Developer List (name and computing IDs): Pam Gregoretti (prg5rv), Amy Zhao (awz4pc)

Device Name / Platform: Pink iOS

Project Title: Foodie

Project Pitch (one paragraph describing what the app does):

The purpose of our application is to find local restaurants based on seeing images of food. When the application first starts, the user is prompted to log in with Google account. Afterwards, the application stores the Google account information to local storage and uses this to welcome the user. The user then sees an image grid view that is populated with images from a Yelp API call based on the GPS coordinates of the phone. Clicking on an image takes the user to a restaurant information screen that shows the restaurant name, categories, distance, address, rating, and an image of the restaurant. The user can then go back to the gridview by clicking back. Clicking "random" prompts a message that tells the user that if they shake the device while on the grid view screen that a random nearby restaurant information page will show. Shaking the device prompts this segue.

Platform Justification (a discussion as to why you chose the platform you did):

We chose iOS because it is easier to make an application consistent with iOS visuals compared to Android. We both had macs so developing on Xcode was not an issue, and we liked how the storyboard in Xcode is more intuitive compared to the XML layouts in Android. For example, we like how in Xcode, you can use the control-click feature to connect buttons and other UI objects to the corresponding code. We found that the tutorials in iOS on how to make tables was easier than in Android. Also, we like the overall visual design aesthetics of iOS because it looks more streamlined compared to Android.

Key Features (include short descriptions of each):

Google log in: The user logs in with Google so the application can save their Google account info to local data and use that to welcome the user

Local restaurant feed: The user can see images of local restaurants in a grid image feed that users can click on to see more information about the restaurant

Shake for random restaurant: Users can shake the device while on the grid view to get a random local restaurant information page to show up on their screen

Testing Methodologies (what did you do to test the app):

As we developed we would test what we had on the emulator or device. We would also use print statements to see where in our code the device was hitting, like making sure the Yelp API call was successful, testing the GPS coordinates, and see when the grid view was reloading its data. When we had significant work done, we would upload to the GitHub for version control in case huge bugs were created later so we could recover earlier versions if needed.

Usage (including any special info we need to run the app (username/passwords, etc.)):

We have a weird bug where after logging in with your Google account you have to click the Google log in button again to continue. We asked Professor Sherriff about this and he said that since it is not a specifically tested part of the required functionality in the rubric, this small bug shouldn't be penalized. There are no required usernames/passwords but you need to log into Google to continue. You also need to enable GPS location (which is prompted when the user first uses the app).

Lessons Learned (at least half a page):

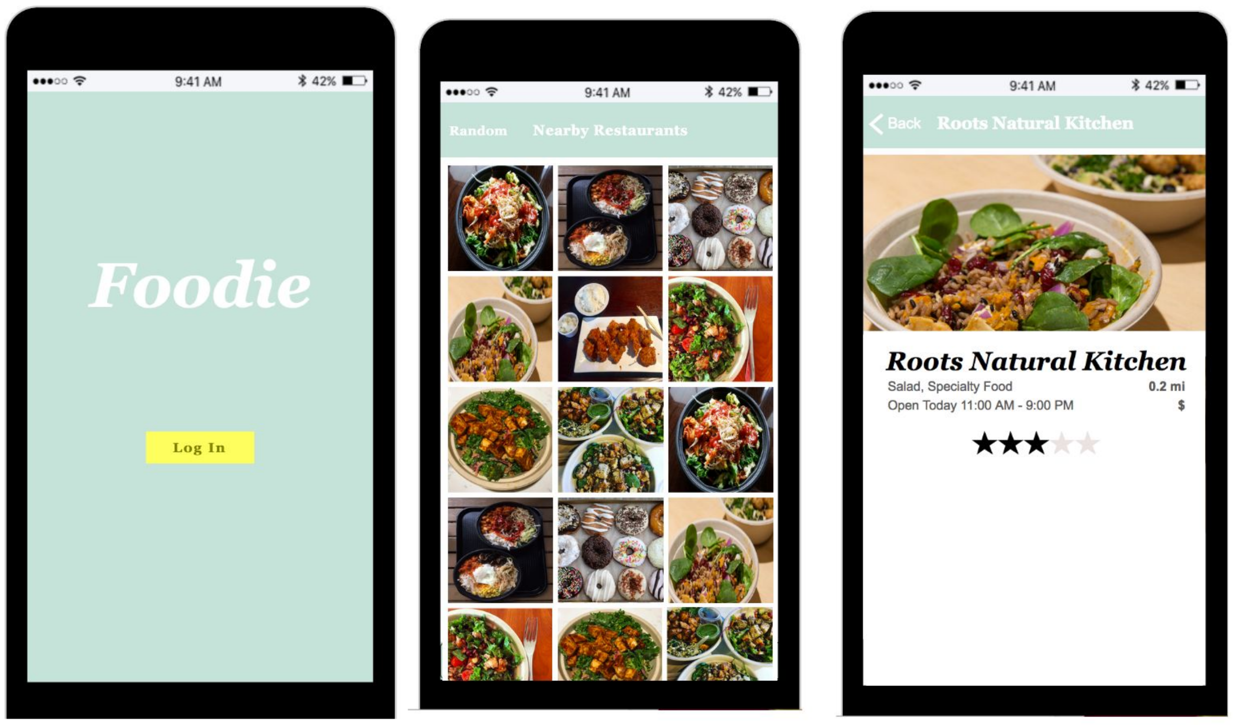
Pam:

I became a lot more comfortable with iOS development and REST API calls. I feel more comfortable performing things like segues and building UICollectionViews or UITableViews on my own. I also understand a lot better how RESTful APIs work and why they are useful. I had some difficulty at first implementing the Yelp API call because it requires an OAuthentication and that wasn't covered in the materials in class, but I managed to find a solution online. I also ran into an error where the UICollectionView of images would load before the location manager was initiated and passed the information for the Yelp API call to run, so I had to somehow get the UICollectionView to reload after this was done. You can accomplish this with a reloadData() method for the UICollectionView, but since it runs on the main thread it can run before you expect it to, and I found the best way to fix this was to put the method at the end of my Yelp API call logic after it has found the results but still on the Yelp API call thread. After that, building the rest of the application was fine. I see how easy it is to get shake to recognize and to have messages pop up. I also learned more about the info.plist file and how it is involved with things like authenticating location managers or allowing things like Google log in.

Amy:

Through this project, I became much more comfortable with iOS development as a whole from working with storyboards to REST API calls. I feel a lot more comfortable working with the storyboard and linking UI elements such as buttons to the corresponding code. I also feel more comfortable working with unwind segues now compared to when we did the iOS mini project. Now, I have a better understanding of how to work with different views and how to build UICollectionViews and UITableViews to serve the purposes we want them to serve. I also learned about how to use motion gestures such as the shake gesture to create a random restaurant chooser feature for the app. We ran into some difficulties with implementing the Yelp API call due to the fact that certain aspects of it were not discussed in class. For example, the OAuthentication was not covered by Professor Sherriff in class, but we had to use it in order to correctly implement the Yelp API call. Online resources helped with figuring this out. I am also now more comfortable with the info.plist file and better understand its role in authentication. For example, when it was used to authenticate the Google log in and location managers. Through this project, I have also become more familiar with setting constraints for the UI elements in a view. We had run into a problem where if you rotate the app, one of the texts on the screen disappeared. Learning more about how constraints work helped us fix the problem.

Wireframe:



This wireframe reflects changes that were made as we developed our application. The main idea about seeing nearby restaurants is the same but we simplified a lot of the functionalities to make development and debugging easier. This wireframe better reflects our final product. The user is prompted to log in, and then sees a feed of nearby restaurant images where they can tap or shake to see more restaurant info.