A report on

FANCY FOODIE : A FOOD REVIEW APP FOR IPHONE

Done by

Full Name	Xinjiang Shao
UIN	675469866

Advisor	Jakob Eriksson
Secondary Committee	Ugo Buy

UIC UNIVERSITY OF ILLINOIS AT CHICAGO Department of Computer Science COLLEGE OF ENGINEERING

Department of Computer Science
University of Illinois at Chicago

Spring 2013

CERTIFICATE OF APPROVAL: MASTER'S PROJECT

UIC GRADUATE COLLEGE
UNIVERSITY OF ILLINOIS AT CHICAGO
601 S. Morgan Street (MC 192)
http://grad.uic.edu/

Student Name: _____ UIN: _____
Last First M.

Degree Program: _____ Degree Sought: _____ Program Code: 20FS

Project Title: _____

Results:

_____ SATISFACTORY Project meets the standards of scholarly performance expected of master's candidates in the field.

_____ UNSATISFACTORY Project lacks the minimum criteria for approval.

Grade Change:
(Indicate below any research courses requiring a grade change upon completion of the project.)

Subject	Course #	Course Ref#	Credit Hours	Term	Current Grade	Final Grade

AUTHORIZED APPROVAL

Major Advisor: _____ Date: _____

Committee: _____ Date: _____

_____ Date: _____

_____ Date: _____

_____ Date: _____

_____ Date: _____

Department DGS: _____ Date: _____

Graduate College: _____ Date: _____

Abstract

Food review is becoming trendy among young people since smart phones are getting more and more popular. However, not many tools available for people to review a cuisine and publish their comment to social media conveniently in the market. This project is meant to develop an iOS App called *Fancy Foodie* which could take picture of food, attach meta data such as location, rate, comment, date, tags to the picture and share with friends via social media including Facebook, Twitter, Weibo etc.. The App also provides ways to search events by address and review statistic of all events.

Contents

List of Figures

Chapter 1

Introduction

1.1 Scope of Project

Mobile applications(apps) are making our life more and more interesting than ever before. A lot of young people love to take photos with their smart phone, make comments to it and share with their friends about what they ate. I personally find it would be useful if there is a tool for anyone to create food event and share events with friends.

The aim of the project is to be able to review food. Before you start eating, a user would take a picture of the food they are going to eat and give some basic information like location, tags, date etc. Tags are used to describe the type of the food such as "Chinese, Bun" or "Tea, Classic, Jasmine Green". After finish eating, the user would make comments to the cuisine. This app also lets the user search nearby location or location defined by the user, which will show a list of pins on the map indicating the user had been there before, to decide what the user want to eat. When the user saves a food event, he or she also have the ability to share the review and photo of food to Facebook, Twitter or to an email address. For the user to see the history records, the app provides a way to view statistics data which answers to questions like how many places the user has been, how the rate is, how many tags the user used etc. ?

1.2 Motivation

Seeking for places to eat is a thing in my genuine. I always want to keep track of my food adventures so that next time I could have better sense of what kind of cuisine I should order. I could also give recommendations to

my friends about the adventures. Luckily, it turns out that I am not the only person who want to do such kind of thing. A few of my friends tell me that people nowadays love to take photos of the food before they eat, and post their photos to all kinds of social media such as Facebook, Twitter, Weibo (Chinese Twitter) etc.. Especially for Asian students, they have really a strong motivation to take pictures of food before they eat.

After searching the apps available in apple app store, I couldn't find any suitable choice for my need. So I come up with an idea that I need to make best use of skills and build a good App for people who like to explore food world with friends.

Currently, various of apps related to food are available in Apple's app store. But most of them focused on the whole store/restaurant review like *Yelp*. This could be inaccurate sometimes because you might just hate one cuisine so you hate the restaurant. Things can be totally opposite if you love one cuisine but still hate the restaurant. So I'm hoping we could have an app only evaluate the cuisine itself. And there are also a few recipe-related apps like *Evernote Food*. Food app from Evernote did a good job showing the meals. The user interface is really friendly. Adding tags to the meal is easy and choosing place information is convenient in Food. The app also provided functionality to add a cuisine recipe as well. But people who like to take pictures of their food are not always into making food. So in our project, we won't do any recipe recording. We want to have a better tool to publish what you eat instead of how that cuisine is made.

1.3 Deliverable

An iOS app called *Fancy Foodie* is built and submitted to Apple's app store via iTunes connect. The version 1.0 is released. The category of the app are "Food Drinks" and "Lifestyle". The default language for *Fancy Foodie* is English. Another web site(http://soleo.github.io/Fancy/) is built to support the app.

1.4 Intended Audience

The target users for *Fancy Foodie* is people who love to try different kind of food and enjoy using social media to publish their experience of food adventures.

1.5 Platform for Deployment

- Operating System: iOS 6.0 or later
- Hardware: iPhone 4, 4S , 5 or iPod Touch 4th Generation

1.6 Development Platform

- Operating System: Mac OS X 10.8.3
- Processor: 2.9 GHz Intel Core i7
- Memory: 8 GB 1600 MHz DDR3
- Integrated Development Environment(IDE): Xcode 4.6 with Automatic Reference Counting(ARC)
- Version Control: git

1.7 Road map

Fancy Foodie 1.0

Core Features

- Taking picture of food, storing all meta data in local database
- Listing all the food events in a table view and detailed view
- Showing basic statistics information of the events
- Searching by using current user location or user input address
- Configuration of fancy foodie

User Interface Design

- Designing app logo, navigation bar, icons and background.
- Using Font-Awesome and Droid Sans font as default in app

Testing

- Using TestFlight do beta testing on the fly
- Testing in iPhone 5, iPod Touch 4th generation

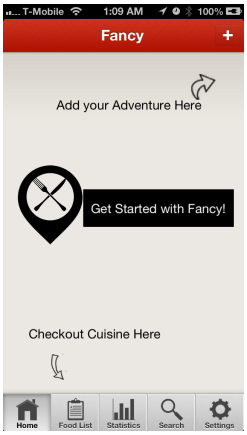
Chapter 2

The Graphic User Interface

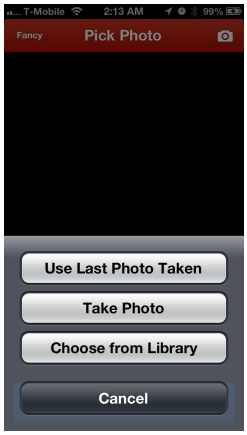
2.1 Home Tab

As shown in Figure 2.1, home tab is created for adding new event to the app. The default view for user when entering the app is the view shown in Figure 2.1(a).

Figure 2.1(a) gives a short tutorial for user. If the user chooses the plus sign in navigation bar, another blank view with black background should show up. After tapping on the camera icon in navigation bar, View in Figure 2.1(b) will display an action sheet including "Use Last Photo Taken", "Take Photo", "Choose from Library" and "Cancel". Here we use "Take Photo" as an example. For instance, we choose "Take Photo", the app should pop up a modal and let user take picture of food. A modal like Figure 2.1(c) should appear for user to move and scale the photo to a right position and size after choosing a picture. Figure 2.1(d) will show up after scaling step. If no photo is chosen, a warning will show up which alerts user to add a photo first. Tapping on next to move to next view as shown in Figure 2.1(e). In this view, a form is created for user to fill in location information, date, tags, comment and rate. Since the user just starts to eat, comment and rate field can be empty for now. Other fields should be filled correctly in this view because the user won't be able to edit all the other fields once saved the event. In Figure 2.1(g), the location list is generated through Foursquare Application Programming Interface(API). Foursquare API is chosen because it has a good reputation in both academic and industry world for rich venues in their database. By passing current longitude and latitude, we're able to have a venue list based the distance.



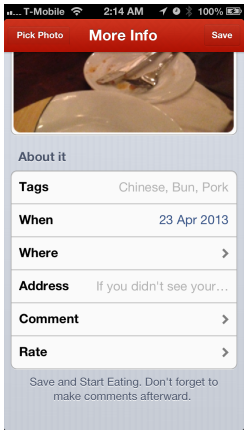
(a) Guide View



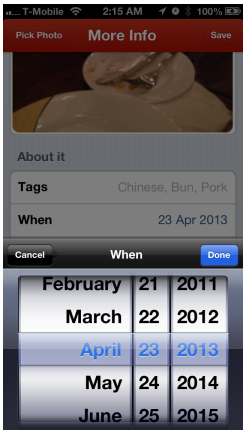
(b) Pick Action

(c) Move and Scale

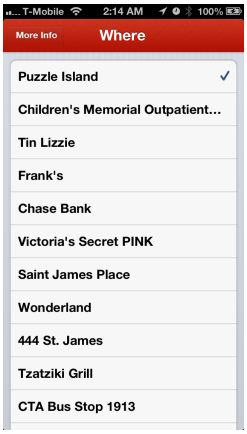
(d) After Picking



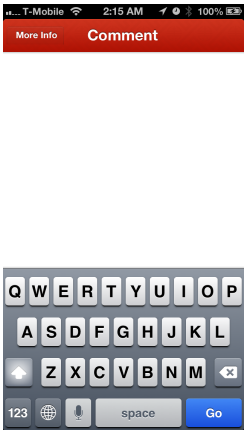
(e) Event Form



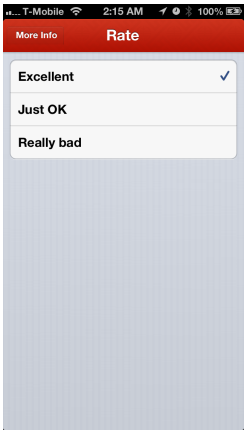
(f) Date Picker



(g) Location Picker



(h) Comment Editor



(i) Rate Picker

Figure 2.1: Home Tab View

2.2 Food List Tab

In food list tab, it fetches a list of food events ordered by creation date. Figure 2.2(a) shows some events. On each table cell, it shows a thumbnail, relative date and event location name. A green menu pop up button is created for sharing with social media(Figure 2.2(e)), updating comment(Figure 2.2(c)) and changing rate(Figure 2.2(d)).

After choosing one event, a detail view of the event will appear as shown in Figure 2.2(f) and Figure 2.2(g). You could see all the detailed information here which stored in the local database. In the top navigation bar, a sharing button displays in order to let user publish the event conveniently. If the user chooses Twitter, it'll check if he is logged in Twitter or not. After making sure the user have a Twitter account, Figure 2.2(h) will appear. The comment and photo is generated by the app. Simply tapping on send, and the event will be shared.

2.3 Statistics Tab

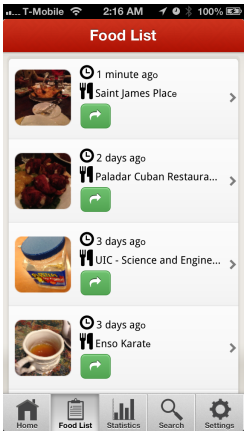
Statistics tab is supposed to give the user an overview of all events he created. As shown in Figure 2.3, "Rates" are clustered in different categories. All the tags are counted and shown in the table. The total number of places are shown in the table as well. In this way, the user could easily get an idea of what kind of places and food he likes.

2.4 Map Tab

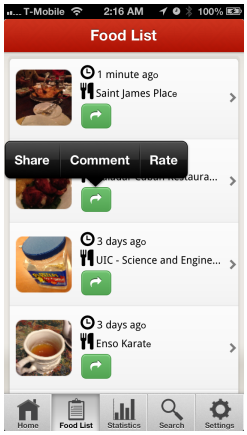
In Figure 2.4(a), all the event locations are labeled with red pins. By default, it is showing the events around your current location. But you could also change your view by passing address in Figure 2.4(b). After choosing a location in the list, the table view will be dismissed and the new central region will be the location you chose.

2.5 Setting Tab

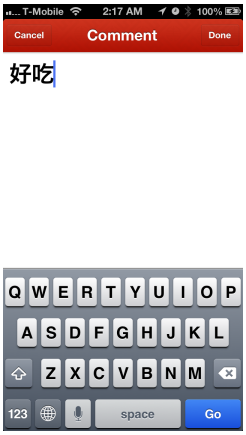
Setting tab provides a view to setup configurations of the app (Figure 2.5). The view in this tab is a static table view built in storyboard. A web-view controller is created to show the app website and author introduction. By



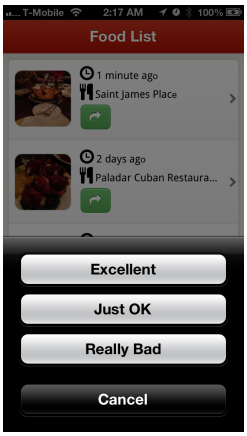
(a) Food List



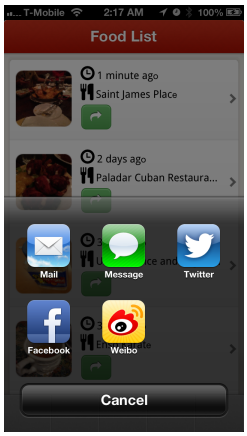
(b) Menu Pop-up



(c) Comment Editor

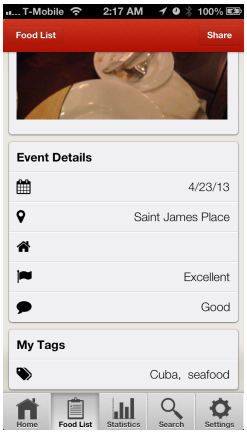


(d) Rate Action Sheet

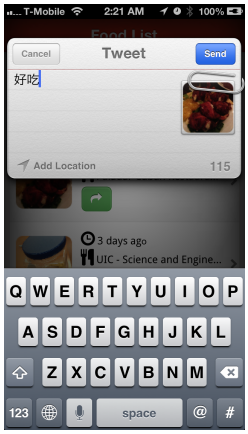


(e) Sharing Activity Controller

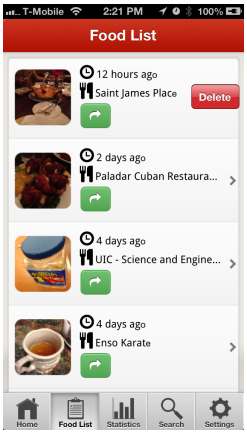
(f) Detail View



(g) Detail View Cont'd



(h) Twitter



(i) Delete View

Figure 2.2: Food List Tab View

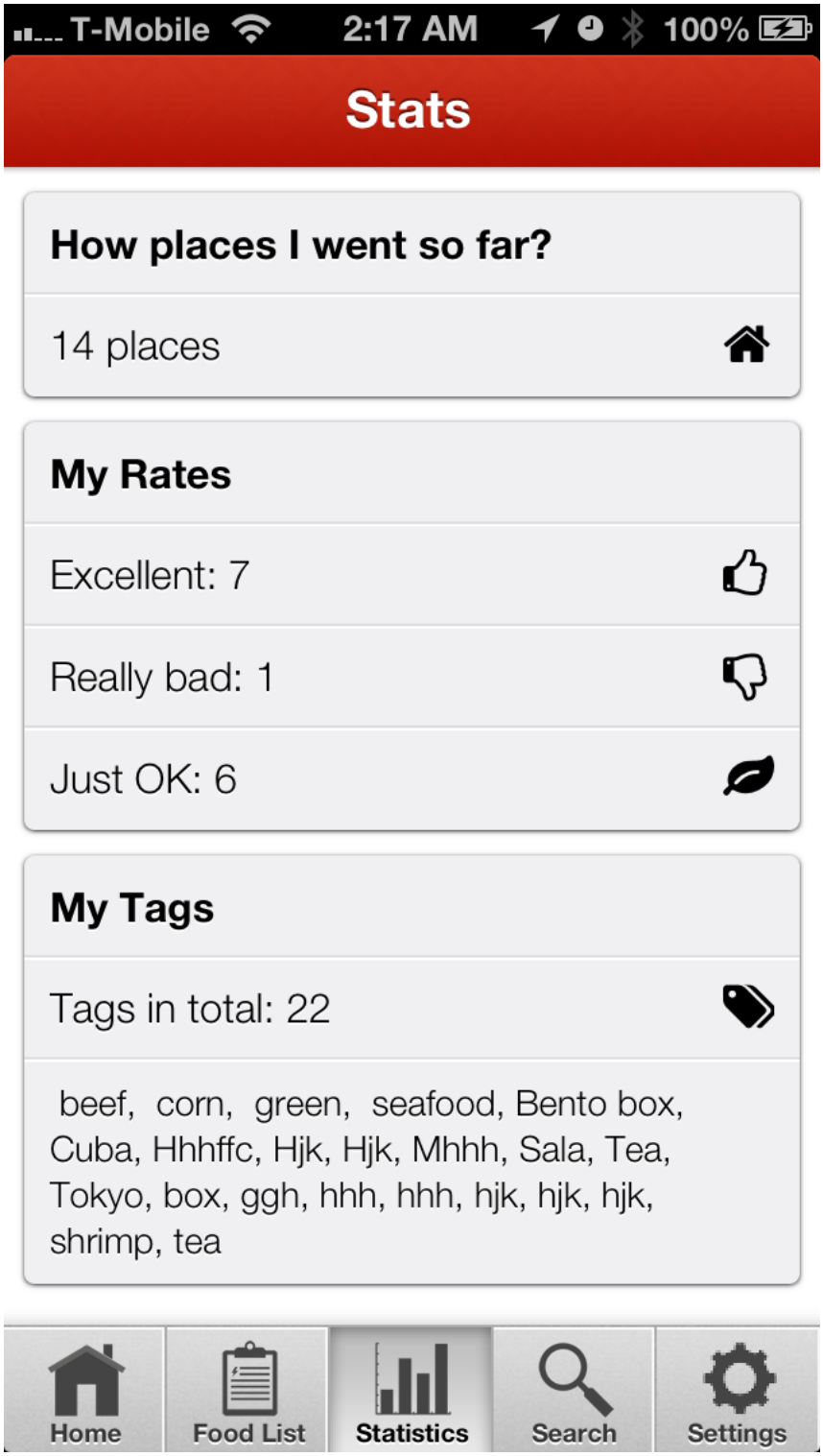
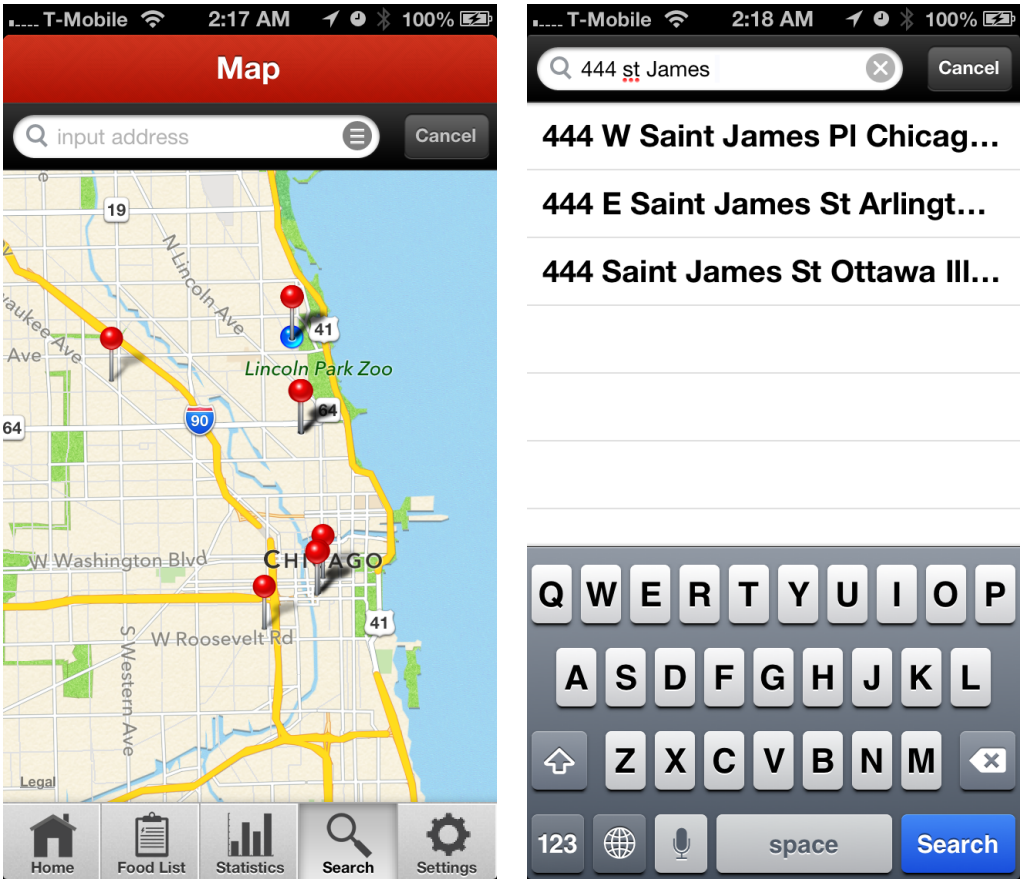


Figure 2.3: Statistics Tab View



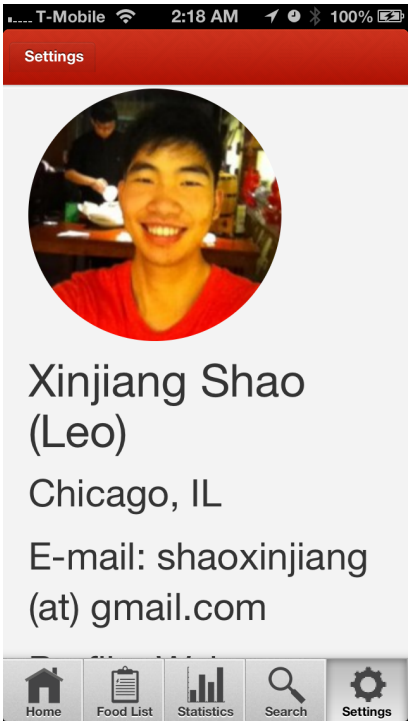
(a) Map (b) Address Searching

Figure 2.4: Map Tab View

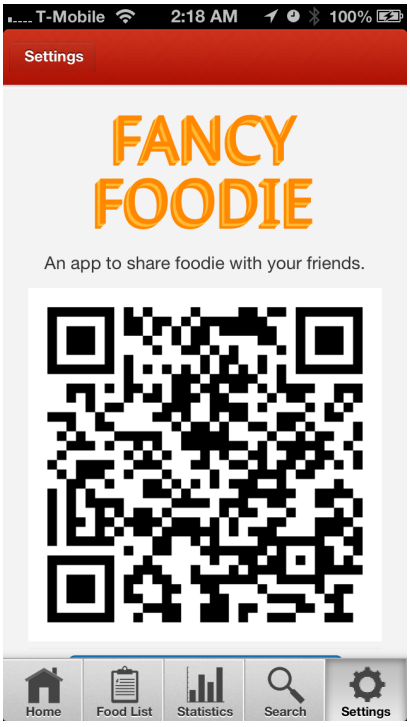
turning on the *save photo* option, it will save the photo to an album. The feedback option is used for user to submit feedback through TestFlight. TestFlight is a online tool to do open beta testing on the fly. By hooking with TestFlight, we will be able to see all the crash reports, time durations for each session, and even ask questions to users when a checkpoint is reached.

(a) Settings

(b) Save to Album



(c) Author Web-view



(d) Website View

Figure 2.5: Setting Tab View

Chapter 3

Design

3.1 Software Architecture

3.1.1 Relationship with Cocoa Touch Framework

Core Data, Core Location, WebKit, MapKit, UIKit are used in this project. As shown in Figure 3.1, NSManagedObject, NSObject, UITableViewController, UIViewController are inherited by different classes.

3.1.2 Database Schema

We used three tables for storing all the information from user. As Figure 3.2 shows, Events is the main table in the app. It provides fields "address", "comment", "creationDate", "latitude", "longitude", "locationName", "rate", "thumbnail", "photoBlob" and "tags". "address" field is used when user didn't find their "locationName" in location list which fetched from Foursquare API v2. "Latitude" and "longitude" is used for adding annotations in Map View. A 80*80 resolution thumbnail is stored for each event in order to accelerate loading food list table.

"photoBlob" has one to one relationship with "photo" in PhotoBlob table. Using a separate table should also help speeding up when we don't need to load photo while we still need to get the meta data of the event.

Field "tags" has many to many relationship with "photos" in Tag table because one photo can labeled with many tags and one tag can relate to many photos.

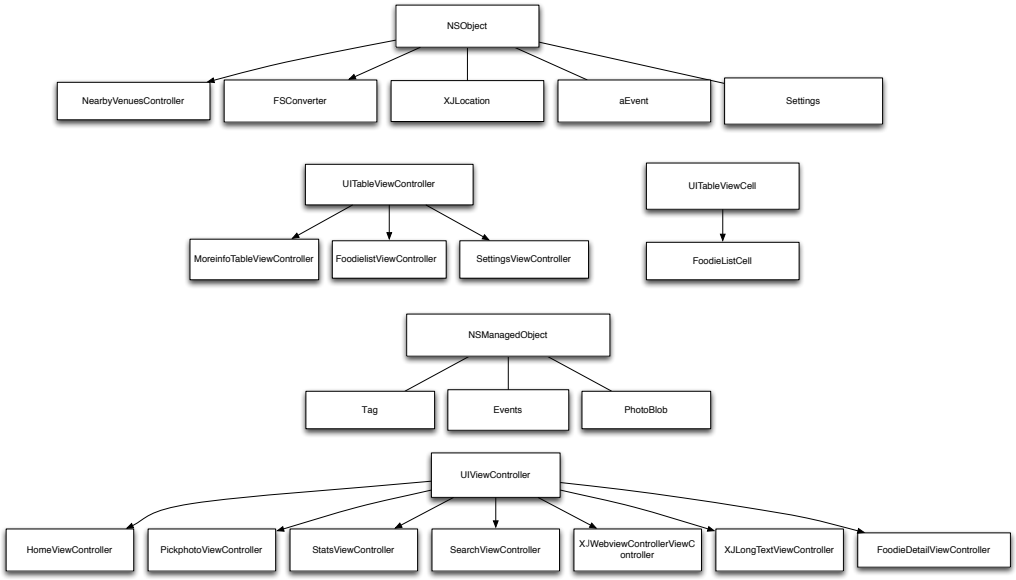


Figure 3.1: Relationships with Cocoa Touch Framework

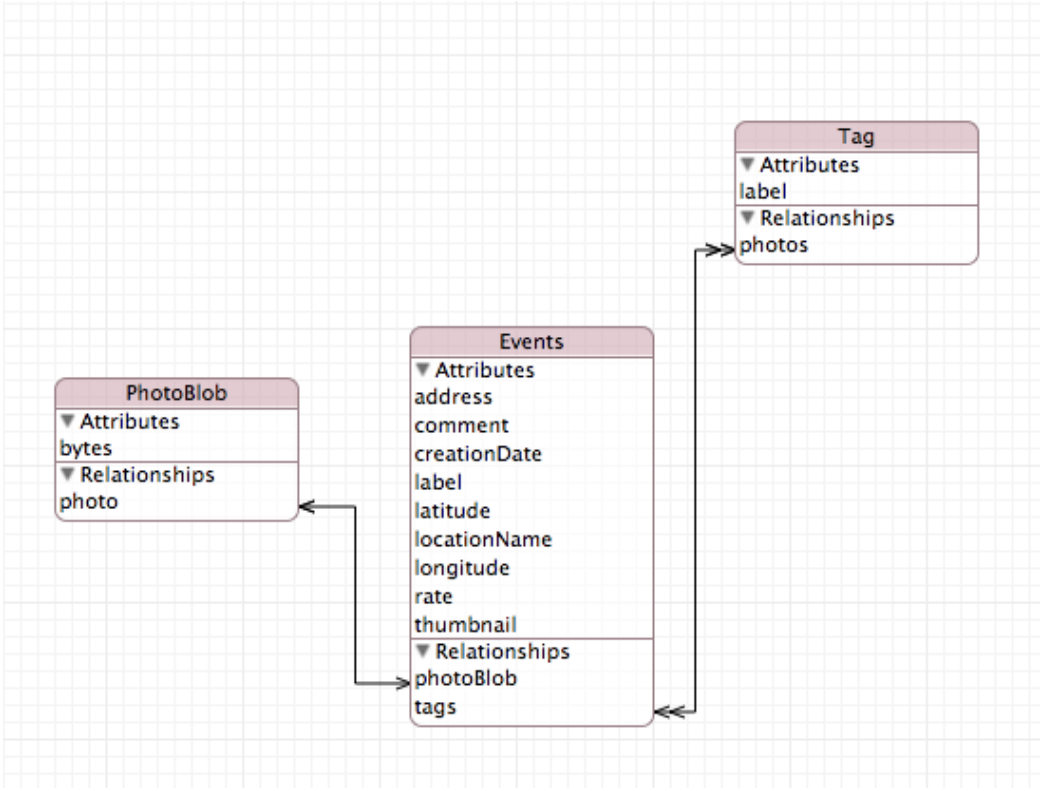


Figure 3.2: Database Schema

3.1.3 Settings Property List

When developing under iOS, we can also use another way to store the data we need. *Property List* is used to store all information related to the app itself. For instance, using *Property List* to store app display name, app version are commonly used in all apps for iPhone.

"Fancy Foodie" uses release number as main version string and git hash tag of the release as build string, so "1.0 (build 3df55bb)" is shown in Figure 2.5 which means that the main version number is 1.0 and build hash tag is 3df55bb.

This app also use *Property List* to store whether we need to save photo to album locally. If this option is enabled, the app will create an album named "Fancy Foodie Photos" and put photos when creating the event in this album as shown in Figure 2.5(b).

3.2 Modules

Figure 3.3 shows all the classes and public methods in the project.

3.2.1 View Controllers

Figure 3.4 shows main views in storyboard. In this project, we build it as a tab based application. Five tabs are created for different purposes. Home tab plays as a guide to create a food event; Food list tab shows all the events created before; Statistics tab shows all aggregated data; Searching tab is used to show all past events and searching by address to find the events; Setting tab is for configurations.

The correspond viewControllers are list as follows.

HomeViewController Control views for users to start using the app.

PickphotoViewController Present a preview of the photo chosen.

MoreinfoTableViewController Control a form with all different types of information need to create a event.

FoodieListViewController Display a table view to show all food events with thumbnail, relative date and location info. Control the logic of updating comments and rates.

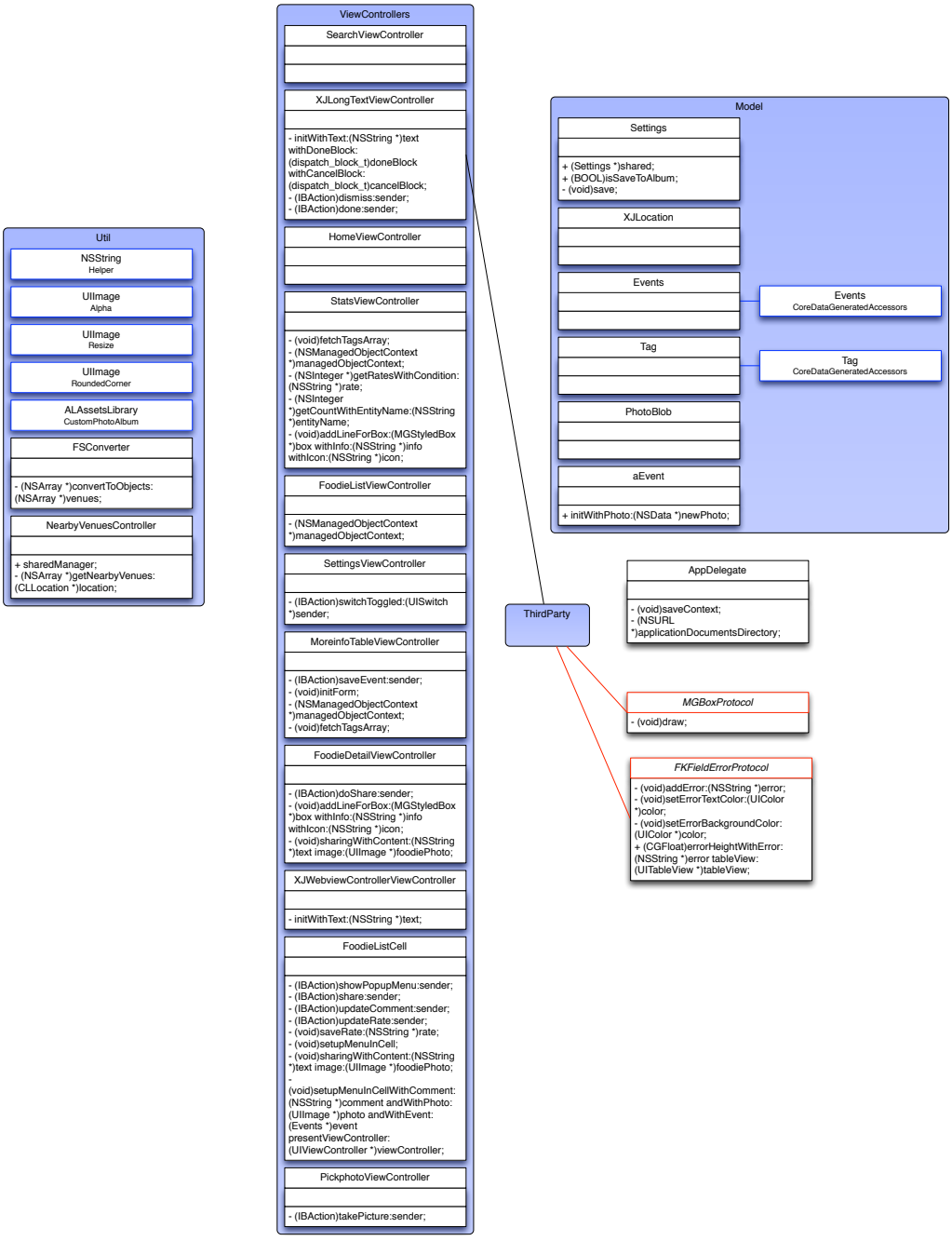


Figure 3.3: Class-diagram of the project

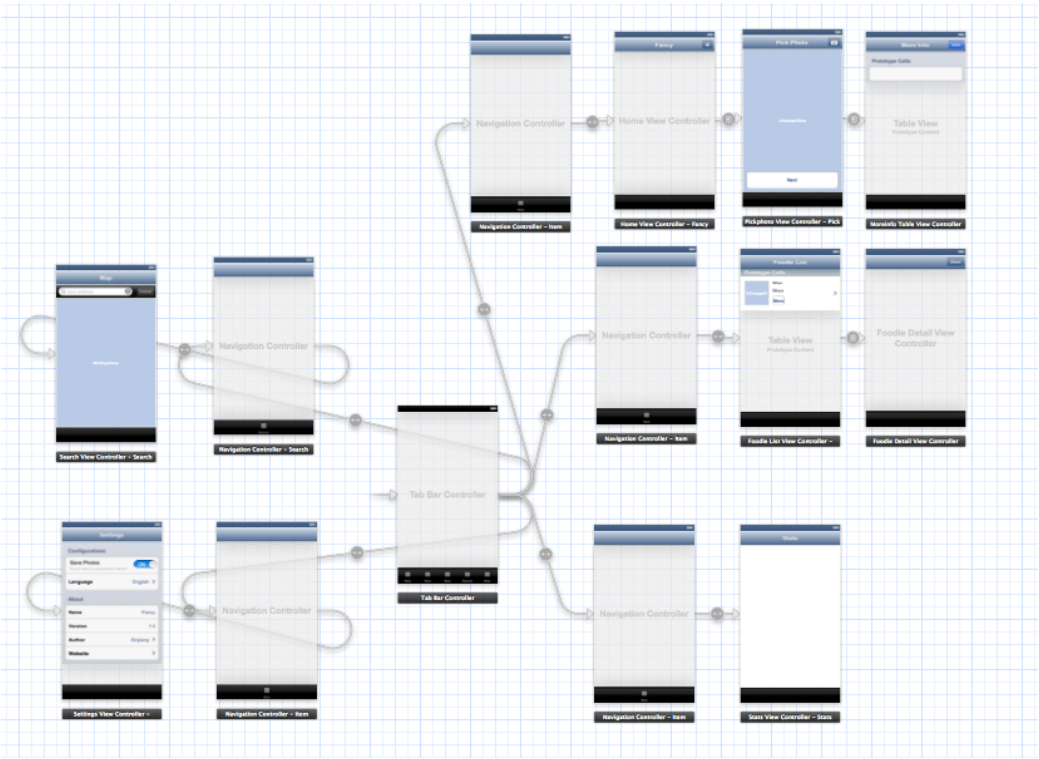


Figure 3.4: Storyboard of Overview Fancy Foodie

XJLongTextViewController Embedded a UITextView in UIViewController.

FoodieDetailViewController Display all information of a particular event.

StatsViewController Show statistics of all events.

SearchViewController Provide a MKMapView, UISearchDisplayController and UITableView in this controller.

SettingsViewController Setting Options such as "Save photo","language". App name, version, author and support website are controlled here as well.

XJWebviewControllerViewController Provide a UIViewController embedded with UIWebView.

3.2.2 Third Party Libraries and Utility

The following is a list of third party libraries used in the project.

- TestFightSDK: Beta Testing on the fly.
- BaseKit: Tools to create singleton class and better Location Manager.
- QBPopupMenu: Pop-up Menu User Interface
- CZPhotoPickerController: Preview of photos and better photo picker.
- FormKit: Form style tableview creation.
- BButton: Button with twitter bootstrap color schema.
- Foursquare2: Foursquare API version 2.
- MGBox: Table style boxes creation.
- SORelativeDateTranformer: Convert date to relative date such as "One day ago".
- FontAwesome: Adding icons to user interface.
- SVProgressHUD: Notification messages display.

A few important utility classes are listed as follows.

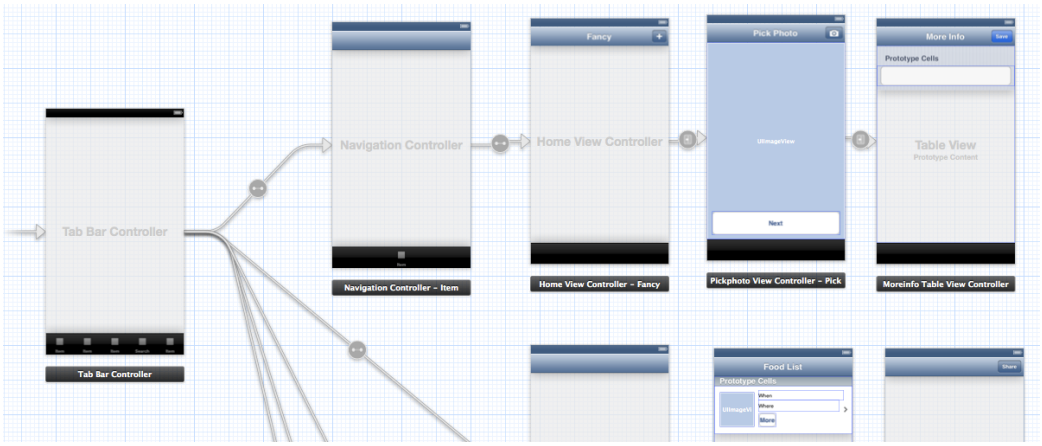


Figure 3.5: Storyboard of Home Tab View

- NearbyVenuesController: Making request to Foursquare to get a list of locations nearby.
- UIImage(Resize), UIImage(Alpha), UIImage(RoundedCorner): Tools to resize and modify images.
- ALAssetsLibrary(CustomPhotoAlbum): Saving images to custom album.
- FSConverter: Converting Foursquare feedback to an object.
- NSString(Helper): Providing convenient methods of NSString.

3.3 Key Methods

3.3.1 Insert New Event

Figure 3.5 shows the procedure to add a new event. - (IBAction)saveEvent:(id)sender; in class MoreinfoTableViewController is called when a user finished editing all information in the form. First of all, we checked whether the save photo option is on or not. If it is on, we should save the image to an album named "Fancy Foodie Photos". Otherwise, go ahead and create an instance of Events called newEvent. Afterward, we assign all the information from user interface and save. Next step is to use the navigationController to pop to root view controller which is HomeViewController.

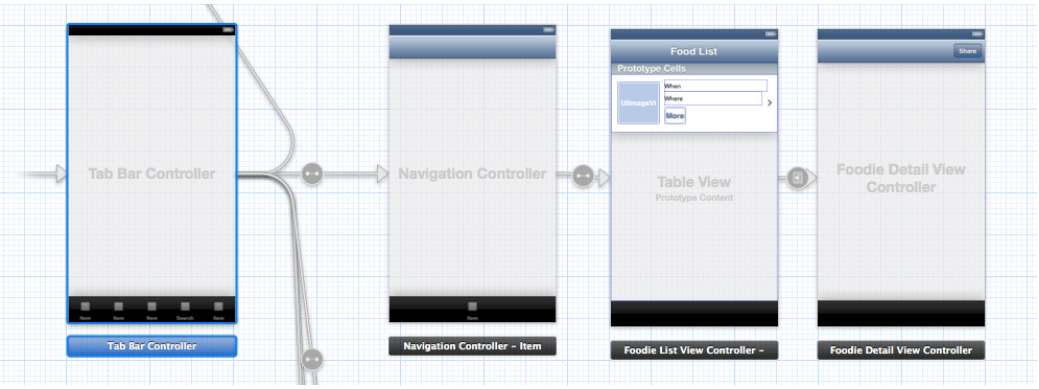


Figure 3.6: Storyboard of Food List Tab View

3.3.2 Update Event

Updating event is actually updating rate or comment. When a user tapped on the pop up menu and chose comment option in Figure 2.2(b), - (IBAction)updateComment:(id)sender; in class FoodieListCell is called. Class XJLongTextViewController is used and presented as a modal (Figure 2.2(c)). After typing in new comment and done button is tapped , it uses NSManagedObjectContext to save the event and stored the updated data in the local database.

3.3.3 Delete Event

Delete event is very simple in this app. In food list tab, you'll see a list of events. Swiping from right to left enable the delete button as shown in Figure 2.2(i). After tapping on the button, - (void)tableView:(UITableView *)tableView commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath *)indexPath; in class FoodieListViewController is called. In the method, it uses NSManagedObjectContext deleteObject to delete the event and save the result in local database after save function is called. Afterward, current tableview is refreshed as well.

3.3.4 Searching Events

Searching events is done with MapKit API. All the events are fetched in during loading time. The events are annotated with red pins when the tab is loaded.
As shown in Figure 2.4(b), as the user is typing in the address, the searching string is passed to -(BOOL)searchDisplayController:(UISearchDisplayController

*)controller shouldReloadTableForSearchString:(NSString *)searchString; , so in that function, we're using CLGeocoder to guess the possible locations for that string. The possible locations are displayed in a table view. After the use choose one possible location, the central region will be focused on that area. At this time, the nearby events stored in database will appear in front of the user.

Chapter 4

Testing

4.1 Unit Test, Integration Test and System Test

All the modules are tested individually in separate Xcode project, and integrated into our main project *Fancy Foodie* afterwards. After making sure the features are integrated with the main project, we build the project nightly, and test the function systematically with both simulators and real cellphone whose model is iPhone 5.

4.2 Beta Testing

Fancy Foodie use TestFlight to do beta testing on the fly. TestFlight provides a SDK for us to integrate it to the app. After integration, crash reporting will happen automatically. By logging in the dashboard of TestFlight, we could see all the information for each session of users as shown in Figure 4.1.

4.3 iTunes

Fancy Foodie is created in iTunes Connect on April 17, 2013. First submission was uploaded on April 17, 2013. But minor changes are made after that, so another submission was upload on April 23, 2013. Currently, *Fancy Foodie* is waiting for review. After review, https://itunes.apple.com/us/app/fancy-foodie/id638036832?ls=1mt=8 should be available to U.S. users. Anyone could download the App via that link.

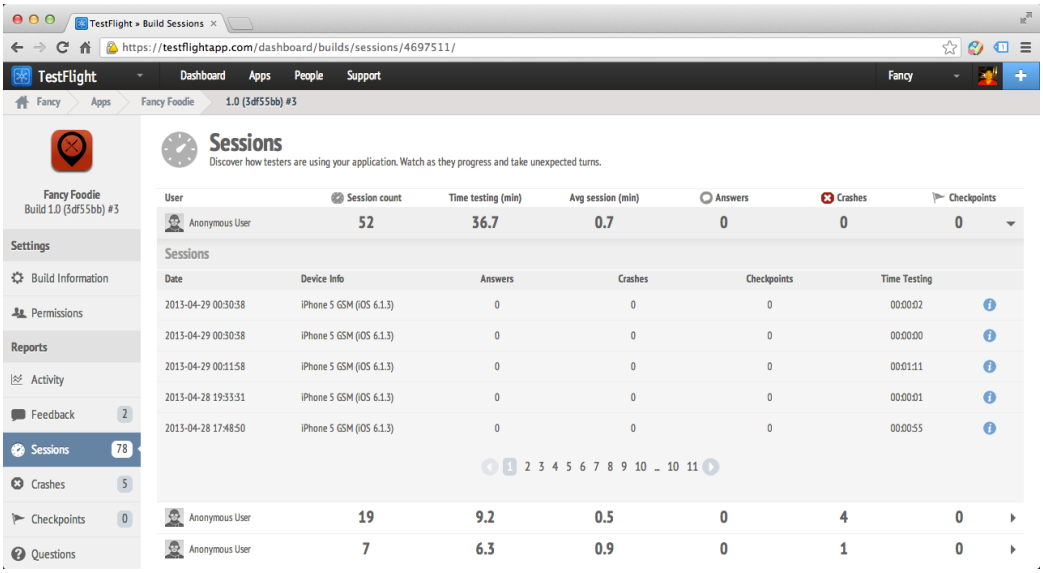


Figure 4.1: TestFlight Dashboard

4.4 Future Testing

Since it is ongoing project, future testing will be done by using TestFlight as well. We would set up checkpoint and create questionnaire to get better feedback from users.

Chapter 5

Conclusion

This project is successfully finished in time with good functionalities overall. It could provide users to take picture of food they like and create tags, location information, rates, and comments to the food. At the time, it provides convenient methods for users to share those food with their friend by social media such as Facebook, Twitter, Weibo etc.. A map view and statistics view are created for users to have better understanding of their preferences.

There are still more features could be done in future. Since Yelp is a good resource of restaurant information, the app could pull more information from Yelp in order to give users more about the place they eat at. A central database is needed to making users communicate with each other such as making comments to friends food inside of the app. Chinese is my mother tongue, so I'm hoping to adding multi-language feature for the App. In this way, more people could be able to use the App. Last but not least, a larger pool of testers should be included in beta testing process.

Acknowledgments

I would like to thank open source community. Without them, I probably need much more time to finish this project.

I would also like to thank Jakob Eriksson and Ugo Buy. Prof. Eriksson gave me a lot of valuable suggestion for my project. Prof. Buy also provided helpful suggestion on the report. Both Prof. Eriksson and Prof. Buy give a good evaluation of my work.

Special thanks to all my friends who made tremendous comments about my app and all my colleagues in EXACT Sports LLC who slow me down in the right way so that I can make the project better.

Xinjiang Shao
April 2013
University of Illinois at Chicago

References

[1] iOS Cocoa Touch, https://developer.apple.com/technologies/ios/cocoa-touch.html

[2] Apple Developer Center, https://developer.apple.com/devcenter/ios/index.action

[3] Cocoa Controls, https://www.cocoacontrols.com/

[4] AppCoda: iOS Programming Course, http://www.appcoda.com/ios-programming-course/

[5] Code for App, http://code4app.com/