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

Indoor location service with iBeacon

LYU 1402 Final Year Project Report
Spring 2015

Wan Ka Ki 1155030692

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 as shown in Fig1.1. This case is very common especially in those free apps. Some of the advertisements content are changing from time to time.

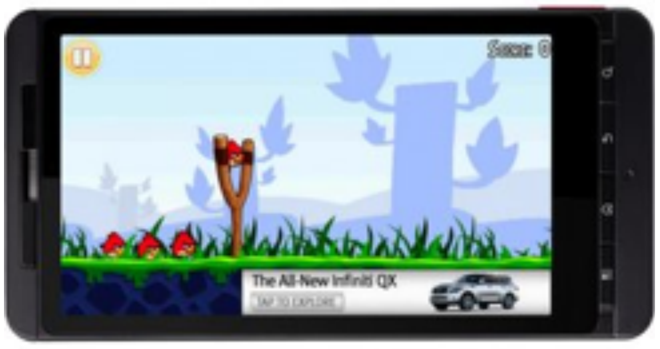


Fig1.1 The advertisement banner is placed at the bottom of the app.

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 as shown in the Table2.1.

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.

Table 2.1 Specifications of iBeacon

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. Fig 2.1 shown some transmitters with plastic covered.

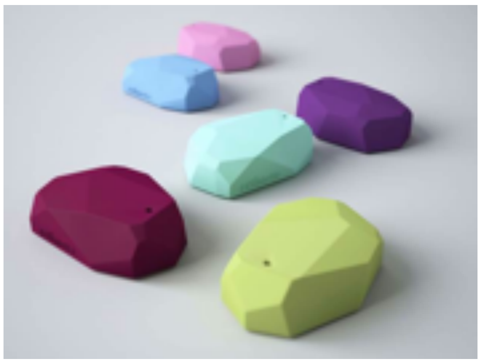


Fig 2.1 Example of iBeacon transmitter with plastic covered.

- **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

The below Table 2.2 shown the comparison of NFC and BLE technologies.

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

Table 2.2 Comparison of NFC and BLE

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

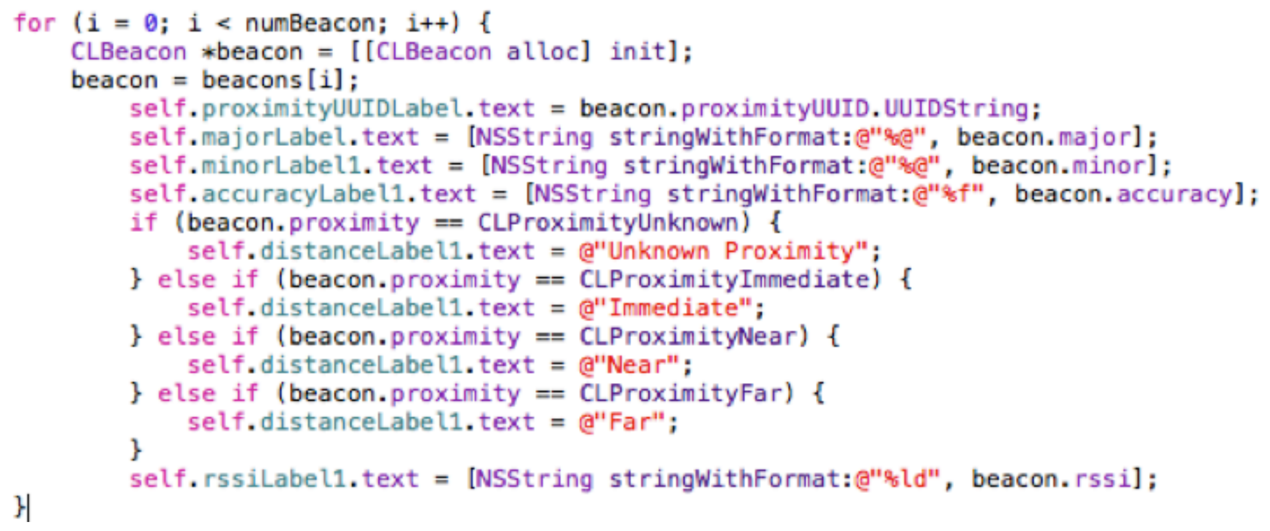
- Disadvantages

- 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)

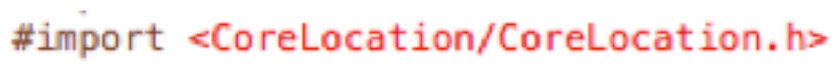
Chapter 3 Study on IOS iBeacon framework

3.1 IOS iBeacon package

In Xcode, all triggers related to iBeacon are handled by a Location Manager of CoreLocation (refer to Fig 3.1).



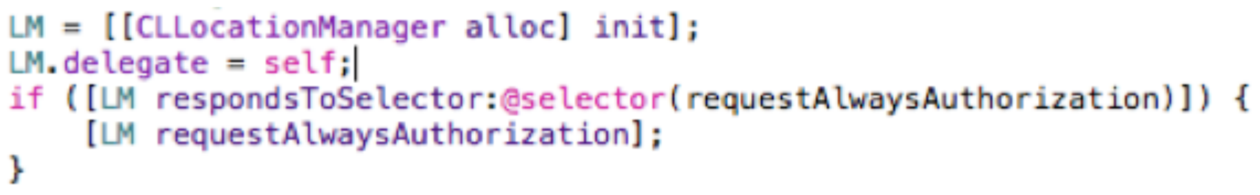
```
for (i = 0; i < numBeacon; i++) {
    CLBeacon *beacon = [[CLBeacon alloc] init];
    beacon = beacons[i];
        self.proximityUUIDLabel.text = beacon.proximityUUID.UUIDString;
        self.majorLabel.text = [NSString stringWithFormat:@"%@", beacon.major];
        self.minorLabel1.text = [NSString 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];
}
```



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

Fig 3.1 Code of CoreLocation

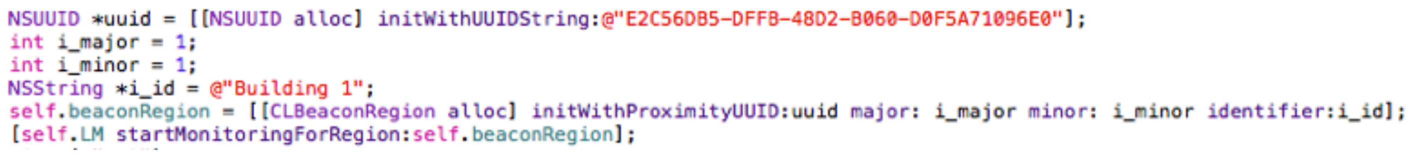
Though the Location Manager can be declared anywhere, it is better to initialise it at the beginning of the program. If we need it to monitor the iBeacon in background, we have to declare it's function in AppDelegate.m, which exists in every apps and allows programmer to control the app when the app finished launching, when entering background or when entering foreground(refer to Fig 3.2).



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

Fig 3.2 Code for declaration in AppDelegate.m

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(refer to Fig 3.3). Those variables' contents can be changed during runtime.



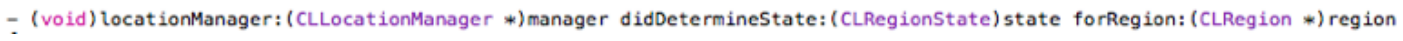
```
NSUUID *uuid = [[NSUUID alloc] initWithUUIDString:@"E2C56DB5-DFFB-48D2-B060-D0F5A71096E0"];
int i_major = 1;
int i_minor = 1;
NSString *i_id = @"Building 1";
self.beaconRegion = [[CLBeaconRegion alloc] initWithProximityUUID:uuid major: i_major minor: i_minor identifier:i_id];
[self.LM startMonitoringForRegion:self.beaconRegion];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

Fig 3.3 Code for setting a 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 (refer to Fig 3.4).



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

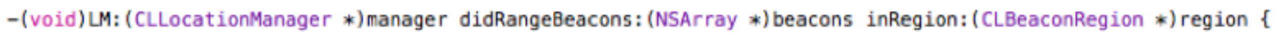
Fig 3.4 Code for didDetermineState

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 (refer to Fig 3.5).



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

Fig 3.5 Code for didRangeBeacons

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.)

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. The results of the experiment have been shown in Table 4.1 and Table 4.2 respectively.

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

Table 4.1 Response time of entering and leaving the beacon when the app is running at foreground

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

Table 4.2 Response time of entering and leaving the beacon when the app is running at background

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 as shown in Fig 5.1. 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.

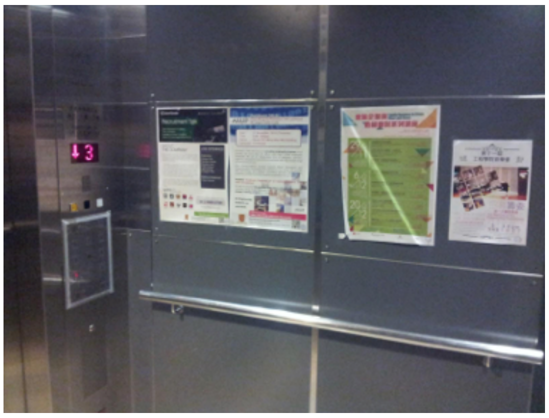


Fig 5.1 Posters are placed inside the lift.

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

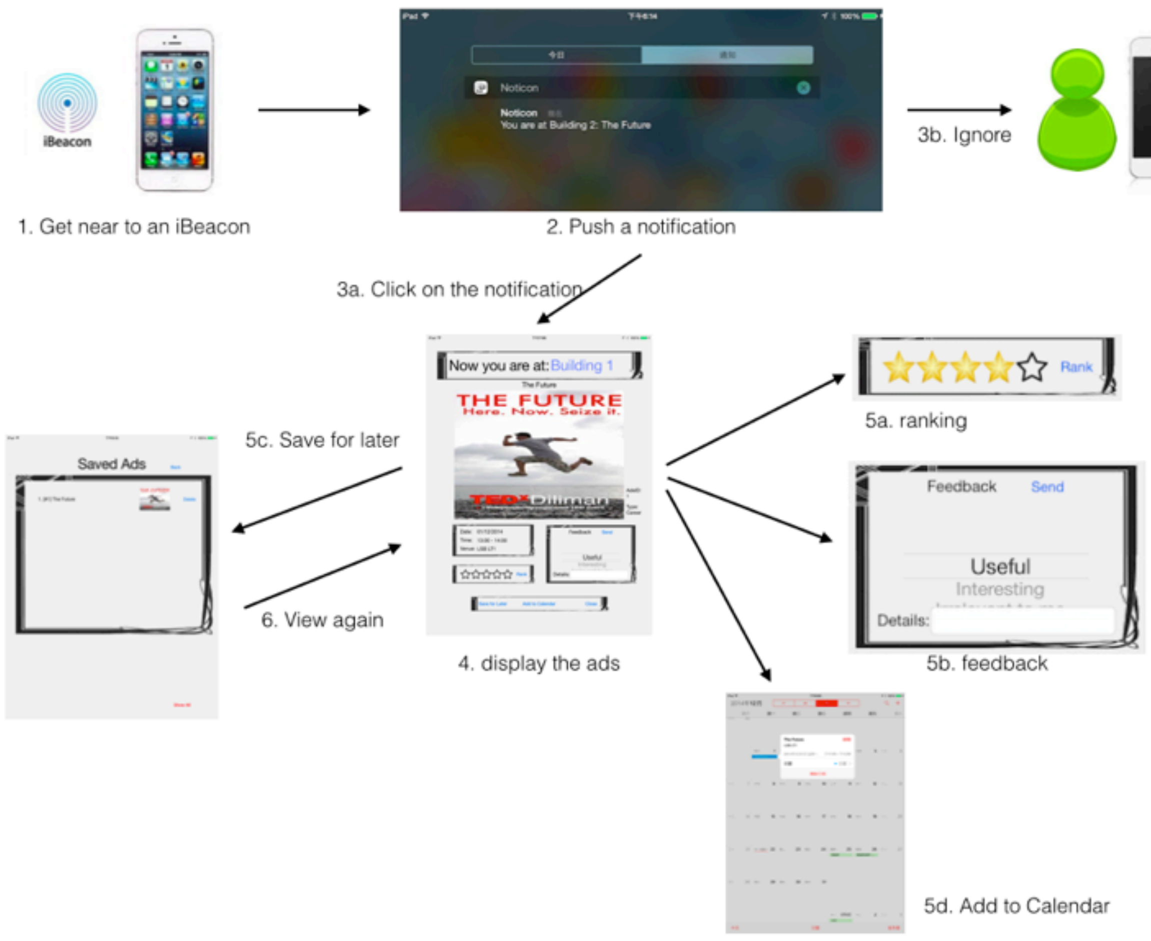


Fig 5.2 Program flow of Noticon ver 1.0

The above Fig 5.2 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

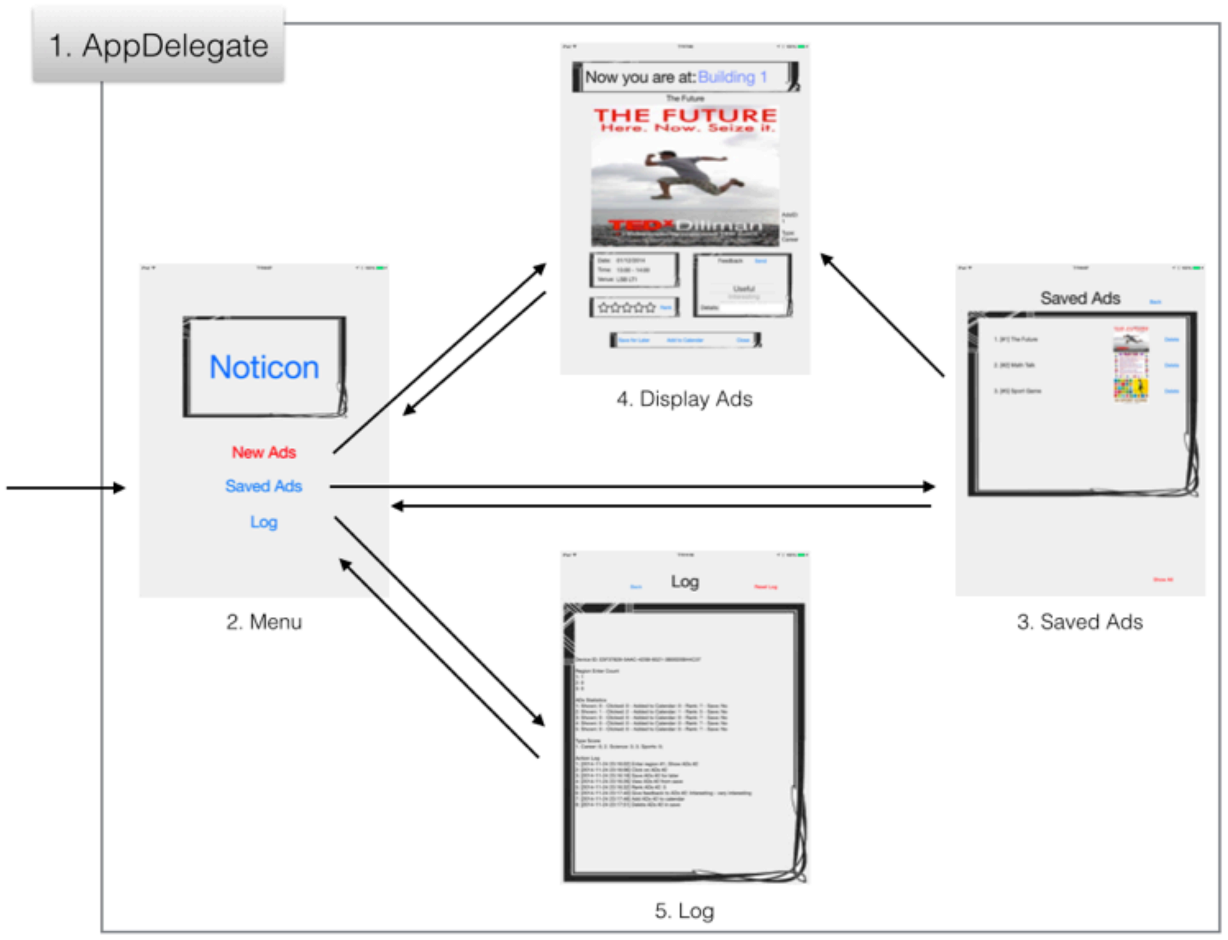


Fig 5.3 User interface of the Noticon ver 1.0

Above Fig 5.3 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

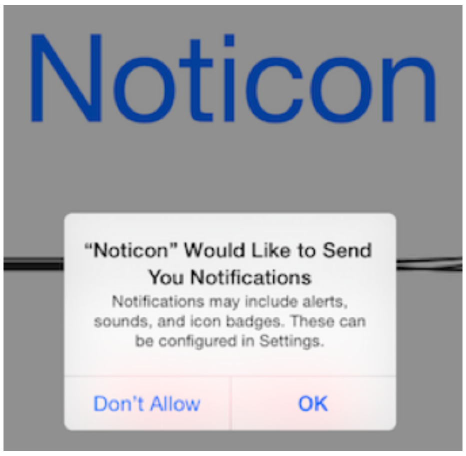
This interface exists in every apps. It allows programmer to control the app when the app finished launching, when entering background or when entering foreground. Core Location manager should also be setup in this interface.

- **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];
}
```



"Noticon" Would Like to Send You Notifications

Notifications may include alerts, sounds, and icon badges. These can be configured in Settings.

Fig 5.4 Screen capture of an iPad ask user to give authority to send notifications

The above Fig 5.4 shown IOS asks the permission of sending notifications.

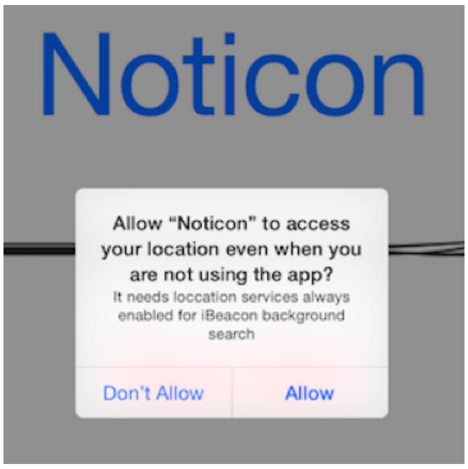
2.Set Location Manager

```
//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];
}
```



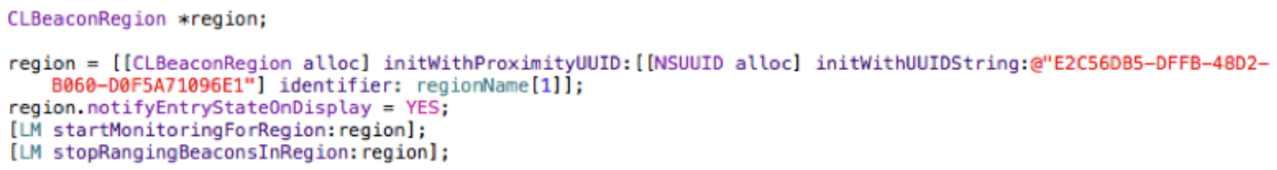
Allow “Noticon” to access your location even when you are not using the app?

It needs loccation services always enabled for iBeacon background search

Fig 5.5 Screen capture of an iPad ask user to give authority to obtain user location information

The above Fig 5.5 shows IOS asks 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];
```

Fig 5.6 code for creating regions

The above Fig 5.6 shows how to 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

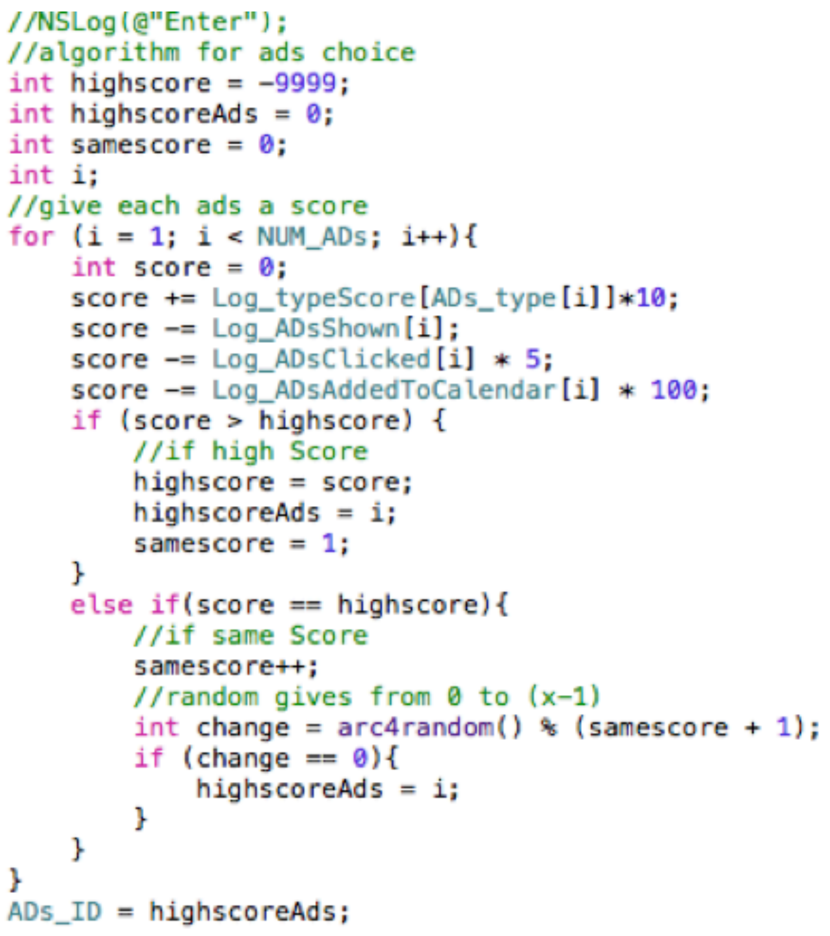
Determine if the device is entering or leaving the region and determine the region index

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;
```

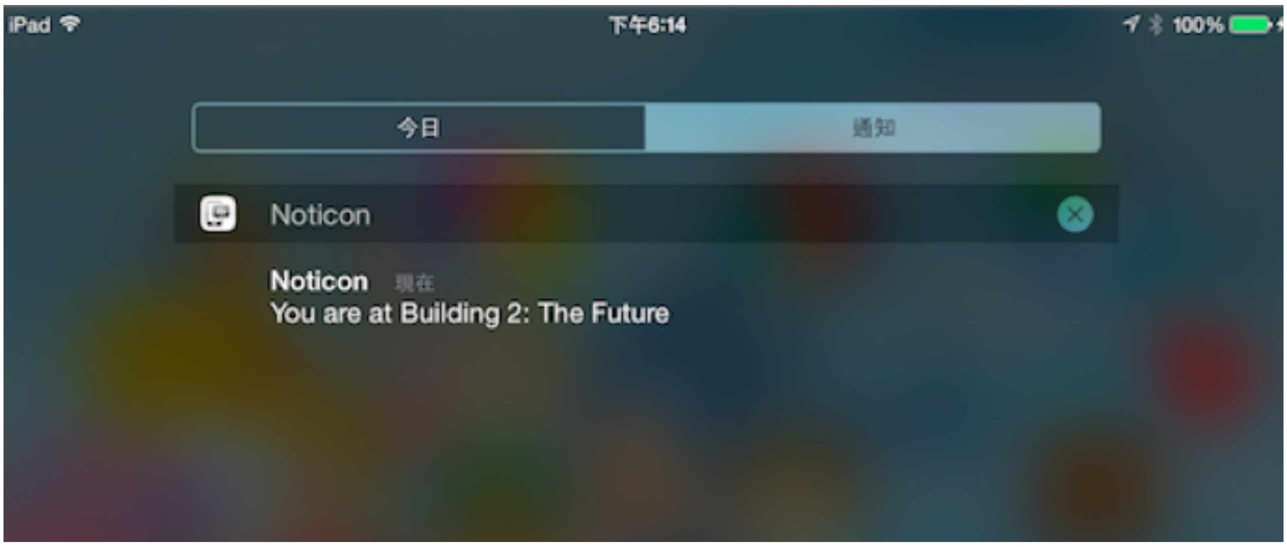
The idea is to choose one of the highest score ads with equal probability in O(n). The calculation of score will make sure that those already shown ads have a lower chance to display, while those ads of the type that user would be interested have a higher chance to display.

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];
```

After created an notification, a message will be created as shown as Fig 5.7.



Noticon 現在
You are at Building 2: The Future

Fig 5.7 Screen capture for the notification message

5.5.2 Menu

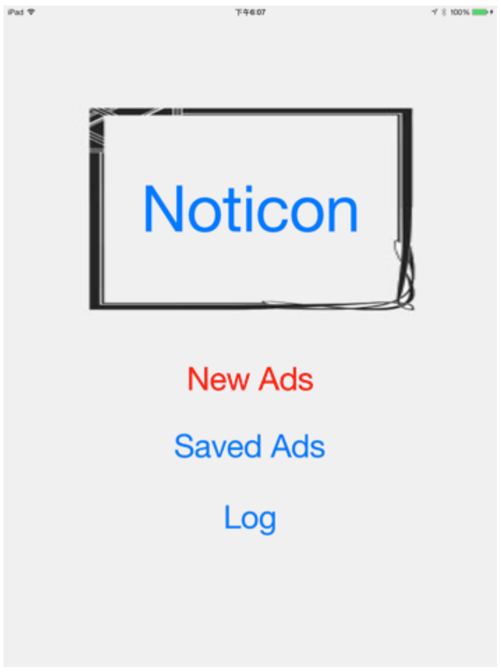


Fig 5.8 The main menu of Noticon ver 1.0

The above Fig 5.8 shows 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)**

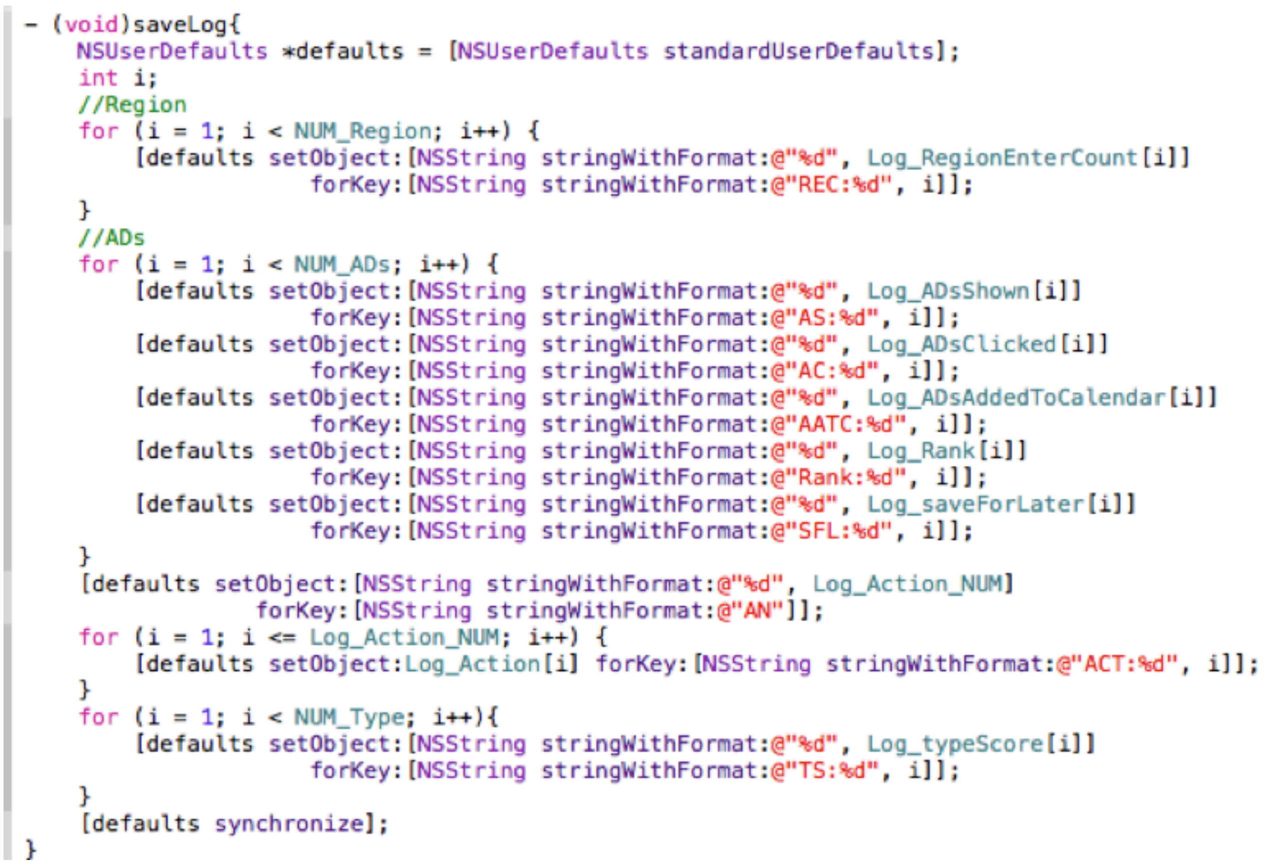
When user enters a region and an ads is given to user, user may click on the notification bar. Then this function will be called and redirect the user to 5) Display Ads automatically. For debug purpose, we can press on "New Ads" to call this function too.

- **(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. The following Fig 5.9 shows the code of saveLog.

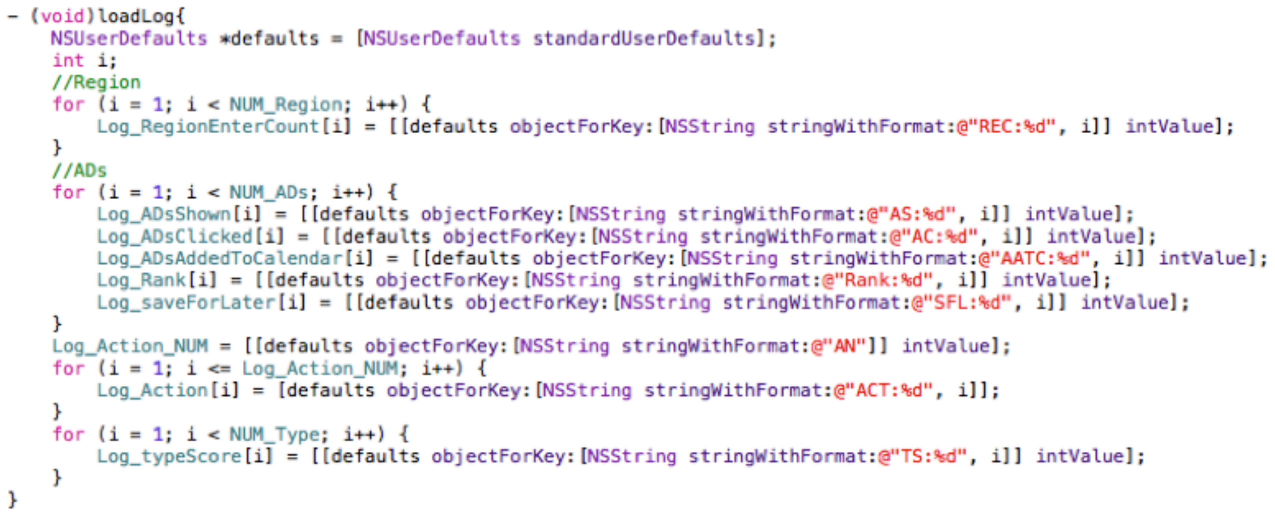


```
- (void)saveLog{
    NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
    int i;
    //Region
    for (i = 1; i < NUM_Region; i++) {
        [defaults setObject:[NSString stringWithFormat:@"%d", Log_RegionEnterCount[i]]
                     forKey:[NSString stringWithFormat:@"REC:%d", 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];
}
```

Fig 5.9 Code of saveLog

- (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. The following Fig 5.10 shows the code of loadLog.

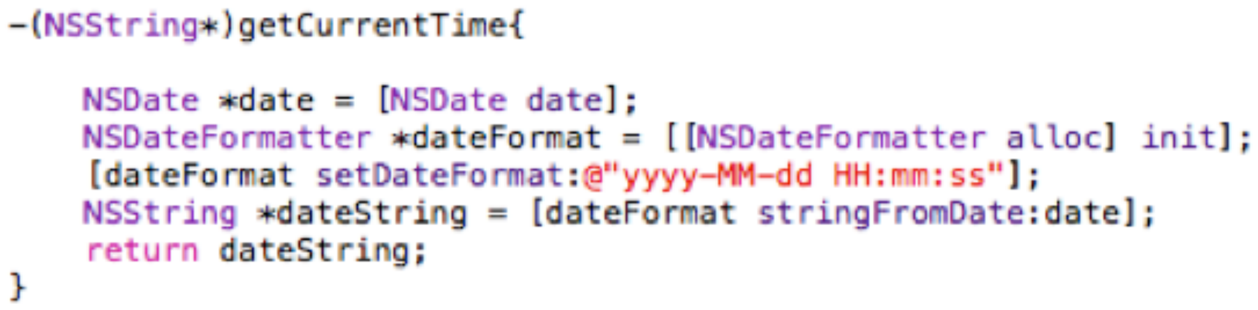


```
- (void)loadLog{
    NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
    int i;
    //Region
    for (i = 1; i < NUM_Region; i++) {
        Log_RegionEnterCount[i] = [[defaults objectForKey:[NSString stringWithFormat:@"REC:%d", i]] intValue];
    }
    //ADs
    for (i = 1; i < 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];
    }
}
```

Fig 5.10 Code of loadLog

- (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, see Fig 5.11.



```
-(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;
}
```

Fig 5.11 Code of getCurrentTime

5.5.3 Saved Ads

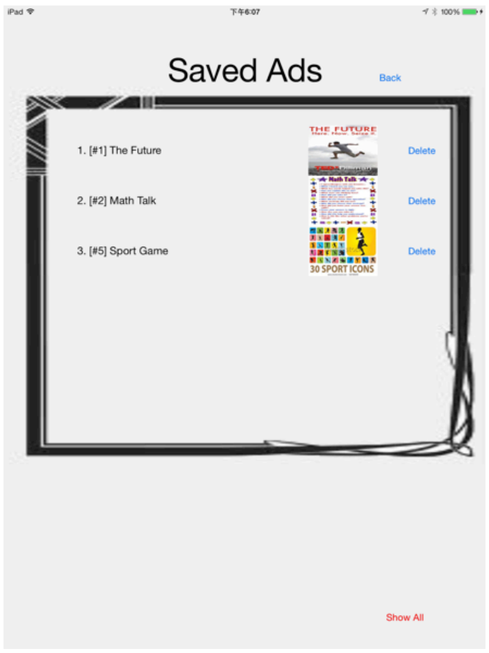


Fig 5.12 Screen capture of saved Ads page

The above Fig 5.12 shows the saved Ads page of Noticon ver 1.0

- **(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.

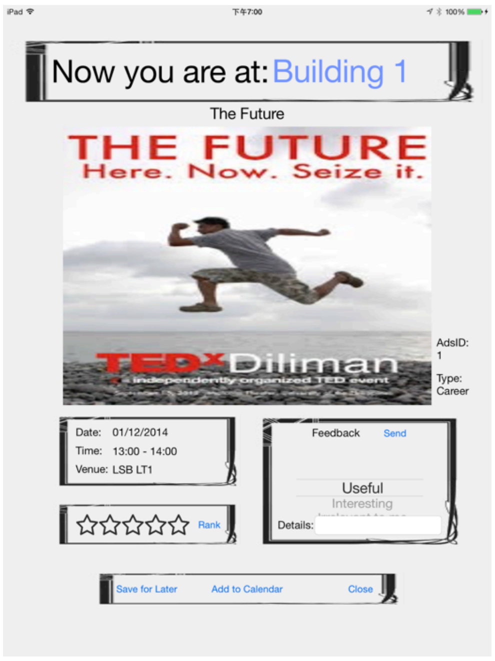
- **(void)Delete an Ads(int id)**

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



Now you are at: Building 1

The Future

AdsID: 1

Type: Career

Date: 01/12/2014
Time: 13:00 - 14:00
Venue: LSB LT1

Useful
Interesting

Details:

Fig 5.13 Screen capture of Ads page

In this view, user can view the detail information of the ads. User can also interact with the ads in different ways. The over view of this page has been shown in Fig 5.13.

- **(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

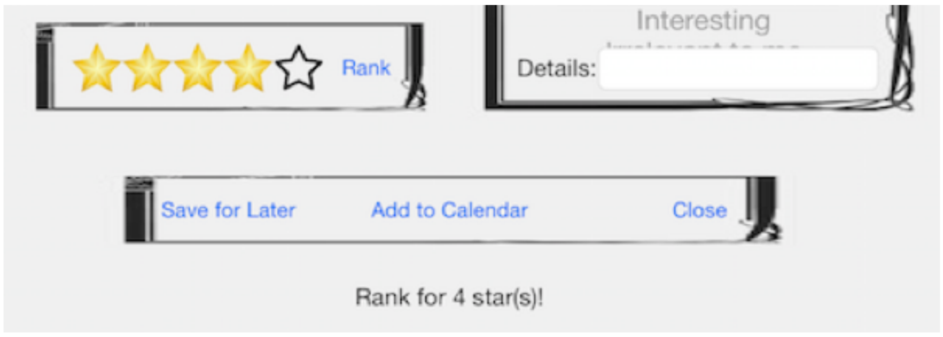


Fig 5.14 Screen capture of ranking an Ads

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 as shown in Fig 5.14. 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

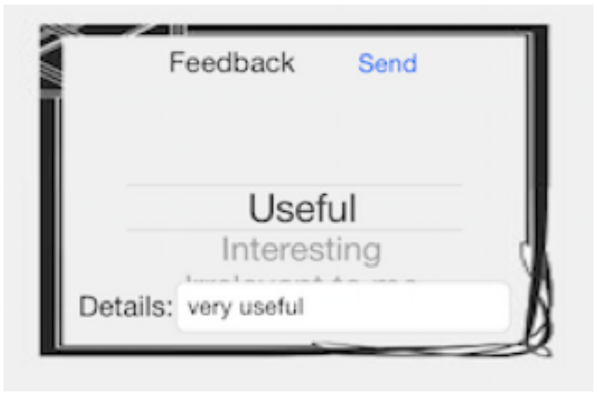
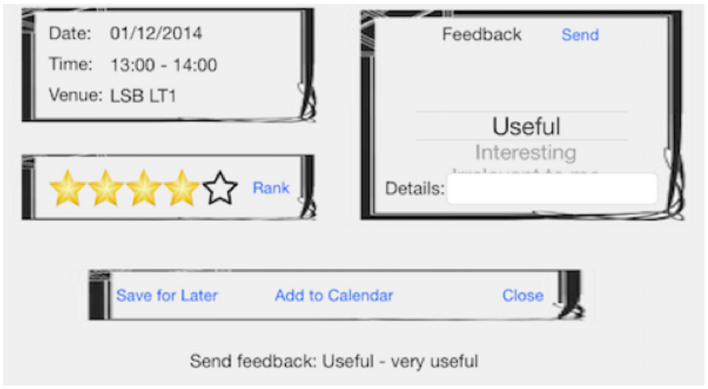


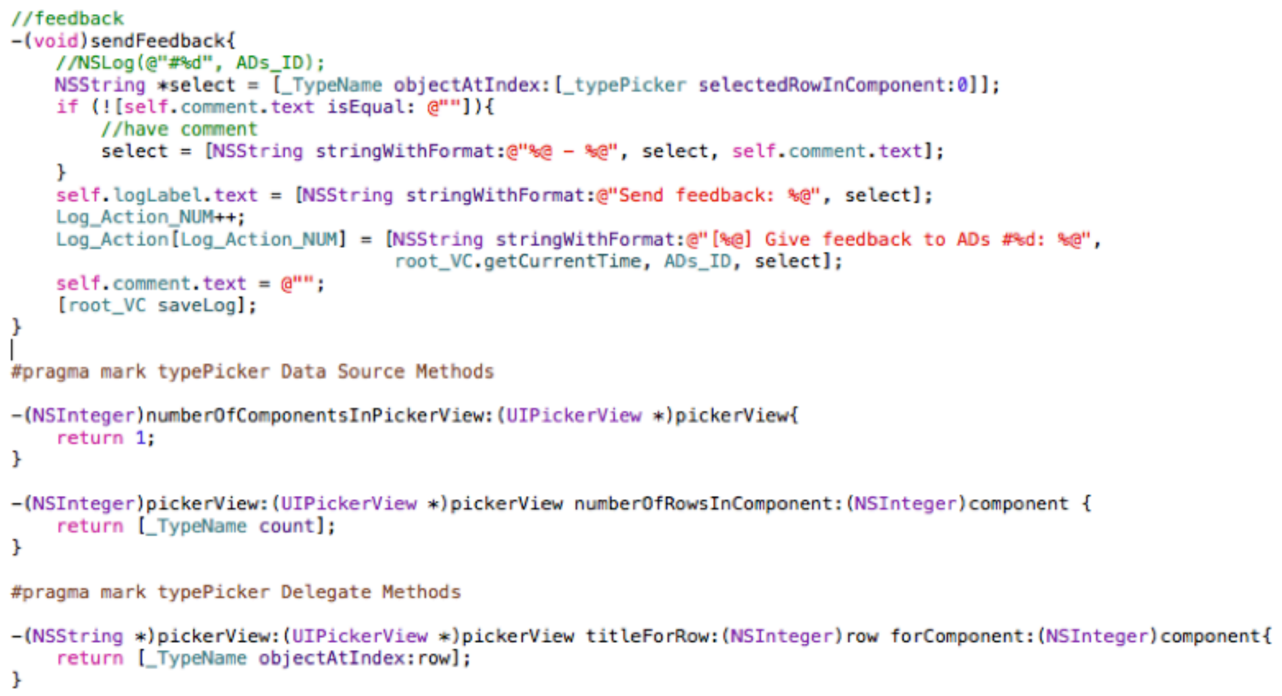
Fig 5.15 Screen capture of the feedback



Send feedback: Useful - very useful

Fig 5.16 Screen capture of the successful send feedback message

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 as shown in Fig 5.15. Then the user can press "send" to send the feedback. After pressing the "send", a message will be returned, see Fig 5.16. For the coding part of this function, refer to Fig 5.17.



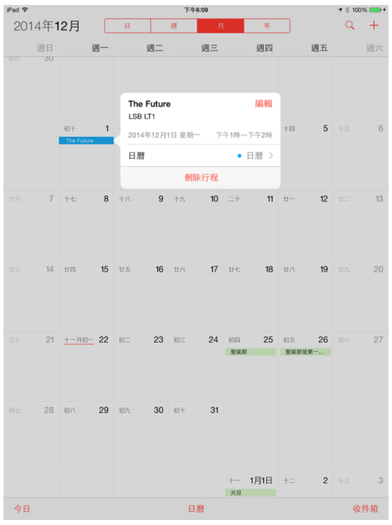
```
//feedback
-(void)sendFeedback{
    //NSLog(@"#%d", ADs_ID);
    NSString *select = [_TypeName objectAtIndex:[_typePicker selectedRowInComponent:0]];
    if (![self.comment.text isEqual: @""]){
        //have comment
        select = [NSString stringWithFormat:@"%@ - %@", select, self.comment.text];
    }
    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];
}
```

Fig 5.17 Code of sendFeedback

- (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 as shown in Fig 5.18. We use <eventKit> to do this, refer to Fig 5.19.

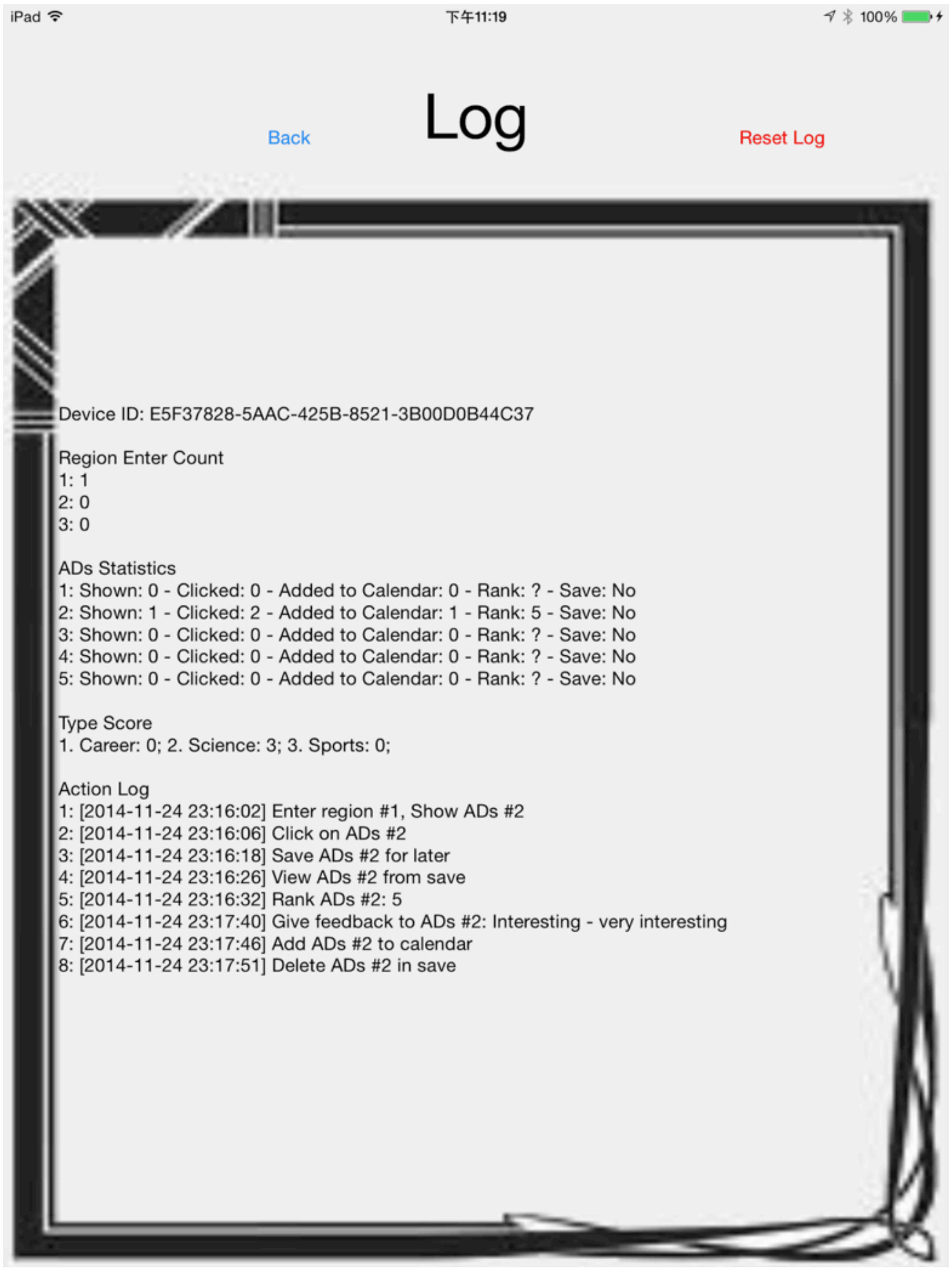
Fig 5.18 Screen capture of adding to calendar

```
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];
            }
        });
    }];
}
```

Fig 5.19 Code of event Kit

5.5.5 Log

In this view, it displays the contents of the Log class variables as shown in Fig 5.20. 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.



```
Device ID: E5F37828-5AAC-425B-8521-3B00D0B44C37

Region Enter Count
1: 1
2: 0
3: 0

ADs Statistics
1: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No
2: Shown: 1 - Clicked: 2 - Added to Calendar: 1 - Rank: 5 - Save: No
3: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No
4: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No
5: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No

Type Score
1. Career: 0; 2. Science: 3; 3. Sports: 0;

Action Log
1: [2014-11-24 23:16:02] Enter region #1, Show ADs #2
2: [2014-11-24 23:16:06] Click on ADs #2
3: [2014-11-24 23:16:18] Save ADs #2 for later
4: [2014-11-24 23:16:26] View ADs #2 from save
5: [2014-11-24 23:16:32] Rank ADs #2: 5
6: [2014-11-24 23:17:40] Give feedback to ADs #2: Interesting - very interesting
7: [2014-11-24 23:17:46] Add ADs #2 to calendar
8: [2014-11-24 23:17:51] Delete ADs #2 in save
```

Fig 5.20 Screen capture of Log page

- **(void)Back to Menu**

Press "Back" and it goes to 5.5.2 main menu.

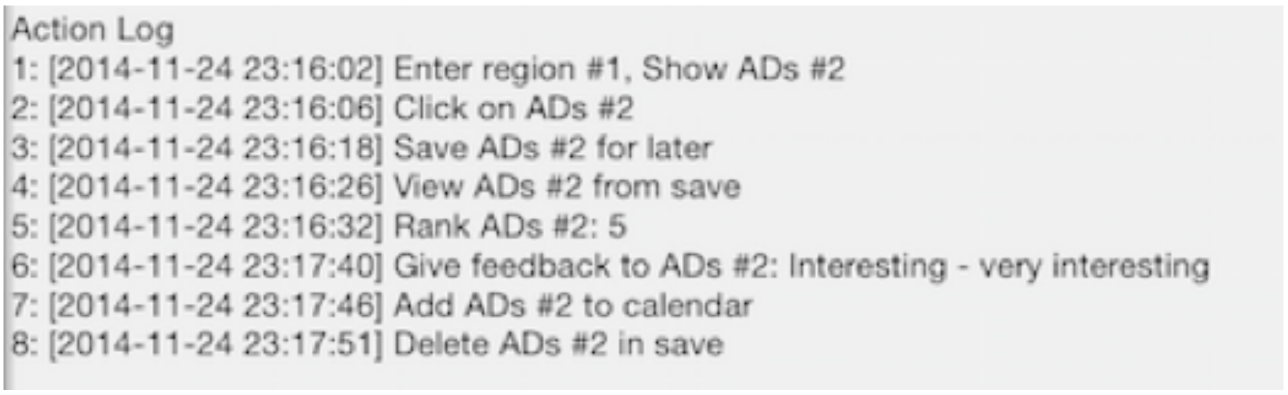
- **(void)Display log**

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**



```
Action Log
1: [2014-11-24 23:16:02] Enter region #1, Show ADs #2
2: [2014-11-24 23:16:06] Click on ADs #2
3: [2014-11-24 23:16:18] Save ADs #2 for later
4: [2014-11-24 23:16:26] View ADs #2 from save
5: [2014-11-24 23:16:32] Rank ADs #2: 5
6: [2014-11-24 23:17:40] Give feedback to ADs #2: Interesting - very interesting
7: [2014-11-24 23:17:46] Add ADs #2 to calendar
8: [2014-11-24 23:17:51] Delete ADs #2 in save
```

Fig 5.21 The content of action log

The above log(refer to Fig 5.21) 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
   (refer to Fig 5.22)

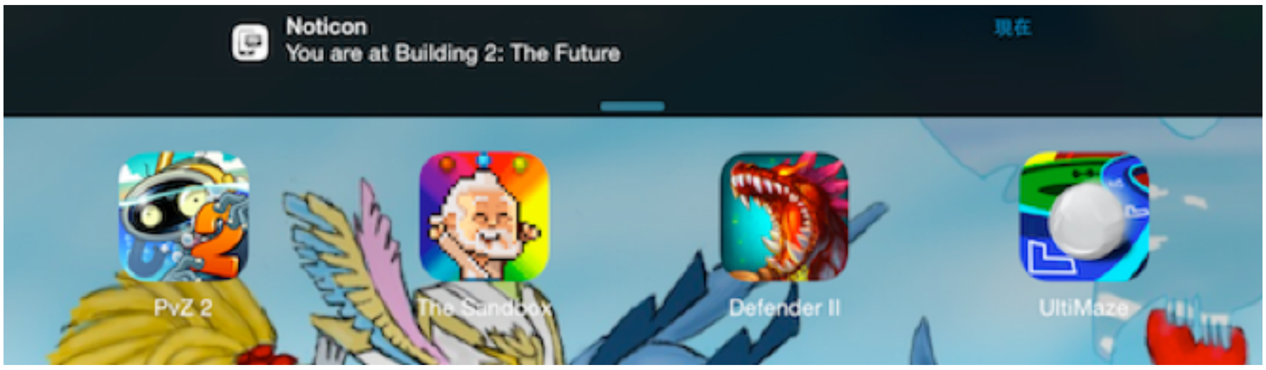


Fig 5.22 Screen capture of IOS Notification bar

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

Chapter 6 Noticon version 2.0

6.1 Overview

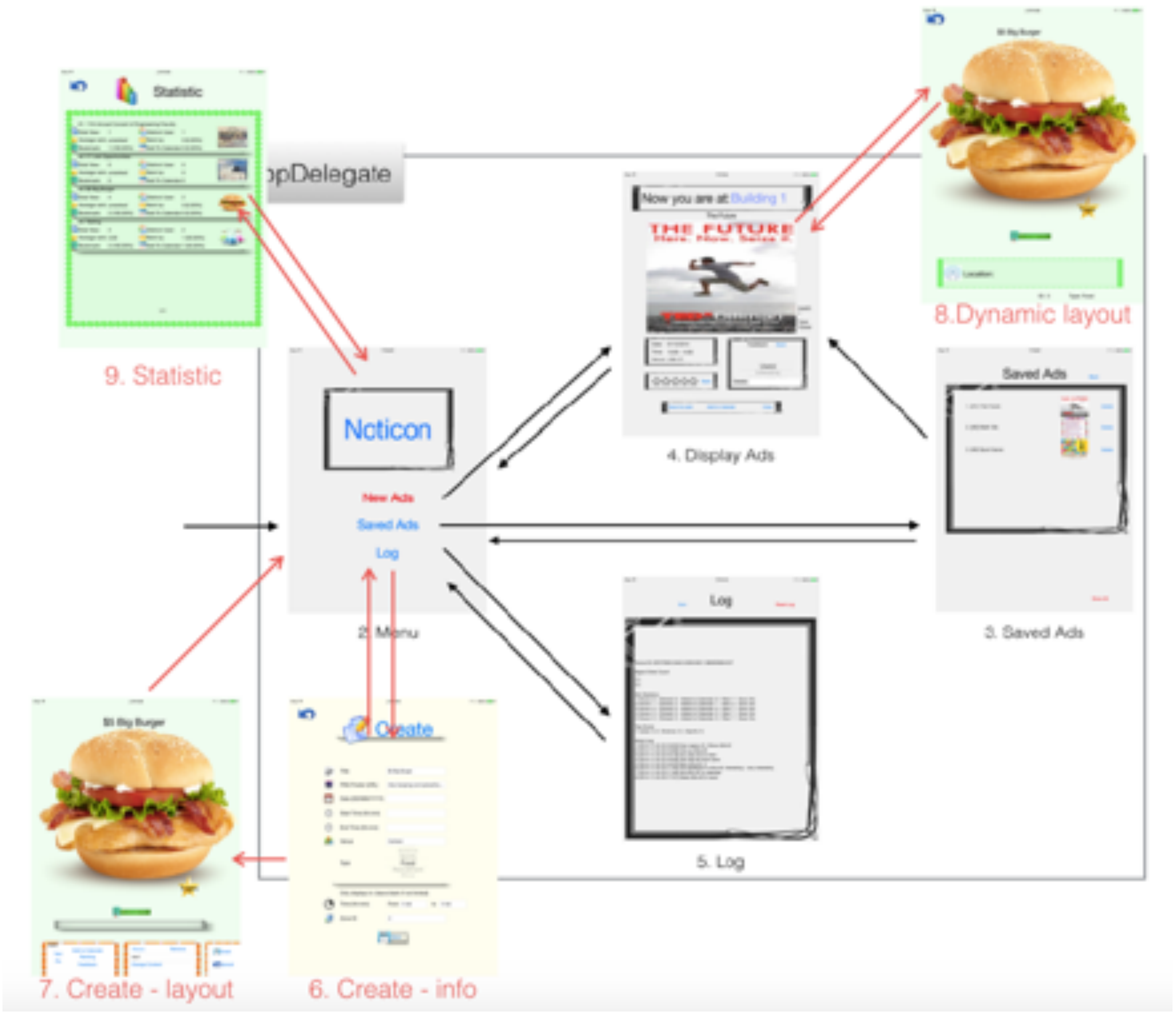


Fig 6.1 Overview of Noticon ver 2.0

In this semester, we enhanced the function in Noticon(refer to Fig 6.1). Moreover, we implemented new function (in red). We allow the administrator to create new “Noticon”(An advertisement) in <6. Create - info>. After that, if the administrator do not like the default layout of the “Noticon”, a dynamic layout can be created specially for that “Noticon” in <7. Create - layout>. As dynamic layouts are created, when user open the “Noticon” in <4. Display>, the app will check if a dynamic layout is created. If yes, the app will display the information accordingly in <8. Dynamic layout>. Administrator can also view statistics of all “Noticon” in <9. Statistic>.

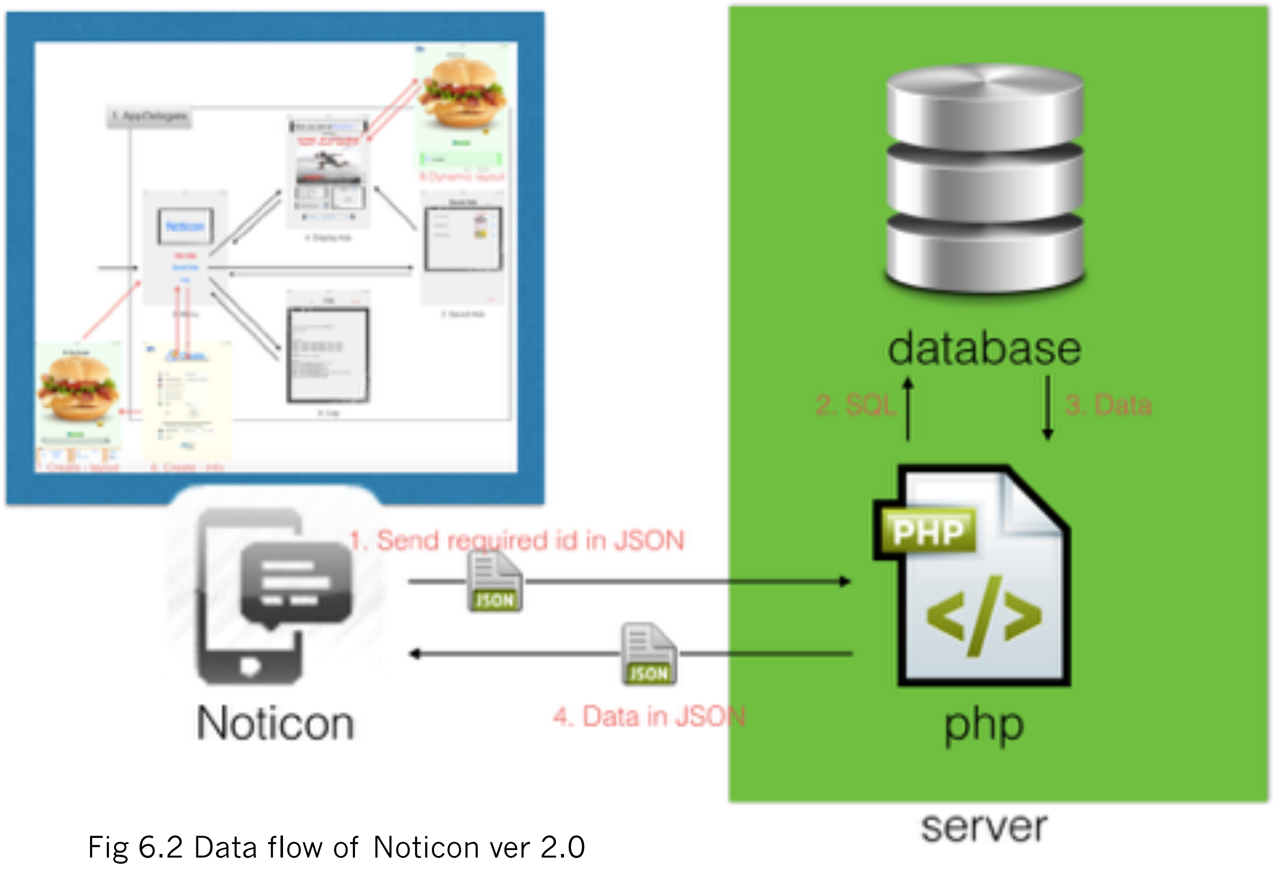


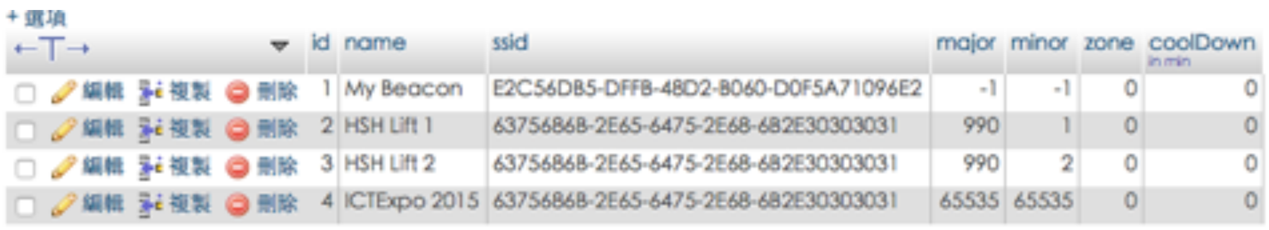
Fig 6.2 Data flow of Noticon ver 2.0

Also, we are no longer hardcoding the data. We are now connecting to database to download "Noticon", a piece of information that will be sent to the user, as well as uploading the log and new "Noticon". Since Xcode does not allow to directly connect to a database, we need to access to php file and "POST" inputs in json format <Step 1>. Then in php files, we do SQL <Step 2> in database and get data <Step 3>. Then in the php we echo the data in json format. The app will decode json and save data in memory <Step 4>, the details have been shown in above Fig 6.2.

6.2 Database implementation

Different schemas have been created to facilitate the Noticon's features.

6.2.1 Region



+ 選項

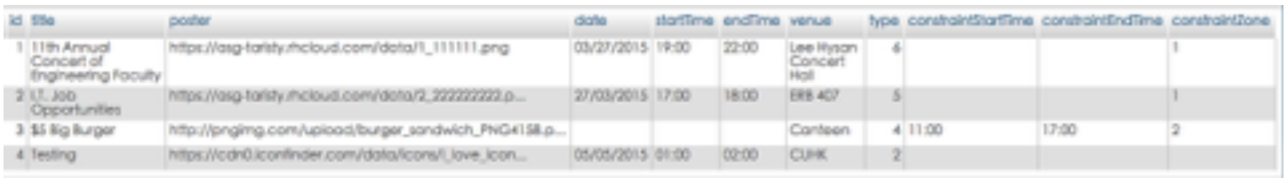
	id	name	ssid	major	minor	zone	coolDown in min
編輯 複製 刪除	1	My Beacon	E2C56DB5-DFFB-48D2-B060-D0F5A71096E2	-1	-1	0	0
編輯 複製 刪除	2	HSH Lift 1	6375686B-2E65-6475-2E68-682E30303031	990	1	0	0
編輯 複製 刪除	3	HSH Lift 2	6375686B-2E65-6475-2E68-682E30303031	990	2	0	0
編輯 複製 刪除	4	ICTExpo 2015	6375686B-2E65-6475-2E68-682E30303031	65535	65535	0	0

Fig 6.3 Schema of Region

Store the region for ibeacon (refer to Fig 6.3)

- id: id of region
- name: name of region
- ssid: SSID of iBeacon to be monitored
- major: Major of iBeacon to be monitored, with -1 as wildcard
- minor: Minor of iBeacon to be monitored, with -1 as wildcard.
- zone: Group different regions together as a zone for selecting "Noticon" to display
- coolDown: If the user stay in the same zone, the coolDown time(in minute) for not displaying new "Noticon" so that the user will not be overwhelmed by notifications

6.2.2 Ads



id	title	poster	date	startTime	endTime	venue	type	constraintStartTime	constraintEndTime	constraintZone
1	11th Annual Concert of Engineering Faculty	https://asg-taristy.rhcloud.com/data/1_111111.png	03/27/2015	19:00	22:00	Lee Hysan Concert Hall	6			1
2	I.T. Job Opportunities	https://asg-taristy.rhcloud.com/data/2_222222222.p...	27/03/2015	17:00	18:00	ERB 407	5			1
3	$5 Big Burger	http://pngimg.com/upload/burger_sandwich_PNG4158.p...				Canteen	4	11:00	17:00	2
4	Testing	https://cdn0.iconfinder.com/data/icons/i_love_icon...	05/05/2015	01:00	02:00	CUHK	2			

Fig 6.4 Schema of Ads

Store the “Noticon” information (refer to Fig 6.4)

- id: id of “Noticon”
- title: title of “Noticon”
- poster: URL of downloading the poster
- date: date of event for adding to calendar
- startTime: Start time of event for adding to calendar
- endTime: End Time of event for adding to calendar
- venue: Venue of event for adding to calendar
- type: id of type of the “Noticon” (see <3. Type>)
- constraintStartTime, constraintEndTime: The “Noticon” will only be chosen to display in this time period of a day. Empty for no constraint.
- constraintZone: The “Noticon” will only be chosen to display if the user get into that zone. Empty for no constraint.

6.2.3 Type

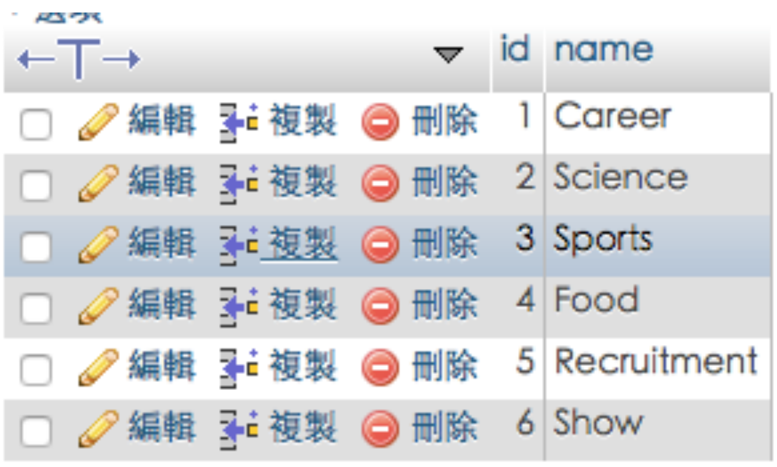

	id	name
←T→ ▼		
☐ 編輯 複製 刪除	1	Career
☐ 編輯 複製 刪除	2	Science
☐ 編輯 複製 刪除	3	Sports
☐ 編輯 複製 刪除	4	Food
☐ 編輯 複製 刪除	5	Recruitment
☐ 編輯 複製 刪除	6	Show

Fig 6.5 Schema of Type

Store the type of "Noticon", which is used to identify the favors of users and selecting "Noticon" to be displayed (refer to Fig 6.5).

- id: id of type
- name: name of the type

6.2.4 Layout1

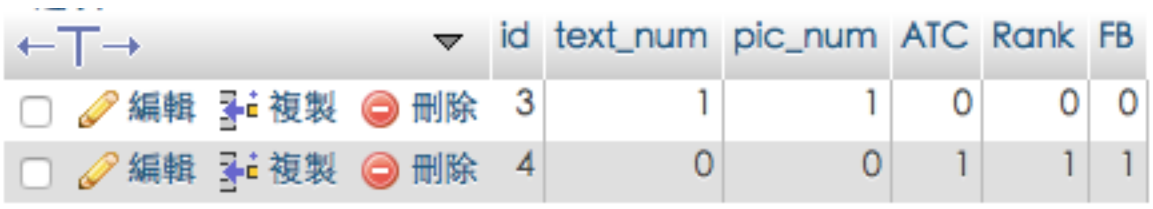
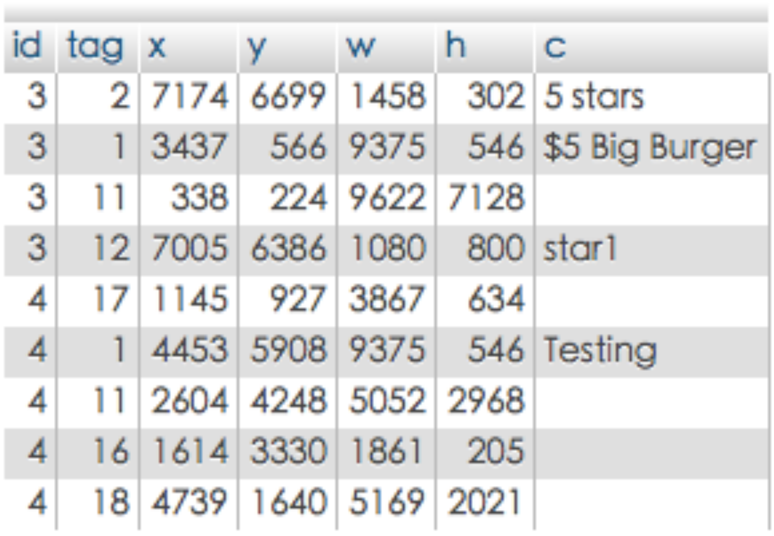

	id	text_num	pic_num	ATC	Rank	FB
☐ 編輯 複製 刪除	3	1	1	0	0	0
☐ 編輯 複製 刪除	4	0	0	1	1	1

Fig 6.6 Schema of Layout1

Store the dynamic layout information of "Noticon" (refer to Fig 6.6).

- id: id of "Noticon". If the id have no record, the "Noticon" uses default layout.
- text_num: Number of text added except the title
- pic_num: Number of picture added except the main poster image
- ATC: If "add to calendar" function is added
- Rank: If "Ranking" function is added
- FB: If "feedback" function is added

6.2.5 Layout2



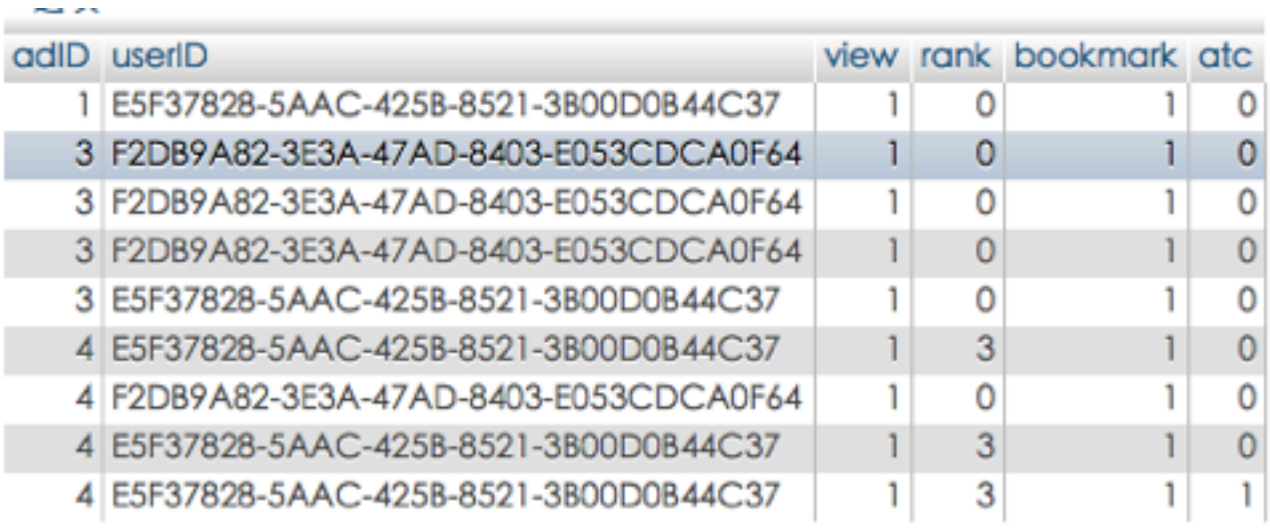
id	tag	x	y	w	h	c
3	2	7174	6699	1458	302	5 stars
3	1	3437	566	9375	546	$5 Big Burger
3	11	338	224	9622	7128	
3	12	7005	6386	1080	800	star1
4	17	1145	927	3867	634	
4	1	4453	5908	9375	546	Testing
4	11	2604	4248	5052	2968	
4	16	1614	3330	1861	205	
4	18	4739	1640	5169	2021	

Fig 6.7 Schema of Layout2

Store the dynamic layout tags information of “Noticon” (refer to Fig 6.7).

- id: id of “Noticon”.
- tag: Representation id of an object.
  - 1: title
  - 2-10: text
  - 11: main poster image
  - 12-15: image
  - 16: ATC
  - 17: Rank
  - 18: FB
- x: relative x-coordinate of location to screen size
- y: relative y-coordinate of location to screen size
- w: relative width to screen size
- h: relative height to screen size
- c: Content of object.

6.2.6 Statistic

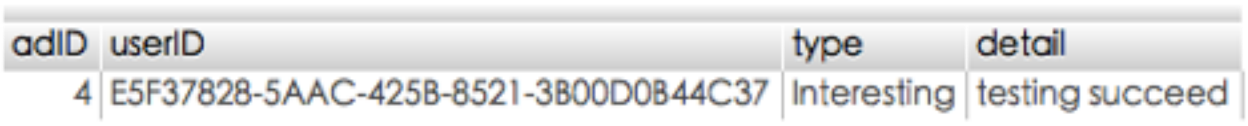


adID	userID	view	rank	bookmark	atc
1	E5F37828-5AAC-425B-8521-3B00D0B44C37	1	0	1	0
3	F2DB9A82-3E3A-47AD-8403-E053CDCA0F64	1	0	1	0
3	F2DB9A82-3E3A-47AD-8403-E053CDCA0F64	1	0	1	0
3	F2DB9A82-3E3A-47AD-8403-E053CDCA0F64	1	0	1	0
3	E5F37828-5AAC-425B-8521-3B00D0B44C37	1	0	1	0
4	E5F37828-5AAC-425B-8521-3B00D0B44C37	1	3	1	0
4	F2DB9A82-3E3A-47AD-8403-E053CDCA0F64	1	0	1	0
4	E5F37828-5AAC-425B-8521-3B00D0B44C37	1	3	1	0
4	E5F37828-5AAC-425B-8521-3B00D0B44C37	1	3	1	1

Fig 6.8 Schema of statistic

Store some log of users uploaded (refer to Fig 6.8).
- adID: id of "Noticon".
- userID: Device id of uploading the log
- view: If opened to view the "Noticon"
- rank: Number of stars ranked(1-5), with 0 of unranked
- bookmark: If added to bookmark
- atc: If added to Calendar

6.2.7 Feedback



adID	userID	type	detail
4	E5F37828-5AAC-425B-8521-3B00D0B44C37	Interesting	testing succeed

Fig 6.9 Schema of Feedback

Store feedback of users uploaded (refer to Fig 6.9).
- adID: id of "Noticon".
- userID: Device id of uploading the log
- type: Type of feedback
- detail: detail of feedback

6.3 New/enhanced Function of Noticon version 2.0

Below Fig 6.10 shows the program flow of Noticon:

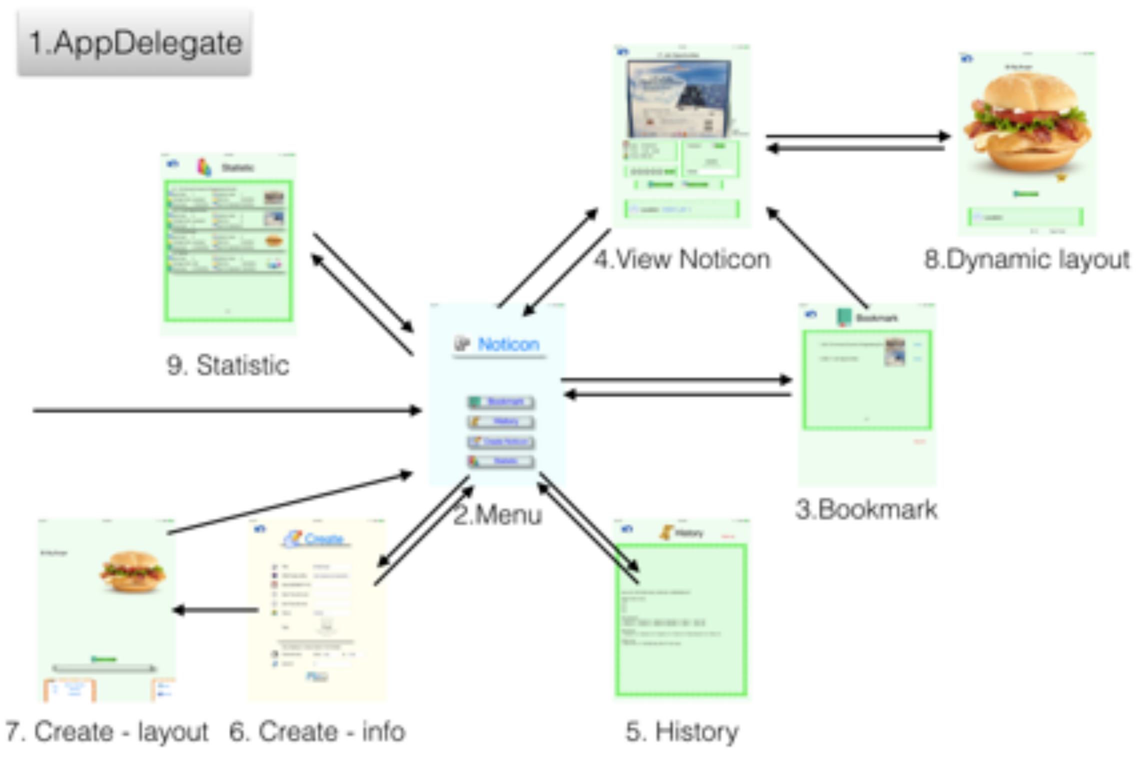


Fig 6.10 Program flow of Noticon ver 2.0

1. AppDelegate
   Control the background flow.
2. Menu
   Main Menu.
3. Bookmark
   Bookmark for user to review those seen “Noticon”.
4. View Noticon
   View and interact with the “Noticon” with default layout.
5. History
   Statistics and log of the user
6. Create - info
   (Administrator only) Create a new “Noticon”.
7. Create - layout
   (Administrator only) Create the layout of “Noticon”.
8. Dynamic layout

View and interact with the "Noticon" with dynamic layout created by <7. Create - layout>.

9. Statistic

(Administrator only) View statistic of all "Noticon".

6.3.1 AppDelegate

Function done in semester 1:

A. didFinishLaunchingWithOptions
   1. Ask for authority of notification
   2. Set Location Manager
   3. Ask for authority of location service
   4. Set Region
      - Now we set region according to database

B. locationManager:didDetermineState
   1. Determine state and regionIndex
   2. Create notification
      - Now we select the "Noticon" to be displayed with the consideration of the zone, the cool down time, the constraint of time and zone
      - Below is the algorithm:

```
if (enter a region)
    if (zone != pervious zone & current time - pervious time > cool down)
        give each Ads a score
        if (the Ads belongs to other zone or current time do not belong to
        the display time zone)
            score = -9999
        choose one of the highScore Ads to display
        create notification
```

The change for

1. Do not want to overwhelm users with too much notifications
2. Give better and more relevant advertisement

There are a lot of ibeacon installed in HSH building. If we use the original algorithm, each time the user go pass a ibeacon, a notification will pop. As a result, the user may get over ten notification within a minute, which may annoy the user. With the new algorithm, the user will only get 1 notification when he/she just get in HSH and get out within several minutes.

In HSH engineering building, we may expect viewing posters about academic, while when we pass a canteen, we may expect seeing information about the food provided. Therefore, we have the "zone" concept that binding the "Noticon" with the region.

Also, the canteen provide different food at different period of time. With the method of binding the "Noticon" with time, we can achieve that.

6.3.2 Menu

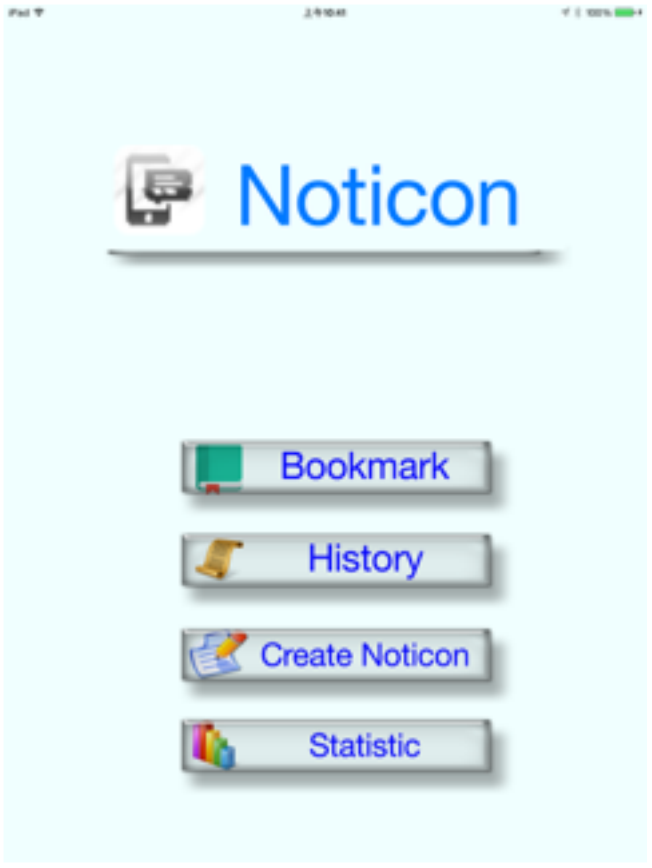


Fig 6.11 Main menu of iPad/Administrator version

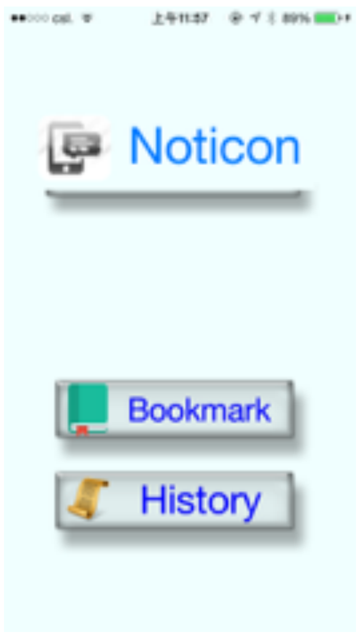


Fig 6.12 Main menu of iPhone/User version

The above Fig 6.11 shows the interface of iPad version and Fig 6.12 shows the interface of iPhone version of Noticon.

Function done in semester 1:

A. to different pages

B. (void)Save Log

C. (void)Load Log

D. (string*)Get Current Time

Function done in this semester:

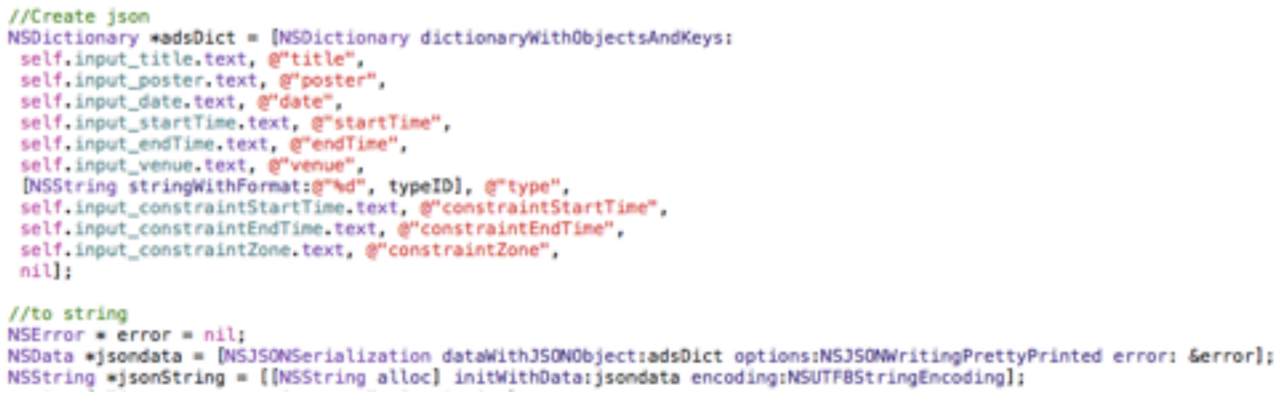
E. download data

We get the data of region, "Noticon", type from database.

The procedure of getting data from database is:

- In Xcode

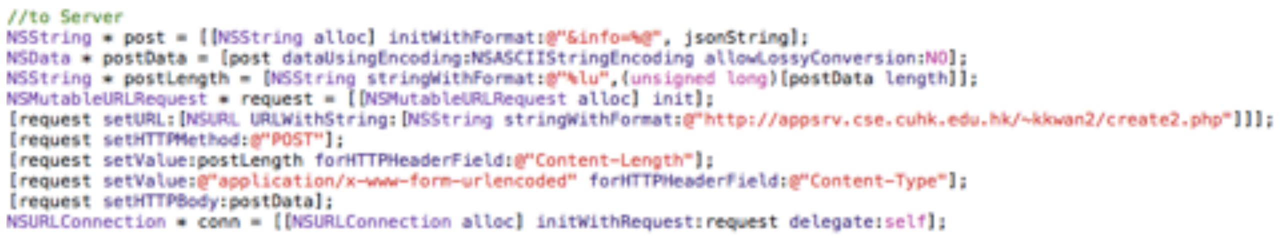
i. Create json element to post to php (refer to Fig 6.13)



```
//Create json
NSDictionary *adsDict = [NSDictionary dictionaryWithObjectsAndKeys:
 self.input_title.text, @"title",
 self.input_poster.text, @"poster",
 self.input_date.text, @"date",
 self.input_startTime.text, @"startTime",
 self.input_endTime.text, @"endTime",
 self.input_venue.text, @"venue",
 [NSString stringWithFormat:@"%d", typeID], @"type",
 self.input_constraintStartTime.text, @"constraintStartTime",
 self.input_constraintEndTime.text, @"constraintEndTime",
 self.input_constraintZone.text, @"constraintZone",
 nil];

//to string
NSError * error = nil;
NSData *jsondata = [NSJSONSerialization dataWithJSONObject:adsDict options:NSJSONWritingPrettyPrinted error: &error];
NSString *jsonString = [[NSString alloc] initWithData:jsondata encoding:NSUTF8StringEncoding];
```

Fig 6.13 Code for creating json element

ii. Mark a url request to connect to php(refer to Fig 6.14)

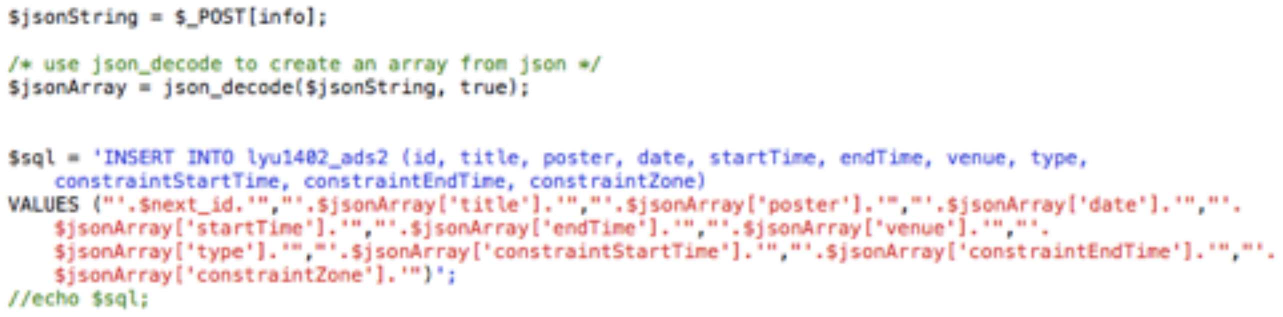


```
//to Server
NSString * post = [[NSString alloc] initWithFormat:@"&info=%@", jsonString];
NSData * postData = [post dataUsingEncoding:NSASCIIStringEncoding allowLossyConversion:NO];
NSString * postLength = [NSString stringWithFormat:@"%lu",(unsigned long)[postData length]];
NSMutableURLRequest * request = [[NSMutableURLRequest alloc] init];
[request setURL:[NSURL URLWithString:[NSString stringWithFormat:@"http://appsrv.cse.cuhk.edu.hk/~kkwan2/create2.php"]]];
[request setHTTPMethod:@"POST"];
[request setValue:postLength forHTTPHeaderField:@"Content-Length"];
[request setValue:@"application/x-www-form-urlencoded" forHTTPHeaderField:@"Content-Type"];
[request setHTTPBody:postData];
NSURLConnection * conn = [[NSURLConnection alloc] initWithRequest:request delegate:self];
```

Fig 6.14 Code for marking a url request

- In corresponding PHP

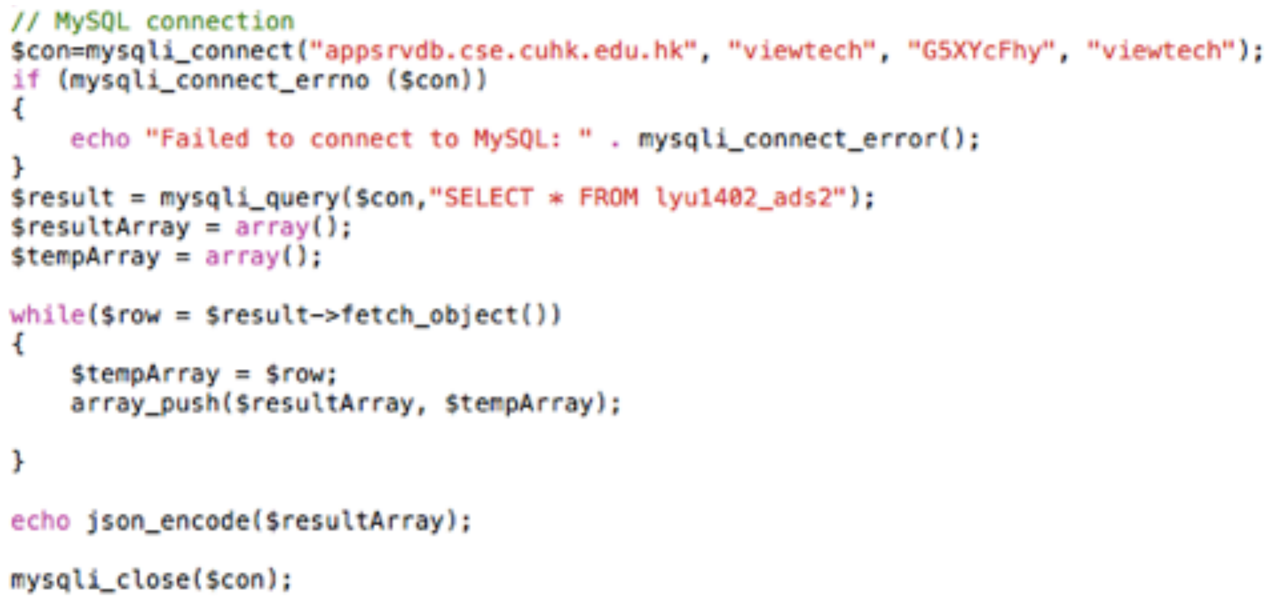
iii. make a query with posted data (refer to Fig 6.15)



```
$jsonString = $_POST[info];

/* use json_decode to create an array from json */
$jsonArray = json_decode($jsonString, true);


$sql = 'INSERT INTO lyu1402_ads2 (id, title, poster, date, startTime, endTime, venue, type,
    constraintStartTime, constraintEndTime, constraintZone)
VALUES ("'.$next_id.'","'.$jsonArray['title'].'","'.$jsonArray['poster'].'","'.$jsonArray['date'].'","'.
    $jsonArray['startTime'].'","'.$jsonArray['endTime'].'","'.$jsonArray['venue'].'","'.
    $jsonArray['type'].'","'.$jsonArray['constraintStartTime'].'","'.$jsonArray['constraintEndTime'].'","'.
    $jsonArray['constraintZone'].'")';
//echo $sql;
```

Fig 6.15 Code for making a query

iv. echo the result in json format (refer to Fig 6.16)

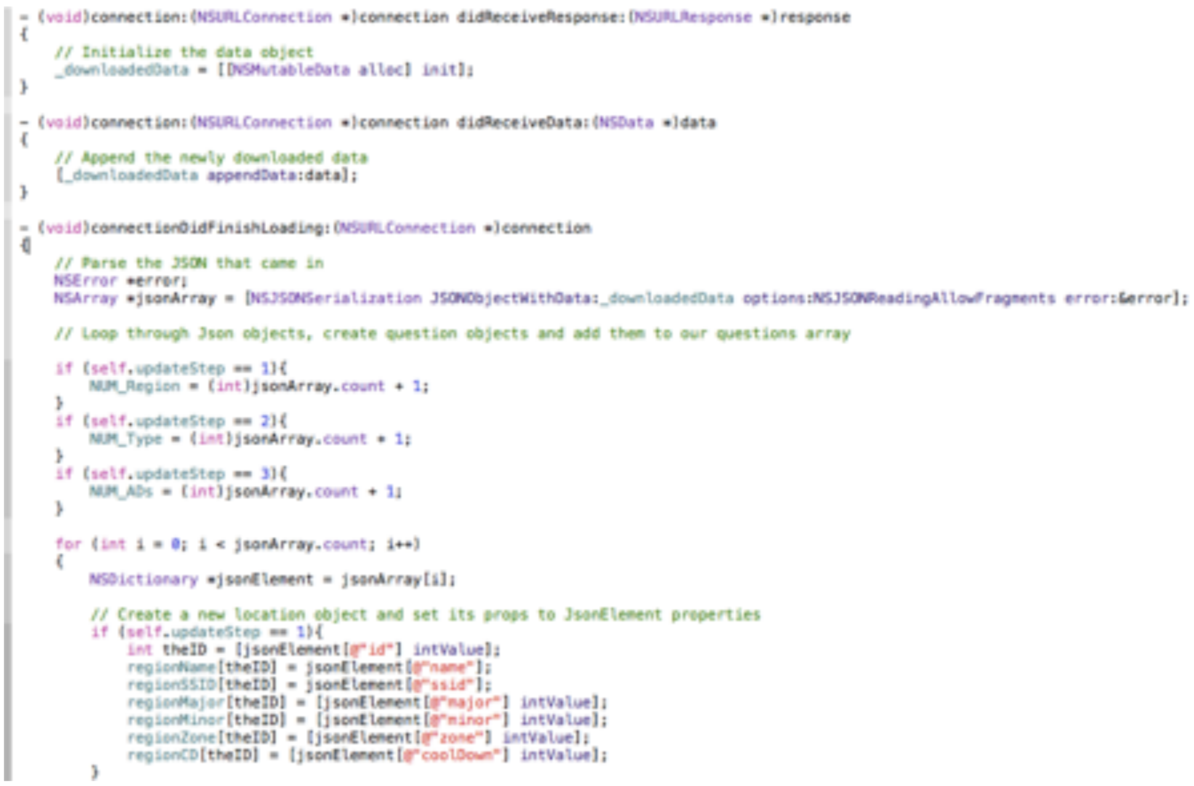


```
// MySQL connection
$con=mysqli_connect("appsrvdb.cse.cuhk.edu.hk", "viewtech", "G5XYcFhy", "viewtech");
if (mysqli_connect_errno ($con))
{
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
}
$result = mysqli_query($con,"SELECT * FROM lyu1402_ads2");
$resultArray = array();
$tempArray = array();

while($row = $result->fetch_object())
{
    $tempArray = $row;
    array_push($resultArray, $tempArray);

}

echo json_encode($resultArray);

mysqli_close($con);
```

Fig 6.16 Code for echo the results in json format

- In Xcode

v. Wait for result, decode json and store in memory



```
- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
{
    // Initialize the data object
    _downloadedData = [[NSMutableData alloc] init];
}

- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
{
    // Append the newly downloaded data
    [_downloadedData appendData:data];
}

- (void)connectionDidFinishLoading:(NSURLConnection *)connection
{
    // Parse the JSON that came in
    NSError *error;
    NSArray *jsonArray = [NSJSONSerialization JSONObjectWithData:_downloadedData options:NSJSONReadingAllowFragments error:&error];

    // Loop through Json objects, create question objects and add them to our questions array

    if (self.updateStep == 1){
        NUM_Region = (int)jsonArray.count + 1;
    }
    if (self.updateStep == 2){
        NUM_Type = (int)jsonArray.count + 1;
    }
    if (self.updateStep == 3){
        NUM_ADs = (int)jsonArray.count + 1;
    }

    for (int i = 0; i < jsonArray.count; i++)
    {
        NSDictionary *jsonElement = jsonArray[i];

        // Create a new location object and set its props to JsonElement properties
        if (self.updateStep == 1){
            int theID = [jsonElement[@"id"] intValue];
            regionName[theID] = jsonElement[@"name"];
            regionSSID[theID] = jsonElement[@"ssid"];
            regionMajor[theID] = [jsonElement[@"major"] intValue];
            regionMinor[theID] = [jsonElement[@"minor"] intValue];
            regionZone[theID] = [jsonElement[@"zone"] intValue];
            regionCD[theID] = [jsonElement[@"coolDown"] intValue];
        }
```

Fig 6.17 Code for decoding json

*Note: All the connection result call the same function when finished. So we need an extra variable(self.updateStep) to remember which php is connected and dual with the result accordingly(refer to Fig 6.17).
In this function, we connected to "getRegion2.php", "getType.php" and "getAds.php", as well as download image if not exist (<F. download image>).

An icon "Updating..." is displayed at the bottom-left to indicate the app is updating(refer to Fig 6.18)

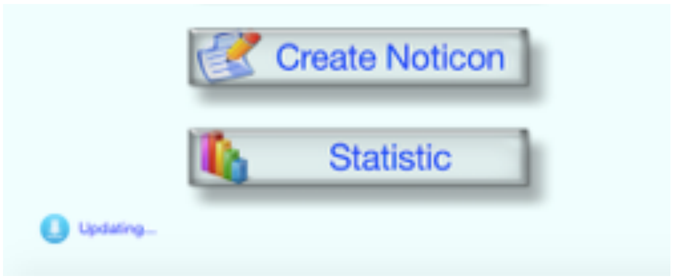
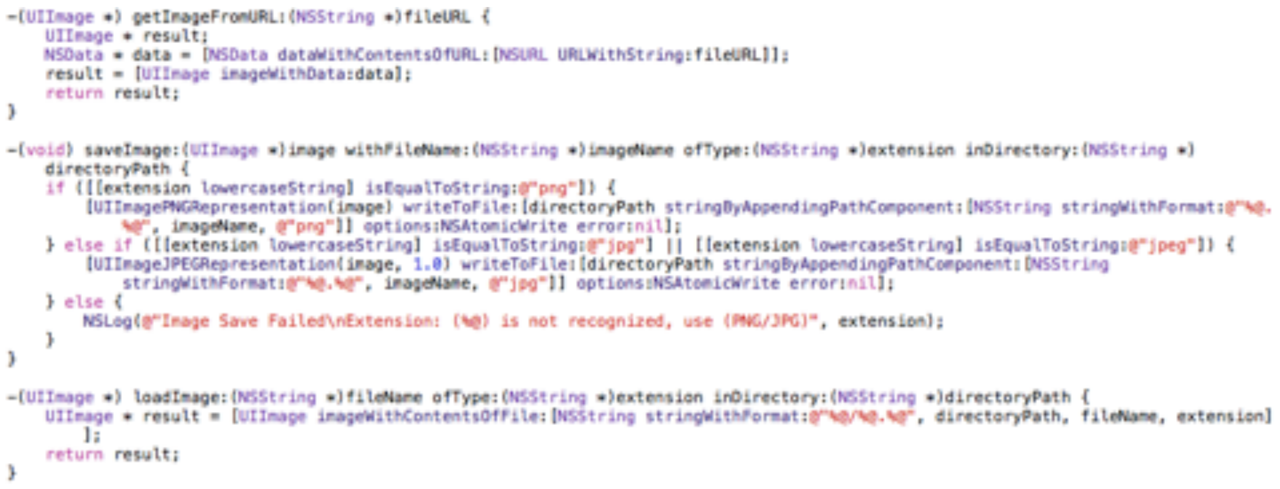


Fig 6.18 Screen capture showing "updating" message

F. download image

This function is for downloading image with a url and store to memory.

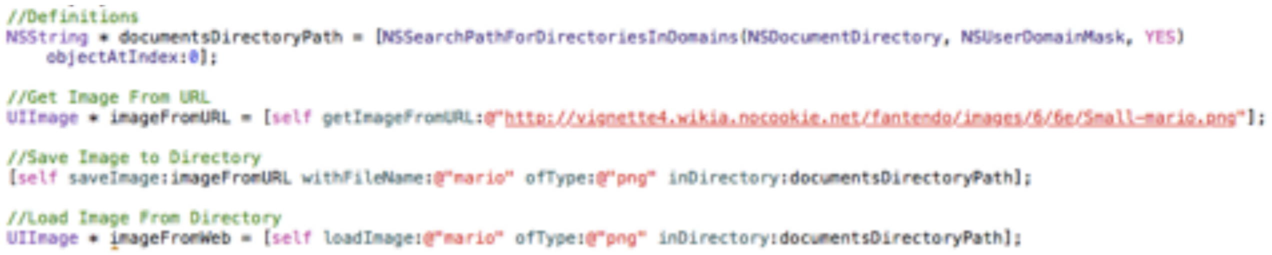


```
-(UIImage *) getImageFromURL:(NSString *)fileURL {
    UIImage * result;
    NSData * data = [NSData dataWithContentsOfURL:[NSURL URLWithString:fileURL]];
    result = [UIImage imageWithData:data];
    return result;
}

-(void) saveImage:(UIImage *)image withFileName:(NSString *)imageName ofType:(NSString *)extension inDirectory:(NSString *)
    directoryPath {
    if ([[extension lowercaseString] isEqualToString:@"png"]) {
        [UIImagePNGRepresentation(image) writeToFile:[directoryPath stringByAppendingPathComponent:[NSString stringWithFormat:@"%@.
            %@", imageName, @"png"]] options:NSAtomicWrite error:nil];
    } else if ([[extension lowercaseString] isEqualToString:@"jpg"] || [[extension lowercaseString] isEqualToString:@"jpeg"]) {
        [UIImageJPEGRepresentation(image, 1.0) writeToFile:[directoryPath stringByAppendingPathComponent:[NSString
            stringWithFormat:@"%@.%@", imageName, @"jpg"]] options:NSAtomicWrite error:nil];
    } else {
       NSLog(@"Image Save Failed\nExtension: (%@) is not recognized, use (PNG/JPG)", extension);
    }
}

-(UIImage *) loadImage:(NSString *)fileName ofType:(NSString *)extension inDirectory:(NSString *)directoryPath {
    UIImage * result = [UIImage imageWithContentsOfFile:[NSString stringWithFormat:@"%@/%@.%@", directoryPath, fileName, extension]
        ];
    return result;
}
```

Fig 6.19 Code for downloading image with url

The above Fig 6.19 shows 3 functions, they are:
getImageFromURL: download the image from fileURL
saveImage: save the image into directory
loadImage: get the image from directory

The sample usage has been shown in Fig 6.20



```
//Definitions
NSString * documentsDirectoryPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES)
    objectAtIndex:0];

//Get Image From URL
UIImage * imageFromURL = [self getImageFromURL:@"http://vignette4.wikia.nocookie.net/fantendo/images/6/6e/Small-mario.png"];

//Save Image to Directory
[self saveImage:imageFromURL withFileName:@"mario" ofType:@"png" inDirectory:documentsDirectoryPath];

//Load Image From Directory
UIImage * imageFromWeb = [self loadImage:@"mario" ofType:@"png" inDirectory:documentsDirectoryPath];
```

Fig 6.20 Sample usage of Fig 6.19 codes

6.3.3 Bookmark

Improved UI as shown in Fig 6.21 and Fig 6.22

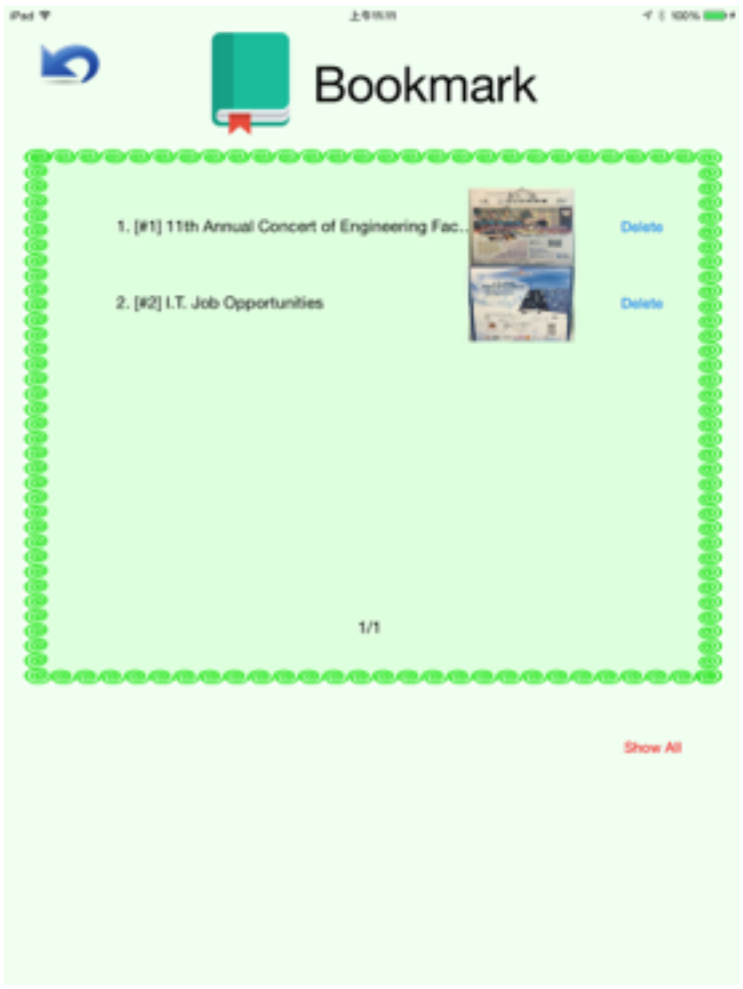

Bookmark
1. [#1] 11th Annual Concert of Engineering Fac...
Delete
2. [#2] I.T. Job Opportunities
Delete
1/1
Show All

Fig 6.21 Screen capture of iPad version

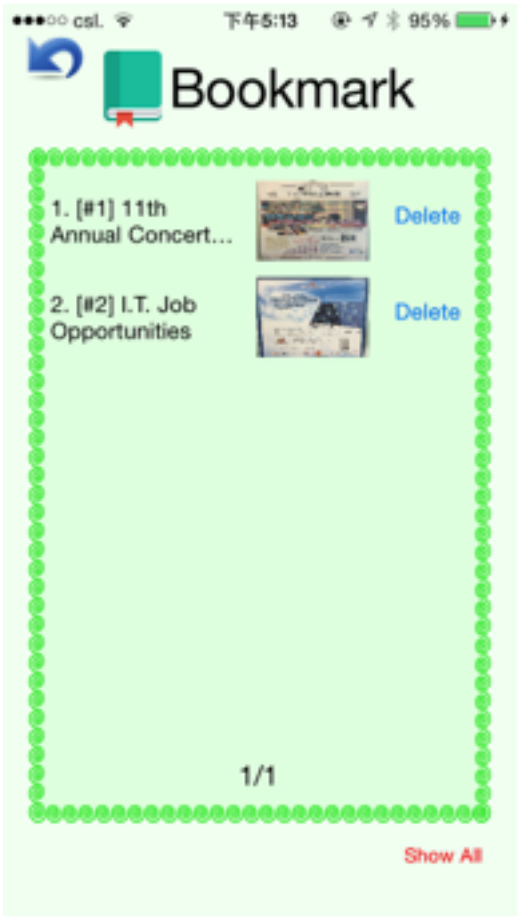

●●●○○ csl.
下午5:13
95%
Bookmark
1. [#1] 11th Annual Concert...
Delete
2. [#2] I.T. Job Opportunities
Delete
1/1
Show All

Fig 6.22 Screen capture of iPhone version

6.3.4 View Noticon

Improved UI as shown in Fig 6.23 and Fig 6.24.

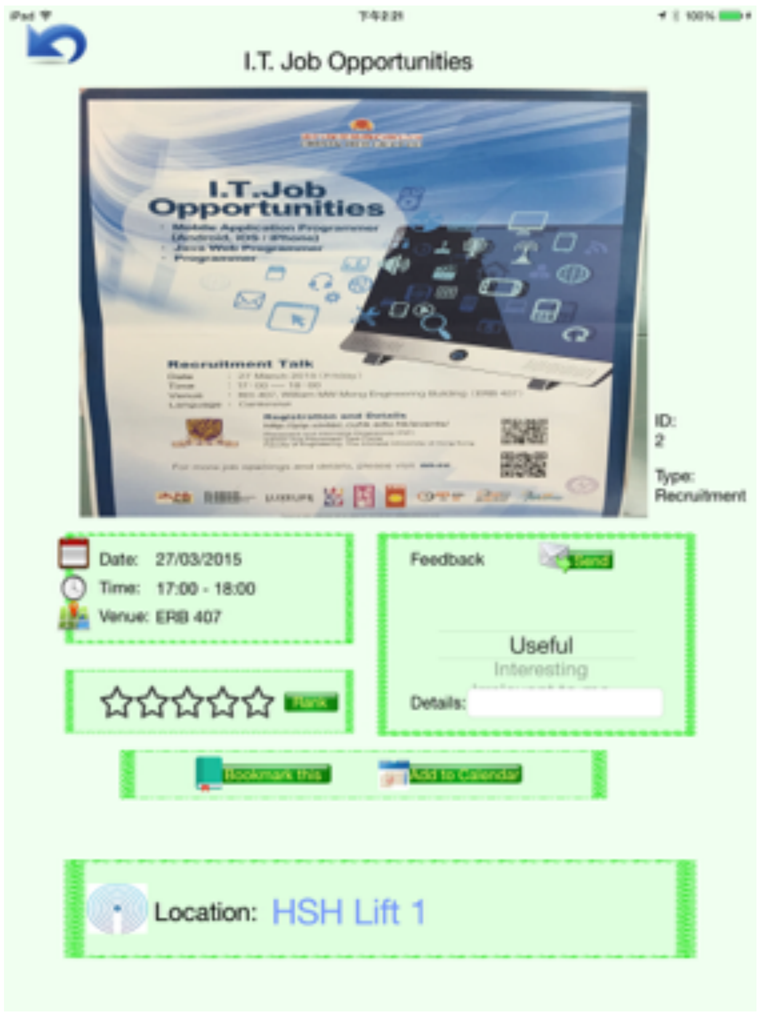


Fig 6.23 Screen capture of iPad version

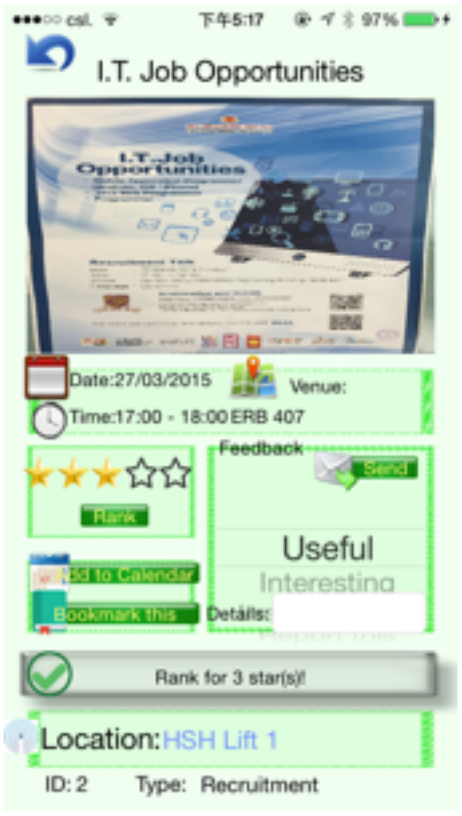
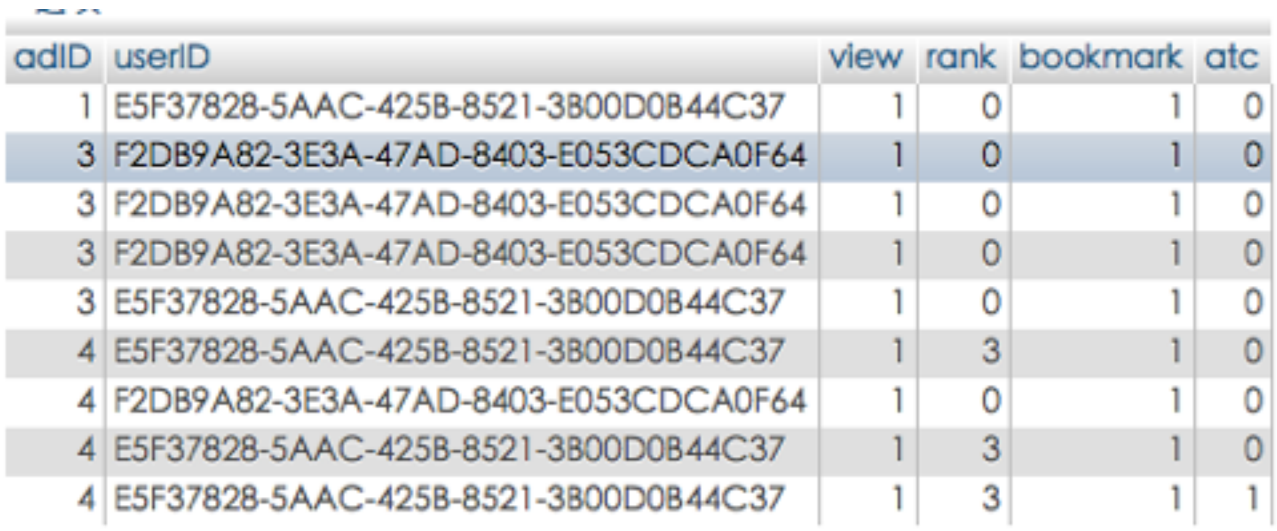


Fig 6.24 Screen capture of iPhone version

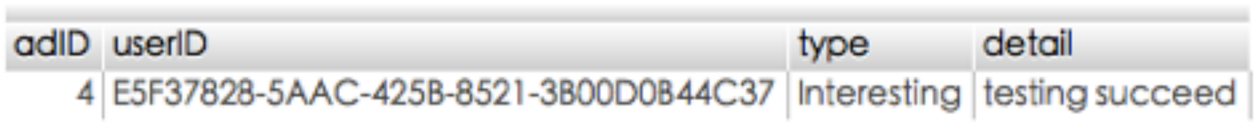
The app now send a log to the server after the user viewed the app. The app insert the log as a record to Table Statistic through "upStat.php"(refer to Fig 6.25).



adID	userID	view	rank	bookmark	atc
1	E5F37828-5AAC-425B-8521-3B00D0B44C37	1	0	1	0
3	F2DB9A82-3E3A-47AD-8403-E053CDCA0F64	1	0	1	0
3	F2DB9A82-3E3A-47AD-8403-E053CDCA0F64	1	0	1	0
3	F2DB9A82-3E3A-47AD-8403-E053CDCA0F64	1	0	1	0
3	E5F37828-5AAC-425B-8521-3B00D0B44C37	1	0	1	0
4	E5F37828-5AAC-425B-8521-3B00D0B44C37	1	3	1	0
4	F2DB9A82-3E3A-47AD-8403-E053CDCA0F64	1	0	1	0
4	E5F37828-5AAC-425B-8521-3B00D0B44C37	1	3	1	0
4	E5F37828-5AAC-425B-8521-3B00D0B44C37	1	3	1	1

Fig 6.25 Record samples in Statistic schema

Also, if the user press "send" in feedback function, the app will insert a record to Table Feedback by "upFeedback.php"(refer to Fig 6.26).

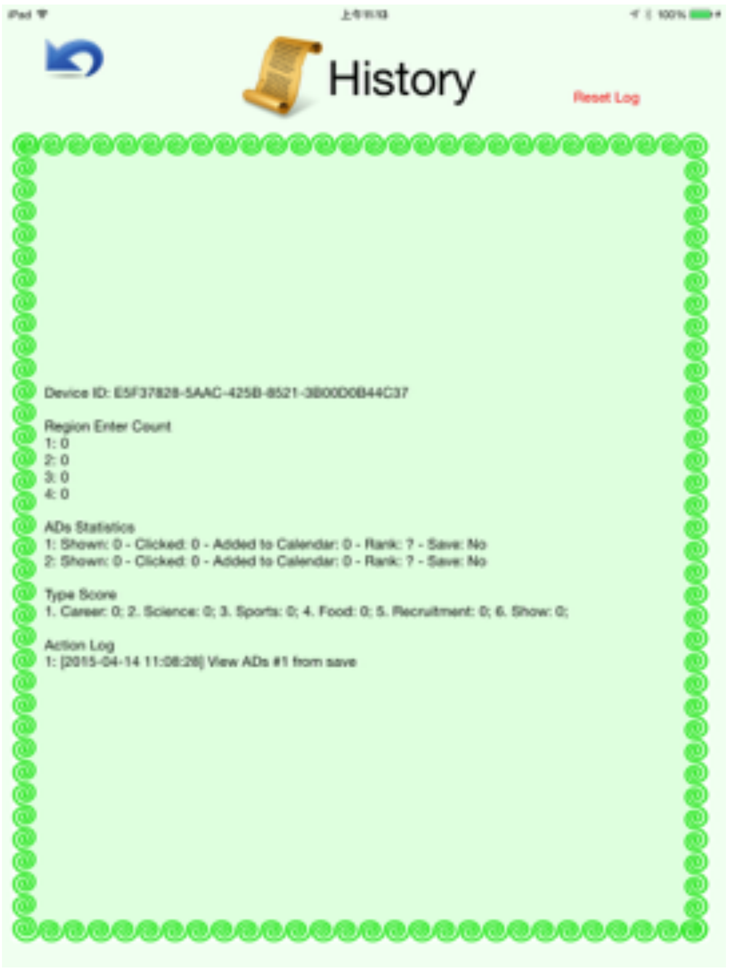


adID	userID	type	detail
4	E5F37828-5AAC-425B-8521-3B00D0B44C37	Interesting	testing succeed

Fig 6.26 Record sample in Feedback schema

6.3.5 History

Improved UI as shown in Fig 6.27 and Fig 6.28.



Device ID: E5F37828-5AAC-425B-8521-3B00D0B44C37

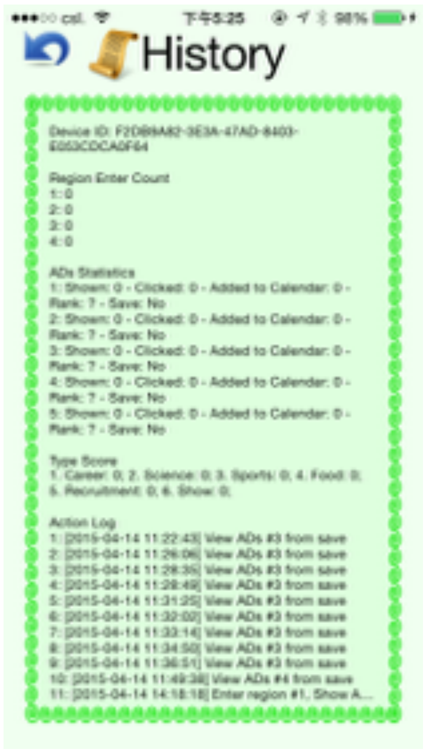
Region Enter Count
1: 0
2: 0
3: 0
4: 0

ADs Statistics
1: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No
2: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No

Type Score
1. Career: 0; 2. Science: 0; 3. Sports: 0; 4. Food: 0; 5. Recruitment: 0; 6. Show: 0;

Action Log
1: [2015-04-14 11:08:28] View ADs #1 from save

Fig 6.27 Screen capture of iPad version



Device ID: F2DB9A82-3E3A-47AD-8403-E053CDCA0F64

Region Enter Count
1: 0
2: 0
3: 0
4: 0

ADs Statistics
1: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No
2: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No
3: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No
4: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No
5: Shown: 0 - Clicked: 0 - Added to Calendar: 0 - Rank: ? - Save: No

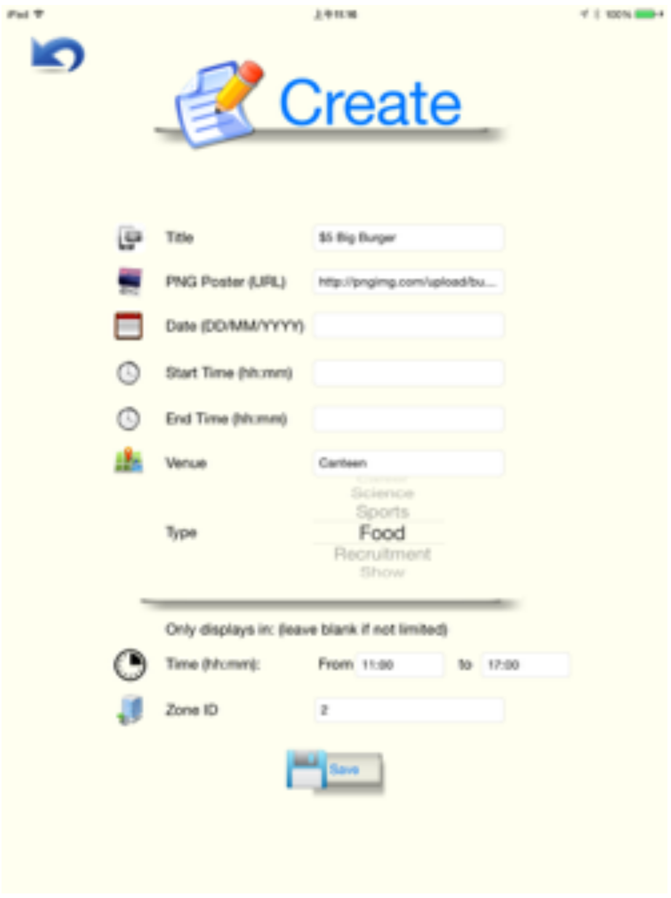
Type Score
1. Career: 0; 2. Science: 0; 3. Sports: 0; 4. Food: 0; 5. Recruitment: 0; 6. Show: 0;

Action Log
```
1: [2015-04-14 11:22:43] View ADs #3 from save
2: [2015-04-14 11:26:06] View ADs #3 from save
3: [2015-04-14 11:28:35] View ADs #3 from save
4: [2015-04-14 11:28:49] View ADs #3 from save
5: [2015-04-14 11:31:25] View ADs #3 from save
6: [2015-04-14 11:32:02] View ADs #3 from save
7: [2015-04-14 11:33:14] View ADs #3 from save
8: [2015-04-14 11:34:50] View ADs #3 from save
9: [2015-04-14 11:36:51] View ADs #3 from save
10: [2015-04-14 11:49:38] View ADs #4 from save
11: [2015-04-14 14:18:18] Enter region #1, Show A...
```

Fig 6.28 Screen capture of iPhone version

6.3.6 Create - info

The Fig 6.29 shows the create page of Noticon(Administrator only).


Create

Title	$5 Big Burger
PNG Poster (URL)	http://pngimg.com/upload/bu...
Date (DD/MM/YYYY)	
Start Time (hh:mm)	
End Time (hh:mm)	
Venue	Canteen
Type	Food

Only displays in: (leave blank if not limited)

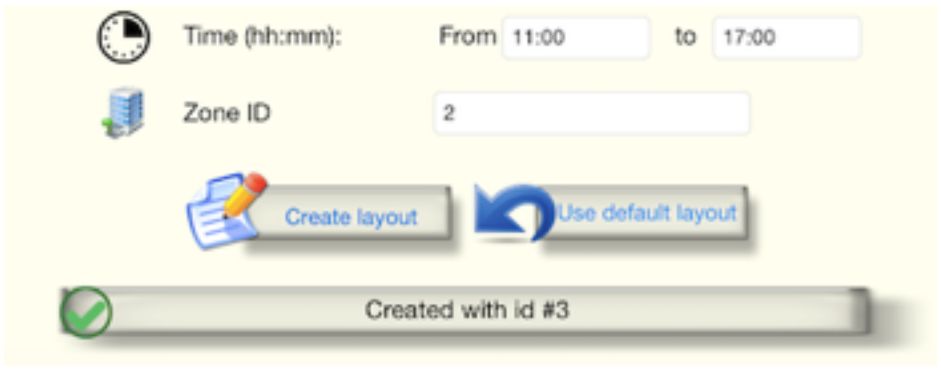
Time (hh:mm):	From	11:00	to	17:00
Zone ID	2			

Fig 6.29 Create page in iPad version

In the page, it provides a form for entering the basic info of a new “Noticon”. After clicking save button.

The app will insert a new record in Table Ads using “create2.php”. The id of the newly created “Noticon” will be returned.

Then two option appears:


Time (hh:mm): From 11:00 to 17:00

Zone ID 2

Created with id #3

Fig 6.30 Two options for creating a new ads

“Create layout” go to <7. Create - layout> while “use default layout” goes back to main menu (refer to Fig 6.30).

6.3.7 Create - layout

(Administrator only)

After entering the page, only the title and main poster image are provided in the layout(refer to Fig 6.31).

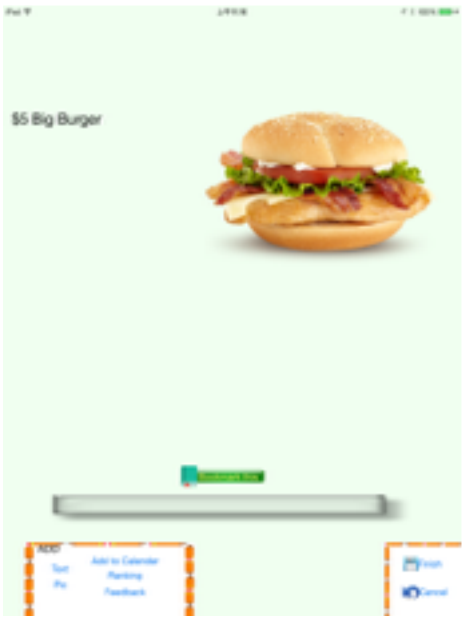


Fig 6.31 Default layout of the ads

The user can customize the layout with the following action:

1. Add text - Add a new text object (maximum of 9)
2. Add pic - Add a new pic object (maximum of 4)
3. Add “Add to Calendar” - Add a “ATC” object
4. Add “Ranking” - Add a ‘Ranking” object
5. Add “Feedback” - Add a “Feedback“ object
6. Move mode - Touch on an object and move it
7. Resize mode - Touch on an object and resize it
8. Change content - Touch an object, type the content and press “Change content”

    text object will change the display text

    pic object will change to the image in images.xcassets (see appendix)
9. Remove - Remove the last added text/pic/ATC/Ranking/Feedback
10. Save - Insert records into Table Layout1, Layout2 with “Noticon” ID to “CreateLayout1.php”, “CreateLayout2.php”
11. Cancel - Return to main menu without saving

6.3.8 Dynamic layout

We use currently display ID to “getLayout1.php” to check if there is a dynamic layout provided. If so, we enter this page. We use “getLayout1.php” and “getLayout2.php” to get data in Table Layout1, Layout2. We will then render the layout according to the coordination, with respect to the screen size. Fig 6.31 shows an example of created layout in <7. Create - layout> and Fig 6.32 and 6.33 shows the display in iPad and iPhone:

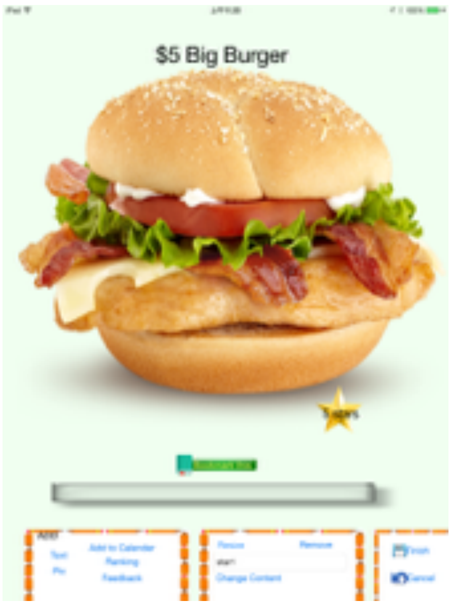


Fig 6.31 Page in create mode

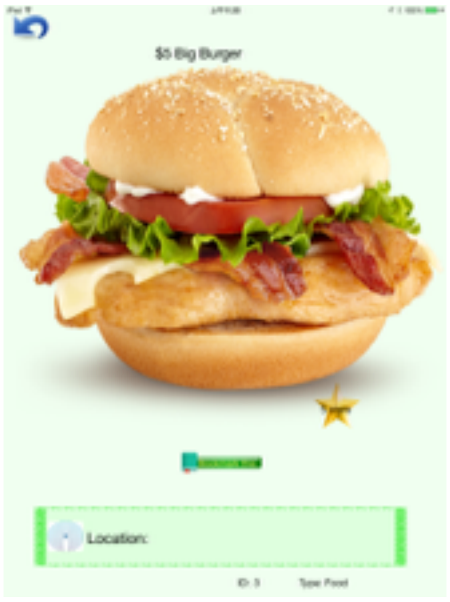


Fig 6.32 Page in iPad display mode

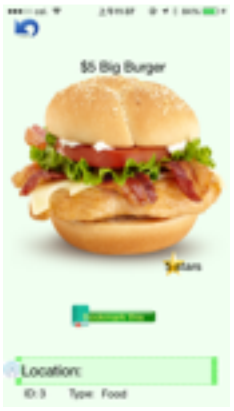


Fig 6.33 Page in iPhone display mode

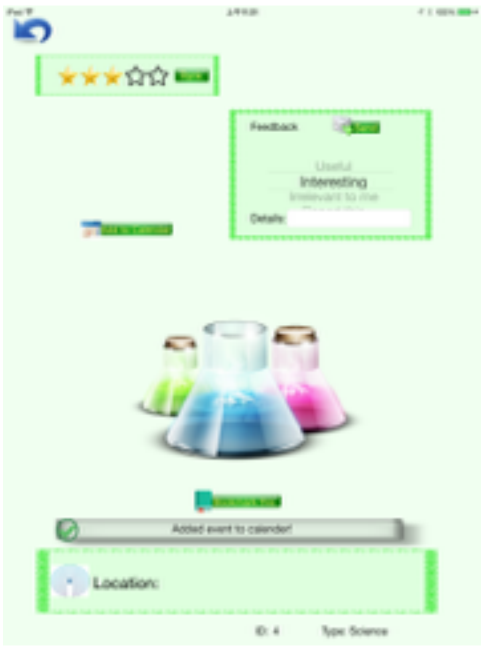
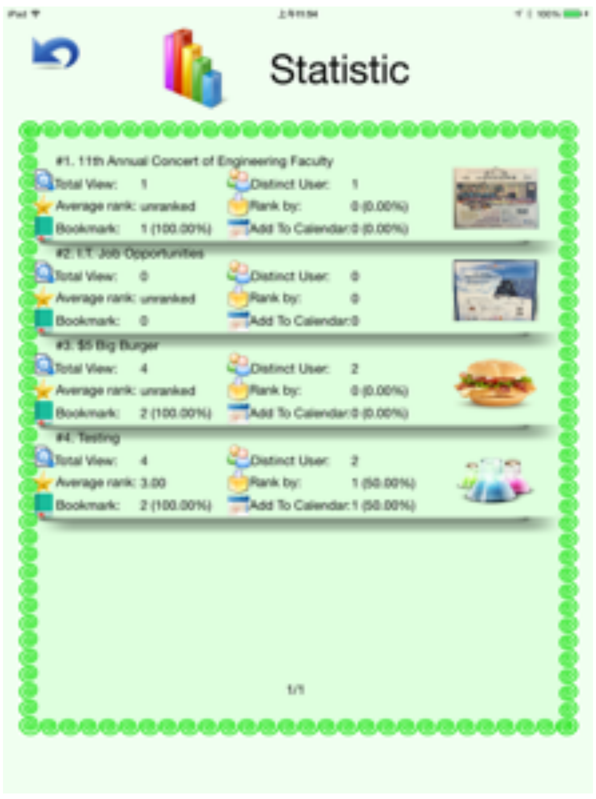


Fig 6.34 Create mode page

Moreover, functional objects (Ranking, add to calendar, feedback) will be created. It does the same function as in default view as shown in Fig 6.34.

6.3.9 Statistic

We provide a way for the administrator to view the statistic in app. The app get data in Table Statistic by "getStat.php" and display in the view (refer to Fig 6.35).


Statistic

#1. 11th Annual Concert of Engineering Faculty
Total View: 1 | Distinct User: 1
Average rank: unranked | Rank by: 0 (0.00%)
Bookmark: 1 (100.00%) | Add To Calendar: 0 (0.00%)

#2. I.T. Job Opportunities
Total View: 0 | Distinct User: 0
Average rank: unranked | Rank by: 0
Bookmark: 0 | Add To Calendar: 0

#3. $5 Big Burger
Total View: 4 | Distinct User: 2
Average rank: unranked | Rank by: 0 (0.00%)
Bookmark: 2 (100.00%) | Add To Calendar: 0 (0.00%)

#4. Testing
Total View: 4 | Distinct User: 2
Average rank: 3.00 | Rank by: 1 (50.00%)
Bookmark: 2 (100.00%) | Add To Calendar: 1 (50.00%)

1/1

Fig 6.35 Screen capture of Statistic page

For each "Noticon", it shows the id, title and main poster image.
The six statistics have the following meanings:

Total View: Total number of time of opening the "Noticon"

Distinct User: Number of distinct users (determined by device ID) have viewed the "Noticon"

Average rank: Average number of stars all users ranked (Range of 1-5stars)

Rank by: Number of users who have ranked the "Noticon", with percentage of number of distinct users

Bookmark: Number of users who have bookmark the "Noticon", with percentage of number of distinct users

Add to Calendar: Number of users who have add the "Noticon" to calendar, with percentage of number of distinct users

Chapter 7 Application of Noticon 2.0

Since Noticon is a powerful notifying platform using ibeacon technology, it can be proposed to apply in different innovative applications.

7.1 E-noticeboard

Noticon can be introduced to the E-noticeboard application. Some ibeacon transmitters can be installed to the physical notice board, when people walk near the board, they will be notified with the relevant poster. The lifetime of the poster and the available board areas can be defined. In other words, the staff can specify a poster to the particular notice board within a particular time. Apart from the basic information of the poster, the staff also can create some interactive actions to the poster and collect the corresponding feedbacks and statistics.

Using Noticon as an E-noticeboard application brings us a lot of advantages. First, the poster can be shown to more people. Since nowadays most of people will miss or skip the poster that posted on the board, Noticon helps to poster message delivery. Even someone miss the board, his device still being notified when he walk near the board. As a result, he can view the poster after he passed the board and obtain the poster information.

Second, the E-noticeboard allows posters change dynamically from time to time. Since we can set the lifetime of the posters, we can create different posters with different content and lifetime to enhance the promoting effect.

Third, the interactive components such as “add to calendar” feature enrich the user experience. Since we can add some interactive components to the poster, for example, user can just click the add to calendar button to add the event’s details to his own calendar. They also can give ranking to the particular noticon, this action will be recorded to the system for the statistics and further noticon selecting process use.

7.2 Improve the medical services quality

Due to the demand of medical getting higher, the workload of a doctor increase drastically. Sometimes a doctor may miss or forget which patient should follow up. Noticon can be used to improve this situation. We can install ibeacon to every bed, while the doctor walk near the bed , he will be notified and get the updated information of the patient, given that one noticon represent one patient. The nurse will take control of the Noticon management portal, they can update the content of the noticon, and schedule of the noticon. Using Noticon instead of the traditional flows help to protect the information of patient. In the old day, all the patient information are placed at the end of the bed. It can be easily accessed by everybody. Since only doctor and nurse installed the Noticon app, so it can restrict people to access those information. Moreover, reminder can be set to remind doctor when to visit the patient.

7.3 E-advertising platform

Noticon can be used to act as an advertising platform.When a person approach the shop or some locations, we can push them a message with the content of the advertisement. We also can combine with the scheduler and the statistics analysis to do the promotion such as issue a coupon to customers. In this case, the ranking component also helps the business to understand what their target customer thought.

Chapter 8 Contribution of Work

Summer

During the summer break, I have done some research on ibeacon in order to start the final year project. I searched some application of ibeacon in the market and try to think of the probability of ibeacon. Moreover, I am a beginner of IOS programming, so i buy a Macbook, install Xcode, and try to feel the IOS program. At the same time, due to Xcode simulator cannot test the ibeacon features, I spend time on configure the IOS developer account and try to perform device build.

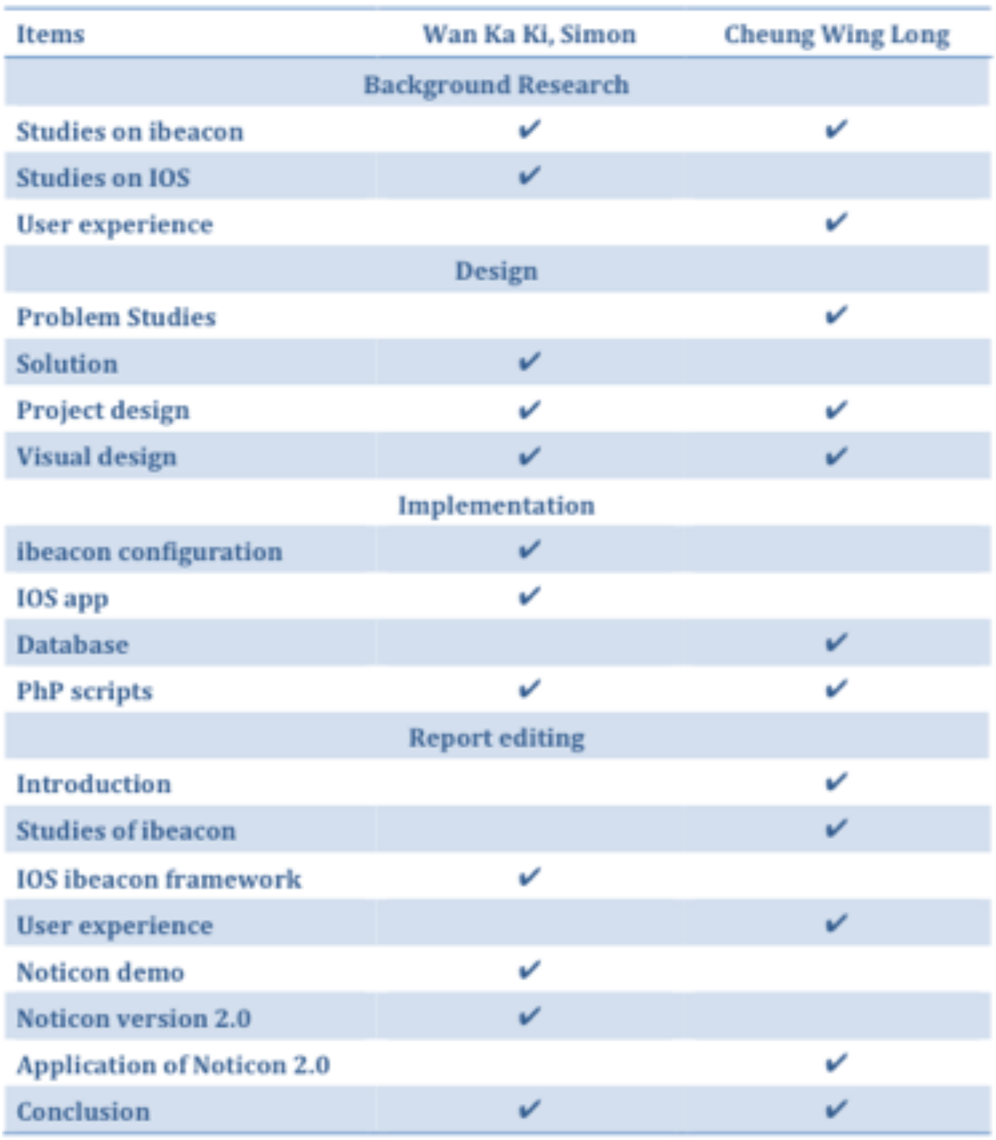
Fall 2014

Starting of the semester, every week we have a meeting with our supervisor and Mr. Edward Yau of ViewLab. Therefore, my partner and I stayed together to work for the project for at least one day per week. We discussed about new ideas, different scenario we may encounter. In this semester, we have already implemented an IOS app called Noticon. I am responsible to the IOS programming.

Spring 2015

At the beginning of spring 2015, we have evaluated the work done in last semester. For improve the app, I am trying to connect external database. During the implementation, i have spent time on the ways to store data and get data through the database, the algorithm to search for data. On the other hand, i have changed the User interface in a clear manner. We also though that using iPad as a administration management app and iPhone as a client app. The reason is that most of the people won't bring a tablet all the time, if we choose using iPad, the ibeacon notifying effect will be diminished. Finally, a Noticon version 2.0 app has been implemented in this semester.

The below Fig 7.1 shows the work divisions details.



Items	Wan Ka Ki, Simon	Cheung Wing Long
	Background Research	
Studies on ibeacon	✓	✓
Studies on IOS	✓	
User experience		✓
	Design	
Problem Studies		✓
Solution	✓	
Project design	✓	✓
Visual design	✓	✓
	Implementation	
ibeacon configuration	✓	
IOS app	✓	
Database		✓
PhP scripts	✓	✓
	Report editing	
Introduction		✓
Studies of ibeacon		✓
IOS ibeacon framework	✓	
User experience		✓
Noticon demo	✓	
Noticon version 2.0	✓	
Application of Noticon 2.0		✓
Conclusion	✓	✓

Fig 7.1 Work divisions details

Chapter 9 Future Works

9.1 Create a library

Since it is hard to motivate people to install our app, so building a library for developers to build their own apps is a suitable way to use ibeacon technology.

9.2 Create a CMS

For every noticon creation, we can create a CMS system to the administrator to manage their contents via a website instead of iPad.

9.3 Deployment of ibeacon

We can make a management system for the administrator to manage the beacon information, not only the UUID, but also the location and the signal strength etc. Also we can think of a effective way for how to set the UUID of a beacon.

9.4 Enrich the interactive components

Currently we provided few interactive components for creating an noticon, we can provide more creative and useful interactive plugin to enrich the user experience.

9.5 Support different platform

Currently Noticon only support IOS, for future work we can also support different such as android or Windows phone.

Chapter 10 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 11 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 12 : 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.

Appendix

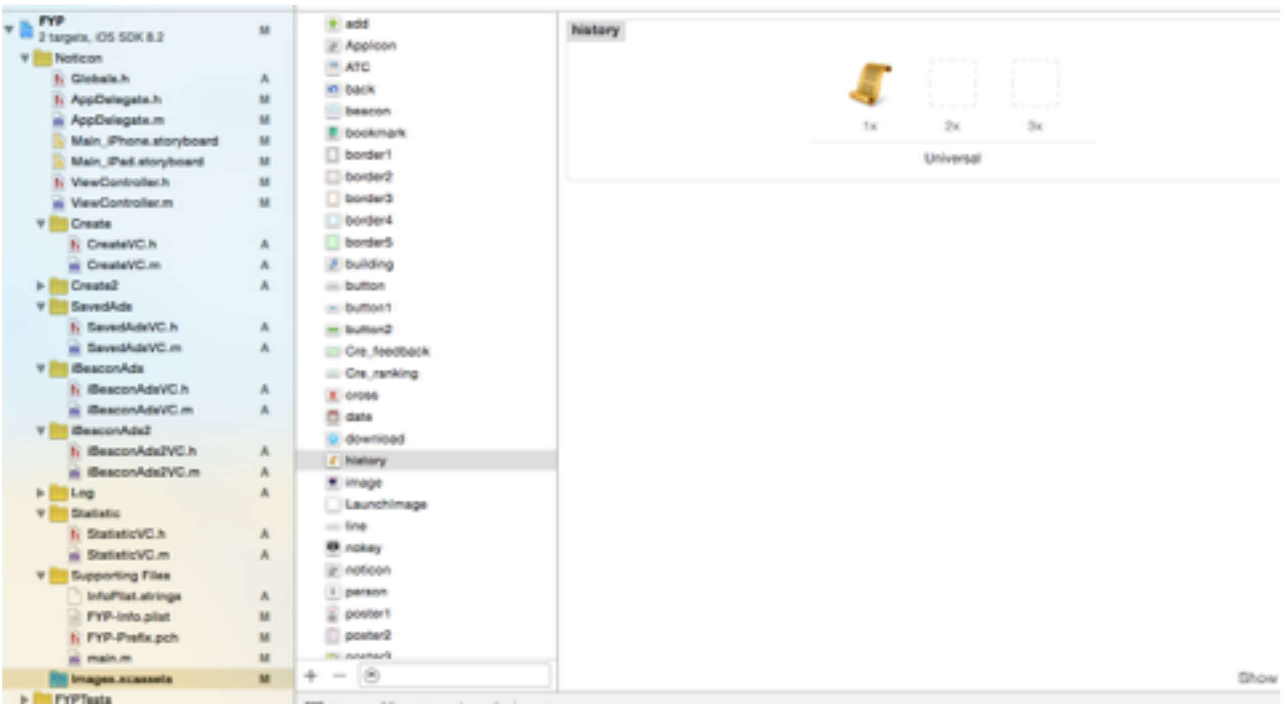
We can change the image in the layout by changing the content to a following “name” (refer to Table 12.1):

Icon	Name
➕	add
📅	ATC
↶	back
📗	bookmart
🏢	building
▭	button
❌	cross
📅	date
⬇	download
📜	history
🖼	image
—	line
⌨	nokey
💾	save
✉	send
☆	star0
★	star1
📊	stat
✓	tick
🕒	time
🕒	time2
👥	user
🗺	venue
🔍	view
🗳	vote
📝	write

Icon	Name
	XXXX
	noticon
	database
	php
	json

Table 12.1 png icons and the corresponding name

We can add more custom png images by adding the png image to images.xcassets (refer to Fig 12.2):



```
▼ FYP
  2 targets, iOS SDK 8.2                M
  ▼ Noticon
      Globals.h                         A
      AppDelegate.h                     M
      AppDelegate.m                     M
      Main_iPhone.storyboard            M
      Main_iPad.storyboard              M
      ViewController.h                  M
      ViewController.m                  M
    ▼ Create
        CreateVC.h                      A
        CreateVC.m                      A
    ▶ Create2                           A
    ▼ SavedAds
        SavedAdsVC.h                    A
        SavedAdsVC.m                    A
    ▼ iBeaconAds
        iBeaconAdsVC.h                  A
        iBeaconAdsVC.m                  A
    ▼ iBeaconAds2
        iBeaconAds2VC.h                 A
        iBeaconAds2VC.m                 A
    ▶ Log                               A
    ▼ Statistic
        StatisticVC.h                   A
        StatisticVC.m                   A
    ▼ Supporting Files
        InfoPlist.strings               A
        FYP-Info.plist                  M
        FYP-Prefix.pch                  M
        main.m                          M
      Images.xcassets                   M
  ▶ FYPTests
```

```
add
AppIcon
ATC
back
beacon
bookmark
border1
border2
border3
border4
border5
building
button
button1
button2
Cre_feedback
Cre_ranking
cross
date
download
history
image
LaunchImage
line
nokey
noticon
person
poster1
poster2
```

Fig 12.2 Directories of images.xcassets