FOOD? : Streamlined meal planning and invitation application for iOS

Dominic Rocco
Computer Science Department
Brown University
Providence, RI 02912
Dominic_Rocco@brown.edu

May 15, 2013

Abstract

Figuring out when to get lunch, where to go, and who to eat with is a common dilemma that plagues the average nine-to-fiver every weekday. Workers resort to emailing, texting, and chatting up their friends in an attempt to ascertain where people are going and when they are free. Internet searches are sometimes run to find the best spot to satisfy new cravings. This flurry of activity is often disorganized and ad-hoc, which can lead to people being accidently left out, or plans getting mixed up. This problem is not just pertinent to everyday workers; some college students face this predicament for nearly every meal and even friends from outside the workplace might wish for a better way to plan get-togethers for dinner or drinks. This is where FOOD? aims to help. FOOD? is a streamlined meal planning and invitation application for iOS devices. It allows events to be created, friends to be invited, and restaurants to be researched and chosen with just a few taps. By combining all of the steps into one simple application, deciding Thursday's lunch plans should be a breeze.

1. Introduction

FOOD? is a streamlined meal planning and invitation application built for iOS devices. The application is centered on food outings. Users can create their own outings or accept invitations to outings their friends created. Each outing can be optionally associated with a date-time, and a restaurant; null values are assumed to be determined later. Restaurant details can be viewed in-app and local restaurants can be searched for by name, dish, or genre. By combining event details, restaurant information, and an invite system, FOOD? makes planning an outing easy.

2. System Architecture

FOOD? was built for any iOS device, but was designed to look best on iPhones or iPod Touches. It uses a MySQL database to save event details and relationships between users and events, a PHP server is used as an interface between the iOS application and database, Facebook integration is used for user identification and access to the user's friends, and Yelp integration is used for retrieving restaurant details and locating nearby restaurants.

- Back-end DB: MySQL [1]
- Web server: PHP server [2]
- Front end: Cocoa Touch [3]
- User credentials: Facebook integration [4]
- Restaurant Database: Yelp Api [5]

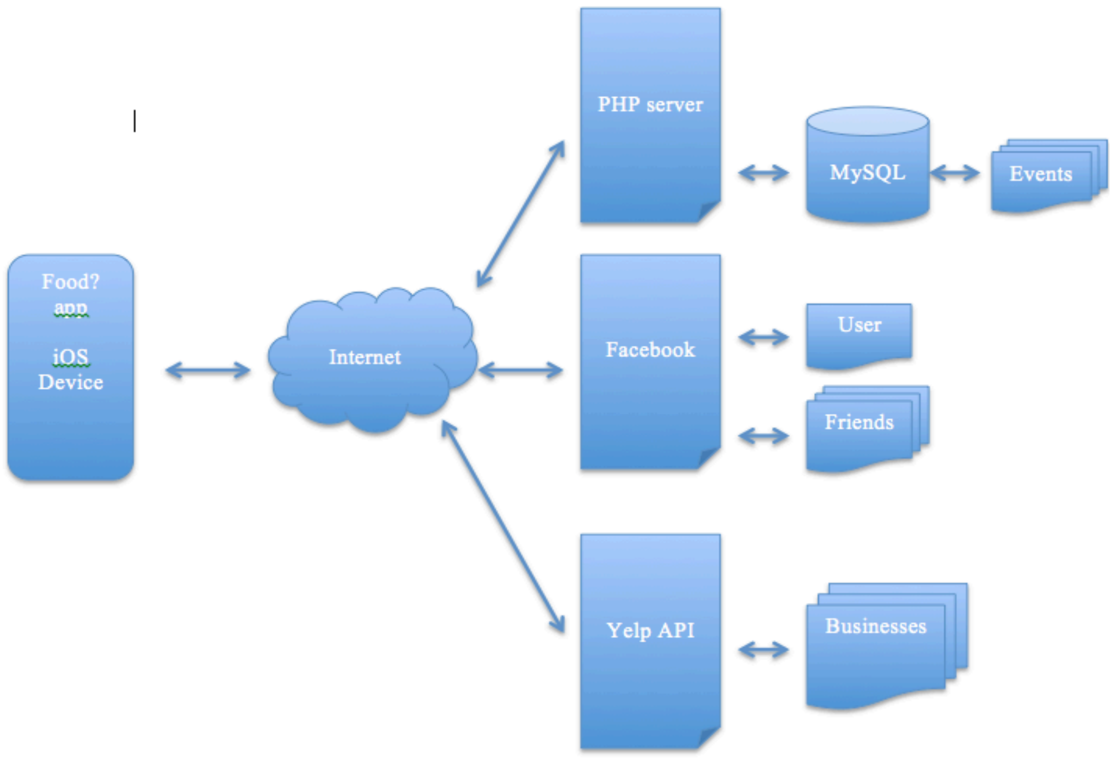


Figure 1. System Architecture

3. Facebook and Yelp Integration

3.1 Facebook Login

Building a user base and establishing a web of relationships between users is a hard task. Friends are slow to make accounts and the absence of relationships between users makes for a boring experience. If you have no friends to invite to your events, the application becomes much less useful. Using Facebook login solves some of the adoption problem. As long as the user has an active Facebook account (which encompasses a large majority of the target audience), the user will be able to start inviting friends from the moment they log in. The user's current Facebook friends become the friends they can invite using the FOOD? application. Facebook integration also allows for profile pictures to be integrated into the application, improving the user interface.

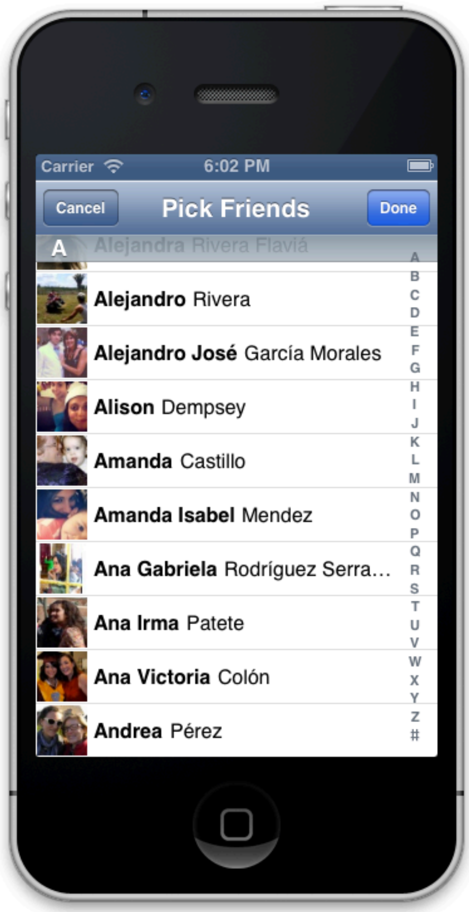
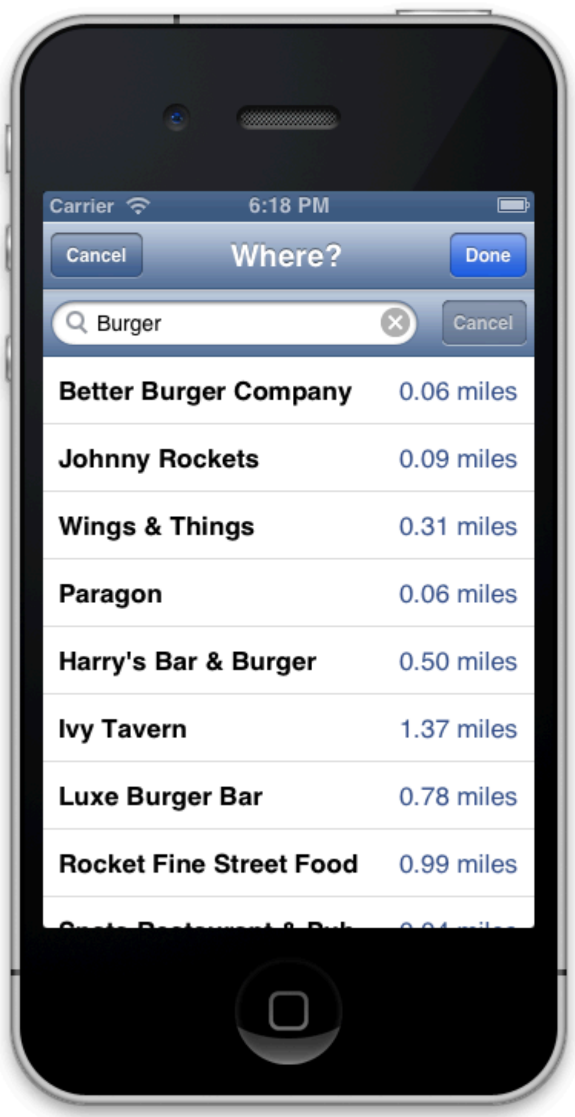


Figure 2. Facebook Integration

3.2 Yelp Integration

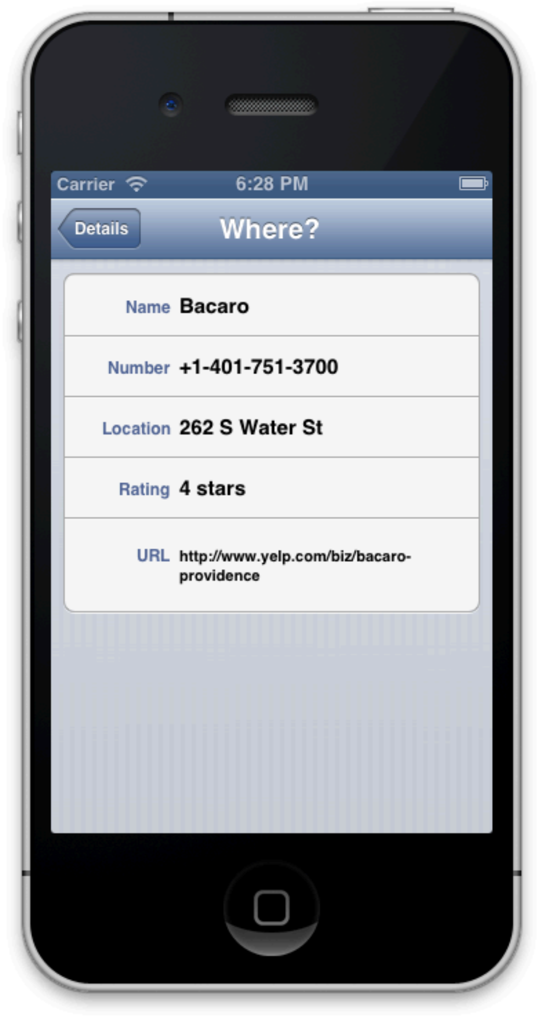
Another feature of FOOD? is Yelp integration. Simply writing out the name of already known restaurants or food genre descriptions would create a boring application. Utilizing Yelp's massive database of restaurants, bars, and businesses creates a richer user experience. The user doesn't just specify sushi as the destination, he is able to perform a location based search for sushi and find the highest rated and nearest restaurants.



Better Burger Company	0.06 miles
Johnny Rockets	0.09 miles
Wings & Things	0.31 miles
Paragon	0.06 miles
Harry's Bar & Burger	0.50 miles
Ivy Tavern	1.37 miles
Luxe Burger Bar	0.78 miles
Rocket Fine Street Food	0.99 miles

Figure 3. Yelp Search

Yelp also provides more details about the restaurant. The application is able to list the restaurant's address, phone number, and Yelp URL.



Name Bacaro
Number +1-401-751-3700
Location 262 S Water St
Rating 4 stars
URL http://www.yelp.com/biz/bacaro-providence

Figure 4. Yelp Detail View

4. Trace of Execution Sample

To provide more depth into the internal workings of the application, I'll walk through a sample interaction a user might have with the application and what happens under the hood.

When the user first opens the application, an active session is made with the Facebook servers. If the user has logged into Facebook through the application before, Facebook will authenticate the application without any prompt from the user, however, if the user is opening for the first time, the application will start by prompting the user to log into Facebook and give the application permission to access basic profile information and friends list.

Once an active session with Facebook is established, the application uses the user's Facebook ID to make an asynchronous query to the PHP server for any events that he/she is either the creator of, is attending, or is invited to. When the query completes these, results are processed by the application: the date strings are parsed back into Date objects, events are sorted by status and time, and date strings are created in a human friendly format. After this processing happens in the background, the UI is updated back on the main thread and the results are displayed to the user.

If the user were to now click on an event he created, he is taken to a new view that displays more information on the event. This information is still cached from the previous request, but can be updated if the user scrolls the view up. If the user is now curious about who is coming to the event, he can click on the table cell listing the attendees. This loads another view that sorts the attendees and displays them in a table. This table includes all of the users' profile pictures, which are attained through asynchronous calls to Facebook servers.

After the user notices that one of his good friends is not included on the list, he return to the event screen and clicks to edit the event. This brings up a new view that allows him to edit any field. He chooses to add his friend and this brings up a new table view (similar to the previous one) that lists all his Facebook friends. He finds his friend and clicks to add. This brings him back to the edit page where his friend's name now appears on the details.

Seeing that the restaurant chosen is a burger joint, he realizes he will have a problem. The friend he just invited is a vegan. He decides to update the location. He clicks the location field and a new view with a search bar appears. He types in Vegan and hits search. The phone now turns on the GPS, finds his location to the nearest 100 meters, authenticates a new session with the Yelp API using OAuth1.0 protocol, and then sends a GET request to Yelp with the search query, location, and authentication token. When this query returns, the table view is now filled with 20 locations for the user to choose from. The user picks the first result, Kabob & Curry, because he has visited the restaurant once before and liked the meal. He now clicks done and returns to the edit view.

When the user now clicks on save, he is returned to the previous view of all events, and a background thread is started that will update the PHP sever to update the database. The background thread determines what has changed from the previous event and the new event (location and one friend added). It generated a request to the PHP server asking it to update the event and waits to see if the request was accepted.

The user's friend becomes hungry and opens his application. He now sees that he has been invited to eat at Kabob & Curry. He clicks on the invite to see who else is coming, and decides he would like to go. He presses the "accept" button and another asynchronous call is made to the PHP server to change his status from invited to attending.

Everyone now meets at the specified time and enjoys a nice meal together.

5. Work Process and What I Learned

Before starting this project, I had zero knowledge of objective-C, and had never set up my own webserver or a program with a GUI from scratch. Moreover, I had never had to pull old libraries from Github, alter them, build them into a framework, and add that into my project in order to get a certain protocol working. I know have a great deal more experience in all of the above and more.

Working on this project outside the strict confines and guidance of a typical CS class has helped me learn a lot of practical skills. I am better able to work my way through documentation, search the web, and skim examples to help me through a problem I am having.

Teaching myself objective-C was the first step of my project. Already having a background in C, this wasn't too hard of a challenge. The hardest part for me to grasp was dealing with reference counting. Knowing when to retain, release, or autorelease objects was hard in the beginning but I soon grew to know when to apply each. It was not long after I got the hang of reference counting that I was introduced to Apple's ARC (Automatic Reference Counting) and I realized most of what I learned was no longer pertinent, but I am still glad to have a better understanding of the inner workings of the language.

I next embarked on figuring out the Cocoa Touch and the iOS UI. I spent a good deal of time trying out different layouts, views, and different animations between them. After some feedback from users, I began to realize that less was often more. A simple, streamlined, UI was more approachable than a customized one. Apple's built-in features and controls are more familiar and comfortable for the users since they know how it works on most applications and expect yours to do the same.

Once the UI was set up with static/fake data. I purchased some web hosting and started to set up my PHP server. This was a pretty simple server, just a bunch of PHP files that read in request parameters and executed them through PDO objects on a MySQL database. In the future, I would like to add some authentication to ensure that only the application is utilizing the server, but as of now it is set so that anyone can call the methods (which I know is not optimal).

After getting the PHP server up and making calls to get "real" data, the application became slow and cumbersome. I was making all the calls synchronously, and never cached much data locally in the app. The app was usable, but not enjoyable to use. I realized, from reading some documentation, that I needed to relegate any slow calls or long processing to background threads so that the app didn't become unresponsive.

After getting the background threads up and running, the application was much more responsive and easy to use, but it kept crashing randomly. I soon realized that iOS UIkit is not thread-safe and that I needed to only update the UI from the main thread. This involved even more restructuring of the code but before long the application was back into a stable working state.

The last and one of the hardest steps in the project was getting an OAuth authentication library working so that I could integrate with the Yelp API. All of the initial libraries and frameworks I tried failed. Some just didn't work, others didn't support newer iPhones, and some just refused to link. After many, many hours of searching, I was able to find an outstanding pull request for one of the frameworks and I looked through the changes he made. I followed his example and was able to alter one of the framework sources I had downloaded so that it would compile and run with my iPhone and iOS6.1.

After getting OAuth working, the remainder of the project was just integrating the data I received from Yelp into the UI and back-end, and then squashing any bugs I found along the way. Overall I feel that this project has given me more “real-world” experience than most other school projects. Working from scratch, adjusting as I go, and piecing together different frameworks, APIs, and libraries to get the job done correctly, has taught me a great deal.

6. Future Work

While FOOD? is in a working state, there is always room for improvement. Being built on the Cocoa Touch framework, the application already has a smooth and attractive interface, but there are still areas that I would like to improve. FOOD? needs a logo to be used for the application’s home screen icon, launch image, and throughout the application as branding.

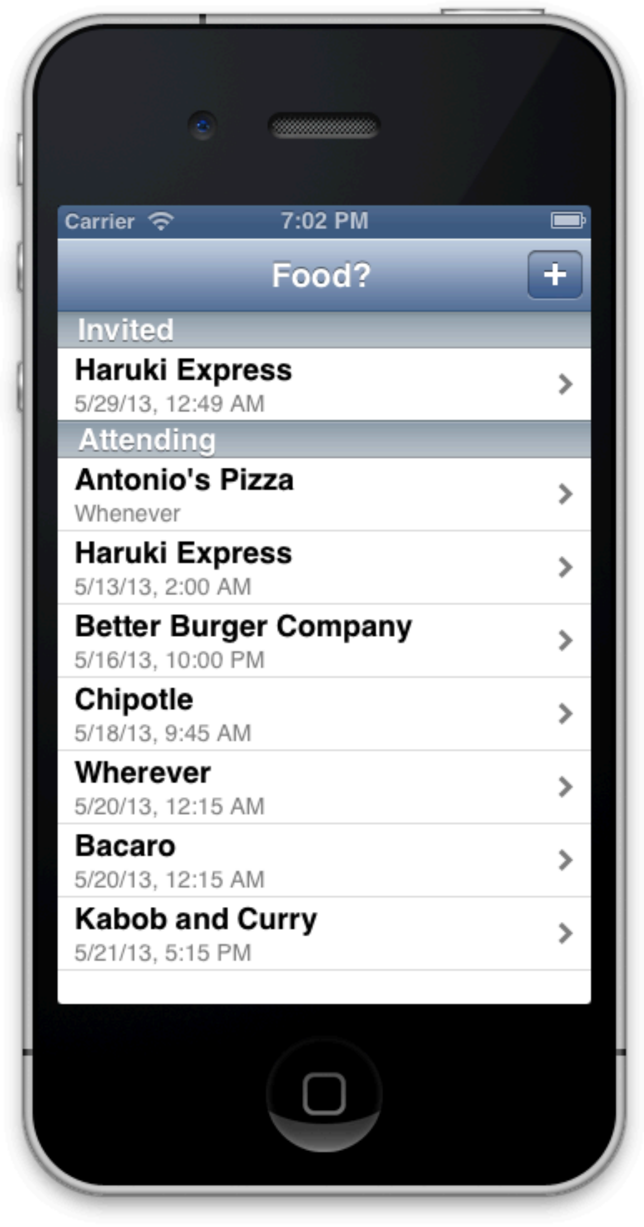
The application is fairly responsive due to fetching all URL requests in background threads and utilizing many of Cocoa Touch’s features (such as scroll down up to reload data), but I would like to tie in some more functionality. The user should be able to tap the phone number on the restaurant detail screen and have the phone ask to dial the number. I would like to also enable push notifications of event invites, but this would require an upgraded server and some certificates from a Certificate Authority.

Another feature that I intend to implement in the near future is the handling of event suggestions. I would like for users who are invited to events (not the creator of the event) to suggest time or location changes and to be able to suggest more friends to invite. The back end for this is already set up on the server but I haven’t designed a user interface for viewing, aggregating, and accepting/denying suggestions that I find acceptable. I believe some custom views and pictures might be needed to attain the interface that I want.

Before releasing that application to the public, I would also need to figure out locations to add required logos of Yelp. Yelp requires active branding so that users are aware of where data is coming from. Yelp also returns deals and specials in some of their search results. I would like to be able to check for deals when a restaurant is chosen and notify the user in case he/she wants to use them.

6. Acknowledgments

I would like to thank Steve Reiss and members of Fall 2012's class of "Human Factors and User Interface Design" who gave me helpful feedback on my user interface while it was evolving and even helped me decide which features would actually be useful and which were just superfluous.



Haruki Express
5/29/13, 12:49 AM

Antonio's Pizza
Whenever

Haruki Express
5/13/13, 2:00 AM

Better Burger Company
5/16/13, 10:00 PM

Chipotle
5/18/13, 9:45 AM

Wherever
5/20/13, 12:15 AM

Bacaro
5/20/13, 12:15 AM

Kabob and Curry
5/21/13, 5:15 PM

Figure 5. Main Event View

7. References

[1] MySQL `http://www.mysql.com/`

[2] PHP `http://us.php.net/`

[3] Cocoa Touch `http://developer.apple.com/technologies/ios/cocoa-touch.html`

[4] Facebook SDK `http://developers.facebook.com/ios/`

[5] Yelp API
`http://www.yelp.com/developers/documentation/v2/overview`