Eksibition - Virtual Exhibition Guide on Smart Phone

LYU 1503

Jin Peng (1155014559)
Han, Zuohao (1155014316)

Supervised by Michael R. Lyu

Contents

Abstract

Eksibition is a mobile smart guide who can provide a great tour experience for visitors in museums and exhibitions. Our goal of the project is to implement the full features of this application on iOS devices.

In this report, we will go through all the topics that we have been studied, all the achievements we have made and all the problems we have encountered within this semester.

The first part is the introduction of this project, indicating the motivation of this project and how we are going to implement this idea. The second part is the original wireframe design of this application, briefly showing the features of it. Then we will describe our studying process of the Bluetooth Low Energy (BLE) technology. Since we are going to make use of iBeacon™, a sub-category of BLE device, to provide the guide service, a study on the hardware is very necessary. The rest part are the showcases of current features consisting of all the technical problems we overcame. After a semester-long development, there are still some limitations and features remained to be tackled in the coming semester.

1. Introduction

1.1. Overview

Apple introduced a protocol called iBeacon™ at the Apple Worldwide Developers Conference in 2013 [1]. iBeacon™ is a protocol allowing the Bluetooth Low Energy (BLE) devices to broadcast their identifier information to nearby portable electronic devices including BLE enabled smart phones [2]. Although it seems quite simple, applications can still do a lot of things by taking advantage of this technology. Our application Eksibition is one of them. Imaging a mobile application can know where you are in a museum and introduce the item or artwork in front of you through both text and audio, visiting a museum could be a more interesting activity even without a real guide explaining what is happening. Eksibition is not only a tour guide, but also an integrated smart extension for the museums, which can help visitors buy tickets, schedule visit time and redeem tickets, etc.

1.2. Motivation

Nowadays, people don’t appreciate exhibitions and museums in a way it used to be. Sometimes when time is limited, we just go to the museum to see one or two items we are interested in, but we don’t know the exact location of them. Sometimes when we see our favorite item, it may be inconvenient for us to learn the interesting background of it unless we Google it or take an extra device from the museum to listen to it. Or for some people, they usually have a real tour guide to guide them in the museum and introduce to them, with a cost ranging from 20 USD/hour up. On account of these facts, we feel it is really necessary to have an application that can make everything easier and more interesting when visiting a museum.

When we are trying to figure out how to make an application for users as if a guide will be walking next to them, we found the iBeacon™ and the Bluetooth Low Energy (BLE) technology very useful, which perfectly help to enhance the features of our application.

Such that, we had this project started, named Eksibition - Virtual Exhibition Guide on Smart Phone.

1.3. Objectives

As stated in the last section, we are developing an application to fulfill the demands from both visitors and museum organizations and also to improve the whole visiting experience. The features below are to be implemented:

- Ticketing system: Users can purchase tickets within our application and retrieve the tickets as Ticket Pass, which can be stored in the newly introduced Apple Wallet (it's called Passbook before iOS 9). The electronic tickets on the user's side can later be redeemed at the entrance of the museum. We also developed a redeem application, which is a separated companion application distributed to museum staffs to redeem users' tickets at the entrance. These whole process is paperless and environment-friendly.
- Showing nearby items: Users can see the nearest item in our application when getting close to it. They can also choose any one of the already visited items to view the detail information, which includes more images and descriptions. Users can even listen to the introduction soundtrack of items.
- Like and share: Users can like an item and share the related information to social networks.
- Map view of the museum: Users can see his location and all the items on a map of the museum, so that one can decide where to explore his favorite items.
- Smart route guide (to be implemented in next semester): Application can help generate a routine based on the items selected by users and how much time they have.
- News notification system (to be implemented in next semester): Users can get notifications from the museum promoting the coming events when they are not physically in the museum.
- Analyzing of users' behavior (to be implemented in next semester): By analyzing the users' behavior from the recorded cumulated time they spent on each item, we can draw the conclusion of which items are users' favorites and what is the most popular routine, which can help the museums improve their service.

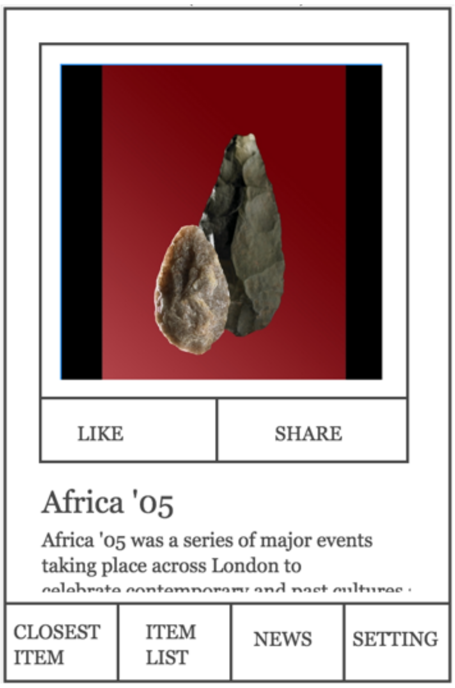
1.4. Milestones

To achieve those objectives above, there are some technical milestones required to be reached:

- A well-designed backend server with all the applications programming interfaces (API) implemented to handle the possible requests from the mobile client side.
- A Bluetooth signal analyzing mechanism within the application to detect the distances from iPhone to items.
- A map system to show the customized indoor map of the museum, which also carries information of the items in exhibition.
- A positioning system analyzing the users' estimated position according to the different signals broadcasted by iBeacons.
- A ticketing system allowing the users to purchase and redeem tickets within our app.
- Apple Pay integration, which helps to make the purchasing process easier.
- Supporting tickets to be added and managed by Apple Wallet.
- An analyzing mechanism to collect the related information and analyze the users' behavior.

2. Wireframe Design

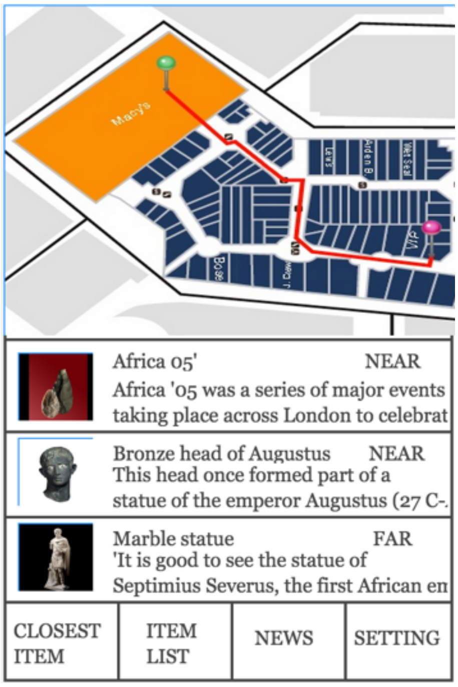
This part shows the initial wireframe design of our application in the beginning. Some basic features are shown.



Africa '05

Africa '05 was a series of major events taking place across London to celebrate contemporary and past cultures ·

Figure 1: Closest tab shows the image and detail introductions of the nearest item, allowing users to like or share the related information.

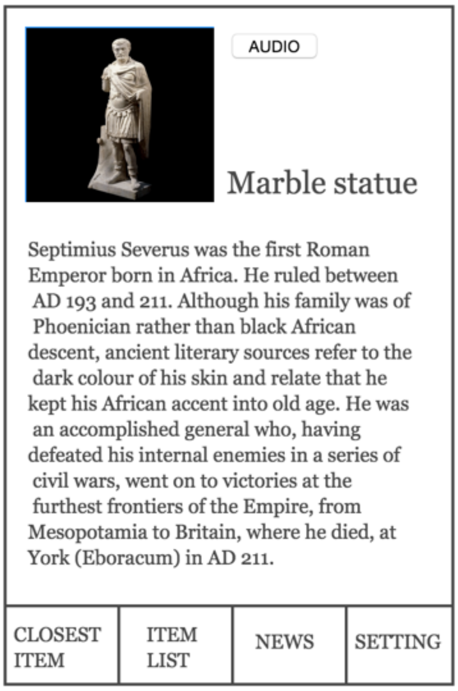


Africa 05' NEAR
Africa '05 was a series of major events taking place across London to celebrat

Bronze head of Augustus NEAR
This head once formed part of a statue of the emperor Augustus (27 C-.

Marble statue FAR
'It is good to see the statue of Septimius Severus, the first African em

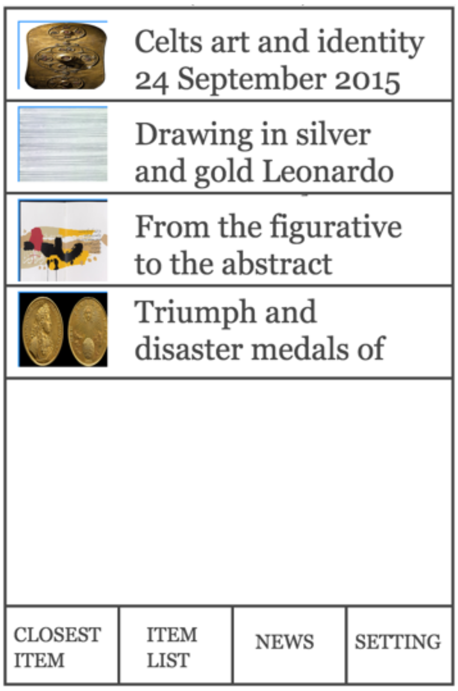
Figure 2: Item list tab has a zoomable map view on the top area showing the museum map and item locations. Under the map is an item list showing all the items in the museum.



Marble statue

Septimius Severus was the first Roman Emperor born in Africa. He ruled between AD 193 and 211. Although his family was of Phoenician rather than black African descent, ancient literary sources refer to the dark colour of his skin and relate that he kept his African accent into old age. He was an accomplished general who, having defeated his internal enemies in a series of civil wars, went on to victories at the furthest frontiers of the Empire, from Mesopotamia to Britain, where he died, at York (Eboracum) in AD 211.

Figure 3: Item detail page let users check the detail information of this item



Celts art and identity 24 September 2015

Drawing in silver and gold Leonardo

From the figurative to the abstract

Triumph and disaster medals of

Figure 4: News tab shows the the promotions of museum events

3. Study of BLE Technology

3.1. Hardware

Bluetooth Low Energy (BLE) is one kind of Bluetooth technology developed by the Bluetooth Special Interest Group. BLE devices broadcast signals in a specific area.[3] Different from the traditional Bluetooth technology, BLE can provide consistent signal with very low energy consumption.

Devices adopting iBeacon™ protocol are typically called beacons. In our development, we use Estimote beacons as out BLE devices. Smartphones, tablets and other smart devices will get notified when they are in close proximity to a beacon.[2]

3.1.1. Advertisement Frame

Beacons works in an advertisement mode, in which it broadcasts its signal and notify its presence to those smart devices who can receive the signal.[4] The signal a beacon broadcasts follows a fixed format defined by Apple's iBeacon™ protocol prefixed together with UUID, major and minor identifiers[5]. Here is an example of what an iBeacon™ advertisement frame will be like:

```
fb0b57a2-8228-44 cd-913a-94a122ba1206 Major 1 Minor 2
```

With this frame, we can make the signal broadcasted by each beacon carries different information. For example, we can set the UUID to represent the building where the beacon locates and set the Major and Minor numbers to represent different locations in this building.

3.1.2. Ranging

Using the strangeness of the signal broadcast by beacons, iOS SDK will give an estimated distance between beacons and the phone. The result is categorized into the following 3 distinct ranges [6]:

Immediate: within 0.5 meters.

Near: within 3 meters.
Far: within 70 meters.

3.2. Estimote Beacons

In this project, we use Estimote beacons as the BLE devices. Estimote beacons and stickers are small beacons that can be attached to any location or object. By making use of the Estimote SDK, our application can know the proximity of nearby beacons and recognize their type, ownership, approximate location, temperature and motion [7].

estimote beacons
real-world context for your apps
Developer Kit
Bluetooth Low Energy Beacons

Figure 5: Estimote beacons packaging

Besides the Estimote SDK, Estimote also provides another SDK especially for indoor positioning - Estimote Indoor Location SDK, which is a set of tools for building precise, blue-dot location services indoors [8]. We can upload the location's map to Estimote cloud after we set up the target location using at least 4 beacons.

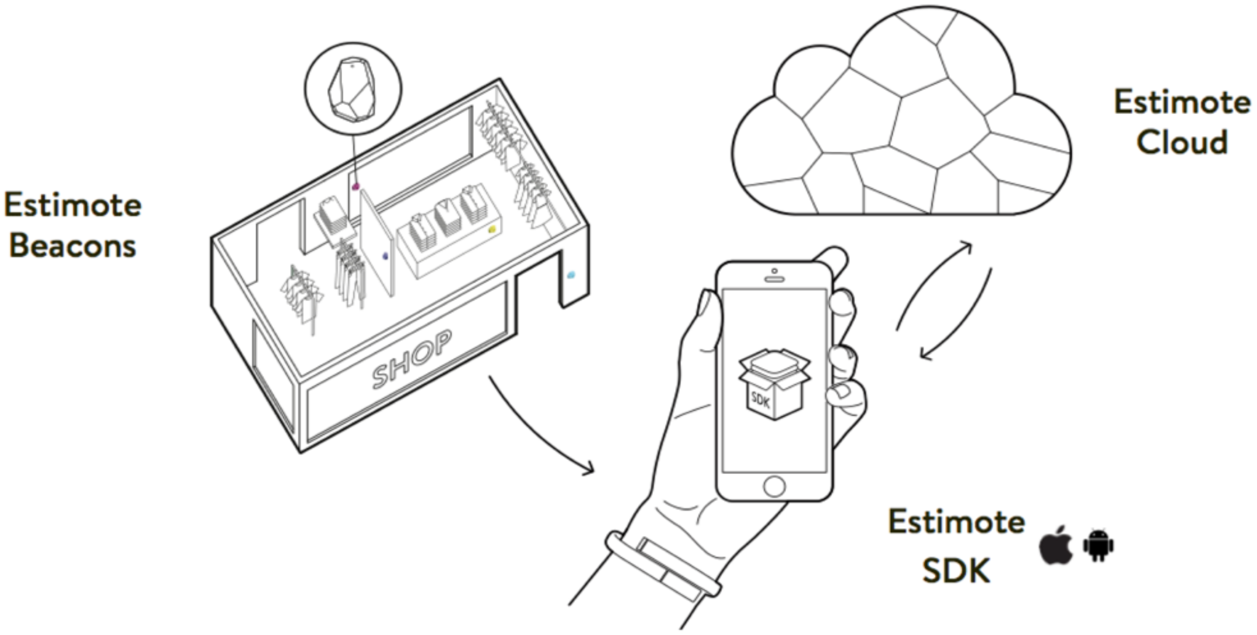


Figure 5: Estimote SDK

4. Study of Indoor Positioning

4.1. Approach by Analyzing RSSI

We started to do research on indoor positioning at the very beginning of our project. The first approach we did to calculate the indoor location is to analyze the received signal strength indicator (RSSI) from each beacon.

4.1.1. Trilateration

Trilateration is a positioning technique using different distances from different points to determine the position of a certain point. This system has to measure the distance from the target point to at least 3 reference points whose positions are already known. Using these distances, we can form 3 circles surrounding each point. Then the only intersection of these three circles is position of the target point.

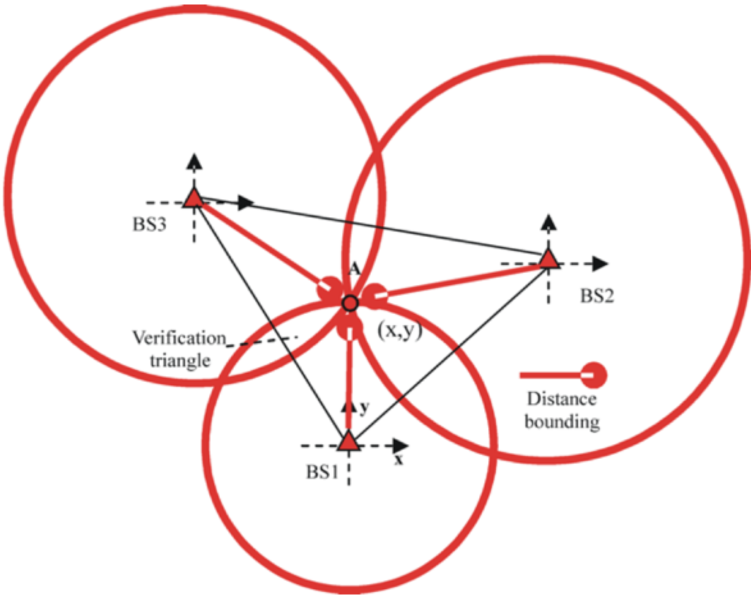


Figure 6: Trilateration [15]

Theoretically, we can get the position of the target point by collecting at least 3 relative distances of the reference points. As can be seen from the picture, every two circles form two intersection points, one of which is our target point. But in real situation, we know that the distance calculated from the value of RSSI may not be accurate. Thus the 3 circles may not have a common intersection. To ensure we can get a single solution, we made an optimization on this approach:

Step 1: calculate two possible position p1 and p2 by the distances from two of the reference points.

Step 2: calculate the distance d3 between the target point and a third reference point.

Step 3: get the distance d1’ between p1 and the third point, d2’ between p2 and the third point.

Step 4: if d1’ is closer to d3 then p1 is the result position. If not then p2 is the result position.

4.1.2.Accuracy

As we can see, it is not difficult to get the target position from 3 distances of reference points, but it is still a problem to calculate the distance from the value of RSSI accurately. According to the researches we did, we found that the value of RSSI follows a lognormal distribution over distance [12]:

$$PL = P_{Tx_{dBm}} - P_{Rx_{dBm}} = PL_0 + 10\gamma \log_{10} \frac{d}{d_0} + X_g,$$

So the same RSSI may result in different estimated distance in indoor environment, the relation between the value of RSSI and the distance is different. So that if we really want to get an accurate value of the distance, we need to do multiple testing under the target environment which is unrealistic and time-consuming.

Finally, we gave up this approach and changed to another one.

4.2. Approach by Using Estimote Indoor Location SDK

As we mentioned above, Estimote provides its own SDK for developers to implement indoor positioning by creating their own indoor map in Estimote cloud. Estimote Indoor Location SDK requires the following things [9]:

- 1 OS X computer with Xcode 7.
- 1 iOS device preferably newer than 5.
- 1 Estimote Account.
- At least 4 Estimote beacons related to the account.

The map creation steps are as follows [9]:

- Use the Estimote Indoor Location app to map the room into Estimote cloud and tune the location setups to make the positioning more accurate.
- Install the Estimote Indoor Location SDK through CocoaPods.
- Add Indoor Location Manager.
- Connect the app to Estimote Cloud and then fetch and update the location.

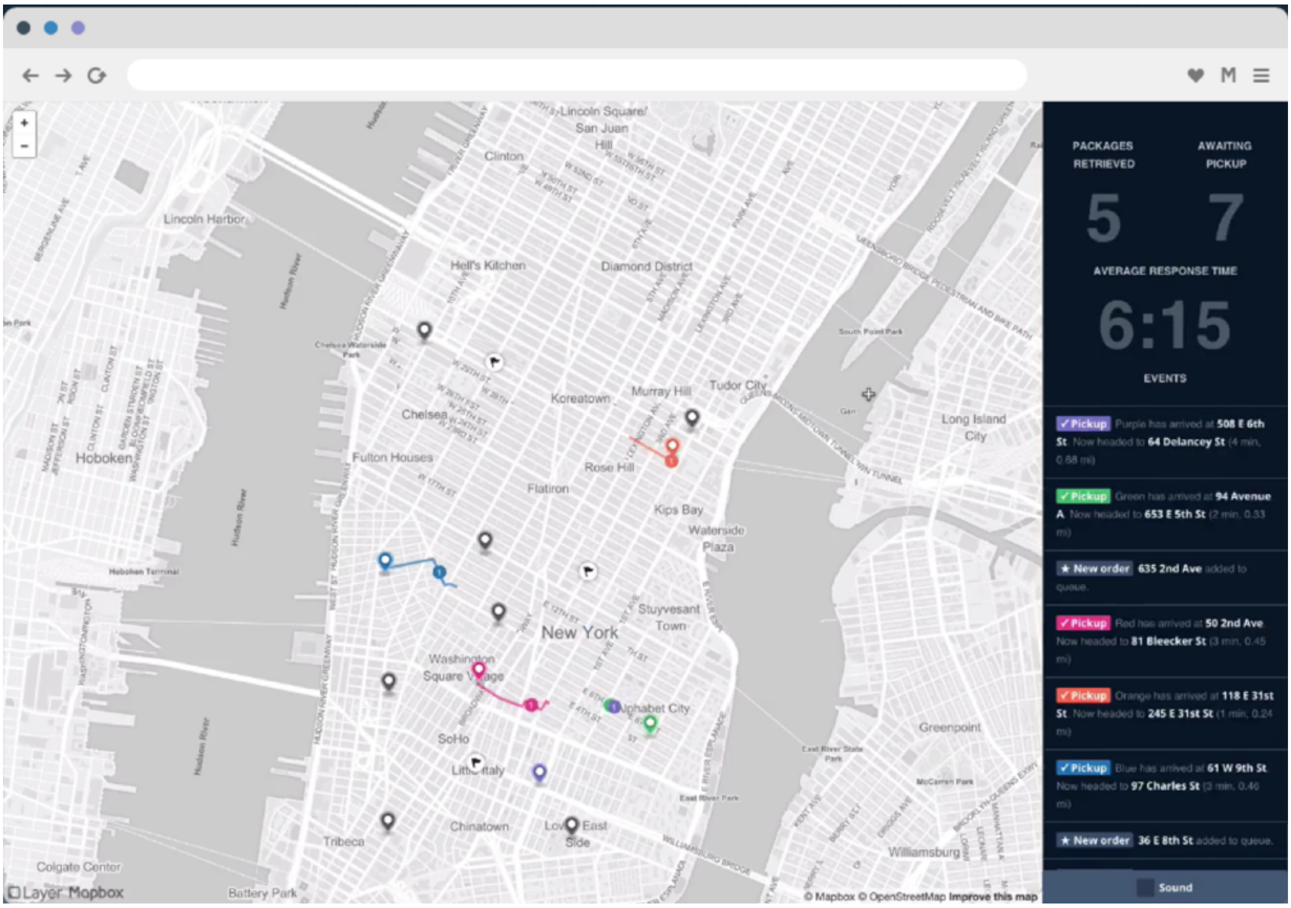
Figure 7: The developer is mapping his room to Estimote Cloud using the Estimote Indoor Location app. [7]

This approach requires at least 4 Estimote beacons while we just have 3 beacons until the end of this semester, so that this approach will be left to next semester's development.

5. Study of Custom Indoor Map Design

5.1. Approach by Using Mapbox

The first approach we did to implement our customized indoor map is to use an third party platform called Mapbox. Mapbox is a map platform for developers to design their own map customized with their own style and data.



PACKAGES RETRIEVED 5

AWAITING PICKUP 7

AVERAGE RESPONSE TIME 6:15

EVENTS

✓ Pickup Purple has arrived at **508 E 6th St**. Now headed to **64 Delancey St** (4 min, 0.68 mi)

✓ Pickup Green has arrived at **94 Avenue A**. Now headed to **653 E 5th St** (2 min, 0.33 mi)

★ New order **635 2nd Ave** added to queue.

✓ Pickup Red has arrived at **50 2nd Ave**. Now headed to **81 Bleecker St** (3 min, 0.45 mi)

✓ Pickup Orange has arrived at **118 E 31st St**. Now headed to **245 E 31st St** (1 min, 0.24 mi)

✓ Pickup Blue has arrived at **61 W 9th St**. Now headed to **97 Charles St** (3 min, 0.46 mi)

★ New order **36 E 8th St** added to queue.

□ Sound

Figure 8: Mapbox online studio interface

Basically, developers can do two things on this platform. The first one is to customize the style of the map. Developers can change the color and size of some geographical utilities, which makes the map looks different from the original one. The second one is to add their customized data like markers, lines and polygons over the map.

Customized Style & Data

Since no matter how we change the style of a map, it is still a world map with the coordinate system based on longitude and latitude. So we have two alternatives here. One is to set the style to blank and transparent the world coordinate system to our own indoor coordinate system and then draw some data as the map of the indoor map. In another word, it is using the world as a room. Another solution is to keep the style on the map and draw the data on the exactly position of the room indicating the in door structure of it.

During implementing this two solutions, we found the shortcomings of each solution.

- **Cons of the first solution**: According to the Mapbox SDK documentation, Mapbox draws the data onto the map by rendering the provided geojson data so that when we are trying to draw an indoor map as large as the earth, it will be very CPU consuming and fail sometimes.
- **Cons of the second solution**: There are two failures of this solution. One is that it is nearly impossible to location the room accurately. When we try to render our customized data to show the indoor map, we have to provide the border coordinates of the room in latitude and longitude, in which all the points owns nearly the same value of latitude and longitude because it will be only different in 0.00001. Another problem is that even though we manage to construct the room above the world's map in the right position, the map cannot be zoomed to the level where the detail of the room is shown well.

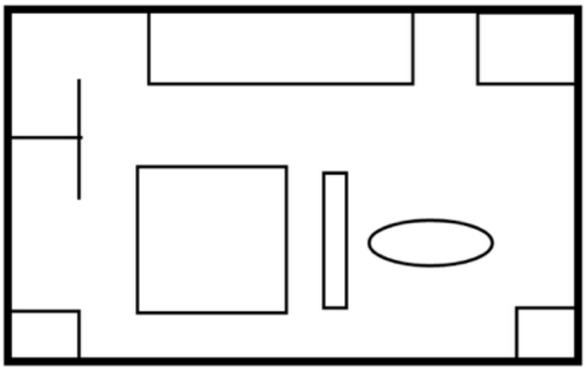
Finally we gave up this approach and chose a simple way to design our own indoor map.

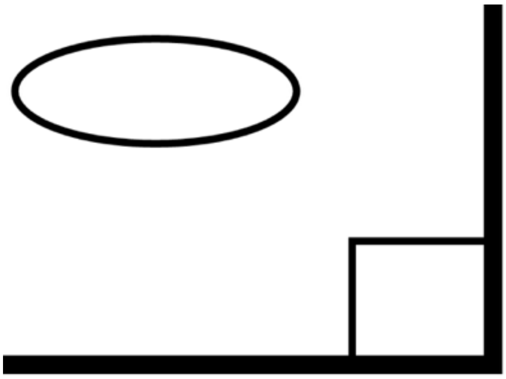
5.2. Approach by Using a High-resolution Image

The reason why we chose Mapbox as the first approach is based on the following points:

- Map box can provide a map view through its SDK implemented with the functions like panning, zooming and rotating.
- The data we drew on the map will be shown in a vector way which means it won't change its resolution no matter how large the user zoom it.
- The map is stored in a JSON file which can be edited conveniently by just adding or deleting a few keys and values.

Based on the reasons mentioned above, we were looking for another solution which has the same advantages. At last, we chose to use a high-resolution image as the map since it can be zoomed smaller without losing its sharpness.





From the comparison between the two images above, we can seen that this image remain clear after being zoomed in.

6. Study of Developing Backend Using Node.Js

Node.js is an open-source framework for developing network applications such as server-side web application. Node.js is distinguished for its cross-platform environment and event-driven architecture. Additional, in the framework of Node.js, developers do not have to take care of the thread-relative problem since Node.js is not a thread-based framework.

6.1. Building RESTful APIs With Express4

Express is a module in Node.js helping developers handle routing when developing APIs. Making use of the event-driven architecture, we can implement an API in following steps:

- Create a .js file with the same name as the target router.
- Handle all the possible RESTful request to this router in this .js file by implement the event handlers.
- Export this .js file as a module and add pass it a handler to the Express module.

Here is a simple code snippet implementing an API on router ‘/’.

```
router.get(‘/‘, function(req, res) {
     res.json({ message: ‘hooray! welcome to our api!’});
});
```

Up to the end of this semester, we have implemented 11 APIs over 3 routers. Here is the documentation for those APIs.

Router	API	Method	Arguments	Usage
/ticket	/redeem	PSOT	name: *ticketId* type: String stored in: body	To let the server check is the ticket with ticket number *ticketId* is valid.
	/getTicketId	GET	no	To generate a ticket number for testing.
	/generatePass	POST	name: *ticketId* type: String stored in: body	To let the server create, sign and return a pass according to the ticket number *ticketId*.

Router	API	Method	Arguments	Usage
	/orderHistory	GET	name: *deviceId* type: String stored in: body	To get all the tickets that has been purchased by the device with *deviceId*.
	/setUp	GET	no	To set up database of types for testing.
/ticket	/pay	POST	name: *stripe_token* type: String stored in: body name: *amount* type: String stored in: body name: *description* type: String stored in: body name: *contact* type: Dictionary stored in: body name: *deviceId* type: String stored in: body name: *quantity* type: Integer stored in: body	To buy amount number of tickets from the server providing enough information. The server will return the client a string as ticket number if succeed.

Router	API	Method	Arguments	Usage
/item	/:uuid/:major/:minor	GET	name: *deviceId* type: String stored in: headers	To get an item object containing all the related information. The argument *deviceId* is for server to know if this device has liked one item.
	/all	GET	name: *deviceId* type: String stored in: headers	To get all the items of the museum in an array.
	/like	PUT	name: *deviceId* type: String stored in: body name: *_id* type: String stored in: body	To indicate the server that an item with identifier *_id* is liked by a device with identifier *deviceId*.
	/share	PUT	name: *_id* type: String stored in: body	To indicate the server that an item with identifier *_id* is shared by a device with identifier *deviceId*.
	/view	PUT	name: *_id* type: String stored in: body	To indicate the server that an item with identifier *_id* is viewed by a device with identifier *deviceId*.

6.2. Using MongoDB to Save Data

Different from the traditional SQL database, MongoDB is a NoSql database which is implemented in a document-oriented way. There are several reasons why we chose MongoDB to store our data instead of the traditional SQL databases:

- MongoDB is perfectly integrated with Node.js. We can add MongoDB into our modules by adding just one line of code in file package.json.
- MongoDB stores data in document with JSON style which is very readable and editable to developers. JSON style data is also easy to use for client side.

- Developers do not have to set the property like primary key or default value when creating a schema for a table. The only thing developers have to do is to set the index name and the value type of each column.
- Updating data becomes easier. We can update a row of data by calling the method ‘save()’.
- We do not have to care about if one of the column value is not given when adding date into each row of the table.

We use MongoDB to store our item data and order data, and here is the schema of these two table.

```
var ItemSchema = new Schema({
   beaconUUID: String,
   beaconMajor: String,
   beaconMinor: String,
   title: String,
   coverImage: String,
   author: String,
   country: String,
   description: String,
   soundtrack: String,
   images: Array,
   coordinateX: Number,
   coordinateY: Number,
   introduceTime: Date,
   inExhibition: Boolean,
   views: [String],
   likes: [String],
   shareCount: Number
});
```

```
var OrderSchema = new Schema({
     email: String,
     amount: Number,
     purchaser: String,
     generateDate: Number,
     redeemedDate: Number,
     generateDateObject: Date,
     redeemedDateObject: Date,
     ticketId: String,
     valid: Boolean,
     deviceId: String
});
```

7. Developing Application With Swift

7.1. Model-View-Controller

We designed our application in a Model-View-Controller(MVC) pattern which is to separate objects into three roles: Model, View and Controller. Model objects is the representation of data. Views are what users can see and interact with. Controller objects will be in charge of updating model and view according to user's actions. MVC is a good pattern to design our application for the following reasons.

- It makes most of our objects reusable. The interfaces between objects are well defined.
- Our application will become more extendable since it would be relatively easy to add new features by add a set of Model-View-Controller.
- We can use many Cocoa technologies that are based on MVC pattern.

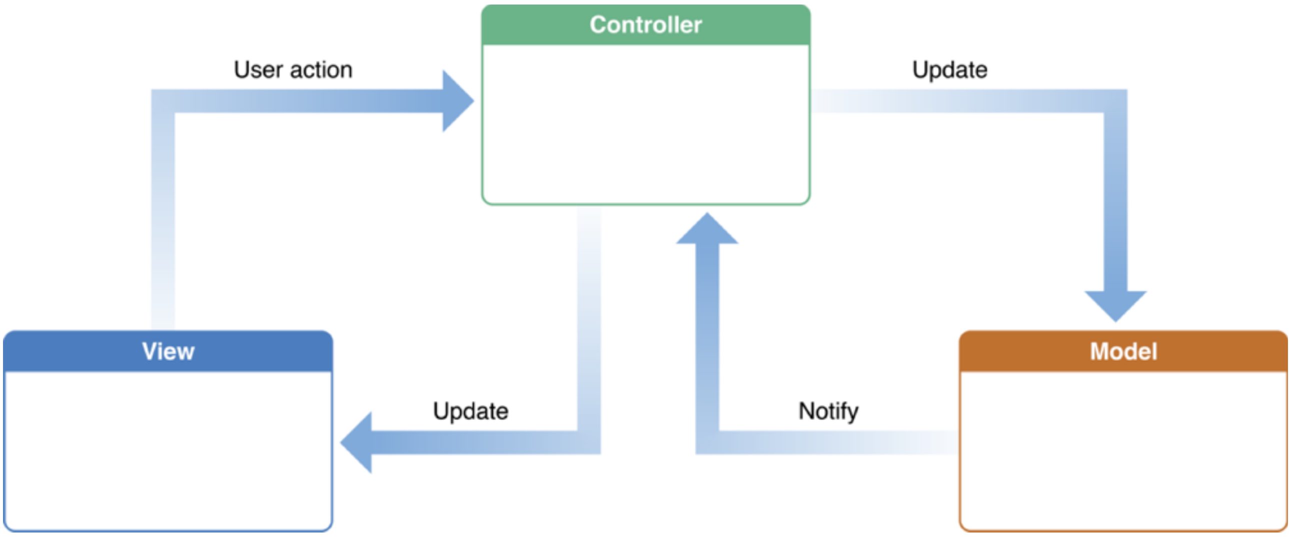


Figure 9: MVC illustration [10]

7.2. 3D Touch

3D touch is a new technology Apple introduced this year together with iPhone 6s / 6s plus and iOS 9. This technology allows iPhone 6s / 6s plus to know how hard a user is pressing the screen. Integrating with 3D Touch technology can make users have more actions over our application.

7.2.1.Home Screen Quick Actions

The traditional way to launch an application is by tap it on the home screen. With the help of 3D Touch, users can have quick actions on our application by pressing our icon in the home screen and select a quick way to launch our application, which can anticipate and accelerate a user’s interaction with our application.

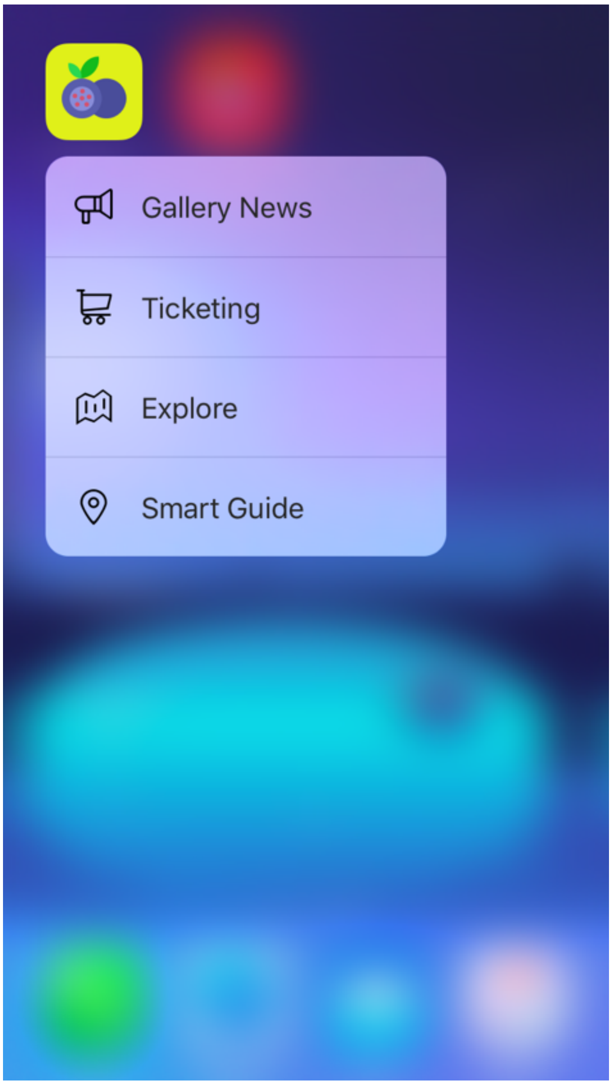
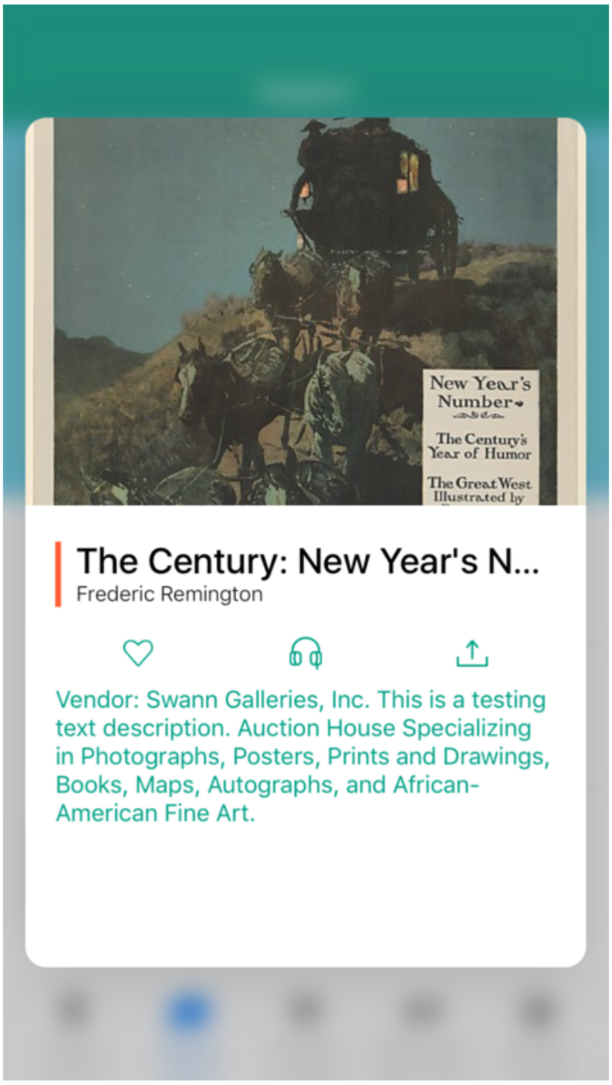


Figure 10: Quick Actions

7.2.2.Peek and Pop

“Peek and Pop” is a new way for users to interact with iOS device supporting 3D Touch. It allows users to take a peek or have a preview of the view, which is going to be opened.

We can make our application work in a more efficient way by implement this feature. From our perspective, we think a user should not spend too much time on operating his iPhone when he or she is visiting a museum. When a user is interested in one of the items in the item list, he or she can decide whether to look into detail of it by having a peek. This can make a user save more time and get exactly what he or she needs from our application.



The Century: New Year's N...

Frederic Remington

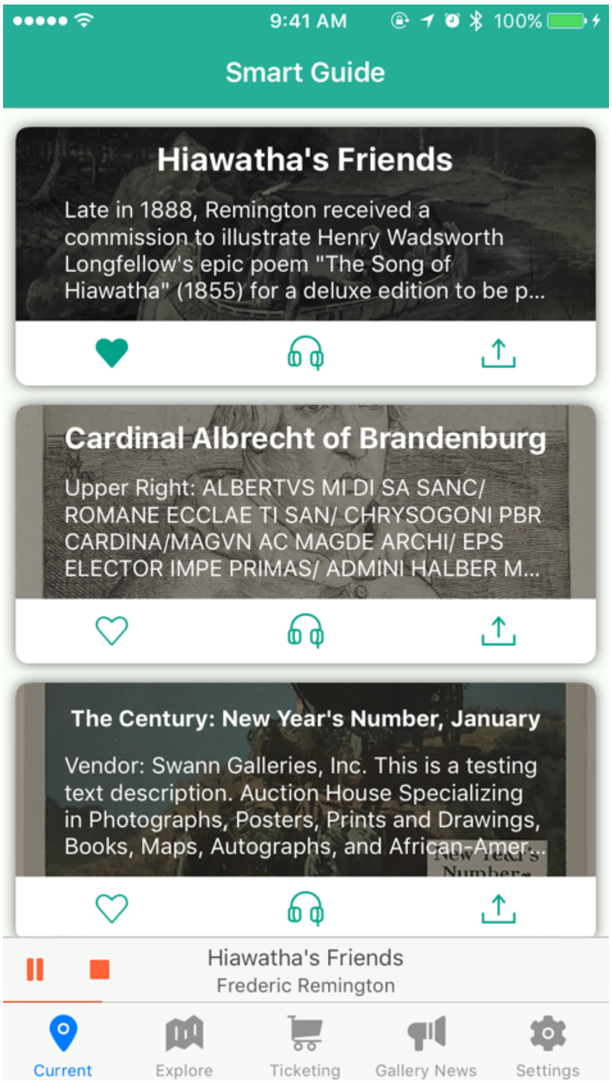
Vendor: Swann Galleries, Inc. This is a testing text description. Auction House Specializing in Photographs, Posters, Prints and Drawings, Books, Maps, Autographs, and African-American Fine Art.

Figure 11: Peek and Pop

7.3. Centralized Soundtrack Player

Our application has a function to play the item introduction soundtrack. Since we are not allowed to play multiple soundtracks at the same time, we need only one instance of the player object to do the playing job.

We implemented a singleton player inside SoundtrackPlayer class which can only be instanced once. Although the player can be seen in every tab of our application, it is only controlled by the methods in class AppDelegate, which is the heart of the application.



Hiawatha's Friends

Late in 1888, Remington received a commission to illustrate Henry Wadsworth Longfellow's epic poem "The Song of Hiawatha" (1855) for a deluxe edition to be p...

Cardinal Albrecht of Brandenburg

Upper Right: ALBERTVS MI DI SA SANC/ ROMANE ECCLAE TI SAN/ CHRYSOGONI PBR CARDINA/MAGVN AC MAGDE ARCHI/ EPS ELECTOR IMPE PRIMAS/ ADMINI HALBER M...

The Century: New Year's Number, January

Vendor: Swann Galleries, Inc. This is a testing text description. Auction House Specializing in Photographs, Posters, Prints and Drawings, Books, Maps, Autographs, and African-Amer...

Hiawatha's Friends
Frederic Remington

Figure 12: Centralized player

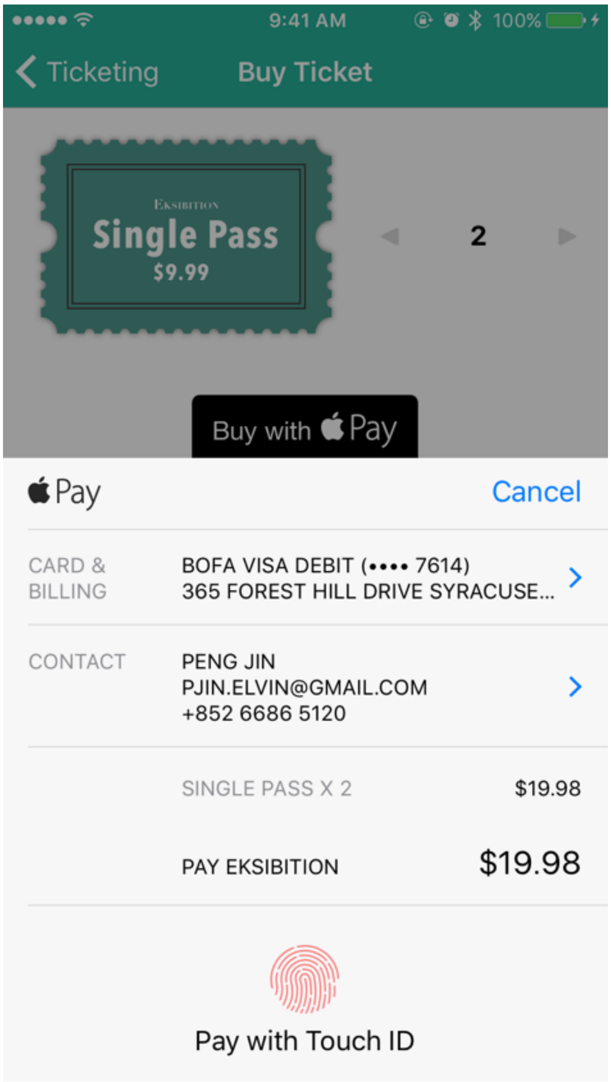
8. Summary at Midterm

After we built up our server and implemented required APIs, our application can work fluently with the following functions:

- Listing all the nearby items.
- Preview an item by pressing harder on the item in the item list.
- Enter the item detail page by pressing even harder on the item or simply tap on the item.
- Every time an user entering the detail page of an item, the server will update the count of viewers of the item.
- Users can like or unlike an item and the server can be notified and update the count of likes of the item.
- The second tab of the application can only show view of the map, since we didn't get enough beacons to implement the indoor positioning.

9. Study of Integrating Apple Pay

Apple first introduced Apple Pay in its conference in September 2014. Tim Cook, the CEO of Apple, illustrated that the tradition payment system like loading a card with a magnetic stripe and verifying a credit card by a combination of several numbers are not that safe. Apple Pay will be the new generation's payment system that can change the whole payment system. Basically, if a user want to use Apple Pay, he or she has to first add his card into his Apple Wallet. After that, the user can authorize a payment by just verifying his Touch ID through Apple Pay in all the retail stores and online stores supporting Apple Pay.



CARD & BILLING: BOFA VISA DEBIT (•••• 7614)
365 FOREST HILL DRIVE SYRACUSE...

CONTACT: PENG JIN
PJIN.ELVIN@GMAIL.COM
+852 6686 5120

SINGLE PASS X 2	$19.98
PAY EKSIBITION	$19.98

Figure 13: Apple Pay

For developers who want to integrate Apple Pay with their applications, there are following prerequisites:

- A developer account with payment processor.
- A valid Merchant Identifier.
- A public key and a private key for encryption and decryption.
- Apple Pay entitlement being included in the application.

9.1. Workflow of Apple Pay

Below is a diagram showing the workflow of Apple Pay in one purchase.

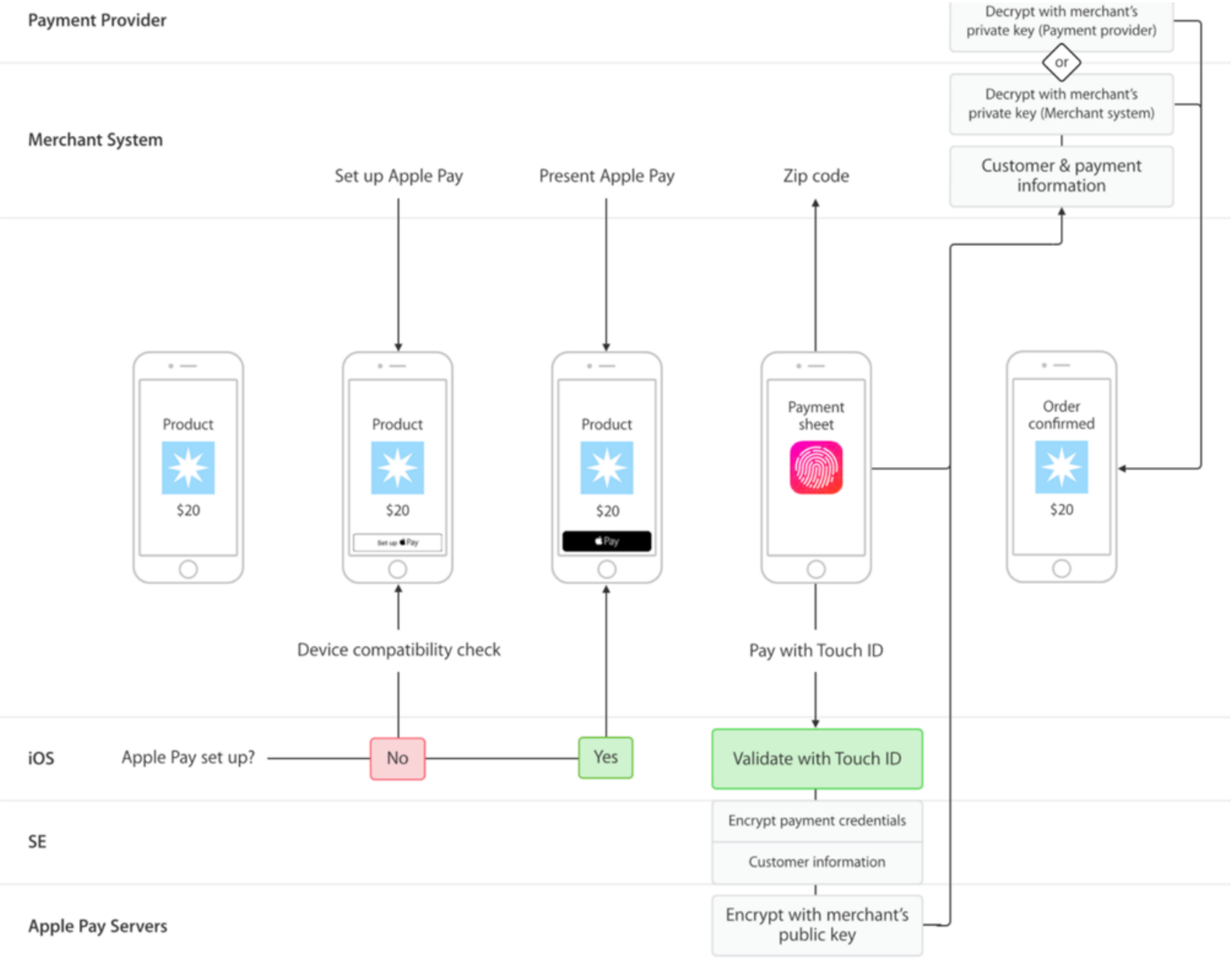


Figure 14: Apple Pay workflow [11]

9.2. Client-side

What we did in the client-side:

- Setting up the project to support Apple Pay by importing Apple Pay developer kit.
- Getting the information of billing, cost, shipping address and others.
- Generating an encrypted token representing this transaction.
- Sending the token to the server and handle the callback.

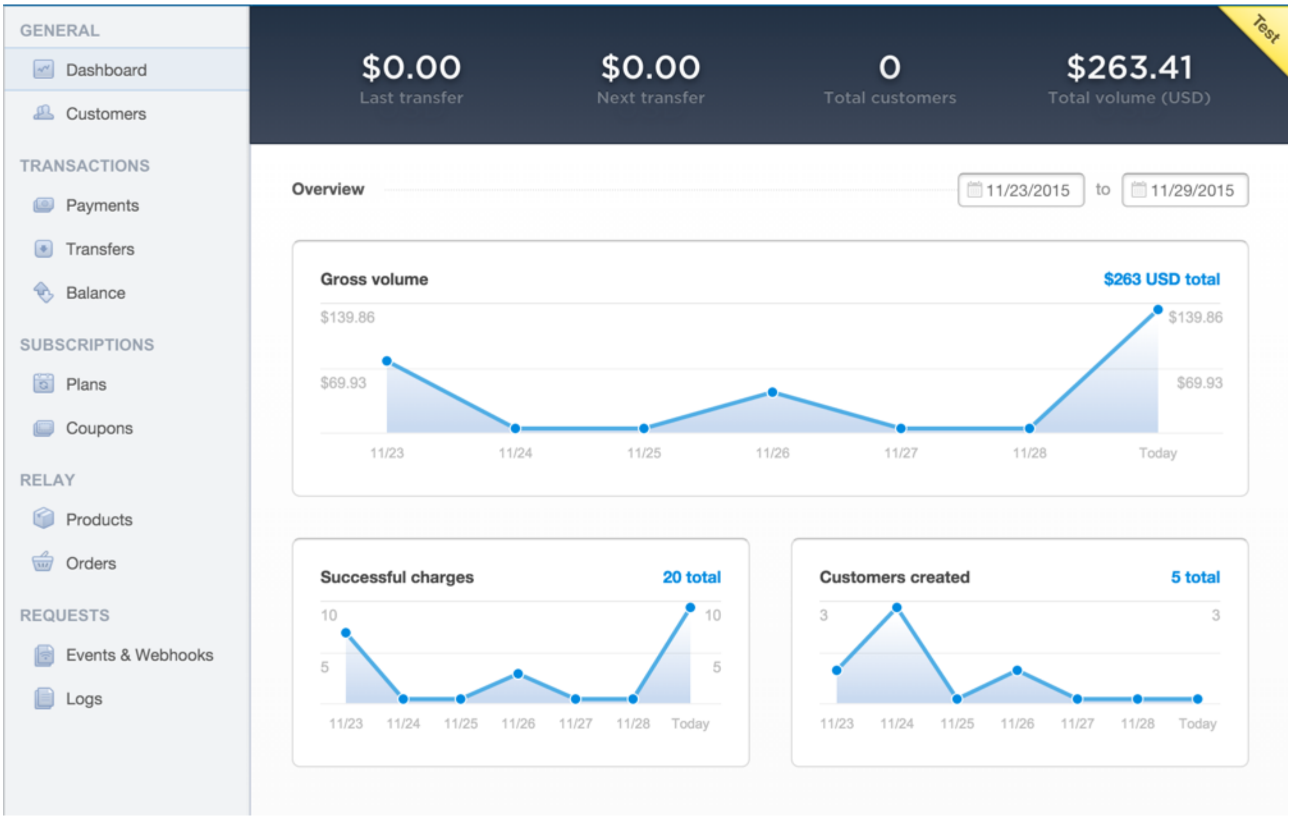
9.3. Server-side

For the server-side, we did the following things:

- Creating an APIs to handle the request from client, which carries the token and other transaction information.
- Decrypting the token and getting the payment credentials.
- Making the charge and send the result of it to the client.

9.4. Stripe

Stripe is a third party payment gateway service provider, who provides a easy platform for business owners to charge clients more easily. By integrating Stripe SDK, implementing the payment process with Apple Pay becomes much easier. Here is the steps we had gone through when integrating Stripe:



$0.00 Last transfer

$0.00 Next transfer

0 Total customers

$263.41 Total volume (USD)

11/23/2015 to 11/29/2015

Gross volume $263 USD total

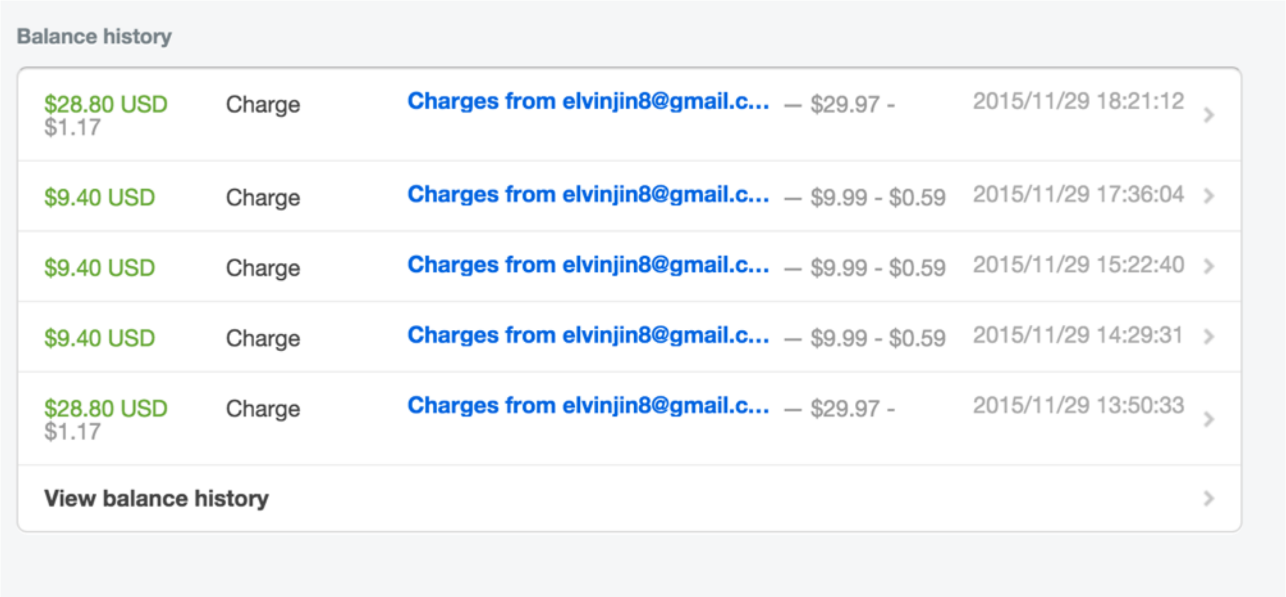
Successful charges 20 total

Customers created 5 total

Figure 15: Stripe dashboard

- Install the Stripe library on both client-side and server-side.
- Get the public API key from Stripe and configure it on both client-side and server-side.
- Get the information of the purchaser from Apple Pay.

After these 3 steps, we can use methods in Stripe library to finish transaction. Here, Stripe speed up the whole development process by handling all encryption, decryption and charging with the banks for the developers.



Balance history

$28.80 USD $1.17	Charge	Charges from elvinjin8@gmail.c... — $29.97 -	2015/11/29 18:21:12
$9.40 USD	Charge	Charges from elvinjin8@gmail.c... — $9.99 - $0.59	2015/11/29 17:36:04
$9.40 USD	Charge	Charges from elvinjin8@gmail.c... — $9.99 - $0.59	2015/11/29 15:22:40
$9.40 USD	Charge	Charges from elvinjin8@gmail.c... — $9.99 - $0.59	2015/11/29 14:29:31
$28.80 USD $1.17	Charge	Charges from elvinjin8@gmail.c... — $29.97 -	2015/11/29 13:50:33

Figure 16: Stripe transaction history

10. Study of Integrating Apple Wallet

Apple Wallet is an application introduced in iOS 8.1. Before iOS 8.1, it is called Passbook, which allows user to store their passes, like coupons, tickets and boarding passes. Apple introduced a new generation of Passbook named Apple Wallet together with Apple Pay in this year's conference. Users can add his own credit cards, debit cards and loyalty cards into his Apple Wallet.

Since we have finished integrating Apple Pay in our application, users can buy tickets right within our application using Apple Pay. What we are going to do now is to enable users to add purchased tickets into Apple Wallet so that users can redeem their tickets by just showing the passes in their Apple Wallet.

Figure 17: “Add to Apple Wallet” badge

10.1.Turning a Ticket into a Pass

Before trying to add a ticket into Apple Wallet as a pass, we have to learn what exactly a pass is. From a user's perspective, a pass is a collection of all the relevant information of an event. From a developer’s perspective, a pass is a package containing a JSON file as the template, which stores all the information and all the image resources that are required to build the pass.

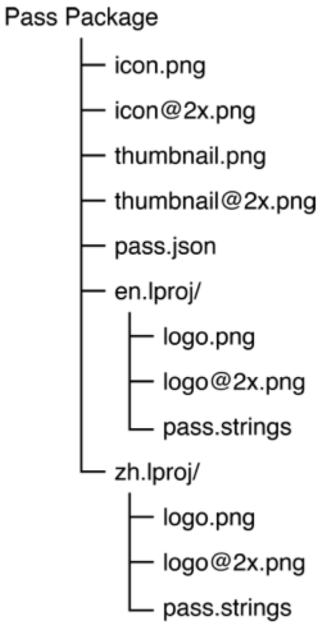
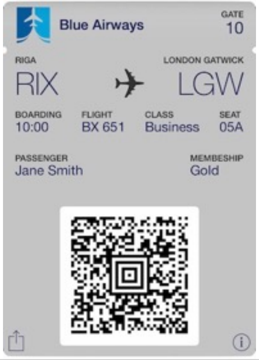
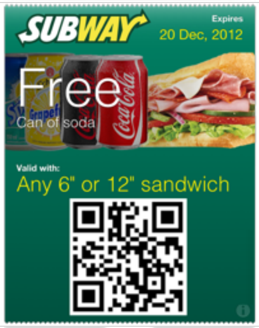
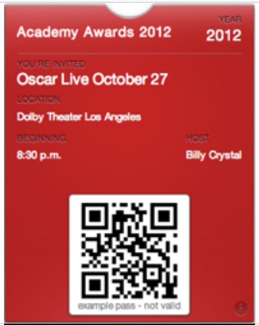
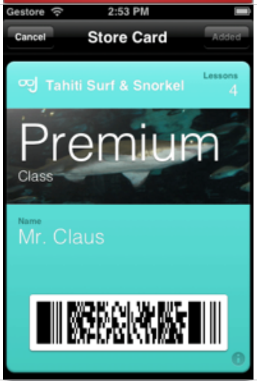
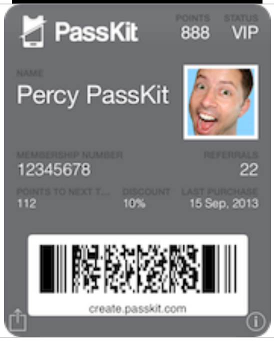
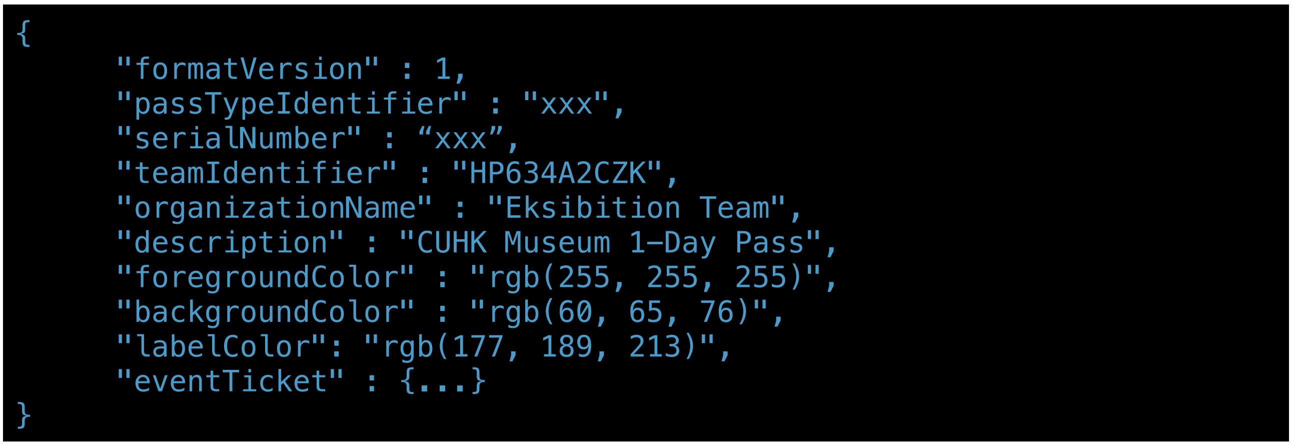


Figure 18: Pass structure

There are 5 different styles of passes:

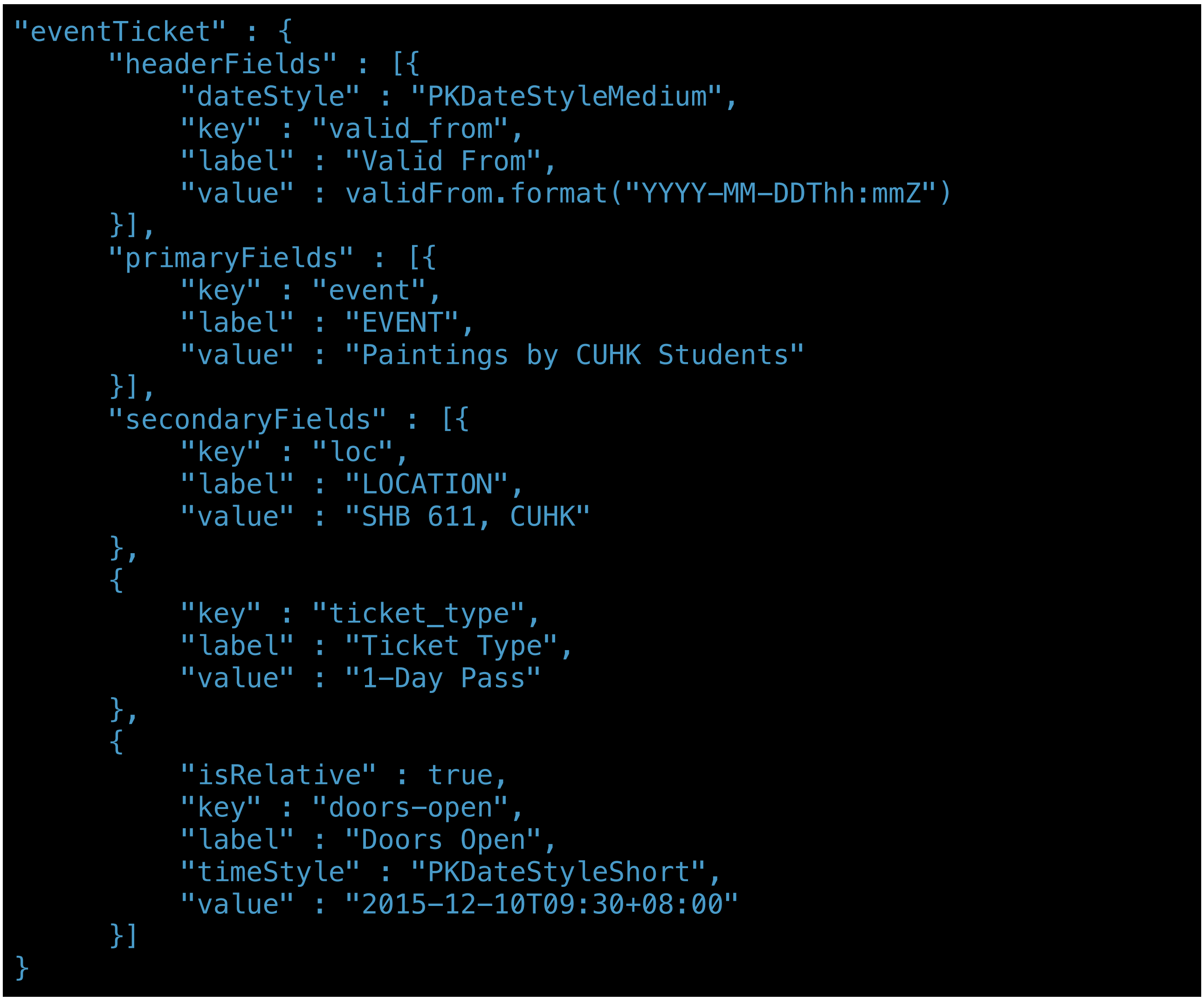
Pass Style	Description	Example
Boarding Pass	Representing a ticket to get discount when buying goods in the certain store.	 Blue Airways GATE 10 RIGA RIX LONDON GATWICK LGW BOARDING 10:00 FLIGHT BX 651 CLASS Business SEAT 05A PASSENGER Jane Smith MEMBESHIP Gold
Coupon	Representing a ticket to get discount when buying goods in the certain store.	 SUBWAY Expires 20 Dec, 2012 Free Can of soda Valid with: Any 6" or 12" sandwich
Event Ticket	Represent a ticket to get access to an event at a particular time and venue.	 Academy Awards 2012 YEAR 2012 YOU'RE INVITED Oscar Live October 27 LOCATION Dolby Theater Los Angeles BEGINNING 8:30 p.m. HOST Billy Crystal example pass - not valid
Store Card	Representing the user's account in a store or club.	 Gestore 2:53 PM Cancel Store Card Added Tahiti Surf & Snorkel Lessons 4 Premium Class Name Mr. Claus
	If the purpose of the pass is not among the above 4 categories.	 PassKit POINTS 888 STATUS VIP NAME Percy PassKit MEMBERSHIP NUMBER 12345678 REFERRALS 22 POINTS TO NEXT T... 112 DISCOUNT 10% LAST PURCHASE 15 Sep, 2013 create.passkit.com

The JSON file in the pass package is for specifying the information and the style of the pass which is in a format like this:



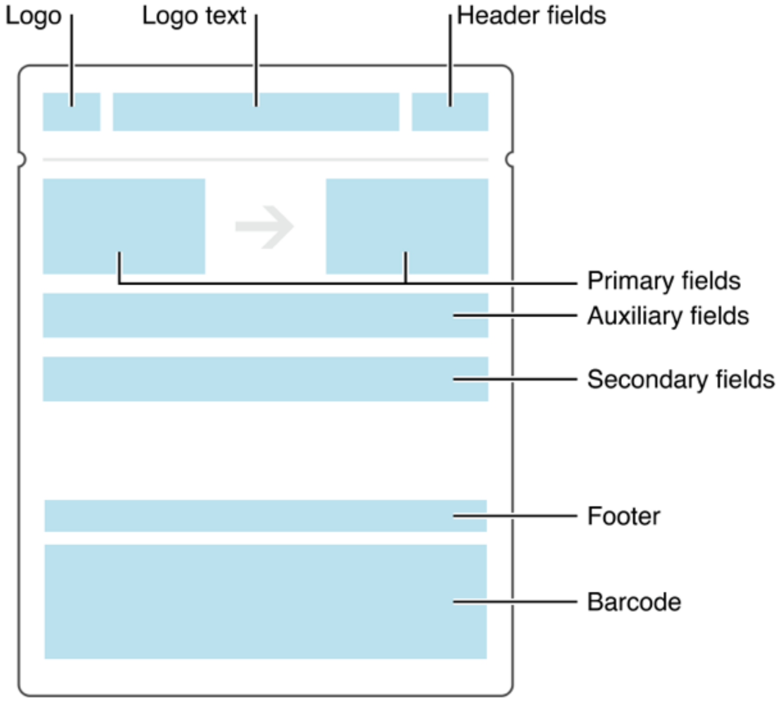
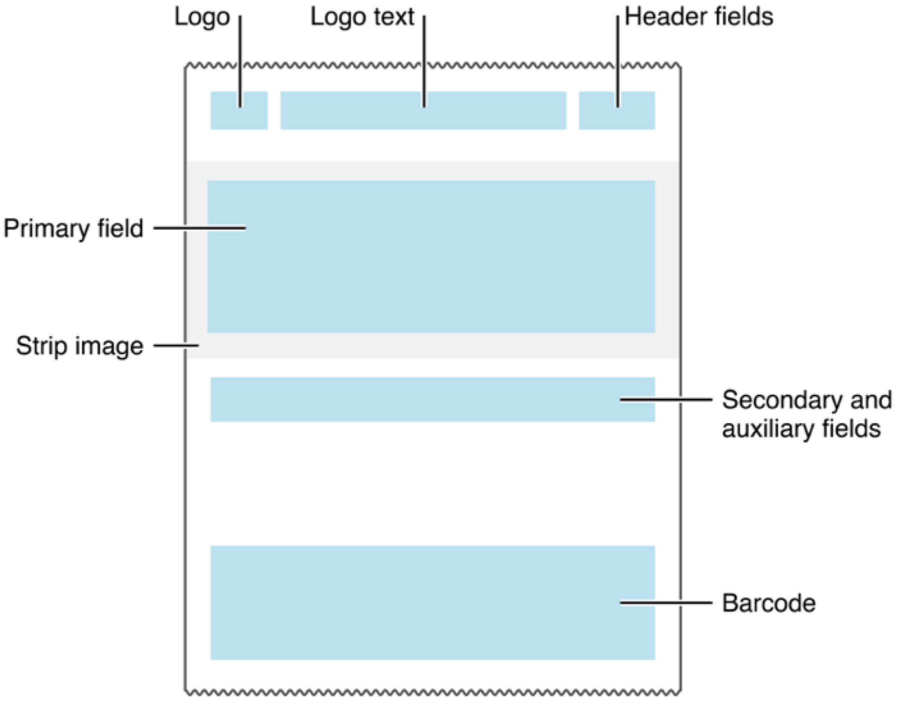
```
{
      "formatVersion" : 1,
      "passTypeIdentifier" : "xxx",
      "serialNumber" : “xxx”,
      "teamIdentifier" : "HP634A2CZK",
      "organizationName" : "Eksibition Team",
      "description" : "CUHK Museum 1-Day Pass",
      "foregroundColor" : "rgb(255, 255, 255)",
      "backgroundColor" : "rgb(60, 65, 76)",
      "labelColor": "rgb(177, 189, 213)",
      "eventTicket" : {...}
}
```

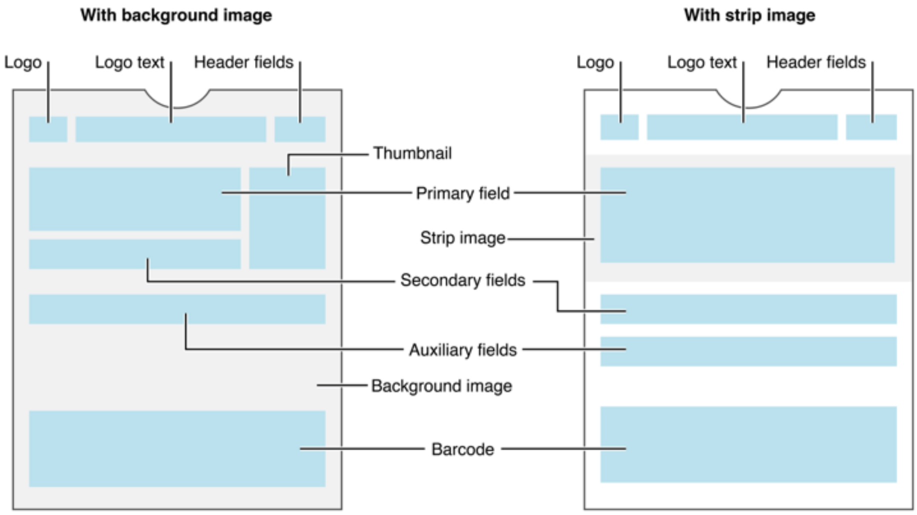
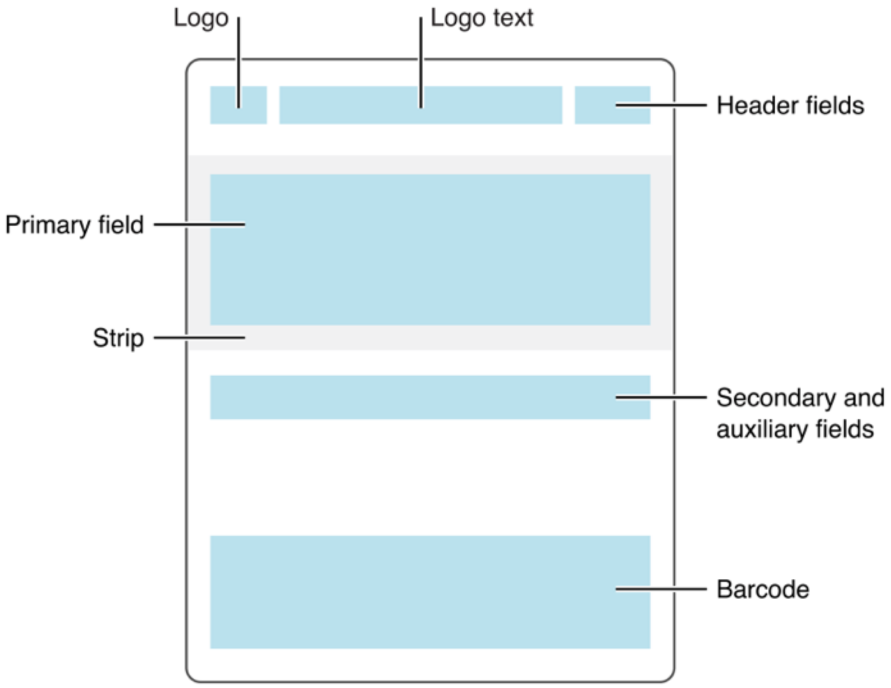
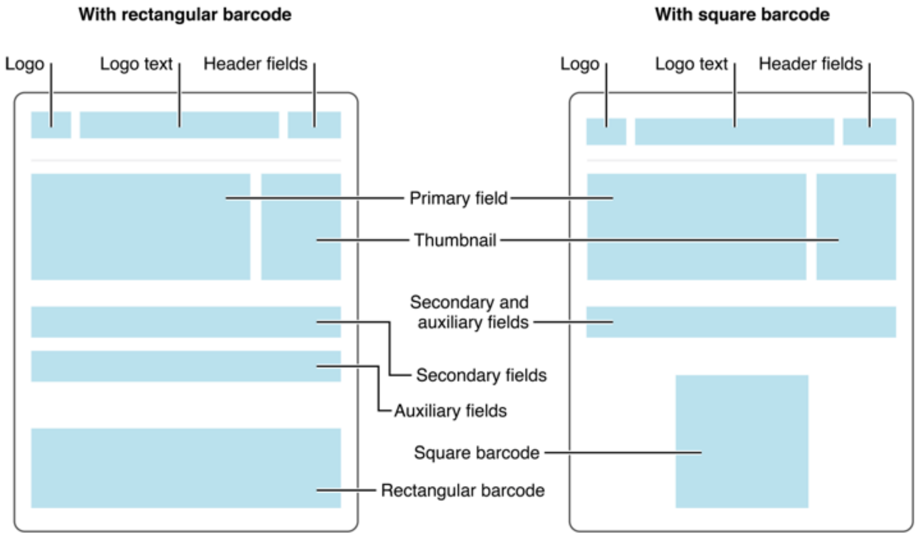
Taking the above code as an example, the field “eventTicket” will contain all the relevant information shows on the front page of the pass. Below is a sample of a event ticket.



```
"eventTicket" : {
      "headerFields" : [{
          "dateStyle" : "PKDateStyleMedium",
          "key" : "valid_from",
          "label" : "Valid From",
          "value" : validFrom.format("YYYY-MM-DDThh:mmZ")
      }],
      "primaryFields" : [{
          "key" : "event",
          "label" : "EVENT",
          "value" : "Paintings by CUHK Students"
      }],
      "secondaryFields" : [{
          "key" : "loc",
          "label" : "LOCATION",
          "value" : "SHB 611, CUHK"
      },
      {
          "key" : "ticket_type",
          "label" : "Ticket Type",
          "value" : "1-Day Pass"
      },
      {
          "isRelative" : true,
          "key" : "doors-open",
          "label" : "Doors Open",
          "timeStyle" : "PKDateStyleShort",
          "value" : "2015-12-10T09:30+08:00"
      }]
}
```

For different styles of pass, the layout of this field is different [16].

Pass Style	Layout
Boarding Pass	 Logo Logo text Header fields Primary fields Auxiliary fields Secondary fields Footer Barcode
Coupon	 Logo Logo text Header fields Primary field Strip image Secondary and auxiliary fields Barcode

Pass Style	Layout
Event Ticket	 With background image With strip image Logo Logo text Header fields Logo Logo text Header fields Thumbnail Primary field Strip image Secondary fields Auxiliary fields Background image Barcode
Store Card	 Logo Logo text Header fields Primary field Strip Secondary and auxiliary fields Barcode
Generic	 With rectangular barcode With square barcode Logo Logo text Header fields Logo Logo text Header fields Primary field Thumbnail Secondary and auxiliary fields Secondary fields Auxiliary fields Square barcode Rectangular barcode

After specifying all these fields in the JSON file and adding all the relevant images in the package, we still have to sign our pass package using the registered Apple Developer private key. We have go through the following steps to be able to sign a pass [13]:

- Go to iOS Developer Portal and add a pass into the pass list of our account.
- Download the certificate from the configuration of this pass and then add the certificate to our Keychain.
- Download the Apple Worldwide Developer Relations Certification Authority (WWDR) certificate and add it to the Keychain.
- Sign the pass package with these two certificates.

10.2.Sign and Distribute Passes

Since the key is private, we can't just have a static pass embedded in the application. We have to sign every single pass on our server whenever the client side request a pass. So we improved our server by adding a new API for signing and distributing passes according to the requests from the client-side.

We use a module called "passbook" in npm to help us do this. The module passbook requires the two certificates in format .pem. So that we have to convert the certificated to the right format by using the command provided by passbook.

```
node-passbook prepare-keys -p keys
```

After this, we went through the following steps to sign and distribute a pass of ticket from server according to the request from the client-side.

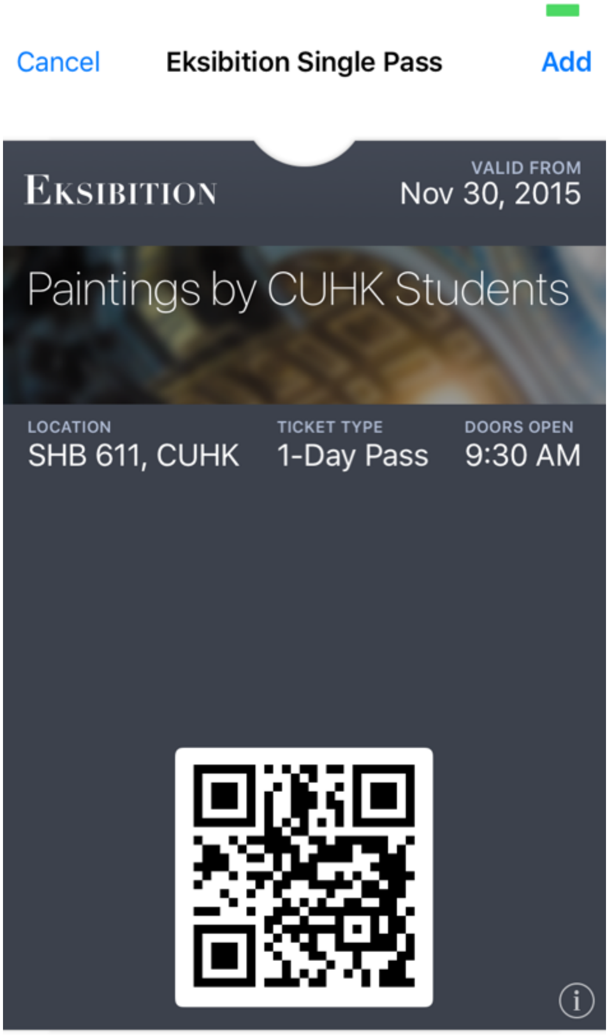
- Get the ticket number from the parameters in the request and get the related information about this ticket from the database providing the ticket number.
- Create a template for this ticket and update this template according to the information got from the database.
- Create a pass package with the template we got and sign this pass package with the two certificates.
- Send the package to the client and delete the local file of it.

11. Redeem System

After implemented the function that a user can add his or her tickets into Apple Wallet, we still need a redeem system for the museum organizer to check if the ticket provided by the user is valid or not. So we designed a redeem system and built another application for museum organizers to check the tickets.

As we can see from Figure 18, each pass contains a QR code at the bottom. The museum can use our redeemer application to scan this QR code to check the validity of the ticket.

The redeemer application will get a ticket number from the QR code and send it to the server. The server will check if this tick number is in the right format and if it is still valid. The server will give the redeemer application a response indicating the validity of the ticket number.



EKSIBITION

VALID FROM
Nov 30, 2015

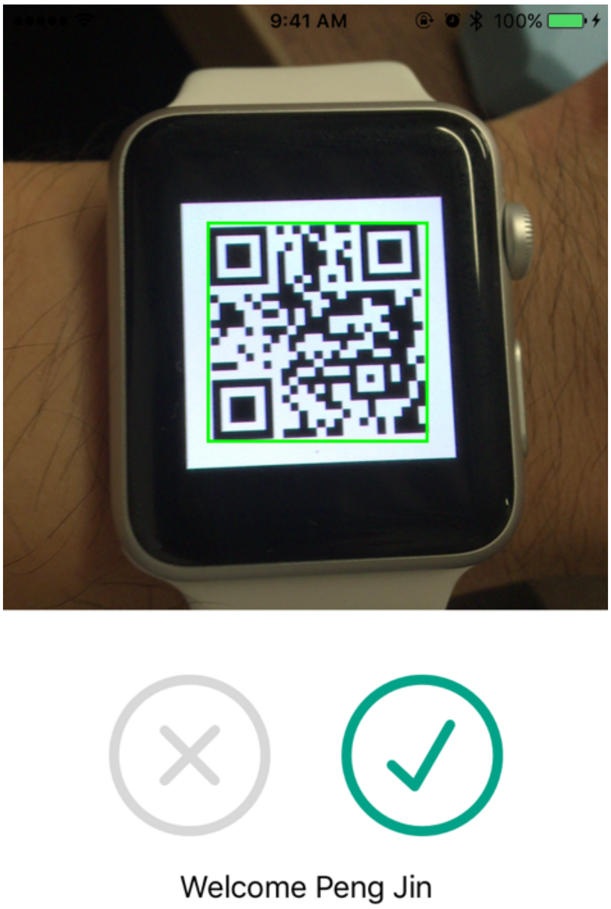
Paintings by CUHK Students

LOCATION
SHB 611, CUHK

TICKET TYPE
1-Day Pass

DOORS OPEN
9:30 AM

Figure 18: Sample Pass



Welcome Peng Jin

Figure 19: Redeem a ticket

12. Limitations and Future Development

Although we have implemented most of the function, there are still some limitations of our application that remain to be tackled.

- Right now all of our functions related to the user’s behavior are device based, which mean all the requests sent from one device will be recognized as one user’s action. This makes the analyzing of user’s behavior less accurate in case of the situation that different users used one device to use our application. We planned to add a user management system to let each user has their own accounts and link their accounts to their social networks. Such that the analyzing of user behavior will become more accurate and meaningful.

- On account of lacking Estimote beacons, we didn’t finish implementing the indoor positioning. But we did much research on how to use Estimote Indoor Location SDK to implement indoor position. We planned to create a coordinate transferring system to convert the relative position of the a certain point into the right position over the map.

- We have not done the research on the indoor path analyzing, which is to calculate a most reasonable path from two given points in the map. This function is highly related to the smart guide function which we are going to develop in the next semester.

- Right now, all the data related to museum items are hard coded into the database. We still need some more APIs for museum users to update the data of the items in the museum. We need to develop a user interface for museum organizers to login and manage their data like updating the information of an item, updating the relationship between items and beacons.

- The map and the positions of the beacons are also hard coded, so we should also develop a system to let museum users to edit their map and the positions of the beacons. How to save the position of the beacons is of great importance.

- The function of new notification is also unimplemented. In addition to the development of the user interface, developing the APIs for museum organizer and designing the format for the news are also of great importance.

- Currently, all the description are in English. We do not support multiple language selection at this time.

- Add a news notification system to notify users about the promotions of events.
- Develop an account management system to manage the account of the users and museum organizers.
- Make our application a platform for different museums to adopt. (If time allowed.)

13. Difficulty of the Project

During doing this project, we have faced the following difficulties which took us a relative long time to solve.

- When doing the research on the indoor positioning and studying on our first approach, we read some paper on this topic and found a solution that are relatively accuract. In order to understand the theory behind that approach, we did research on the features of the bluetooth energy in order to learn the way it broadcast and how its signals are interfered. At the end we got an algorithm and a plan on how to calculate the environment of one room and map it into a parameter for position analyzing, but we gave up this approach because it is nearly impossible to calculate the parameter, which represents the environment of a big and complex exhibition hall.
- Designing the indoor map also cost us a lot of time. As what we have mentioned at the section “Study of Custom Indoor Map Design”, we spend much time on trying to use Mapbox as a tool to create the custom indoor map, in which most of the time were spent on how to draw the map as a .geojson file. At that time, we even developed a quite complicated method to parse the .geojson file and draw it out by ourselves. But at the end, we gave up Mapbox for the reasons mentioned early.
- When building the server for signing and distributing the passes, we kept failing on the final point, which means we could not know what client side get from the server or whether the data generated by the server is correct. Since the error occurred at the final step, we don’t know the reason why the file failed to be downloaded and opened. After a long time fixing, we replaced the certificate and regenerate the .pem file using the command provided by passbook module and change the way we used to update the template of the pass. Finally, we get out pass and can successfully add it to our passbook.

14. Summary

After a one-semester long development, our application has already been equipped with some basic features. Besides those traditional user interactions, we also took advantage of the latest iOS features such as 3D Touch, Apple Pay and Apple Wallet.

We were keeping learning new technologies along the way of developing. We are getting much more familiar with terms like BLE, beacons, RSSI, and RESTFul and tools like Node.js, Stripe and MongoDB. Now we can make great use of these tools to develop better applications. After the learning procedure this semester, we are much more confident for the development next semester. We hope we can implement more powerful functions and fancy features in the future and makes our application better.

At the end of this report, we would like to give our special thanks to Prof. Michael R. Lyu and Mr. Edward Yau, who are willing to take their time to meet with us and offering constructive comments on our project. Without them we wouldn't have tackled this much difficulties.

Reference

[1] "iOS: Understanding iBeacon". Apple Inc. February 2015, https://support.apple.com/en-gb/HT202880

[2] "Beacons: Everything you need to know.". Pointrlabs.com. 18 January 2015. http://www.pointrlabs.com/blog/beacons-everything-you-need-to-know/

[3] Bluetooth.com, (2015). Bluetooth Low Energy | Bluetooth Technology Website. [online] Available at: http://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy [Accessed 28 Nov. 2015].

[4] Warski, B. (2014). How do iBeacons work?. [online] Blog of Adam Warski. Available at: http://www.warski.org/blog/2014/01/how-iBeacons-work/ [Accessed 28 Nov. 2015].

[5] iBeacon.com Insider, (2014). What is iBeacon? A Guide to iBeacons. [online] Available at: http://www.iBeacon.com/what-is-iBeacon-a-guide-to-beacons/ [Accessed 28 Nov. 2015].

[6] Developer.estimote.com, (2015). [online] Available at: http://developer.estimote.com/ [Accessed 28 Nov. 2015].

[7] Estimote.com, (2015). Estimote. [online] Available at: http://estimote.com/ [Accessed 28 Nov. 2015].

[8] Developer.estimote.com, (2015). What is Estimote Indoor Location SDK. [online] Available at: http://developer.estimote.com/indoor/ [Accessed 28 Nov. 2015].

[9] Developer.estimote.com, (2015). Build an App With Indoor SDK. [online] Available at: http://developer.estimote.com/indoor/build-an-app/ [Accessed 28 Nov. 2015].

[10] Developer.apple.com, (2015). Model-View-Controller. [online] Available at: https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html [Accessed 30 Nov. 2015].

[11] Apple Pay, (2015). Getting Started With Apple Pay. [online] Available at: https://developer.apple.com/apple-pay/Getting-Started-with-Apple-Pay.pdf [Accessed 29 Nov. 2015].

[12] Wikipedia, (2015). Log-distance path loss model. [online] Available at: https://en.wikipedia.org/wiki/Log-distance_path_loss_model [Accessed 1 Dec. 2015].

[13] White, T. and White, V. (2013). 10 Must Have Apps That Support Apple's Passbook | BestAppSite. [online] BestAppSite. Available at: http://www.bestappsite.com/10-must-have-apps-that-support-apples-passbook/ [Accessed 30 Nov. 2015].

[15] Mics.org, (2015). NCCR - MICS - Project IP6 abstract. [online] Available at: http://www.mics.org/micsProjects.php?groupName=IP6&action=abstract [Accessed 1 Dec. 2015].

[16] Developer.apple.com, (2015). Wallet Developer Guide: Pass Design and Creation. [online] Available at: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/PassKit_PG/Creating.html#//apple_ref/doc/uid/TP40012195-CH4-SW1 [Accessed 1 Dec. 2015].