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

Indoor location service with iBeacon

LYU 1402 Final Year Project Report
Fall 2014

Wan Ka Ki 1155030692
Cheung Wing Long 1155028797

Supervised by Prof. Michael R.Lyu

This is a blank page.

Abstract

We are going to design and implement a library to achieve notification pushing which triggered by iBeacon. Since there is no similar product all over the world at this moment, it is hoped that our final year project could provide a library for the app developers who want to embed the iBeacon technologies into their app. In this project, we have firstly studied the iBeacon technical specifications and the ways to embed iBeacon into IOS and android app. Then, we decided to do the notification in IOS platform with iBeacon to achieve the goal of our project. We have through many creative ideas to introduce iBeacon into an app. Lastly, we make use of the iBeacon and try to push some advertisements to the devices in a perfect time and places.

Table of Content

Chapter 1 Introduction

1.1 Motivation

In this 21st century, smartphones have become a necessity for many people throughout the world. Today's smart phones are capable of not only receiving and placing phone calls, but also storing data, taking pictures, and even being used as walkie talkies, to name just a few of the available options. One of the key feature inside the phone is app. App is
a short term of "application software", which is a computer program designed to run on smartphone or tablet. Those innovative and creative app redefined the abilities of the phone. We can almost do anything using an app.

It is not only the software development, the hardware of the smartphone also getting more and more powerful. There are many sensors embedded into the phone, for example gyroscope, accelerometer etc. The GPS location technology also enrich the functionality of the phone. We can get the location of the phone using the GPS service.

When playing around the app, we have found that there are some advertisements at the bottom of the screen. This case is very common especially in those free apps. Some of the advertisements content are changing from time to time.

No doubt, this feature provide a good opportunity to the commercial organisation to reach their target customers, but it may gives a bad user experience to them On the other hand, this advertising model only work when the user is using the app. If the user sends the app to the background, those advertisement would not be shown.

Due to the limitations and bad user experience, what comes to our mind is whether if we can implement some mechanism with some new technologies to optimise the effect of promotion, while letting the businessmen reach their target audience more easily, without compromising the user experience at the same time. We are then inspired by the Apple iBeacon indoor location technologies. We come up with the idea of using iBeacon to locate the devices and try to push the advertising notification message instead of only showing the message during the app is running. We think this project would be interesting and it can brings us a new trend of advertising.

1.2 Background

When we are using the app, there are some small advertisements shown at the bottom of the screen .The content of it will be changed from time to time. It is not difficult to find out what it is. Actually, it is all caused by iAd. iAd is a mobile advertising platform which invented by Apple Inc in 2010. One of the major function is allowing the app developer to directly embed the advertisements into their own applications on IOS platform. iAd will try to group all the apple users by iTune accounts and divided them into around 400 target options. Although iAd is very effective and convenient, it compromises the user experience very much.

1.3 Objective

In our final year project, we are going to study about iBeacon, and implement a library for pushing notifications.

There are several objectives in our final year project.

- Study and compare the iBeacon or other location technologies
- Design and implement an algorithm to select the pushed advertisements
- Combine and make the library with coding

1.4 Runtime environment

For our development, we have decided IOS as the platform. Since we believe that iBeacon technology is invented by Apple Inc, the support of the iBeacon will be the most completed.

In fact, Apple provides a framework called CoreLocation for the location determination. Inside the framework, it contains some function calls for monitoring the iBeacon signals.

We use objective-C as programming language with IDE Xcode 6.1, and develop the program for IOS 8.1 platform.

chapter 2 iBeacon

2.1 Introduction

iBeacon was first introduced in 2013 for IOS 7. Actually iBeacon is just a trademark for Apple. In fact, at the behind it uses Bluetooth Low Energy (BLE) technology. This technology brings new possibilities for location awareness for apps. By installing the iBeacon to the environment, the IOS devices can determine if they have entered or left the region, or estimate the proximity to an iBeacon to trigger some specific App functionalities.

2.2 Specification

Because iBeacon is a BLE standard technology, it can be operated with coin cell batteries for a month or longer, depending on size of the battery used. Moreover, an IOS device can also be configured to be a beacon and advertises the signal when the app is running. It brings a flexibility to the app developers when writing an iBeacon embedded app.

For all the iBeacon advertisement signal information via Bluetooth Low Energy, it consists of three major fields.

Fields	Size	Description
UUID	16 Bytes	Application developers should define an UUID specific for their app and deployment use case.
Major	2 Bytes	Further specifies a specific iBeacon and use case. For example, this could define a sub-region within a larger region defined by the UUID.
Minor	2 Bytes	Allows further subdivision of region or use case, specified by the application developer.

An iBeacon identifies itself using three customisable values: Proximity UUID (16 bytes), Major and Minor (2 bytes each); there is also an additional Internal Identifier for your own reference.

Therefore you have three levels to identify a micro-location: only Proximity UUID, Proximity UUID and Major, Proximity UUID and Major and Minor. These levels give a way for managing the iBeacons in a well-organised manner. The app developer can make use of this feature to give some meaning to those identifiers by a subdivision policy. For example, one Proximity UUID represents a museum, a Major represents a specific gallery within the museum and a Minor represents an exhibit within that gallery.

2.3 iBeacon Hardware

iBeacon transmitters come in the form of hardware that run on Bluetooth 4.0 Low Energy (BLE). The BLE specification is used to create BLE chipsets, which are then embedded into devices. These devices are other words known as beacons, transmitters, or broadcasters coming in the form of any type of hardware such as USB dongles, computers, small coin-cell powered gadgets, etc.

2.3.1Choosing a Beacon

When we are planning to buy the beacons, we may need to consider the following conditions:

- **Battery life**

Since the beacons need to broadcast signals in a high frequency, the power consumption will be the major consideration. In fact, beacons have the option of being powered by cells or fixed power sources. One of the common and convenient beacon is powered by coin-cell, the battery life of it can be as short as 2 months. However, this type of beacon also gives an advantage of smaller size so that it is easy to be deployed. On the other hand, for a fixed power beacon, it can be powered by an USB port. This type of beacons needs to be connected with a cable for power supply. Also the size of it is larger than a coin-cell beacon. For reducing maintenance costs, using beacons running on a fixed power source is the most ideal.

- **Beacon Encasing**

The beacons are placed indoor or outdoor, so the physical conditions of it should be good. The challenge comes when beacons are deployed in environments that are susceptible to weather conditions such as rain or humidity.

- **Beacon management**

Some ibeacon manufactures provide management systems to their customers for dealing with the beacon identifying values such as UUID, major, minor etc. Therefore they don't need to process those data manually. If it is a large scale beacons deployment, whether such management system is provided may be a concern.

2.4 iBeacon hardware and software support

Actually, iBeacons technology is cross-platform. Both Apple(with IOS and OS X) and Google(with Android) have committed to support the BLE standard. Since there are many devices that support Bluetooth, the development of app should not only focus on a single OS. For Microsoft, they have added support BLE on Windows 8 and Windows Phone 8. Nokia's Lumia WP8 also announced to add the BLE hardware. Based on those observations, iBeacon definitely has a broad availability and supports on different platforms.

2.5 Comparison between NFC and BLE

	NFC	BLE
Range	4 - 20 cm	20 - 35 m
Platform	Not supported by Apple Devices	Cross Platform
Mode	Active (Need to touch)	Passive

2.6 Pros and Cons of IOS iBeacon

- Advantages

- Background search do not use much battery power as an Android phone
- Using the Passbook of iPhone

- Disadvantages

- Can monitor up to 20 regions (20 UUID)
- Need to specify UUID of the beacon, cannot be triggered by a random beacon (Android allows to do so)
- Cannot scan for unknown UUID in background
- No library for distance estimation between the device and the beacon (Android has some)

Chapter 3 Study on IOS iBeacon framework

3.1 IOS iBeacon package

In Xcode, all triggers related to iBeacon are handled by a Location Manager of CoreLocation.

```
#import <CoreLocation/CoreLocation.h>
```

Though the Location Manager can be declared anywhere, it is better to initialise it at the beginning of the program. If we need it to monitor the iBeacon in background, we have to declare it's function in AppDelegate.m, which exists in every apps and allows programmer to control the app when the app finished launching, when entering background or when entering foreground.

```
LM = [[CLLocationManager alloc] init];
LM.delegate = self;
if ([LM respondsToSelector:@selector(requestAlwaysAuthorization)]) {
    [LM requestAlwaysAuthorization];
}
```

After that, we need to specify regions to be monitor. A region contains UUID, major and minor (optional), identifier. Only those iBeacons match the UUID, major and minor(if any) can trigger the Location Manager. Those variables' contents can be changed during runtime.

```
NSUUID *uuid = [[NSUUID alloc] initWithUUIDString:@"E2C56DB5-DFFB-48D2-B060-D0F5A71096E0"];
int i_major = 1;
int i_minor = 1;
NSString *i_id = @"Building 1";
self.beaconRegion = [[CLBeaconRegion alloc] initWithProximityUUID:uuid major: i_major minor: i_minor identifier:i_id];
[self.LM startMonitoringForRegion:self.beaconRegion];
```

Then we can use LM's triggers to do whatever we want. There are two triggers can be used:

3.1.1 Location Manager trigger — *didDetermineState*

The first is *didDetermineState*, which triggers every time the device enters or leaves the region.

```
- (void)locationManager:(CLLocationManager *)manager didDetermineState:(CLRegionState)state forRegion:(CLRegion *)region
```

state will contain the value of CLRegionStateInside, CLRegionStateOutside or other. CLRegionStateInside appears when it is the first iBeacon of that region enters. CLRegionStateOutside appears when it is the last iBeacon of that region leaves. Other values appear for other cases, for example, the second iBeacon of that region enters.

region is the CLRegion object related to the trigger. We can use region.identifier to determine which region we enter or leave.

3.1.2 Location Manager trigger — *didRangeBeacons*

The second one is *didRangeBeacons*, which triggers once every second.

```
-(void)LM:(CLLocationManager *)manager didRangeBeacons:(NSArray *)beacons inRegion:(CLBeaconRegion *)region {
```

The *beacons* array contains all CLBeacon objects about all iBeacon in the *region*. Since we get the CLBeacon objects, we can get more information than

using *didDetermineState*. We can get the UUID, major, minor, accuracy, proximity and rssi of the iBeacon. However, the order of objects in *beacons* is quite random, and we cannot determine the order of entry of those iBeacons. (However, we can writing a log explicitly to achieve this.)

```
for (i = 0; i < numBeacon; i++) {
    CLBeacon *beacon = [[CLBeacon alloc] init];
    beacon = beacons[i];
        self.proximityUUIDLabel.text = beacon.proximityUUID.UUIDString;
        self.majorLabel.text = [NSString stringWithFormat:@"%@", beacon.major];
        self.minorLabel1.text = [NSString stringWithFormat:@"%@", beacon.minor];
        self.accuracyLabel1.text = [NSString stringWithFormat:@"%f", beacon.accuracy];
        if (beacon.proximity == CLProximityUnknown) {
            self.distanceLabel1.text = @"Unknown Proximity";
        } else if (beacon.proximity == CLProximityImmediate) {
            self.distanceLabel1.text = @"Immediate";
        } else if (beacon.proximity == CLProximityNear) {
            self.distanceLabel1.text = @"Near";
        } else if (beacon.proximity == CLProximityFar) {
            self.distanceLabel1.text = @"Far";
        }
        self.rssiLabel1.text = [NSString stringWithFormat:@"%ld", beacon.rssi];
}
```

Chapter 4 User Experience

4.1 Latency and response time

To investigate the amount of time delay for the device detects a beacon enters and leaves with the app in both foreground and background, we have conducted a test for that. We turn on and off a beacon and measure the time delay before the app notices the beacon appears and disappears. We have 5 trials for each case and below are the records of the response time.

4.1.1Foreground

Enter time/s	4.1	2.3	2.7	5.2	2.4
Leave time/s	34	31.5	32.8	28.6	30.7

4.1.2 Background

Enter time/s	2.2	3.8	2.3	2.2	2.1
Leave time/s	31.2	28.4	26.6	30.8	33.0

After the investigation, we found that the reaction times are quite similar for foreground and background. For the beacons entering, it takes around 2 to 4 seconds, while for the beacons leaving, it takes around 30 seconds.

Since we are now doing the notification, we are more concerning the time for noticing entry of a beacon. It is reasonable and acceptable that the entering time is quite responsive so that the notifying messages can be shown within a few seconds when getting near to the beacon.

Chapter 5 Demo application — Noticon

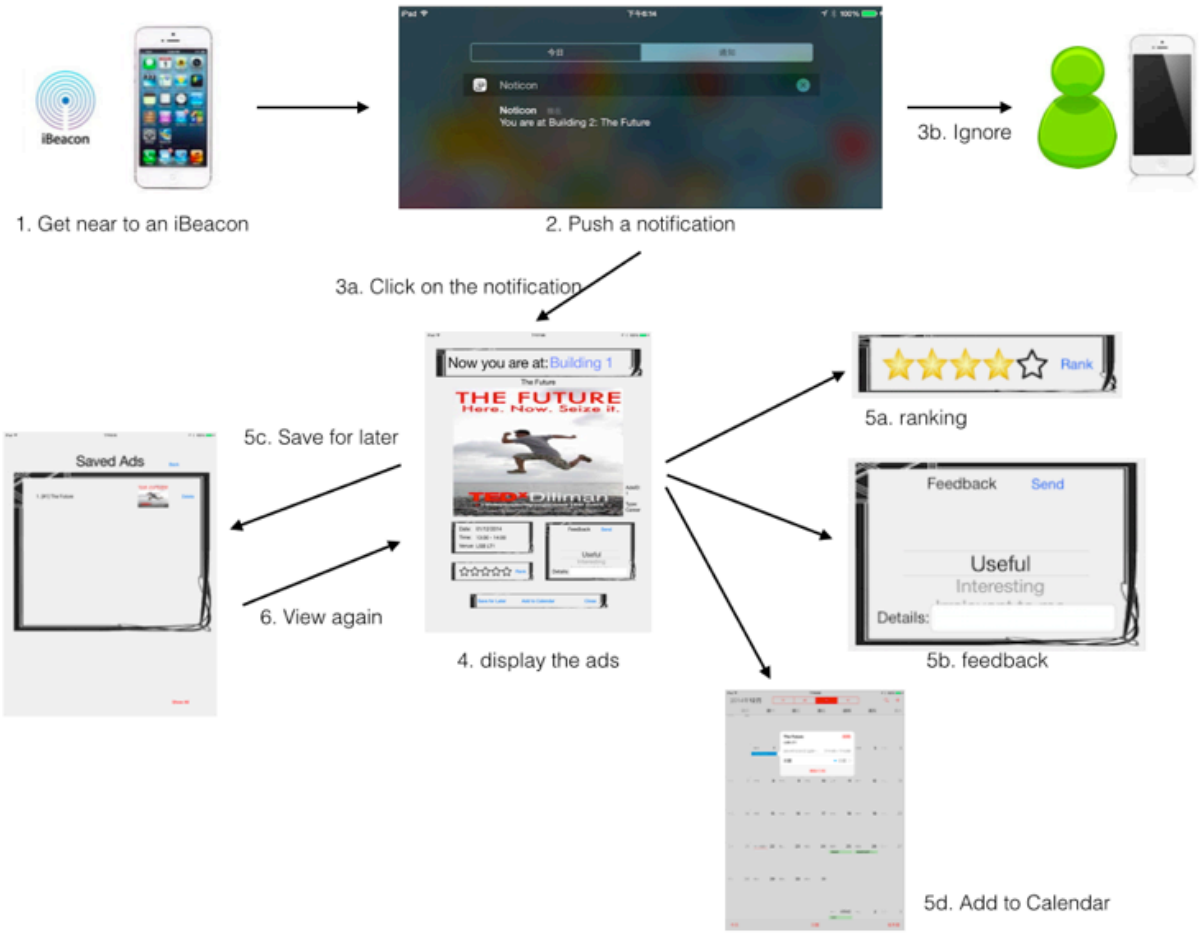
In order to demonstrate the features we would include in the library, we wrote an app called Noticon, which means Notification with iBeacon. Every time the user get near to the iBeacon, the app will push a notification about the users current location and an advertisement.

5.1 Scenario

When we enter a lift, we can see many advertising posters inside the lift. Imagine if now we install an iBeacon inside the lift. Every time we enter the lift, the iBeacon will trigger the phone to push a notification for an ads(short for advertisement), and we can use our phone to view the advertising posters.

With an electronic ads, we can interact with the poster more easily. We can add the event to calendar quickly, get coupons, or giving feedbacks to the poster. For the organization giving the ads, their ads can access to phone users more easily. They can also give coupons to attract people and get statistics and feedbacks about their ads for improvements.

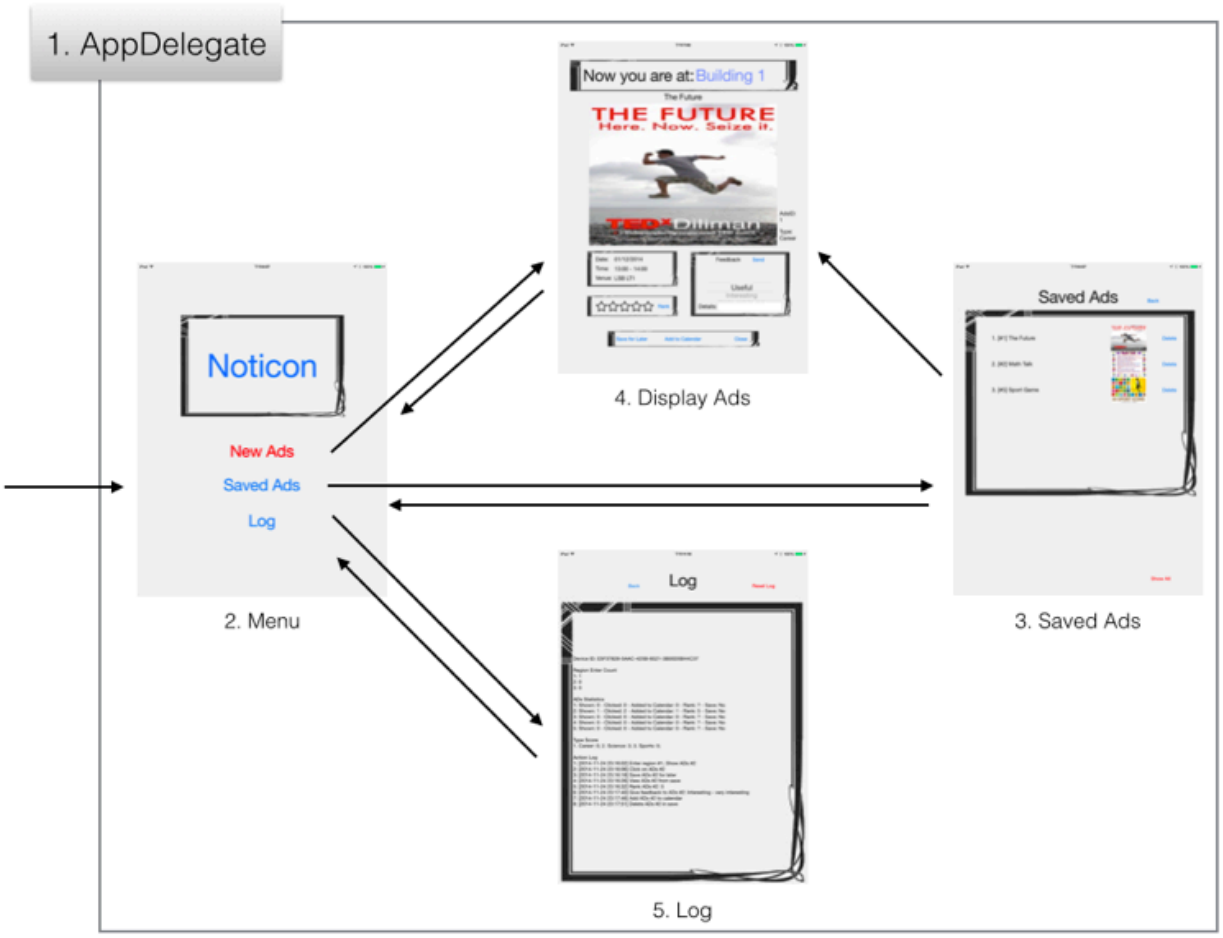
5.2 Program Flow



The above flow shows the program flow when the user get near to an iBeacon.

1. The app notices an iBeacon belongs to the monitoring region.
2. The app chooses an ads to push the notification.
3. User may respond or ignore the notification.
4. If user is interested and clicks on the notification bar, the app will display the details of the ads.
5. User can do several actions with the ads.
6. If the ads is saved, the user can view the ads again later.

5.3 User Interface



Above shows all the UI for user to use. The details will be discussed in 5.5 Function.

5.4 Classes

The following shows the classes and functions in Noticon, which can be included in the library we develop later.

- System Class

The class keeps some variables used around the app.

class System{

int NUM_Region:	total number of region to be monitored
int NUM_ADs:	total number of Ads
int NUM_Type:	total number of types of Ads
CLLocationManager *LM:	the LocationManager to handle all ibeacon monitoring
int current_regionIndex:	index of current region
int ADs_id	index of current displaying Ads

}

- Region Class

The class keeps the identifying values for one region. It also keeps the ids of ads to be shown when entering this region.

class Region{

int id:	id of the region
String Name:	name of the region
String UUID:	uuid of beacon to be monitor, e.g. "E2C56DB5-DFFB-48D2-B060-D0F5A71096E1"
int major:	major of beacon to be monitor (optional). Range from 0 to 2^{16}-1, with -1 as wildcard
int minor:	minor of beacon to be monitor (optional). Range from 0 to 2^{16}-1, with -1 as wildcard
boolean DisplayADs[i]:	if enter this region, whether the i-th ads is available or not e.g. (regionDisplayADs[2] == true) means the ads #2 is available.

}

- ADs Class

The class keeps the data for an ads.

class ADs{

int id	id of the Ads
string title	title of the Ads, e.g. “Career Talk”
string poster	filename of the poster of the Ads, e.g. “poster3.jpeg”
string date	date of the event, in format “dd/MM/yyyy”, e.g. “03/12/2014”
string startTime	start time of the event, in format “HH:mm” in 24 hour time format, e.g. “15:00”
string endTime	end time of the event, no earlier than startTime
string venue	venue of the event, e.g. “LSB LT3”
int type	id of the type of the Ads belongs to

}

- ADs_type Class

The class defines the ads type, which is used for determining interested areas of the users.

class ADs_type{

int id	id of the type
string name	name of the type

}

- Log Class

The class keeps the statistics and action log of a user.

class Log{

NSUUID *UserID	id to distinguish users, get by UIDevice.identifierForVendor
int RegionEnterCount[i]	Count for number of time of entering the region id i
int ADsShown[i]	Count for number of time of choosing the i-th ads to show and notify the user
int ADsClick[i]	Count for number of time of clicking the i-th ads, both from notification bar and view from saved ads
int ADsAddedToCalendar[i]	Count for number of time of adding the i-th event to calendar
int Rank[i]	the rank of the i-th ads given by user. For unrank, 0 for storage and ‘?’ for displaying in log page
int Action_NUM	total number of action taken
String Action[i]	time and action details of the i-th action
int saveForLater[i]	if the i-th ads is saved for later view. 1 is true and 0 is false.
int typeScore[i]	the score of i-th type, used for choosing ads to display

}

5.5 Functions

5.5.1 AppDelegate

This interface exists in every apps. It allows programmer to control the app when the app finished launching, when entering background or when entering foreground. Core Location manager should also be setup in this interface.

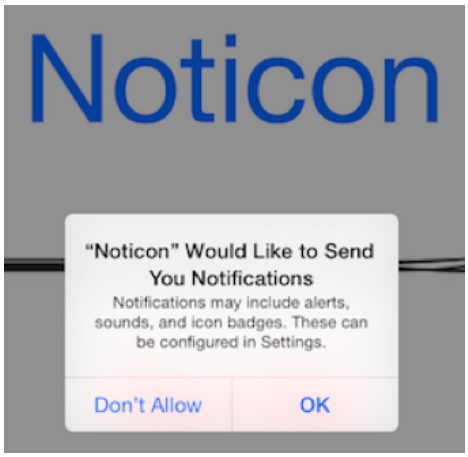
- **didFinishLaunchingWithOptions**

  This function is triggered when the app is started

1.Ask for authority of notification

```
//-- Set Notification
if ([application respondsToSelector:@selector(isRegisteredForRemoteNotifications)])
{
    // iOS 8 Notifications
    [application registerUserNotificationSettings:[UIUserNotificationSettings settingsForTypes:
        (UIUserNotificationTypeSound | UIUserNotificationTypeAlert | UIUserNotificationTypeBadge) categories:nil]]
        ;

    [application registerForRemoteNotifications];
}
```



Ask user to give authority to send notifications.

2.Set Location Manager

```
//LM setting
LM = [[CLLocationManager alloc] init];
LM.delegate = self;
```

Initialize the Location Manager

3.Ask for authority of location service

```
// Check for iOS 8. Without this guard the code will crash with "unknown selector" on iOS 7.
if ([LM respondsToSelector:@selector(requestAlwaysAuthorization)]) {
    [LM requestAlwaysAuthorization];
}
```

Ask user to give authority for always access location for the location manager to scan beacon when the app is in background.

4.Set Region

```
CLBeaconRegion *region;

region = [[CLBeaconRegion alloc] initWithProximityUUID:[[NSUUID alloc] initWithUUIDString:@"E2C56DB5-DFFB-48D2-
    B060-D0F5A71096E1"] identifier: regionName[1]];
region.notifyEntryStateOnDisplay = YES;
[LM startMonitoringForRegion:region];
[LM stopRangingBeaconsInRegion:region];
```

Create regions with the Region class objects and let the Location Manager to start monitoring the regions.

- **locationManager: didDetermineState**

This function is triggered when the device get inside to a region or get outside from a region.

1.Determine state and regionIndex

Determine if the device is entering or leaving the region and determine the region index

2.Create notification

If the device enter a region, we choose one ads to notify the user. Below is the algorithm.
If (enter a region)
give each Ads a score
choose one of the highScore Ads to display
create notification

Code for simple algorithm for choosing a ads:

```
//NSLog(@"Enter");
//algorithm for ads choice
int highscore = -9999;
int highscoreAds = 0;
int samescore = 0;
int i;
//give each ads a score
for (i = 1; i < NUM_ADs; i++){
    int score = 0;
    score += Log_typeScore[ADs_type[i]]*10;
    score -= Log_ADsShown[i];
    score -= Log_ADsClicked[i] * 5;
    score -= Log_ADsAddedToCalendar[i] * 100;
    if (score > highscore) {
        //if high Score
        highscore = score;
        highscoreAds = i;
        samescore = 1;
    }
    else if(score == highscore){
        //if same Score
        samescore++;
        //random gives from 0 to (x-1)
        int change = arc4random() % (samescore + 1);
        if (change == 0){
            highscoreAds = i;
        }
    }
}
ADs_ID = highscoreAds;
```

The idea is to choose one of the highest score ads with equal probability in O(n). The calculation of score will make sure that those already shown ads have a lower chance to display, while those ads of the type that user would be interested have a higher chance to display.

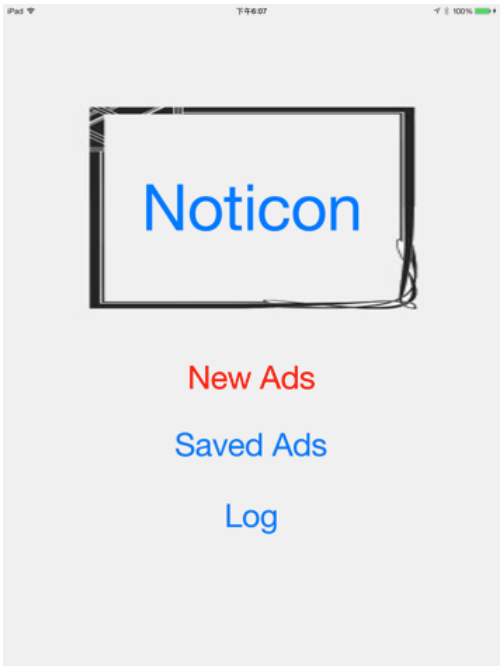
Proof for those highest ads have equal probability to be chosen:
Suppose 5 ads have the same score,
then P(5th is chosen) is 1/5 (when sameScore = 5)
P(4th is chosen) = 1/4 (when sameScore = 4) * P(5th is not chosen)
= 1/4 * 4/5
= 1/5
P(3rd is chosen) = 1/3 (when sameScore = 3) * P(4th is not chosen) * P(5th is not chosen)
= 1/3 * 3/4 * 4/5
= 1/5
and so on…

Code for create notification:

```
UILocalNotification *notification = [[UILocalNotification alloc] init];
notification.alertBody = [NSString stringWithFormat:@"You are at %@: %@",
                          region.identifier, ADs_title[ADs_ID]];
[[UIApplication sharedApplication] presentLocalNotificationNow:notification];
```

Screen cap for the notification message

5.5.2 Menu



The main menu of the app.

- **(void)To Save Ads**

When user press “Saved Ads”, it goes to 5.5.3 Saved Ads.

- **(void)To Display Ads (for debug)**

When user enters a region and an ads is given to user, user may click on the notification bar. Then this function will be called and redirect the user to 5) Display Ads automatically. For debug purpose, we can press on “New Ads” to call this function too.

- **(void)To Log**

When user press “Log”, it goes to 5.5.5 Log.

- (void)Save Log

The function to save the data of Log class variables with the function NSUserDefaults.setObject, which can store data in non-volatile memory. This is called every time a change is made to the Log.

```
- (void)saveLog{
    NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
    int i;
    //Region
    for (i = 1; i < NUM_Region; i++) {
        [defaults setObject:[NSString stringWithFormat:@"%d", Log_RegionEnterCount[i]]
                     forKey:[NSString stringWithFormat:@"REC:%d", i]];
    }
    //ADs
    for (i = 1; i < NUM_ADs; i++) {
        [defaults setObject:[NSString stringWithFormat:@"%d", Log_ADsShown[i]]
                     forKey:[NSString stringWithFormat:@"AS:%d", i]];
        [defaults setObject:[NSString stringWithFormat:@"%d", Log_ADsClicked[i]]
                     forKey:[NSString stringWithFormat:@"AC:%d", i]];
        [defaults setObject:[NSString stringWithFormat:@"%d", Log_ADsAddedToCalendar[i]]
                     forKey:[NSString stringWithFormat:@"AATC:%d", i]];
        [defaults setObject:[NSString stringWithFormat:@"%d", Log_Rank[i]]
                     forKey:[NSString stringWithFormat:@"Rank:%d", i]];
        [defaults setObject:[NSString stringWithFormat:@"%d", Log_saveForLater[i]]
                     forKey:[NSString stringWithFormat:@"SFL:%d", i]];
    }
    [defaults setObject:[NSString stringWithFormat:@"%d", Log_Action_NUM]
                 forKey:[NSString stringWithFormat:@"AN"]];
    for (i = 1; i <= Log_Action_NUM; i++) {
        [defaults setObject:Log_Action[i] forKey:[NSString stringWithFormat:@"ACT:%d", i]];
    }
    for (i = 1; i < NUM_Type; i++){
        [defaults setObject:[NSString stringWithFormat:@"%d", Log_typeScore[i]]
                     forKey:[NSString stringWithFormat:@"TS:%d", i]];
    }
    [defaults synchronize];
}
```

- (void)Load Log

The function to read the data of Log class variables from memory using the function NSUserDefaults.objectForKey. This is called every time the app is launched.

```
- (void)loadLog{
    NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
    int i;
    //Region
    for (i = 1; i < NUM_Region; i++) {
        Log_RegionEnterCount[i] = [[defaults objectForKey:[NSString stringWithFormat:@"REC:%d", i]] intValue];
    }
    //ADs
    for (i = 1; i < NUM_ADs; i++) {
        Log_ADsShown[i] = [[defaults objectForKey:[NSString stringWithFormat:@"AS:%d", i]] intValue];
        Log_ADsClicked[i] = [[defaults objectForKey:[NSString stringWithFormat:@"AC:%d", i]] intValue];
        Log_ADsAddedToCalendar[i] = [[defaults objectForKey:[NSString stringWithFormat:@"AATC:%d", i]] intValue];
        Log_Rank[i] = [[defaults objectForKey:[NSString stringWithFormat:@"Rank:%d", i]] intValue];
        Log_saveForLater[i] = [[defaults objectForKey:[NSString stringWithFormat:@"SFL:%d", i]] intValue];
    }
    Log_Action_NUM = [[defaults objectForKey:[NSString stringWithFormat:@"AN"]] intValue];
    for (i = 1; i <= Log_Action_NUM; i++) {
        Log_Action[i] = [defaults objectForKey:[NSString stringWithFormat:@"ACT:%d", i]];
    }
    for (i = 1; i < NUM_Type; i++) {
        Log_typeScore[i] = [[defaults objectForKey:[NSString stringWithFormat:@"TS:%d", i]] intValue];
    }
}
```

- **(string*)Get Current Time**

Get the current time in the format “yyyy-MM-dd HH:mm:ss” from NSDate and return it as a string.

```
-(NSString*)getCurrentTime{

    NSDate *date = [NSDate date];
    NSDateFormatter *dateFormat = [[NSDateFormatter alloc] init];
    [dateFormat setDateFormat:@"yyyy-MM-dd HH:mm:ss"];
    NSString *dateString = [dateFormat stringFromDate:date];
    return dateString;
}
```

5.5.3 Saved Ads

- **(void)Back to Menu**

Press “Back” and it goes to 5.5.2 main menu.

- **(void)Display Info**

For rendering the view. It displays the titles and posters of the saved ads. It is called when the view is entered or by the function Delete an Ads.

- **(void)Open an Ads**

When user click on the title or poster, it sets the ADs_ID to the clicked ads ID and goes to 5.5.4 Display Ads to view the ads.

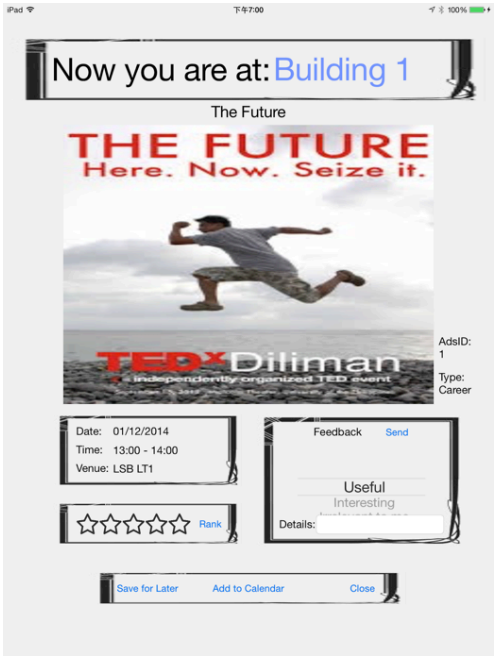
- **(void)Delete an Ads(int id)**

When the corresponding “Delete” is clicked, the ADs will be deleted from the saved ads and Log.saveForLater[id] changes back to false. Then the function Display info will be called to refresh the view.

- **(void)Show all Ads (for debug)**

For debug purpose, press “Show All” and all ads will be displayed in this page. (whole Log.saveForLater[] array is set to true.)

5.5.4 Display Ads



In this view, user can view the detail information of the ads. User can also interact with the ads in different ways.

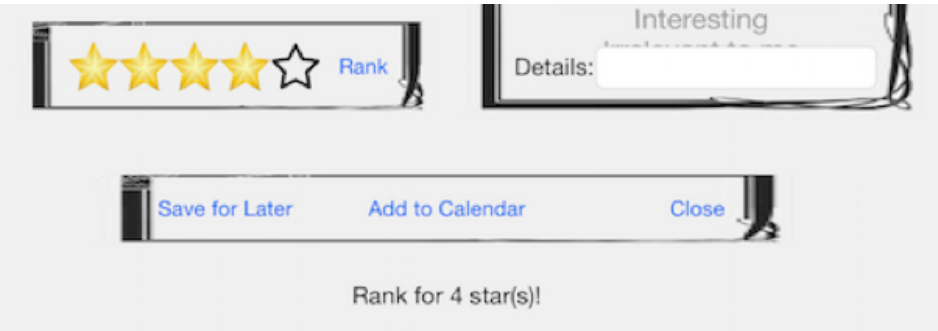
- **(void)Back to Menu**

Press “Close” and it goes to 5.5.2 main menu.

- **(void)Display information**

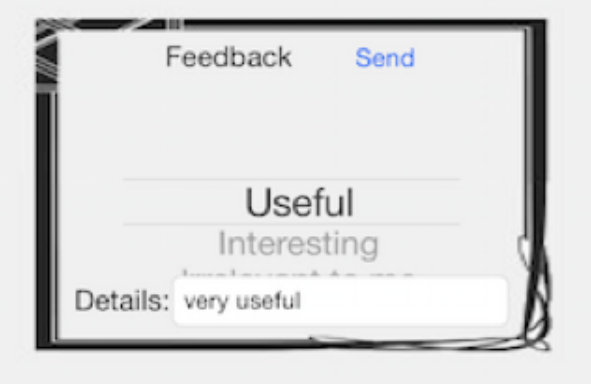
For rendering the view. It displays the name of the current region and the chosen Ads informations with the ADs Class object.

- **(void)Rank the Ads with stars**

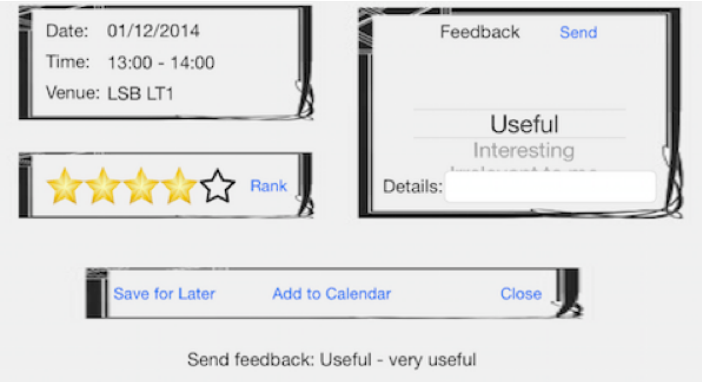


User can click on the stars and the stars will change between empty and gold according to which star is clicked. The leftmost is 1 star and the rightmost is 5 stars. Then the user can click on the “Rank” button to send the ranking. Notice that every ads can only be ranked once by a user. It is fixed and cannot be changed after the user ranked.

- **(void)Send Feedback**



The user can also give a feedback about the ads. The user can choose an item from the picker view and type the detail comment in the text field. Then the user can press "send" to send the feedback.




```
//feedback
-(void)sendFeedback{
    //NSLog(@"#%d", ADs_ID);
    NSString *select = [_TypeName objectAtIndex:[_typePicker selectedRowInComponent:0]];
    if (![self.comment.text isEqual: @""]){
        //have comment
        select = [NSString stringWithFormat:@"%@ - %@", select, self.comment.text];
    }
    self.logLabel.text = [NSString stringWithFormat:@"Send feedback: %@", select];
    Log_Action_NUM++;
    Log_Action[Log_Action_NUM] = [NSString stringWithFormat:@"[%@] Give feedback to ADs #%d: %@",
                                  root_VC.getCurrentTime, ADs_ID, select];
    self.comment.text = @"";
    [root_VC saveLog];
}

#pragma mark typePicker Data Source Methods

-(NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerView{
    return 1;
}

-(NSInteger)pickerView:(UIPickerView *)pickerView numberOfRowsInComponent:(NSInteger)component {
    return [_TypeName count];
}

#pragma mark typePicker Delegate Methods

-(NSString *)pickerView:(UIPickerView *)pickerView titleForRow:(NSInteger)row forComponent:(NSInteger)component{
    return [_TypeName objectAtIndex:row];
}
```

- **(void)Save for later**

If the users want to view this ads later, they can press “Save for Later” and later to view this again in 5.5.3 Saved Ads.

- **(void)Add to Calendar**

If the users want to join the event in the ads, they can press “Add to Calendar” to add the event into their calendars so that they will remember to join this event. We use <eventKit> to do this.

```
EKEventStore *eventStore = [[EKEventStore alloc] init];
if ([eventStore respondsToSelector:@selector(requestAccessToEntityType:completion:)])
{
    // the selector is available, so we must be on iOS 6 or newer
    [eventStore requestAccessToEntityType:EKEntityTypeEvent completion:^(BOOL granted, NSError *error) {
        dispatch_async(dispatch_get_main_queue(), ^{
            if (error)
            {
                // display error message here
            }
            else if (!granted)
            {
                // display access denied error message here
            }
            else
            {
                // access granted
                // ***** do the important stuff here *****
                EKEvent *event  = [EKEvent eventWithEventStore:eventStore];
                event.title     = ADs_title[ADs_ID];

                NSDateFormatter *tempFormatter = [[NSDateFormatter alloc]init];
                [tempFormatter setDateFormat:@"dd/MM/yyyy HH:mm"];

                NSString *dateandtime =[NSString stringWithFormat:@"%@ %@", ADs_date[ADs_ID], ADs_startTime[ADs_ID
                    ]];
                NSString *dateandtimeend =[NSString stringWithFormat:@"%@ %@", ADs_date[ADs_ID], ADs_endTime
                    [ADs_ID]];

                event.startDate = [tempFormatter dateFromString:dateandtime];
                event.endDate = [tempFormatter dateFromString:dateandtimeend];
                event.location = ADs_venue[ADs_ID];

                //[event addAlarm:[EKAlarm alarmWithRelativeOffset:60.0f * -60.0f * 24]];
                //[event addAlarm:[EKAlarm alarmWithRelativeOffset:60.0f * -15.0f]];

                [event setCalendar:[eventStore defaultCalendarForNewEvents]];
                NSError *err;
                [eventStore saveEvent:event span:EKSpanThisEvent error:&err];
            }
        });
    }];
}
```

5.5.5 Log

In this view, it displays the contents of the Log class variables. We may analyse the user habits or preferences so that we can give more useful ads to the user. The use of this view may not be use for displaying to the user, but to make a log file send to server for analysis in the future. We may need to give a warning to users to tell them we log their actions though, as there may be a privacy issue.

- **(void)Back to Menu**

Press “Back” and it goes to 5.5.2 main menu.

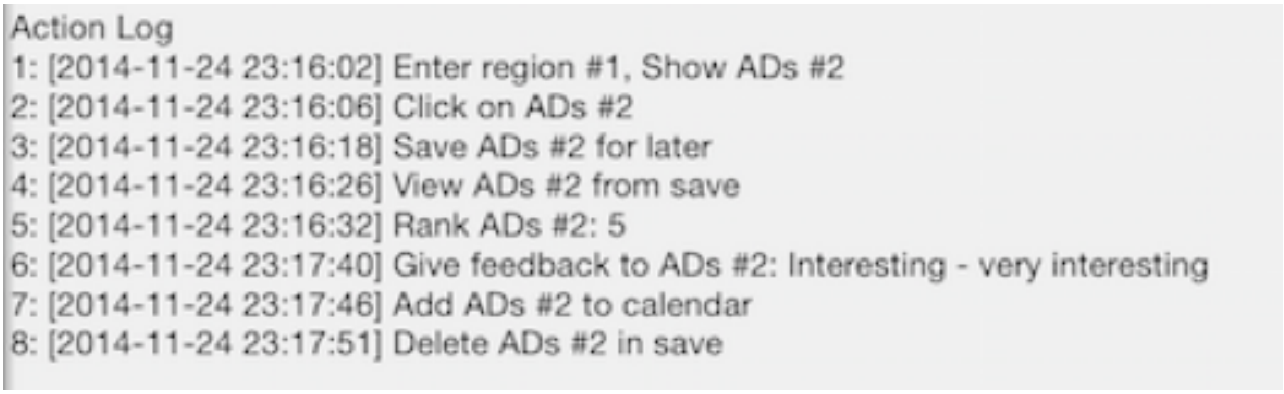
- **(void)Display log**

List the content of Log class variables into a single string and display it.

- **(void)Reset log (for debug)**

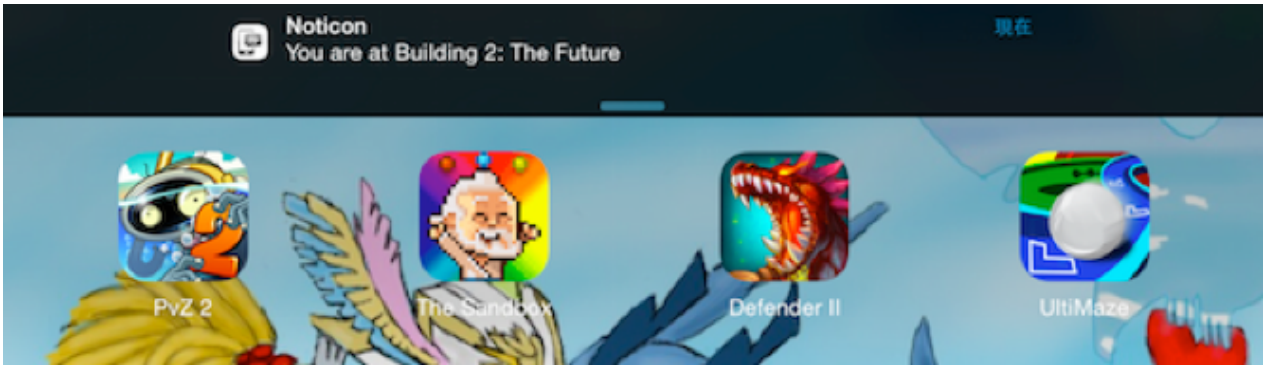
For debug purpose, press “Reset Log” can initialize the log again. This should not be available to users as we don’t want users to wash away their logs.

- **About action log**



The above log shows all the current possible action to be logged. Below lists the corresponding actions:

1. When the user enters a region
   [from 5.5.1 AppDelegate, locationManager: didDetermineState]
2. When the user clicks on a notification bar to view the ads



1. When the user presses “Save for later”
   [from 5.5.4 Display Ads, Save for later]
2. When the user views the ads from the saved ads
   [from 5.5.3 Saved Ads, Open an Ads]
3. When the user presses “Rank” with the stars clicked
   [from 5.5.4 Display Ads, Rank the Ads with stars]
4. When the user presses “Send” the feedback
   [from 5.5.4 Display Ads, Send feedback]
5. When the user presses “Add to Calendar”
   [from 5.5.4 Display Ads, Add to calendar]
6. When the user presses “Delete” for a saved ads
   [from 5.5.3 Saved Ads, Delete an Ads]

Chapter 6 Second Semester Goals

6.1 Create a library

As we mention before, our goal is not writing an apps, but is building a library for developers to build their own apps. We will split the codes into classes with methods and build the library.

6.2 Update region / ads from server

As we have not setup a server yet, we are hardcoding the Regions and Ads class information now. However, it is possible to update those data from a database. We will setup a database server for the apps to update the Ads.

6.3 Send log to server for more analysis

Currently we only do some simple algorithm to choose an Ads for the user. However, we have many statistics to be analysed. The server may also collect many log data from many users to have a larger scale analysis. We will make more uses of those statistics in the future.

6.4 More dynamic layout for displaying the ads

Currently there is only one layout to display the ads. But it is possible to give more dynamic layouts to display the ads in a better way. We will implement dynamic layouts later.

6.5 Include Passbook

iPhone has a special feature called “Passbook”, which allows users to store coupons, boarding passes, event tickets, store cards, credit cards etc. We hope to include this feature into the library so that the users may get coupons or event tickets from the notifications.

Chapter 7 Conclusion

iBeacon is a new technology that will be more common in the market. This is a good chance for us to do a project using iBeacon. In this project, we have done many study or research on iBeacon. During the research, we have learned the use of iBeacon and how does it works.

This is our first time to develop a mobile app on IOS. We have learned a lot of techniques on writing an IOS application and the development environment. We get used of Xcode, Objective-C or even mac OS. During the development, we have faced many problems. For example, we could not get started and do the device build of the application because the apple developer account cannot link with our macbook. As a beginner of IOS app development from 2014, we are now able to write an app with iBeacon technology. Moreover, our problem-solving skills and self-learning skills have also been improved.

Finally, we have tried to design and plan a whole project. From thinking the topic, researching, implementing, and testing. We have understood the software development life cycle (SDLC). Beside, during the project, we know more about each other and understand the importance of time management. We believe that this is a very valuable experience to us.

Chapter 8 Reference

[1]. Location and Maps Programming Guide [Online]. Available:

https://developer.apple.com/library/Mac/documentation/UserExperience/Conceptual/LocationAwarenessPG/RegionMonitoring/RegionMonitoring.html#//apple_ref/doc/uid/TP40009497-CH9-SW1

[2]. Get Started with iBeacons [Online] Available:

https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf

[3]. Apple iBeacon developer page [Online] Available:

https://developer.apple.com/ibeacon/

[4]. News about apple's policy of restricting uuid/ need information available without beacon [Online] Available:

http://beekn.net/2014/05/apple-closed-system-apple-slowly-locking-ibeacon/

[5]. Some people's discussion about the policy [Online] Available:

http://apple.stackexchange.com/questions/134696/does-apple-have-a-policy-restricting-generic-uuid-with-ibeacon-apps-in-the-app-s

[6]. Awwapps's website: they have to remove the manually input uuid feature [Online] Available:

http://blog.awwapps.com/blog/2014/05/20/manual-ibeacon-entry-to-be-removed/

[7]. Beecon on iTune [Online] Available:

https://itunes.apple.com/app/beecon/id822251888

[8]. Estimote(One of beacon manufacturer) website [Online] Available:

https://community.estimote.com/hc/en-us/articles/200868188-How-do-I-modify-UUID-major-and-minor-values-

[9]. Limitations for scan for beacon in background [Online] Available:

http://indoo.rs/insights-from-product-ibeacon-in-the-background/

[10]. Background scanning 1- apple developer forum [Online] Available:

https://devforums.apple.com/message/1027403#1027403

[11]. Background scanning 2- apple developer forum [Online] Available:

https://devforums.apple.com/message/1006867#1006867

[12]. iOS 8 uses M7 chip and motion sensors for accurate indoor positioning [Online] Available:

http://www.idownloadblog.com/2014/06/05/ios-8-indoor-positioning-m7/

Chapter 9 : Acknowledgement

We would like to take this chance to thank our supervisor, Professor Michael Lyu. He gives us many supports and advices for this project. He also reminds us the importance of documentation.

In addition, we also want to express our appreciation to Mr. Edward Yau and Tsz Lung in VIEW Lab. Edward always gives us many valuable advices and ideas while Tsz Lung provides us some technical supports when we faced problems.