COS 333
Spring 2015

Partner 1 : William Bertran
Partner 2 : Gregory Leeper
Partner 3 : Nicholas Pai

FLOCK

Find your flock

Our Original Goals

1. **Make it as hard as possible for friends to lose track of each other at Princeton.**
2. **Make it as easy as possible for users to find out where their friends are and where the party is.**

Milestone Progression

We originally planned to have implemented our fundamental features (Map Page, Friends Page, Club Page) a week before Deans Date, which would have left us a week to add in bonus features such as a message-board for each club and push notifications. However, we didnt account for how long it would take to discover and subsequently resolve all of the small bugs that occurred in our program. Thus, we didnt really get to add in any bonus features. Nevertheless, we are very proud of the functionality of our three main pages. This provides a solid foundation for Flock on top of which we could add on bonus features without having to make any major, structural changes. Most importantly, we addressed our original goals at the minimum.

Major Design Decisions

There were a couple of major design decisions that heavily impacted out implementation of Flock and our programming experiences.

Parse vs Manual Backend: Our backend stores each user upon sign-up and associates their name with their friends-list and current location. We therefore had to figure out how to represent, with our code, the event of a user entering a club. This information had to be easily available to the users friends, and must be easily updated when the users location changes. We ultimately decided to go with Parse, a cloud app platform that takes care of the backend for us, because we didnt want to spend several weeks perfecting our backend. Instead, we wanted to focus on making Flock as easy to use and as powerful for the user as possible. Parse stores each user for us and associates their username with their location. Furthermore, Parse stores each club as a club object and keeps track of its Attendance. Therefore, we can easily update a clubs attendance number as well as pull from this number. By electing to go with Parse, our next decision concerning the login functionality of Flock was made easier for us.

Facebook Login vs. Flock-specific login: Parse integrates Facebook directly into its SDK, so this was an easy decision for us as programmers. We forced users to signup for Parse through Facebook so that their identity (username) was linked with their Facebook ID. By associating each user with their Facebook ID, we were able to quickly access any given users friends list using Facebooks Graph API.

The Facebook login and Parse backend was definitely the right call in retrospect because it let us focus our energy on perfecting the users experience and giving them as much functionality as possible. The Parse backend is incredibly easy to use and it has clear documentation that made it easy for us as programmers as well. By making our app a Facebook app, we were furthermore able to make the users experience better. The main pro for the user was (1) no work on their end to construct their own friends list, for a user could automatically retrieve information about any other Flock user provided they were already Facebook friends, and (2) easy login through Facebook that skipped the creation of yet another username/password/security question to remember.

The Evolution of Our Design

Pages: Our initial design idea, which we stuck with for the entirety of the project, was to have three functional pages along with the home page (See below for a description of each page).

Changing Pages: We wanted the user to be able to switch between pages simply by swiping left or right, but after some trial and error, we found that a slideout menu was significantly easier and widely used by several popular apps (such as Facebook and Venmo). To implement this, we added a button on the top left of the screen that slides out a list of buttons to all of the other pages.

Refreshing: Flock is an app that, when in use, would need to be reloaded quite often because people would constantly be changing their locations. To account for this, we initially wanted each of the pages, other than the Home Page, to automatically reload data after a given time period. We managed to implemented this quite well for the View Friends Page, and even had it up and running when giving our demo presentation.

However, it caused the page to go completely white for a couple of seconds or so every time it reloaded, which could be inconvenient for the user if they are scrolling through their list of friends. So, instead, we decided to implement a pull-down refreshing option on the View Friends Page and on the View Clubs Page. This allows the user, as opposed to an arbitrary timer, to control when the page would reload its data. We also implemented a reload button on the View Maps Page (because that page cannot be reloaded by pulling down due to the scrolling feature of the map). The pages also reload automatically when switching pages, so whenever you visit a page, either for the first time or not, it will contain the most recent data when the page opens or re-opens.

Map Page: Our final design for the Map Page was comparable to our initial idea of what we wanted it to be like. We wanted the user to be able to move throughout the map and easily see how many people and how many friends were at each club. To implement this, we simply made an annotation for each club using MKMapKit and Apple Maps. Each annotation would simply contain the name, number of people, number of friends, and latitude and longitude of a club and would appear as a white box on the page.

Friends Page: As with the Map Page, our final design of the Friends Page met basically all of our initial expectations. When a user switches to the Friends Page, their device iterates through each Facebook friend who has a Flock account and fetches their location from Parse and then displays all of those friend-location pairs in an alphabetized list. We also sorted the friends page alphabetically by first name.

Club Page: The club page actually exceeded our initial expectations. At first, we just wanted to design a list of all of the eating clubs with the number of friends and number of people at each club. However, we eventually decided to create a unique page for each club which shows the status of the club (open/closed), the number of people and friends at each club, and even a slideshow of pictures taken at the club. This slideshow feature, which we demoed in our presentation, allows anyone to take a picture and add it only to the set of photos of their current club.

Major Surprises

The first major surprise we experienced was that Facebook updated their SDK a week after our Spring Break. This led to immediately out of date Facebook documentation that made implementing features such as their login button and graph API extremely difficult. While we spent a frustrating amount of extra hours figuring out updated syntax, we learned valuable lessons about modular programming. In COS classes at Princeton, we are taught to never deviate from a prescribed API, and to aim our comments toward theoretical clients. Modular programming is all about encapsulation. Now that we suddenly found ourselves on the client side of the Facebook SDK, all we wanted to know was (1) what their functions syntaxes were and (2) what exactly we should expect when utilizing these functions. These simple tasks became almost impossible without API documentation. Thus, one takeaway I have as a programmer in general from this project is to always comment clearly and specifically so that future clients/developers can reuse my code without difficulty. Modular programming is a beautiful invention, but can easily become impossible to use.

The second major (bad) surprise we experienced was an update with the Swift syntax. Specifically, Swift changed the way that it dealt with optional types and how they are unwrapped. Unfortunately this update coincided with the middle of our project. So, we had to laboriously go through our code and modify our code. Obviously, if we were further into our development process, then this could have been disastrous. This taught us the incredible danger in changing a languages syntax after its release. If anything, it should be outlawed.

What We Could Improve with More Time

We would definitely focus on adding bonus features. I think we all agree that the most useful feature to add would be push notifications. Luckily, Parse offers easy push functionality for its apps. We would like the user to be able to tap on any of their friends and send them a request for their current location. For example, Will could select Nicks name on the Friends list, which would send Nick a notification on his phone reading, Will Bertrand would like to know where you are. This would allow us to better meet one of our original goals: make losing track of friends at Princeton as hard as possible.

The next feature that we would like to add would be statistics-oriented. We would like to keep track of each clubs nightly attendance numbers. We could then display these attendance statistics on an Analytics/Popularity page on the side-swipe menu. This would let users find out which club is on average the best attended. Or more interestingly, one could find out the answers to questions like, Which club is usually the most popular on Friday nights? or What time of the night do most people go to Cannon?

Lastly, we would definitely work on making the application more attractive for users. We could put a blurred image in the background of the home page as well as implement a better color scheme. These are all things we are thinking about implementing before next Fall, when we would like to release a free version to Princeton students. Based on the feedback from our friends, many Princeton students would use our application and we would definitely like to distribute it in time for the Fall.

What We Would do Differently

First of all, we noticed that we got the most progress done over the last two weeks. Obviously, we had busy schedules and other classes and work that distracted us, but we regret not setting aside more time per week to just focus on the development of Flock. We were amazed at how much work we could get done by just sitting down as a team for several hours. Therefore, we would recommend to future teams that they set aside one chunk of several hours per week to just work on the project.

Secondly, we should have worked out version control before we had written a single line of code. Github combined with Xcode was extremely frustrating due to the nature of Xcode projects. We probably should have used Github for Mac or some other application specifically suited to iOS development source control.

Third, we should have thought a lot more about our menu design before creating the pages. What we ended up doing was converting our existing three page set up to a side-swipe menu structure. This was pretty difficult to go backwards like we did. Instead, if we were smarter we would have decided to use a side-swipe menu from the beginning. This way, we could have focused on setting the menu up first, perfecting it without bugs, and then easily adding on the three pages (clubs, friends, maps). The consequence of our mistakes was a whole host of stack bugs (the menu was implemented with a stack).

Lastly, although we focused on testing step by step and getting as close to perfection on each step, we definitely should have tested more. Testing is the dirty part of programming and something that we definitely have come to value now as a skill. Flock was an especially difficult application to debug because it required us to physically move around, which is quite a time sink. If we had tested rigorously with corner case tests and stress tests earlier, then we would not have spent so much time at the end of the project debugging. For example, we discovered a bug in how we annotated our Map view with rectangles just this past week that we definitely should have addressed when we first set up our map view.

Advice for Future Students

Never underestimate any part of project. Work step by step, perfect each step. If you are also doing an iOS project, then remember to take into account the fact that there are different iPhone versions used in reality. There are several different screen sizes, which can cause surprises in how something looks on an iPhone 4 after testing on an iPhone 6. Furthermore, the older the iPhone, the worst the power and speed. Nick has an iPhone 5, Will and Greg have iPhone 6s. When testing, we would often have weird results (illuminating some bugs) because Nicks GPS was much slower and less powerful than Will or Gregs. Also, if you planning on using Xcode with source control, beware! We used Github for source control, but the way that Xcode projects are developed made source control a constant source of pain for us. Our advice would be for only one person to work on the project at a time, otherwise you will get all sort of weird errors with merges and branches. Because of this constraint it almost seemed as if there was no way to efficiently work on an iOS project as a team.

Acknowledgements

A big thank you to Professor Kernighan for allowing us to do this tremendously educational project; this really felt like the first computer science class at Princeton where we learned how to do real world coding! Finally, a big thanks to Elba for keeping us on pace and giving us valuable advice along the way!