Bryan Gaston. What’s Open: a native iPhone app for finding nearby restaurants organized by open hours. A Master's paper for the M.S. in I.S. degree. April, 2013. 33 pages. Advisor: Paul Jones

Geolocation services in mobile devices provide the opportunity for software developers to write apps that personalize information based on the user’s location. By utilizing GPS, cellular tower location, and information about nearby wireless networks, mobile apps can determine the device’s location and use this information for a variety of purposes. Web-based services such as Google Places provide an easy way to find information about establishments, including their locations, user reviews, price range, and more, and by searching with an app like Google Maps for iPhone, users can find which establishments are near their current location. While there are many iPhone apps intended for finding nearby restaurants and other establishments, there are not many that focus strongly on the open hours of the restaurants. “What’s Open?” is an iPhone app designed to help people find nearby restaurants, based on their open hours, and to view detailed information about them.

Headings:

Location-based services

Mobile device application development

Mobile device applications
WHAT’S OPEN: A NATIVE IPHONE APP FOR FINDING NEARBY
RESTAURANTS ORGANIZED BY OPEN HOURS

by
Bryan Gaston

A Master’s paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina
April, 2013

Approved by:

________________________
Paul Jones
Table of Contents

I. INTRODUCTION 2

II. PROJECT GOALS 3

III. GATHERING DESIGN FEEDBACK 4

IV. DESIGN DECISIONS 7

V. CHALLENGES 17

VI. FUTURE WORK 21

VII. LESSONS LEARNED 23

VIII. CONCLUSION 24

IX. SOURCE CODE 25
I. Introduction

Location awareness in mobile devices has facilitated the development of numerous location-based apps. From movie theater locators\(^1\) to running route mappers,\(^2\) scores of location-based iPhone apps have emerged. The tool that makes this possible is Apple’s Core Location framework, which gives developers access to geolocation services on the iPhone. The iPhone uses several different geolocation technologies, including Global Positioning System (GPS), cell tower triangulation, and Wi-Fi Positioning Service (WPS), providing location accuracy within a few meters in some cases (Mark, Nutting, & LaMarche, 2011).

One case in which location-based searching is particularly useful is when trying to locate food establishments nearby. There are many apps for restaurant searching, and some of the most popular apps are Yelp\(^3\), Urbanspoon\(^4\), and Google+ Local\(^5\). All of these offer the ability to find nearby restaurants and view detailed information about them. Although both Yelp and Google+ Local allow filtering by the restaurants that are open currently, neither app is built around the idea of time as the primary search component. Google+ Local also suffers from a poorly designed, text-heavy user interface that lacks the visual appeal of some other apps. As a night owl, I often want to find restaurants that

---


are open late at night, so being able to see only the restaurants that are currently open is very useful. Also, I regularly wonder if a restaurant will be open later on the same day (i.e. I’m supposed to meet a friend for dinner, but what’s open on a Monday night?), so being able to quickly find restaurants that are open later in the day, even if not currently open, would be very helpful. With these ideas in mind, I set out to create “What’s Open,” a simple iPhone app for finding nearby places that are open now or open later today.

II. Project Goals

I had several goals when I began the project. My primary goal was to greatly improve my Objective-C programming skills through designing and developing a working iPhone app that could help people find nearby restaurants organized by their open hours. In addition, I had several general design requirements that guided my development of the app. I wanted the app to be simple but useful. It had to include enough features to make choosing among restaurants possible, but I wanted to ensure that it would not feel crowded with unnecessary features. Beyond these general guidelines, I considered requirements for the data source and detailed requirements for the user interface.

Data Source

Since gathering detailed data for all restaurants in Chapel Hill would be a daunting task in itself, much less gathering data about restaurants across the country, I wanted to use an existing database of restaurants. Furthermore, I did not want to host the data on a server of my own because that would mean needing to maintain the data and ensure server uptime. My app would need to pull data from a national or international
restaurant database that included information about hours of operation, was kept up-to-date, and had an Application Programming Interface (API) to allow me to access it.

**User Interface**

My main concern with the user interface was that it be simple, intuitive, and clean. This meant avoiding clutter, using controls that were familiar to iPhone users, and displaying text that was easy to read. Since I would be displaying different categories of restaurants (those open now and those open later on the same day), I would need to separate these in a way that provided easy navigation.

**III. Gathering Design Feedback**

To gather feedback about the features and design of “What’s Open,” I recruited a small number of focus group participants from the University of North Carolina at Chapel Hill’s School of Library and Information Science (SILS). All participants were students pursuing a Master’s degree at SILS. We informally discussed the functionality and design of the app, following the Focus Group Guide presented in the appendix. The discussion was based around five main topics: 1) use cases for the app, 2) types of food establishment to include, 3) most important restaurant attributes to display, 4) design of the user interface, and 5) additional features requested. Below, I present a summary of the participants’ feedback for each discussion topic.

**Use Cases**

Participants had several ideas about when the app would be helpful. The suggestion that sparked the most discussion was using the app to find restaurants when
traveling in an unfamiliar area. Another suggestion was to use the app for finding open
restaurants during commute times to and from work when it would either be time for
breakfast (a meal that some restaurants do not serve) or slightly before normal dinner
time. An additional suggested scenario was using the app at times when the nearest
restaurant is several minutes away and it is particularly important to know whether the
restaurant is open or not before spending time traveling to the restaurant.

**Types of Food Establishments to Include**

There was disagreement regarding which types of food establishments to include
in the app, which made it clear that allowing users to filter by types of food
establishments is important. Participants all agreed that grocery stores should not be
included unless they serve prepared food as well as selling groceries. However, they
were torn when it came to the decision about whether or not to include bars and dessert-
only establishments. They suggested separating the food establishments into separate
classes, such as those appropriate for a meal, a dessert, or only a drink. Additional
comments underscored the importance of separating bars and dessert-only establishments
from other food establishments in the app.

**Restaurant Attributes to Display**

I presented the focus group participants with a list of all restaurant attributes
available through Factual’s API in the Restaurants table⁶ and asked for feedback on
which they found the most and least helpful to include in the app. Contact information,
such as street address, phone number, and website were, not surprisingly, their top
choices. Other attributes that they generally agreed were important were cuisine, cash-

---

only payment required, availability of free parking, rating, and whether or not the restaurant serves alcohol. The availability of locally sourced food, though not currently an attribute available through Factual’s API, was also suggested as an attribute to include. Participants had mixed opinions about whether it would be useful to include price level, especially for restaurants that they already knew. Likewise, they were torn on whether or not restaurant reviews were necessary to include. Suggested attire and the availability of private dining rooms were among the attributes that participants found least important.

After discussing which attributes participants thought were most important to include in the app, I asked which they would like to use for sorting or filtering restaurants. Although participants disagreed over the importance of displaying price level, they did generally want the ability to sort restaurants by price level. Sorting or filtering by rating, whether cash payment was required, availability of free parking, availability of alcohol, and whether the restaurant offered takeout were all generally agreed upon.

**Design of the User Interface**

Participants provided an abundance of feedback about how restaurant information should be displayed in the app. Since some people will use the app when they are unfamiliar with restaurant names, participants requested that cuisine information and distance be easy to access when viewing a list of restaurants. When viewing a page of information about an individual restaurant, contact information and a map should be near the top of the page. They considered ratings only moderately helpful but thought that they should be displayed on the restaurant details page. Information about open hours should be prominent, and directions should also be easily accessible. They considered
menus very important to include. Photos of the restaurant and its food were also suggested.

**Additional Features**

At the end of the focus group session, I asked participants which other features they would like in the app. There were not many suggestions, but two involved providing feedback about the restaurant data. Participants discussed a feature whereby users could photograph a sign displaying restaurant hours and upload it to add or correct hours for a restaurant listing. Another feature suggestion related to giving feedback about restaurant data was the ability to flag misclassified restaurants. In addition to these features, participants discussed a feature for saving restaurants in a list of “favorites.”

**IV. Design Decisions**

Focus group participants’ feedback was helpful in guiding my design of the app. Luckily, most of their suggestions were possible for me to implement in this version of the app. Below, I discuss the major design decisions involved with developing “What’s Open.”

**Data Source**

I conducted an internet search of APIs that provide access to restaurant data and found several options. Unfortunately, some of them do not offer information about the open hours of the restaurants. The chart below summarizes the APIs that I considered using, which do include restaurant hours.
Chart 1: Comparison of Restaurant Data Providers

<table>
<thead>
<tr>
<th>Data Provider</th>
<th>Max Read Requests (authenticated)</th>
<th>Max Results/Page</th>
<th>How to Request Business Hours</th>
<th>Hours Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>CityGrid (Places API)7</td>
<td>10 million per month</td>
<td>50</td>
<td>2 queries: Search + detail search with place ID</td>
<td>Non-standardized</td>
</tr>
<tr>
<td>Factual (Restaurant table accessed via Places API)8</td>
<td>10,000 per day</td>
<td>50</td>
<td>1 query</td>
<td>Standardized</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: hours': {'tuesday': ['10:00','14:00'], ['17:00','22:00']}</td>
<td></td>
</tr>
<tr>
<td>FourSquare (Venues API Endpoint)9</td>
<td>5,000 per hour (userless requests)</td>
<td>50</td>
<td>2 queries: Search + venue detail search with venue ID10</td>
<td>Standardized</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Also includes &quot;isOpen&quot; key and a localized, human-readable hours format11</td>
<td></td>
</tr>
<tr>
<td>Google (Places API)12</td>
<td>100,000 per day</td>
<td>20</td>
<td>2 queries: Search13 + detail search14 with place ID</td>
<td>Standardized:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Example: opening_hours: { &quot;open_now&quot; : true, &quot;periods&quot; : [{ &quot;close&quot; : { &quot;day&quot; : 1, &quot;time&quot; : &quot;0200&quot; }, &quot;open&quot; : { &quot;day&quot; : 0, &quot;time&quot; : &quot;1100&quot; } ]}</td>
<td></td>
</tr>
</tbody>
</table>

CityGrid is generous in its allowance of 10 million requests per month, so I considered using their Places API as the source of my app’s restaurant data. Unfortunately, I found the non-standardized format for the open hours to be a serious hassle. Results included strings of text referring to hours on ranges of days, as well as oddities including line breaks and non-matching parentheses. Parsing the data from this field would have been a major undertaking, so I decided that CityGrid would not be a good choice for my app.

FourSquare provides access to very detailed venue data through its Venue API Endpoint, and its cap of 5,000 requests per hour\(^\text{15}\) is quite permissive. Of the four APIs that I considered, FourSquare’s has the most robust business hours information, providing it in both machine-readable format and localized, human-readable format. Furthermore, an “isOpen” key indicates whether or not the venue is open right now, and an “includesToday” key indicates whether the list of open times includes any on the current day. Unfortunately, several test searches around Chapel Hill returned a number of restaurants with misspelled names and other inaccurate information. Although I was very pleased with the FourSquare’s data format for open hours, I realized that I would need to validate the restaurant names with another service to clean the results before displaying them to app users. FourSquare’s policies explicitly prohibit this usage, though.\(^\text{16}\) Despite all of FourSquare’s promising features, I decided not to use it as my restaurant database.

Factual’s Places API is the only one of these four that provides access to restaurant business hours in the initial query results. With other APIs, I had to conduct an


initial search for restaurants nearby and then additional searches for each returned restaurant to determine the business hours. For a list of 50 results of nearby restaurants, this meant conducting the initial proximity query to gather the closest 50 restaurants, parsing the JSON (JavaScript Object Notation) response and saving the row identifier of each restaurant, and then issuing individual queries for each restaurant by row identifier in order to retrieve the business hours for the restaurants. With Factual’s Places API, I can simply request a list of the closest 50 restaurants, and their opening hours are already included in the response. This dramatically reduces the number of read requests that I need to issue, allowing me to support more users since I am in less danger of reaching the daily query quota. It also means less loading time for app users. Since acquiring business hours for nearby restaurants was far easier with Factual than with the other APIs I considered, I chose Factual as my primary data source for the app.

Although Factual was a good choice for my app in terms of acquiring business hours for a large number of restaurants in only one query, the downside was that it does not currently provide photos, reviews, or menus for restaurants. Focus group participants did not value reviews as highly as other restaurant information, but they did say that menus were important to include, so not being able to access menus directly through Factual’s API was discouraging. Furthermore, the app might look bland without restaurant photos, so I needed some way to acquire photos.

For photos, I chose Google’s Places API. The Google Places API provides access to a fairly robust set of restaurant (and other establishments) data, including standardized restaurant hours, ratings, reviews, photos, and other information. Google permits up to 100,000 API requests per day for developers who verify contact information with a credit
card (though Google does not currently charge for use of its Places API). Some types of searches count as more than a single request, however. Since Google had photos for many of the restaurants in Chapel Hill that I tried to locate and because their search system made it fairly easy to match restaurants from Factual’s database (see Section V), I chose it for my restaurant photo data source. It was not ideal as my primary source of restaurant data because it provided too few results per query (only twenty per page) and gave only a boolean “open_now” value, rather than a full set of hours, for restaurants in a standard (non-“Detail”) search.

**Restaurant Data Corrections**

Factual’s restaurant information is fairly accurate, but some restaurants in their database do not yet have associated business hours, and there are also some inaccuracies in the existing listings. Having accurate restaurant data is crucial for “What’s Open,” so I wanted to provide a way for users to correct restaurant listings. Since this is also a feature that focus group participants specifically requested, it was especially important to include. Factual’s API provides two ways of doing this: 1) flagging and 2) submitting or clearing. Flagging is a way of notifying Factual’s team that there is a problem with one of their listings, whether the problem is that the restaurant has gone out of business (“closed”), is a duplicate listing, contains inaccurate information, is listed in the wrong dataset (“inappropriate”), has never existed/is bogus (“nonexistent”), is a spam listing, or has some other problem. On the other hand, “submitting” and “clearing” involve editing rows in Factual’s database (with machine validation instead of editorial review).

Submitting allows users to insert new information or edit existing information about a

---

particular listing by specifying one or more fields and associated information.

I included a flagging feature in the app to ensure that users who find inaccurate data have a way of reporting it. By tapping a flag icon in the top-right of the restaurant details screen (see Figure 1), users can open the problem-reporting screen (see Figure 2).
To make it as easy as possible to report errors, I opted to send requests to Factual as only flagging requests instead of submissions, which would require users to enter information about the specific fields that had inaccurate information. This way, the only information that I require users to report when flagging a restaurant is the type of problem (out of business, duplicate, inaccurate, inappropriate, nonexistent, spam, or other - the Factual-required types) and a comment. This eliminates the tediousness of users correctly formatting new information to submit to Factual but still allows them to report problems and alert Factual’s team that records needs to be updated. Unfortunately, allowing users to upload photos of business hours, as suggested by focus group participants, is not feasible at this time.

**Deciding which “restaurants” to include**

Determining which types of food establishments to include in the app was not straightforward. As discussed in Section III, the focus group participants preferred non-grocery store food establishments, and some thought that bars should also be included. Since my restaurant data are from outside sources, I cannot easily filter by only the establishments that I want to include, but I opted to pull results from subcategories of Factual’s “Food and Dining” category.\(^1\) I grouped Factual’s subcategories into three different query types, as focus group participants requested, so that app users could search for places to order a meal, a dessert, or a drink (see Figure 3). This meant that I could include bars and dessert-only establishments but keep them separate from the other types of food establishments.

---

\(^1\) [http://developer.factual.com/display/docs/Places+API++Categories](http://developer.factual.com/display/docs/Places+API++Categories) (accessed 4 Mar. 2013)
Designing the User Interface

The app has two main views: the list view (see Figure 4) and the restaurant details view (see Figure 1). The list view has tabs to separate restaurants that are open now, those that are open later today, and those for which Factual currently has no hours data. Since the main purpose of the app is to find restaurants that are currently open, the app
displays the “Open Now” tab when initially loaded. By default, results are sorted by distance, but users can tap “Sort by...” to select a different sorting method. Users can refresh the list by pulling the list down, and a spinning activity ball indicates that the new results are loading. Likewise, scrolling to the bottom of the results list and pulling the list upward appends additional results to the list. Users can load a maximum of 500 results (divided among the three tabs) in sets of 50. If a user loads the app in a rural area, though, there are likely to be fewer than 500 total results available.

Since the main focus of the app is the restaurants’ business hours, I wanted to provide as much information about the hours as possible. For the restaurants that are currently open, the app displays a note about when the restaurant will close next, and restaurants closing within the next half hour are highlighted in pale red. Similarly, restaurants displayed in the “Open Later Today” tab have a note to indicate their next opening time, and those opening within the next half hour are highlighted in light green.

When tapping on an individual restaurant, the app displays the details for that restaurant (see Figure 1). I included the attributes that focus group participants found most important with the exception of menus. Tapping “Directions” opens the Google Maps app (if installed) or the Apple Maps app to display an interactive map with directions from the user’s current location to the selected restaurant. Although including in-app directions would have been ideal, Google’s newly released Maps iOS SDK (Software Development Kit)\(^\text{20}\) does not yet offer the directions feature, so loading directions in an outside app was the best option. The “Call Restaurant” button allows users to call the restaurant from within the app, and the “Website” and “Hours” buttons open modal views of the restaurant’s website and full listing of hours, respectively.

\[^{20}\text{https://developers.google.com/maps/documentation/ios/}\]
“More Details” displays a scrollable list of all other available restaurant attributes, such as parking information, whether cash payment is required, and whether the restaurant serves alcohol.

**“Open Later” and “Hours Unknown”**

Although I initially conceived of the app as being exclusively for finding restaurants that are currently open, I realized that including the restaurants open later the same day and those with unknown hours could be useful. Knowing which nearby restaurants are open later the same day could be helpful for anyone with transportation constraints who is making dining plans for later in the day. An obvious example is college or university students. Since they often live in dormitories with other friends or with groups of friends off campus, they tend to make dining plans together and want to meet at a restaurant that is nearby for everyone. Also, there are often restaurants within walking distance near campuses, so knowing which restaurants are both nearby and open later is important. Another example of when it would be important to find nearby restaurants open later the same day is when traveling. Travelers may not have easy access to transportation, so they need to find restaurants that are within walking distance. When planning their schedule for the day, it would be important to find interesting restaurants that are nearby and open when they plan on eating.

Including the restaurants with unknown hours is primarily to encourage users to submit hours information via the flagging feature to improve the app’s data for everyone. My hope is that when browsing for restaurants, some users will find restaurants in the “Hours Unknown” tab, realize that they know the hours for a restaurant, and submit that information. Additionally, some users who are looking for a place to eat may browse
through the restaurants in the “Hours Unknown” tab, find an interesting restaurant, and walk over to see if it is open since it is nearby.

V. Challenges

Number of Query Results

As with any project, this one had its challenges. The biggest problem initially was retrieving a sufficient number of restaurant results to display. When I began the project, I first used the Google Places API before checking into other APIs. Since Places gives me access to only twenty results per page for a request, it was not a good choice of restaurant data source for my app. During some parts of the day, if I requested the closest twenty restaurants and then filtered out the ones that were closed or did not have any listed hours, I was left with only one or two results. This led to a design in which I automatically requested additional “pages” of results matching the initial query since Google provides access to up to three pages of results (each page containing twenty results) by using a page token provided with the first page of results. If the first set of results had fewer than five open restaurants, I automatically requested the second page of results with the page token. This design was suboptimal, though, because Google delays the page token’s activation for a couple seconds after receiving it. To ensure that I was requesting subsequent pages only after the page token had become active, I delayed the automatic requests for the second and third pages by 2.5 seconds. Unfortunately, this made load time unacceptably long.

To gather more than twenty results at a time, I ended up switching from Google Places to Factual’s Restaurants data (see Section IV). Since Factual permits accessing up
to fifty results per page in a query, it was a much better choice for my app. After filtering restaurants by their open times, I had several in each category (open now, open later, and hours unknown) even late at night. Although I had previously considered combining results from Google Places and another provider, I found that the number of results from Factual was sufficient for my needs, especially since users can load up to an additional 450 additional results in the restaurants list. Combining results also had the challenge of proper attribution and ensuring compliance with data providers’ Terms of Service.

**Harmonizing Results from Different Databases**

Since Factual’s Restaurants API does not currently provide access to restaurant photos, I needed to use another provider to obtain photos. Google’s Places API was already familiar to me, so I tested searching Places for photos associated with restaurants obtained from Factual. Many of the restaurants in the Chapel Hill area had photos available through the Google Places API, so I decided to use Places for the photo data source. This meant that I needed to match restaurants obtained from Factual with records in Google’s Places database, though, and this proved to be a bit of a challenge.

Since there is not currently a way to directly translate Factual IDs to Google “references,” I had to match information from restaurant records provided by Factual to records in Google Places. Some fields that I expected to be consistent actually were not the same in the two databases. For example, I found one restaurant that had a different address in the two databases and a couple with different phone numbers (though those restaurants are out of business). Since I found street addresses to be consistent aside from

---

21 Services such as Factual’s Crosswalk (http://www.factual.com/data-apis/places/crosswalk) and FourSquare’s Venue Harmonization Project (https://developer.foursquare.com/overview/mapping) can translate from IDs in one API to others.
that one case, I ended up using a combination of each restaurant’s street address, latitude and longitude, and name to match Factual results to Google results.

Street addresses were formatted inconsistently between the two databases (e.g. 100 E Main St. vs. 100 East Main Street), but luckily this did not cause any problem for matching restaurants. When querying Google Places with either an abbreviated or full street address, it was able to match restaurants properly as long as the restaurant name and point location were also a match. Including latitude and longitude was necessary since those are required parameters in a Google Places “Nearby Search.”

Restaurant names were often inconsistent between the two databases, but I found that when I queried Google Places with a portion of the restaurant’s name included, the accuracy was better than when searching solely by address. To account for the inconsistency in restaurant names, the app parses the full name of the restaurant provided by Factual and uses the first significant word (not “a,” “an,” or “the”) as the name parameter when querying Google. When testing 100 results in the Chapel Hill area, I found that searching with the first significant word in the restaurant name and stripping out dashes and ampersands was sufficient to match the majority of the restaurants in the test set. Unfortunately, there were a few cases in which restaurant names differed enough that I was unable to match the records. For example, some restaurants are recorded in one database with a shortened version of the name, such as “Top of the Hill” as opposed to “Top of the Hill Restaurant & Brewery.” This particular case does not matter for my matching purposes because the only portion of the name that the app would use to query Google Places would be “top.” However, in other cases, names referring to the same restaurant have additional words at the beginning of the name. Since I use the first word

---

other than “a,” “an,” or “the,” restaurants represented in this way will not match if the Factual restaurant name has extra words at the beginning of the name and I query Google with one of those beginning words that it does not include in its representation of the restaurant.

**Restaurant Reviews**

Unfortunately, I was not able to include restaurant reviews in the app. Both Factual and Google provide restaurant ratings, which I did include, but I was unable to find a feasible way to include reviews in the app. The difficulty is that, although Google does have restaurant reviews, Factual does not. When the app conducts an initial query for nearby restaurants, it pulls the restaurant data from Factual’s database. When tapping on a particular restaurant, the app queries Google for the matching restaurant and pulls in a restaurant photo (if available). I considered also pulling in reviews from Google at the same time as loading a restaurant photo, but I was concerned that the rating that I was displaying for the restaurant was the rating from Factual, whereas the reviews were from Google and did not contribute to the rating that I was displaying. Displaying restaurant reviews from Google along with a rating from Factual would be a misrepresentation of the data. To solve this problem, I also considered displaying Google’s rating, along with the reviews from Google, on the restaurant details page. However, Google’s rating on this page might differ from the Factual rating that I display for that same restaurant in the list view, and this would be confusing to users. Not displaying ratings on the list view was another possibility, but focus group participants seemed to weigh rating higher than reviews in terms of necessity of inclusion in the app, and they requested the ability to sort restaurants by rating. This would not be possible if I
did not use Factual’s rating for the restaurants because the app gathers data from only Factual at the point when it displays the list of restaurants. Querying Google for its restaurant ratings at the same time as querying Factual for the initial list of restaurants would mean checking each Factual restaurant against Google’s database to obtain the rating for the restaurants. This scheme would be query-heavy and slow. Unfortunately, these challenges meant that I was unable to include restaurant reviews in the app.

**Menus**

I looked into several data providers for menu information, but usage limits and other restrictions were a major problem. Allmenus,\(^\text{23}\) for example, permits only two calls to its API per second or 100 calls per day. This limit is too low for nearly any app since the API key holder would already reach the daily limit if, for example, 25 app users viewed menus for only four restaurants during the course of the day. OpenMenus,\(^\text{24}\) on the other hand, looks like a promising option since it permits up to 750 API calls per day with its free “Premium” account. Its integration with CrossWalk\(^\text{25}\) means that harmonizing restaurant results from its database and Factual’s database might not be difficult. Unfortunately, without a formal company website for my app, I do not currently qualify for an OpenMenus API key.

**VI. Future Work**

A number of changes and additions in a future version of the app could improve the app’s functionality. Although I was able to include most of the features and design

elements that focus group participants indicated were important, time constraints and other barriers kept me from including everything in the current version of the app.

One of the features that focus group participants said was most important for this app was displaying restaurant menus. As discussed above (Section V), there are several challenges to incorporating menus into the app right now. However, I should qualify to obtain an API key for OpenMenus at some point in the future, and I hope to include menus at that time.

Filtering restaurants by more granular types of cuisine than just “meal,” “dessert,” and “drink” could be helpful. For example, it would be nice for users to have the option to search for restaurants that serve mediterranean food or Chinese food. I was not able to add this feature right now because of the amount of time it would take to create groupings of the roughly 135 cuisine labels\(^\text{26}\) that Factual currently uses. To make filtering by cuisine usable in the app, I would need to limit the options to a much smaller number.

As discussed in Section V, I was not able to include restaurant reviews in this version of the app. This is a feature that I do think would be nice to have in a future version, especially since focus group participants indicated that they would find it useful. Factual recently partnered with TripAdvisor, and I hope that in coming months they will offer access to reviews through their API (Carney, 2013).

Having accurate restaurant information is crucial for the app’s success, so giving users a way to flag inaccuracies was important. Currently, the app allows users to flag a restaurant and provide feedback if the restaurant has inaccurate information. However, providing a way for users to submit information about a restaurant that does not yet exist in Factual’s database could be very helpful for ensuring up-to-date information.

VII. Lessons Learned

Finding good data is challenging

The app’s biggest weakness is the data, a problem that I struggled with since the design phase. I was surprised to learn how difficult it is to find accurate restaurant data, and I found it frustrating to wade through data providers’ Terms of Service, trying to figure out if they would allow me to use their restaurant data in the way that I needed. I spent a significant amount of time researching data providers, and it appears that there is no perfect option for my needs. Factual’s Restaurants API provides robust search capabilities, and I am impressed with Factual’s leniency in regard to use of their data. I feel confident that, with the given options, I made the right choice using Factual’s API for the bulk of the app’s restaurant data. Factual is a young company, and I suspect that their data will only improve in accuracy with time. Users’ feedback through my app will play a role in that improvement.

Classification is not straightforward

When beginning this project, I would not have guessed that defining “restaurant” would present a problem. Surprisingly, everyone seems to have a different idea of how to define the term. As I began looking through query results from different data providers, I realized that some results were not what I expected to find. I questioned what I should include in the app. Should a dessert-only food establishment be considered a restaurant? Should I display smoothie bars among the results? I discussed these concerns with focus group participants, who suggested organizing results by places to get a meal, a dessert, or a drink. This is the organization that I ultimately chose, and I think
that it works well. Even so, there are some establishments that do not neatly fall into only one group. I would have liked to devote more time to investigating ways to more cleanly organize Factual’s results into the three groups that I chose.

**Xcode is finicky**

I spent more time than I anticipated on translating my user interface design ideas into a working interface with Xcode. Although dragging objects onto a mockup of an iPhone screen in the Interface Builder seemed easy, I found that it was challenging to make the objects appear how I wanted them to appear. Setting a UIButton image, a task that should have taken seconds, actually took a couple hours of tracking down why the image looked great in the Interface Builder but became stretched and disproportionate when viewed in the iPhone simulator. I learned that it was safest, in many cases, to use the Interface Builder as a tool for visualizing the interface design, but to define user interface object attributes programmatically to ensure that they were actually set as I intended. Xcode is great once you learn its quirks.

**VIII. Conclusion**

Designing and developing “What's Open” has been a great learning experience. Over the course of the semester, my programming skills have dramatically improved as I have spent many late nights wrestling my code and debugging. There are certainly changes that I could make in future versions of the app to improve its functionality, but even this version of the app exceeds my original expectations and more than fulfills the goals that I had when beginning the project. I encountered challenges with data access limitations, harmonizing restaurant results, classifying food establishments, and using
Xcode’s Interface Builder, but I am pleased with the result: a clean, simple app for finding nearby restaurants organized by their open hours.

**IX. Source Code**

*Source code is available here:*  
https://github.com/gbryan/whatsopen

*Additional screenshots are available here:*

https://github.com/gbryan/whatsopen/wiki/
Works Cited


Appendix

University of North Carolina – Chapel Hill IRB Notice

To: Bryan Gaston  
School of Information and Library Science  

From: Office of Human Research Ethics  
Date: 1/30/2013  

RE: Notice of IRB Exemption  
Exemption Category: 2.Survey, interview, public observation  
Study #: 13-1035  

Study Title: What’s Open: a native iPhone app for finding open restaurants nearby  

This submission has been reviewed by the Office of Human Research Ethics and was determined to be exempt from further review according to the regulatory category cited above under 45 CFR 46.101(b).  

Study Description:  
Purpose: To gather feedback about an iPhone app prototype to improve its design and functionality.  

Participants: ~10 UNC SILS (School of Information and Library Science) Master's students contacted via the SILS Master's students listserv.  

Procedures (methods): Students will discuss iPhone app functionality and design questions in a focus group session.  

Investigator’s Responsibilities:  
If your study protocol changes in such a way that exempt status would no longer apply, you should contact the above IRB before making the changes. The IRB will maintain records for this study for 3 years, at which time you will be contacted about the status of the study.  

Researchers are reminded that additional approvals may be needed from relevant "gatekeepers" to access subjects (e.g., principals, facility directors, healthcare system).  

CC: Paul Jones, School of Information and Library Science  

IRB Informational Message—please do not use email REPLY to this address
University of North Carolina-Chapel Hill
Consent to Participate in a Research Study

IRB Study # 13-1035
Consent Form Version Date: January 28, 2013

Title of Study: What's Open: a native iPhone app for finding open restaurants nearby
Principal Investigator: Bryan Gaston
UNC-Chapel Hill Department: School of Information and Library Science
Faculty Advisor: Paul Jones (919-360-7740)
Study Contact: Bryan Gaston (bryan_gaston@unc.edu)

What are some general things you should know about research studies?
You are being asked to take part in a research study. To join the study is voluntary. You may refuse to join, or you may withdraw your consent to be in the study, for any reason, without penalty.

Research studies are designed to obtain new knowledge. This new information may help people in the future. You may not receive any direct benefit from being in the research study. There also may be risks to being in research studies.

Details about this study are discussed below. It is important that you understand this information so that you can make an informed choice about being in this research study. You will be given a copy of this consent form. You should ask the researchers named above, any questions you have about this study at any time.

What is the purpose of this study?
The purpose of this research study is to gather feedback about an iPhone app to improve its design and functionality. The app may be released publicly to Apple’s App Store, but it will not be used for commercial purposes.

How many people will take part in this study?
If you decide to take part in this focus group, you will be one of approximately ten participants. All participants were selected through the University of North Carolina-Chapel Hill School of Library and Information Science (SILS) Master's student listserv and were chosen because of their expressed interest in the study.

How long will your part in this study last?
Your participation in this focus group will last approximately one hour.

What will happen if you take part in the study?
The focus group participants will be asked to discuss the design of an iPhone app and make suggestions for improvements. No questions will be directed to you individually, but instead will be posed to the group. You may choose to respond or not respond at any
point during the discussion. The researcher will take written notes during the session to analyze later.

**What are the possible benefits from being in this study?**
Participating in this study could help the researcher improve the usability and design of an iPhone app, but your participation may not directly benefit you.

**What are the possible risks or discomforts involved from being in this study?**
We do not anticipate any risks or discomfort to you from being in this study. Even though we will emphasize to all participants that comments made during the focus group session should be kept confidential, it is possible that participants may repeat comments outside of the group at some time in the future. Therefore, we encourage you to be as honest and open as you can, but remain aware of our limits in protecting confidentiality.

**How will information about you be protected?**
Every effort will be taken to protect your identity as a participant in this study. You will not be identified in any report or publication of this study or its results. Your name will not appear on any notes recorded by the researcher, and the notes will not be shared with anyone other than the researchers.

**Will you receive anything for being in this study?**
Participants will be offered a light meal.

**Will it cost you anything to be in this study?**
There will be no costs for being in the study.

**How will my participation in this study affect my status as a student at UNC-CH?**
Taking part in this research is not a part of your University duties as a student, and refusing to participate will not affect your status as a student at UNC-CH. Your participation or choice to not participate will not affect your grades.

**What if you have questions about this study?**
You have the right to ask, and have answered, any questions you may have about this research. If you have questions, or concerns, you should contact the researchers listed on the first page of this form.

**What if you have questions about your rights as a research participant?**
All research on human volunteers is reviewed by a committee that works to protect your rights and welfare. If you have questions or concerns about your rights as a research subject you may contact, anonymously if you wish, the Institutional Review Board at 919-966-3113 or by email to IRB_subjects@unc.edu.

**Participant’s Agreement:**
I have read the information provided above. I have asked all the questions I have at this time. I voluntarily agree to participate in this research study.
Focus Group Guide

Agenda
1) Introduction and Purpose
2) Rules
3) Questions from Participants
4) Group Questions
5) Conclusion and Dismissal

Introduction and Purpose
Thank you for agreeing to participate in this focus group session. My name is Bryan Gaston, a graduate student in the School of Information and Library Science at UNC-Chapel Hill.

The purpose of the focus group session is to gather feedback and suggestions about an iPhone app that will help people locate nearby establishments organized by open hours. I will ask you questions about features and design of the app during our discussion so that I can make improvements in the app for a future version.

You were selected to participate in this focus group because you expressed interest in the study. Keep in mind that there are no wrong answers, so please feel free to speak openly.

Rules
I will be taking written notes during our focus group session, but only I will see the notes. I will not record your name or any identifying information about you. During this focus group session, please respect the other participants’ privacy and do not talk about what each other said after you leave.

Questions from Students
Do you have any questions before we start our discussion?
Group Discussion Questions
Now, we’ll discuss the design and functionality for the iPhone app. Since this is an informal brainstorming session, please give me as much feedback as you can.

1. Discuss use cases for the app.
2. Which class of food establishment is most important to present in the app?
3. Which attributes of a food establishment would you be most interested in seeing in this app?
   a. (Show list of all attributes available through Factual.)
   b. Which in this list would you find most useful? Least useful?
   c. Are there other attributes not listed that I should include?
4. Interface design.
   a. If you were viewing a screen that showed details about a restaurant that you had selected, what would you like to see on that screen?
   b. How would you like the items arranged?
5. What additional features would you like to have in this app?

Conclusion and Dismissal
Thank you for participating! Please remember not to discuss anything that was said during this focus group session after you leave.