Guideo: On my way

Jihan Li (jl4346), Mahd Tauseef (mt2932), Xiaofan Yang (xy2251), Xiaochen Wei (xw2353)

Main goal:

Travelling is an important part in most of people's life. But sometimes people don't know what the best plan of a trip is, so they would have to spend a long time searching the Internet about sites, route and advices from other people. In order to save them time on this, and make their trip safe and fun, we develop an iOS application, Guideo, to provide travellers the advices on sites, routes and events, and the experience from others.

We basically aim at achieving three goals:

- Offering the information of sites near the user;
- Recommending the trip route to go around interesting sites;
- Offering a platform to note and share users' trip advice;
- Offering the information of events near the user;

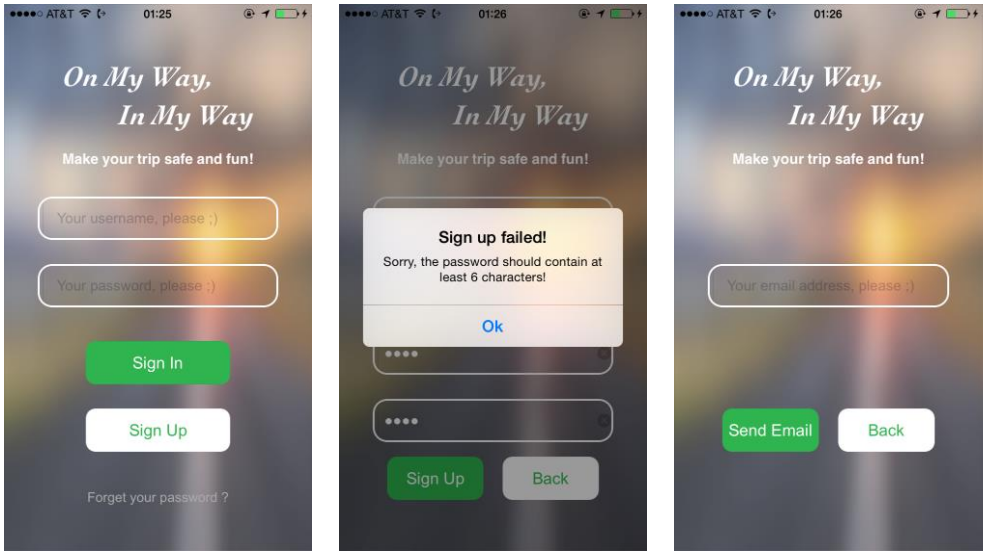
Front End:

Basically, our iOS front end contains five parts: login, sites, routes, events and notes.

✧ Login:

The login parts have three pages: sign in, sign up and retrieve password. We do Gaussian blur on the background image and provide textfields for users to input their user information. In sign in page, we simply check whether the textfields are filled or not and post the information to the back end for validation. In sign up page, we check the username and email address are valid or not based on some regular expression. We also give constraints on password and retyped password. In retrieve password page, we plan to provide password retrieving by email address. The back end will send the user's password to the input email.

Since the information of a user will be used by other sections. We define the user data class as a singleton class, so its properties can be set globally.



✧ Sites:

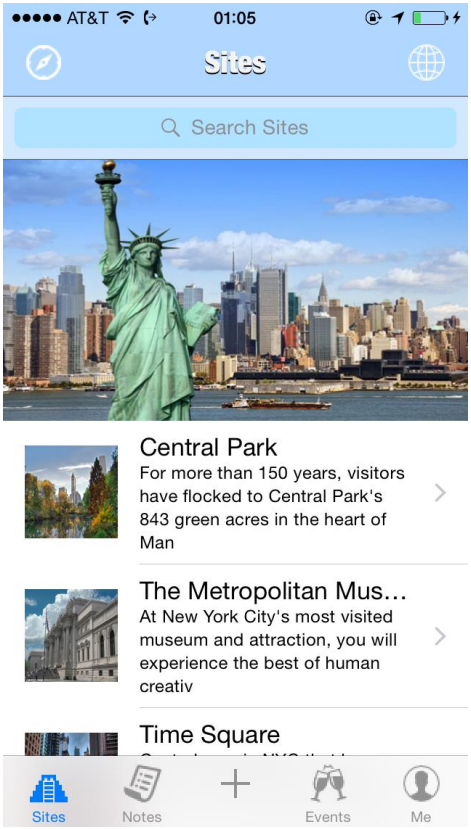
The site parts have two sections: site list and site details.

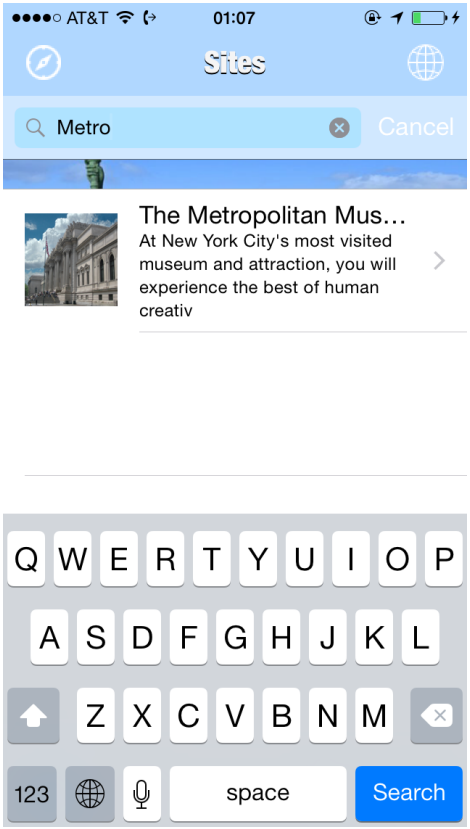
We produce a site list based on user's location when he log in. The sites basic information stored in our database will be listed in a table view. There is a head image for each site. To speed up the process of scrolling the table, we do not download all the head images at the beginning, but download them as the user views the page. We build an image cache in which the key is image url and value is image itself to be showed in the table view. Initially all the values are a default image. Then we use NSOperations to do concurrent downloading of the real images. Once an image is downloaded, we will change the value of the corresponding key to this image in the cache, and the table view will show it immediately.

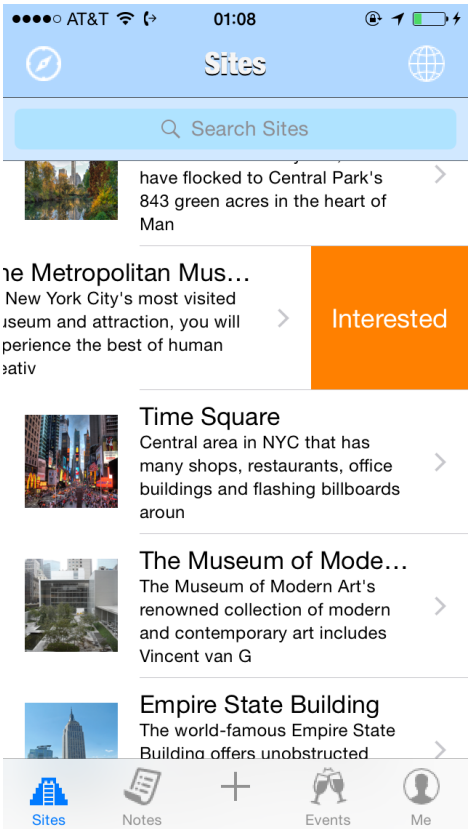
We complete a switch selection effect for users to note which sites they are interested in. We basically define a new cell type which can detect the gesture event. When swipe event happens in a cell, we will show a small side button for the user to note either he is interested in the site or bored with it. We will put all the sites that the user is interested in into a list which can be used by other sections. In this part, we referred to some existing techniques like how to define a swipe-sensitive cell.

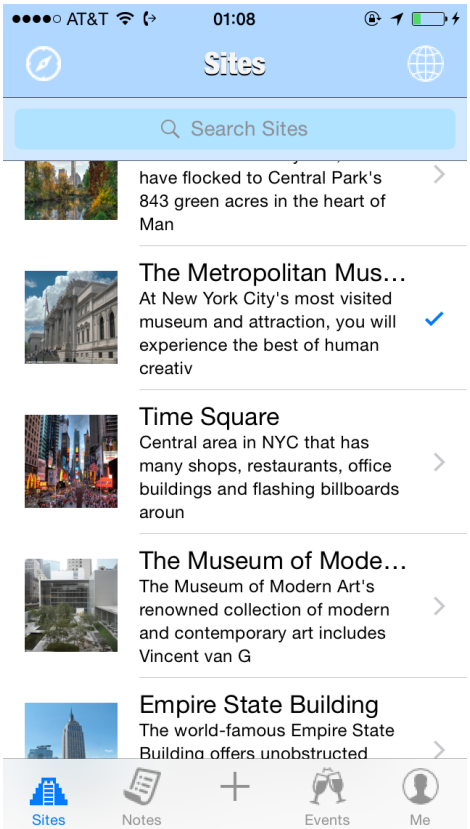
For site details part, we obtain the detail data for the site that the user selects. The detail data includes: site address, popularity (for now we rank the popularity by ourselves), price, opening time, trip time recommended, phone, coordinate and other introductions of the sites. We show the site in a google map based on its coordinate. Other information is showed in a tab view. Each tab contains a specific topic of the site. The user can choose from history, culture, artifact, etc to view as he goes around. We referred to some existing techniques on how to build a tab view and realize the animation effects.

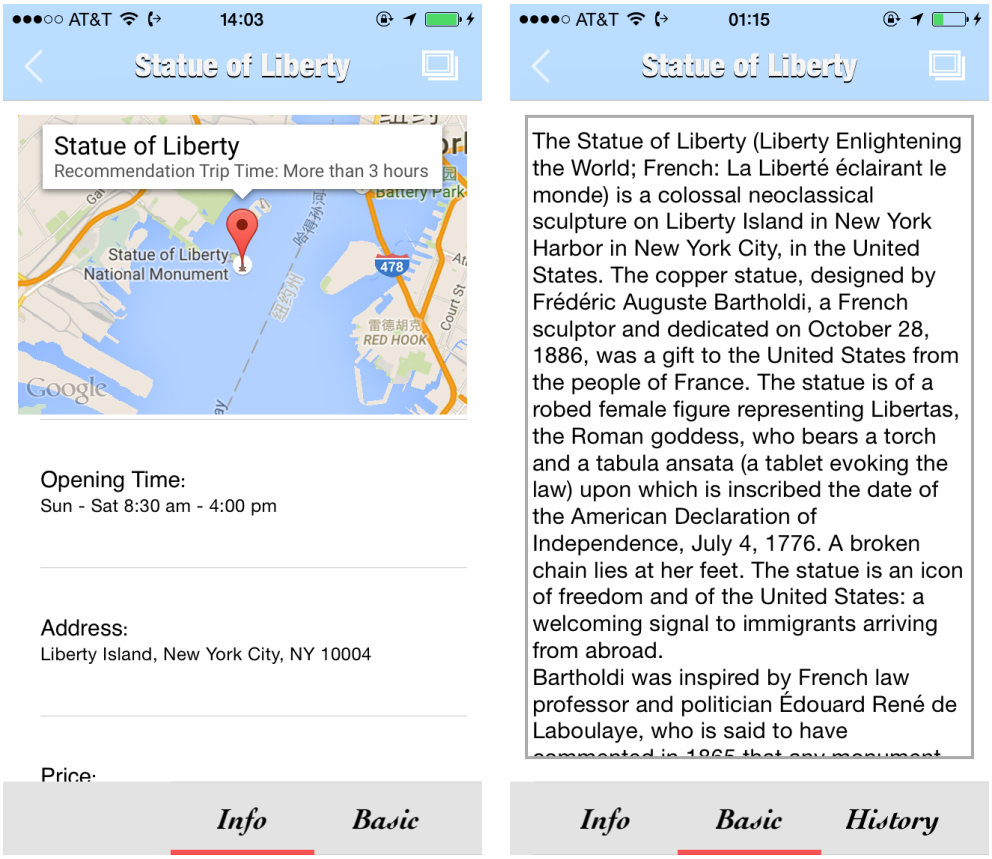
In the future, we will add a Like function to the site, so when a user click Like on a site, he will store it into his table in the database. And we will show it on user's home page.


●●●●● AT&T
01:05
Sites
Search Sites
Central Park
For more than 150 years, visitors have flocked to Central Park's 843 green acres in the heart of Man
The Metropolitan Mus…
At New York City's most visited museum and attraction, you will experience the best of human creativ
Time Square
Sites
Notes
Events
Me


●●●●○ AT&T
01:07
Sites
Metro
Cancel
The Metropolitan Mus…
At New York City's most visited museum and attraction, you will experience the best of human creativ
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
123
space
Search


●●●●○ AT&T
01:08
Sites
Search Sites
have flocked to Central Park's 843 green acres in the heart of Man
ıe Metropolitan Mus…
New York City's most visited
ıseum and attraction, you will
perience the best of human
ːativ
Interested
Time Square
Central area in NYC that has many shops, restaurants, office buildings and flashing billboards aroun
The Museum of Mode…
The Museum of Modern Art's renowned collection of modern and contemporary art includes Vincent van G
Empire State Building
The world-famous Empire State Building offers unobstructed
Sites
Notes
Events
Me


●●●●○ AT&T
01:08
Sites
Search Sites
have flocked to Central Park's 843 green acres in the heart of Man
The Metropolitan Mus…
At New York City's most visited museum and attraction, you will experience the best of human creativ
Time Square
Central area in NYC that has many shops, restaurants, office buildings and flashing billboards aroun
The Museum of Mode…
The Museum of Modern Art's renowned collection of modern and contemporary art includes Vincent van G
Empire State Building
The world-famous Empire State Building offers unobstructed
Sites
Notes
Events
Me



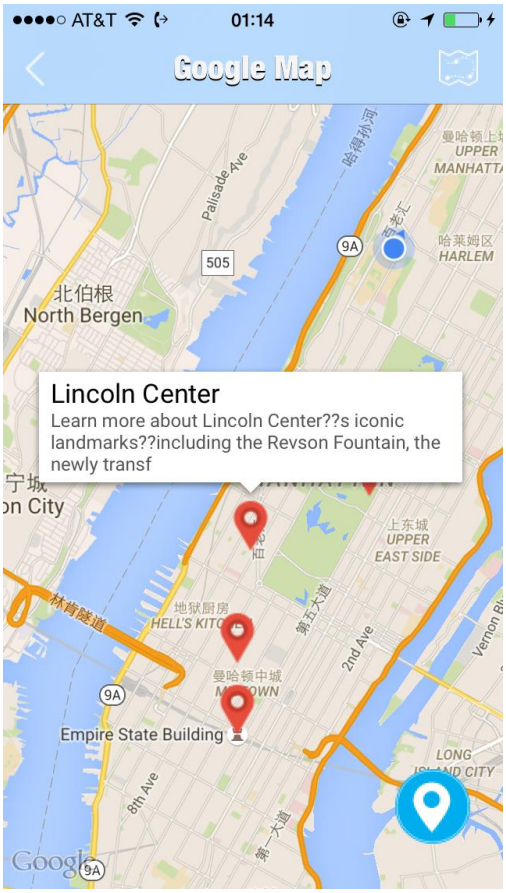
✧ Routes:

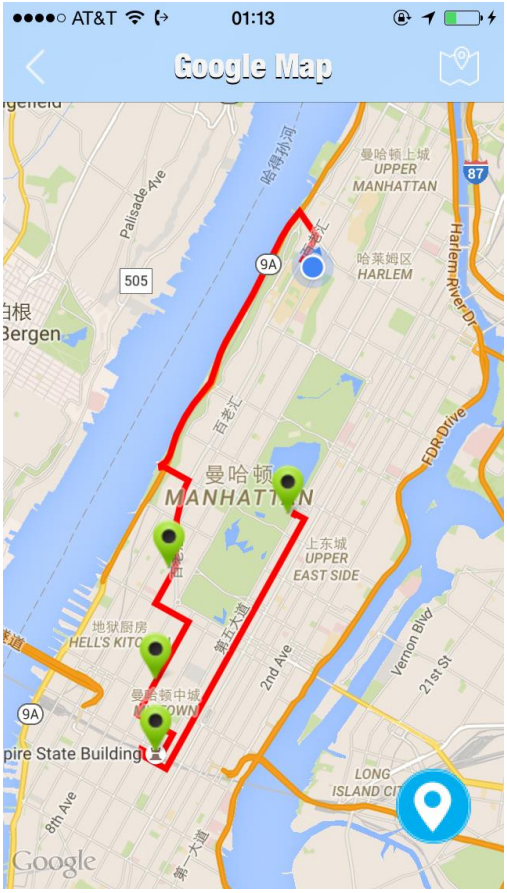
The route section is based on Google Map API for iOS and Google Directions API.

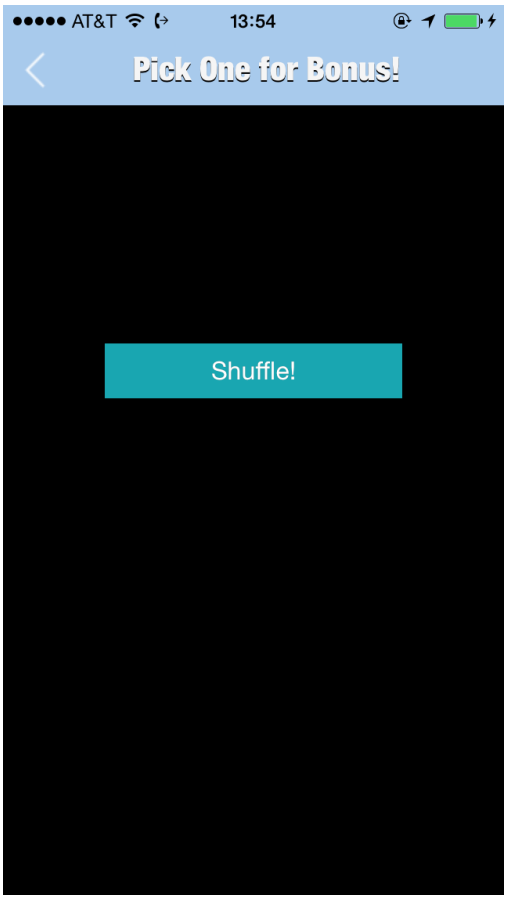
We initialize the camera of the map view based on the user's location. On the map, we only show the sites that the user is interested in. When the user touch the marker, it will pop up a small script of the introduction of the site. When the user tap the button on the right bottom, the camera will focus on the user's location again. We use CLLocation to manage the location services of iOS, such as getting the location of user, tracking the movement of user.

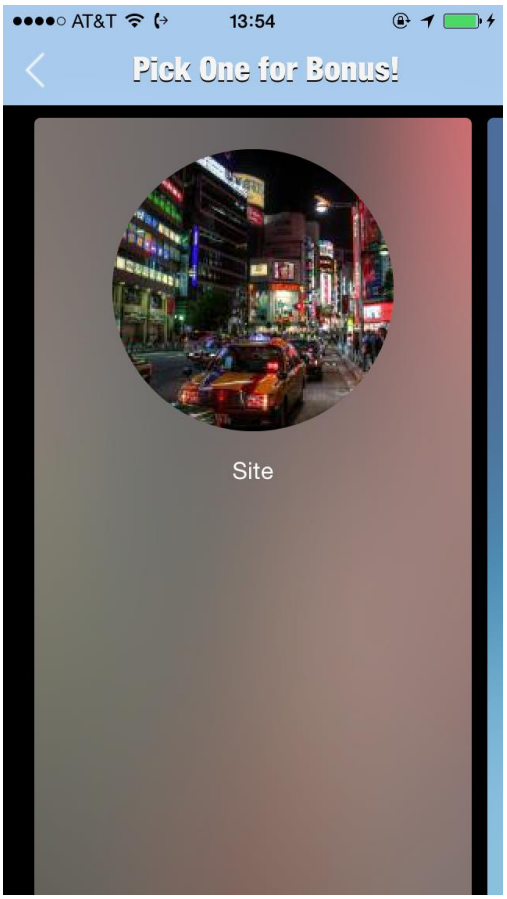
When the user taps the button on the right of the navigation bar, the back end will give back a site list in which the order of the sites are calculated based on the distance, the popularity and the opening time. Then we will show the route using Google Directions API, which gives the indeed route for walking.

We also provide another interesting section called Explore. If the user cannot decide which site is best to go or if he want to try something different, he can try the shuffle function by tapping the left button on the navigation bar. We will randomly select three sites based on the city where the user is. We will give the user some bonus points if he gets there. For now, this section is not completely done yet, but we will make it run in the future.


AT&T 01:14
Google Map
Lincoln Center
Learn more about Lincoln Center??s iconic landmarks??including the Revson Fountain, the newly transf
北伯根
North Bergen
Empire State Building
HARLEM
UPPER EAST SIDE
Google


AT&T 01:13
Google Map
MANHATTAN
UPPER EAST SIDE
HARLEM
Google


AT&T 13:54
Pick One for Bonus!
Shuffle!


AT&T 13:54
Pick One for Bonus!
Site

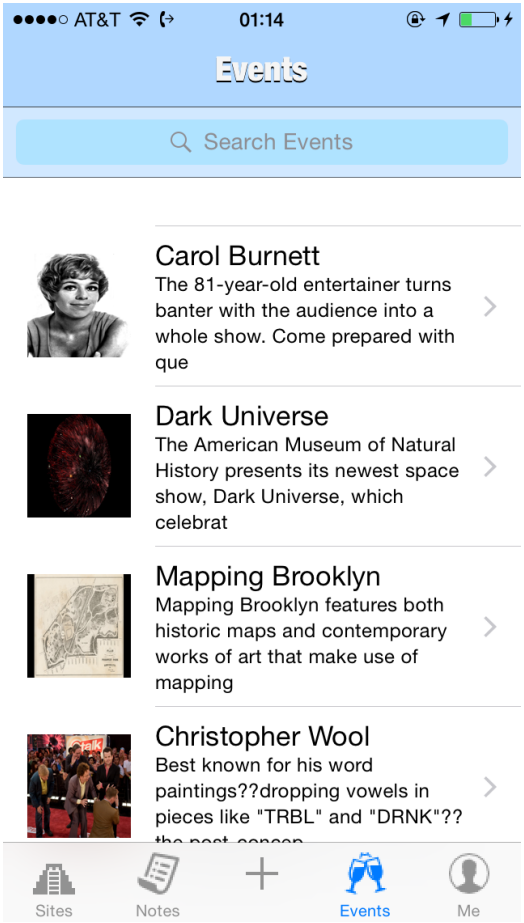
✧ Events:

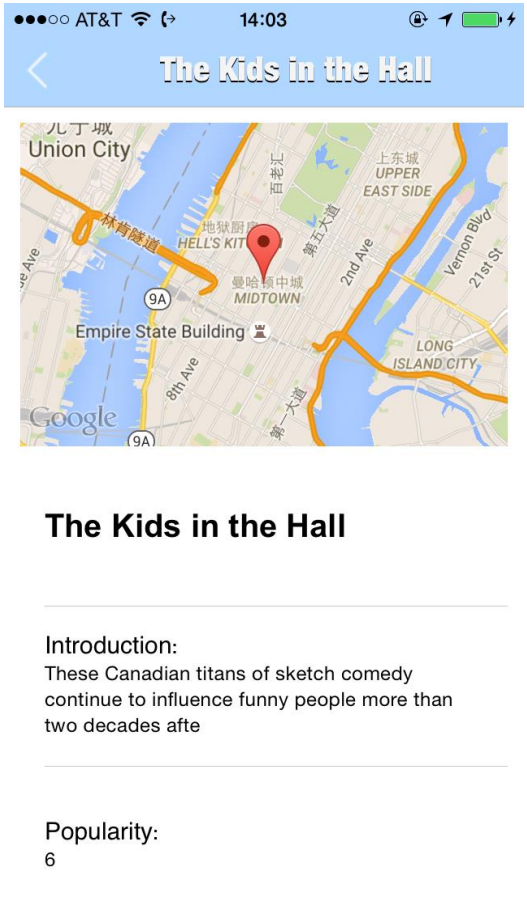
The event parts have two sections: event list and event details.

Event in our case refers to some official events posted by business, such as a musical is being shown in a theatre or the city is celebrating a holiday. So after we obtain these data from the database, we list them in a table view just as what we do in the site section.

After the user tap into the detail view of an event, the page will show event address, popularity, price, time, phone, coordinate and basic introduction of the event.

In the future, we will add a Like function to the event, so when a user click Like on an event, he will store it into his table in the database. And we will show it on user's home page.



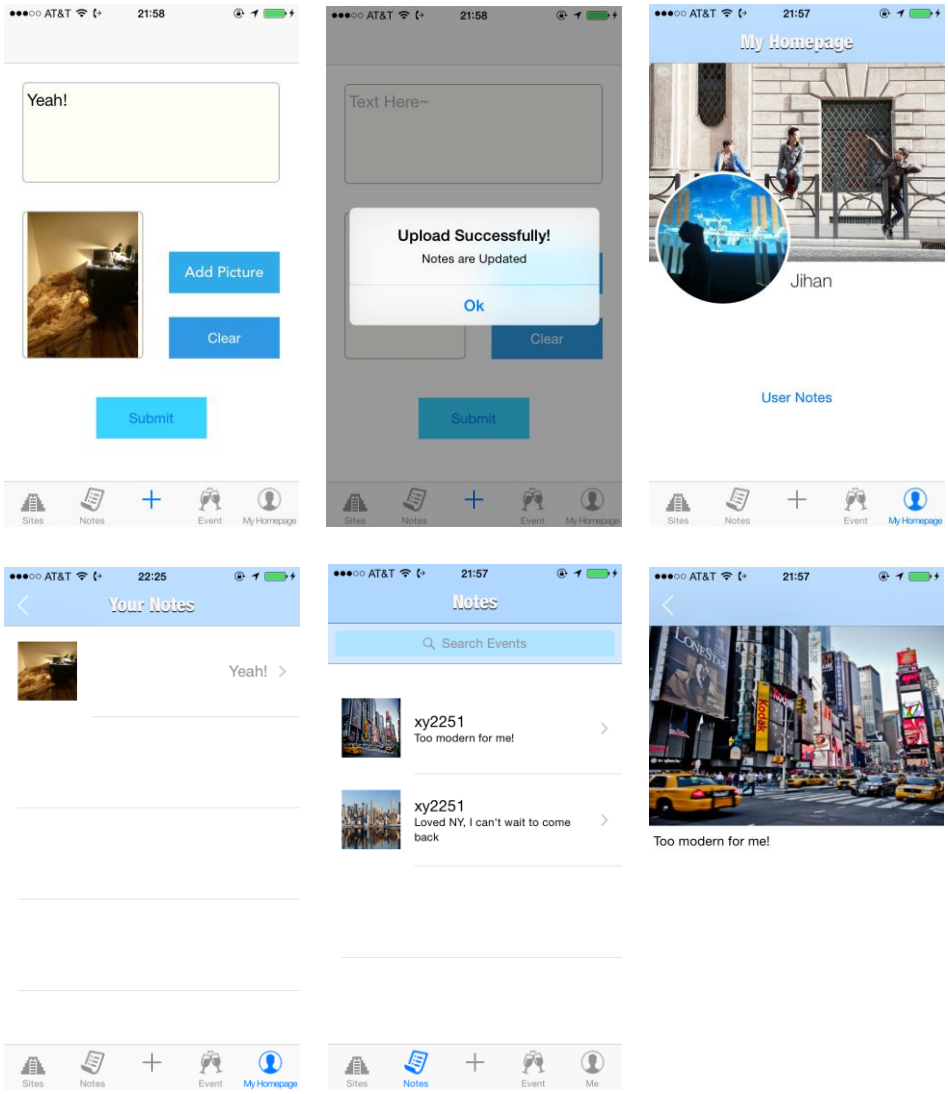


✧ Notes:

We provide a platform for users to post their travel notes to share experiences and comments. User can pick their photos in the gallery and post them with texts.

In note section, we list users' notes in a table view so that users can see travel notes of others. Each cell contains an image and text. In the future, we will show the notes based on user's location so that the user can view the notes which are more relevant to his current trip.

In the user's homepage, he can see his notes posted.



Back End:

The backend technology stack consists of Node.js, Express and MySQL.

Using Node.js had several benefits. It unified the language and data format (JSON), thus allowing us to optimally reuse developer resources. Moreover, Nodes use of non-blocking, event-driven I/O helps our app remain lightweight and efficient. It's capable of handling a huge number of simultaneous connections with high throughput, which equates to high scalability. We felt this was particularly important for our app since it is a product which would work best when being used by a large number of users. Therefore, we chose to develop our backend using Node.js.

Node.js has great support for package management using the NPM tool that comes by default with every Node.js installation. The npm modules are reusable components, available through easy installation via an online repository, with version and dependency management. We use several of these modules which we list below:

- Express.js:

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It provided us with a myriad of http utility methods and middleware which made routing and handling requests easy and creating responses efficient.

- Body-parser:

The bodyparser module makes it easy to parse through and extract relevant elements from the request body.

- Mysql:

This is a node.js driver for mysql that we used for database querying and updating. It provided all the standard function calls and error handling required when interacting with a db.

- Async:

Async is a utility module which provides straight-forward, powerful functions for working with asynchronous JavaScript. We used it to implement better control flow in cases when we needed to serialize the method calls.

- Geolib:

This is a library that provides functions for basic geospatial operations like distance calculation, conversion of decimal coordinates to sexagesimal and vice versa, etc. We use it to calculate distances of Sites from our users and further process that information to return meaningful results

✧ Backend-Frontend communication:

We employ the use of the Express router to help communicate between the front-end and the server. Calls to perform different back-end functions are sent to different URIs/end-points. The server receives the http request generated by the front-end and the Express router helps redirect it to the appropriate handler by looking at the URI targeted in the request. The handler executes the relevant calls and the appropriate db queries and then generates and sends the response back to the front-end. In cases of error, an error message is sent notifying the user about what went wrong.

✧ Path Routing Algorithm:

We basically do routing for sites that user is interested in. The algorithm has three factors to consider: distance, popularity and opening time of the sites.

We take the interesting site list obtained from the site section in front end as the input. Then we define an effective distance to reorder the sites in the list. In each round, we select out the site with minimum distance and put it into the list. Then take the location of the site as the new location of the user and recalculate the effective distance of other sites. We will also update the time by adding the traffic time and the typical tour time of this site to current time. Repeat this process until all the sites in the original list is either selected or taken off. We typically take a site

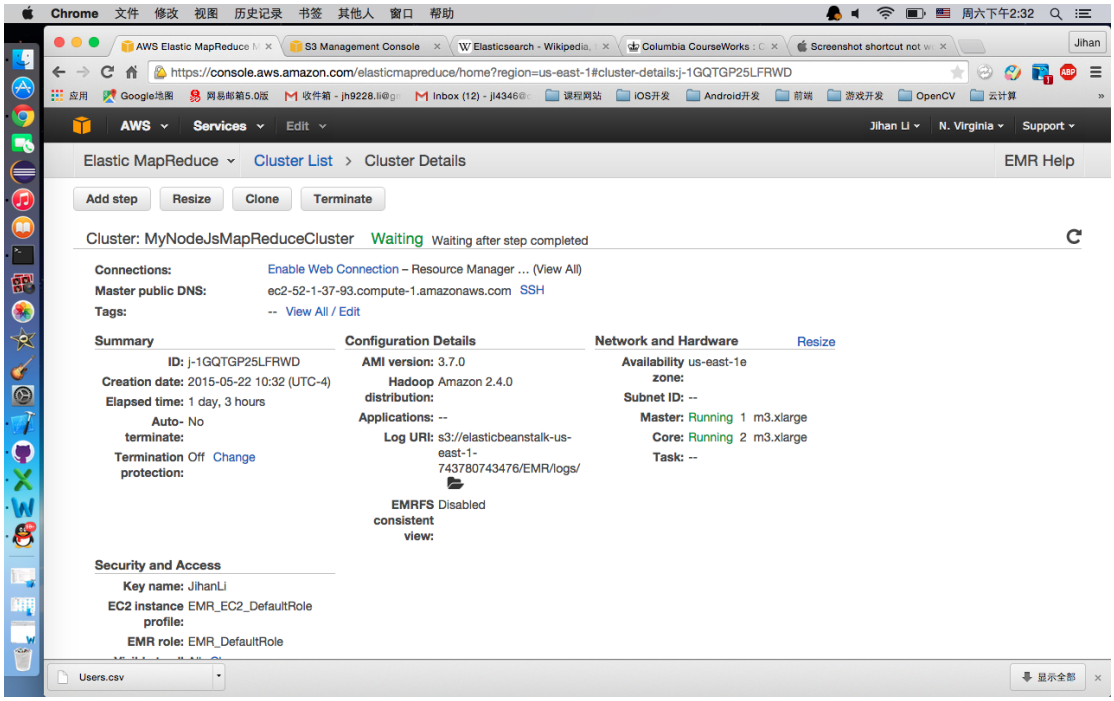
off the list when the current time plus traffic time is not within the opening time of that site.

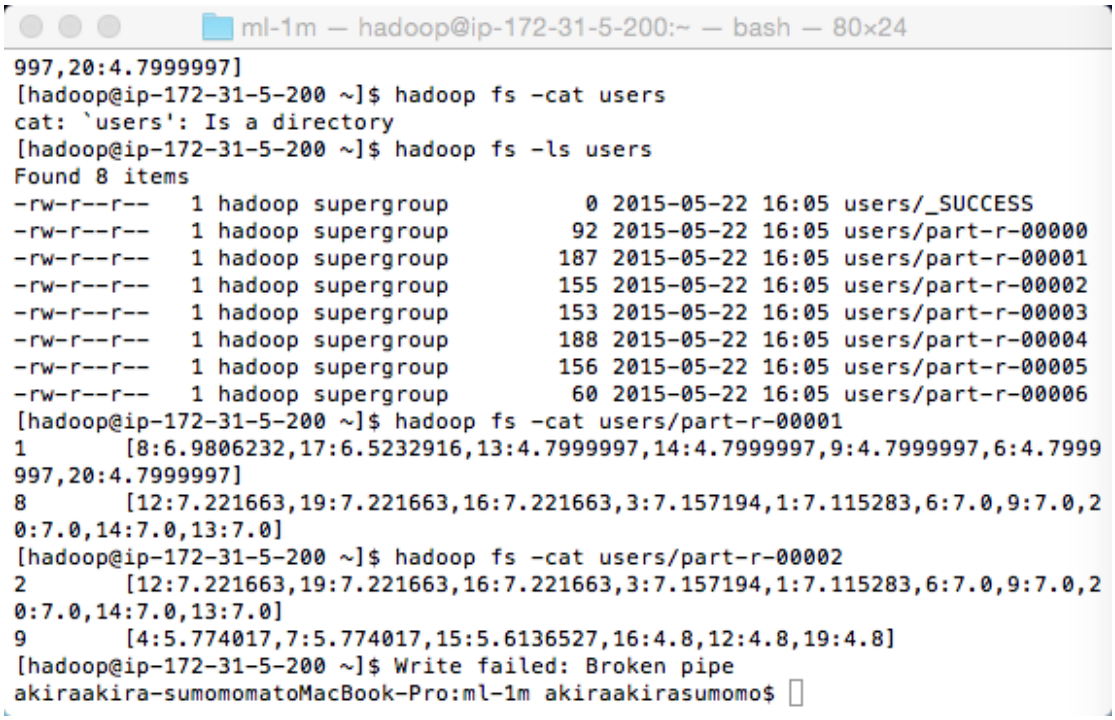
The equation for calculation effective distance is:

$$effectiveDistance = \|sitePosition - currentPosition\|_2 \times \frac{5.0}{popularity} \times \frac{(currentTime < openTime)\,?\,(openTime - currentTime) : (closeTime - currentTime - trafficTime)}{endTime - currentTime}$$

✧ Ranking Algorithm:

For user who has already show his interest of some sites, we will store his interest in the database as (User ID, Site ID, Score). For now, the score is set to 1 as long as the user once selects the site as his interest. After a certain period of time, we will transform the data into csv form, run a simple shell script to ssh into the Amazon AWS EMR Hadoop cluster, and run item-based collaborative filtering algorithm using Apache Mahout framework. The basic idea of this algorithm is to find the similarity of sites based on the interest history of all users. So we will rank the sites with high similarity to his interest history on the top of the list. For now, the ranking result is not very robust since the number of users and their behaviors is not big enough to learn. We will improve this section in the future.





Database:

✧ Data source:

We use Java to crawl information of sites and events from Wikipedia, Tripadvisor and NYCgo website. Information of users, notes and interests are created by users. We use phpMyAdmin to load data into our database built on Amazon RDS (MySQL).

✧ Data structures:

We design the following five tables for our application:

Sites: basic info (name, photo), location info (city, address, zip code, latitude, longitude), contact info (phone number, website, email), description (history, culture, architect), travel notes (open time, recommending visit time, fee, activities) and ranking.

Events: basic info (name, photo), location info (city, address, zip code, latitude, longitude), contact info (phone number, website, email), travel notes (open/end date), brief introduction and full description.

Users: basic info (name, password, email, head photo) and profile (description, preference).

Notes: title, written time, clicked time and access control.

Interests: user ID, site ID, score.

We provide search functions for all five tables. Additionally, For Users, we provide insert and update functions. For Notes and Interests, we provide insert, delete and update functions.

Contributions:

- ✧ Jihan Li: building the front end of sites, routes, events page; designing the routing algorithm; doing ranking of sites based on Apache Mahout techniques; deploying the app on iOS.
- ✧ Xiaochen Wei: building the front end of notes, adding notes, explore and user's home page.
- ✧ Mahd Tauseef: setting up servers; building the back end; doing processing of data in the back end; communicating with the front end and the database.
- ✧ Xiaofan Yang: crawling data; setting up database for users, sites, events, interests and notes; maintaining the database.

References:

- ✧ Data: crawled from Wikipedia, Tripadvisor and NYCgo, including image urls.
- ✧ Sample images: gathered using Google image search.
- ✧ Icons: gathered from Iconfinder and Icons8 for iOS.
- ✧ Map: Google Map API for iOS, Google Directions API, Tutorials from Mano Marks.
- ✧ Swipe selection cell technique: SWTableViewCell, created by Matt Bowman.
- ✧ Tab view technique: ViewPagerController, created by Ilter Cengiz.
- ✧ Card picker technique: MCCardPickerCollectionViewController, created by Michael Chen.
- ✧ Card view: RKCardView, created by Richard Kim.
- ✧ Collaborative filtering technique: Apache Mahout.
- ✧ Back end server: Amazon AWS EC2, Amazon AWS EMR Hadoop Cluster.
- ✧ Database: Amazon AWS RDS.