PrincetEvents : Final Report

Divya Mehta, Maia Ezratty, Alexa Wojak, Mitchell Hamburger, Harry Heffernan

Spring 2016

Introduction

It cannot be understated how important experience is when approaching a large iOS development project like ours. That said, we had absolutely no experience with iOS development entering this project, so much of our process was outrageously backward and hilariously inefficient. However (and this is a HUGE however), despite our frightening lack of experience, we completed our entire app, almost precisely as we envisioned it at the beginning of the semester, in a relatively timely manner. So we would like to begin our final report with a brief message of advice to future generations of COS 333 students, so that they can follow in the (large) footsteps of yours truly, the PrincetEvents Team:

1. When planning your app, do not, even for a second, consider how difficult your functionality will be to implement. Dream big, babies. Shoot for the moon. Even if you miss, you'll land among the stars.
2. Assume that GitHub will be easy. People have been using GitHub for years, so it must be pretty straightforward.
3. Never write descriptive Commit messages. Not even once. Bash the keyboard and hope that you get a consonant followed by a vowel followed by a consonant.
4. Make sure that everyone in your group is working on the app at the same time in different locations. Resolving Merge Conflicts builds character, much like digging holes.

5. Don’t take the initial project outline seriously. App development is like free-form jazz or brain surgery— having a plan will just complicate things.
6. Let other groups get inside your head. If you’re not doing significantly better than every other group you talk to, then you should quit messing around, cut corners, and get in front.
7. Don’t comment. Never comment.

(If you haven’t guessed, these are terrible mistakes to make; please don’t make them.)

Levity aside, we had a great time developing PrincetEvents; it was incredibly satisfying to see a vision we were passionate about materialize into something usable and real. There were certainly hurdles, which we will discuss in the body of this report, but each road block we encountered taught us so much about the iOS development process, and we can say now that we have a confident understanding of Xcode, Swift, FireBase, and a number of HTML scraping frameworks. Beyond these technical skills, we learned invaluable lessons about delegating responsibility, adapting to unusual situations, and managing time in the context of a large project such as this one, and each of us feels confident that we could tackle a similar project in the future.

Overview

We created PrincetEvents in an effort to provide functionality for, to make sense of, and perhaps most importantly, to consolidate the myriad of social events on campus. We noticed that on our hectic campus it can be very hard to keep track of what exciting event is happening on campus next, and even harder to find information about that subject. We felt that listserv spam and lamppost flyers don't do justice in advertising the incredible things that happen at Princeton every day. We envisioned, and created, an application that presents performing arts shows, athletic events, and a number of other types of events in a manner that is easy to use and informative. We envisioned users of this app enlisting it to get information on specific events, as well as to browse upcoming events, in case something looks interesting.

Another issue we identified is one of ticketing. Tired of receiving thousands of emails requesting to buy and sell tickets in the days leading up to a big show, we wanted our app to be able to connect buyers and sellers in a simple way. In order to make inboxes smaller and to reduce the stress of the buyers and sellers, we implemented this functionality: for any event that requires a ticket, our app links buyers and sellers of that ticket, presenting contact information to both parties.

In general, we aimed to develop a hub for the Princeton community where students can go for any student-event-related needs. Whether you want to find the details on a specific event, browse for events that might interest you, or request/post tickets to an event, PrincetEvents serve as a single, easy to use platform for Princeton students interested in events on campus.

Milestones

Our original timeline had us completing small tasks every couple of days, with us gradually reaching larger milestones along the way. Although this process did not, in actuality, materialize in so clean a way, we did find that we were hitting our big milestones on time. In fact, the only large milestone that we missed was integrating our ticket functionality for the Alpha Version. Some of our larger milestones were as follows:

1. Integrating the imported calendar API (for Prototype)
2. Setting up remote database (for Alpha)
3. Scraping athletics data and displaying it in UI (for Alpha)
4. Integrating ticket exchange functionality (for Alpha, but completed for Beta)
5. Finalize UI (for Beta, but completed for Dean's Date)

These large milestones served as anchors for our development process, while smaller milestones, such as integrating CAS or setting up the Ticket Exchange UI, served as stepping stones between these larger destinations. We found that, although some of the smaller milestones proved difficult to implement with exact precision, it was very helpful to have large goals to keep us grounded.

Decisions

A. Why We Have CAS

You might think that anyone should be able to see events going on around Princeton's campus on our app. However, we soon realized a major reason to have every user log in through CAS. When signing up to request or post a ticket, without linking a user to their unique netid, users would almost definitely sign up using bogus names, or sign their friends up, etc. Linking them to their netid provides some amount of accountability. It also provides us with an easy way to allow people to let users know who they've matched with. While probably our main goal in terms of future plans is implementing in-app communication between matches, returning their netids worked well as a temporary solution, since all of our email addresses are our netids.

B. Websites Chosen

We thought that the majority of events that students would be interested in on campus were either arts events or sports events. The university ticketing website provided us with all of the ticketed events on campus -- so not only arts events but also all other events on campus that require tickets. And the goprincetontigers website provided us with all of the sports events on campus.

C. Layout

The two main functions we wanted our app to have were just seeing all of the upcoming events on campus, and also being able to exchange tickets for those events if possible. So, in terms of seeing what events are coming up, we thought that there were two ways users might want to do this: looking at a calendar and seeing what events are on a given day, and looking at just a list of upcoming events all in one list. Then the other main tab is a list of your matches.

Successes

Although we made many decisions that we should not necessarily have made in hindsight, we also made some crucially successful decisions that made the implementation of some features much smoother than it could have been.

A. PDTSimpleCalendar

Perhaps the most pivotal decision we made in our design process was our decision on how to implement PrincetEvents' calendar functionality. We can say, thankfully, that from the very beginning we were set on importing an open source calendar module to customize, rather than designing our own calendar from scratch. We had the foresight to acknowledge in advance just how daunting implementing a comprehensive calendar module from scratch would be. Furthermore, we acknowledged that, in whichever calendar we would select, we would need certain key components. First and foremost, we needed a calendar that would allow us to implement custom actions when a date is selected— many of the calendars we looked at were simply static displays of the current month, allowing users to view dates but not select them. Beyond this key functionality, we needed a calendar that was highly customizable visually; as a group we were very particular about how we wanted the app to look, so a calendar that would be flexible with our aesthetic demands was crucial.

In the end we settled on using the PDTSimpleCalendar module, and it was a resounding success. To the first of our demands (the ability to allow users to select dates), the PDTSimpleCalendar has an excellent system whereby it prepares NSDate information when a specific date is selected. This made it very easy for us to take the information packaged by PDTSimpleCalendar upon selection of a date, pass it through a simple segue, and use this information to query our database for applicable events. To the second of our demands (customizability), the PDTSimpleCalendar module was also excellent. It was hugely convenient that the default appearance of the calendar was very similar to our creative vision, but, beyond that, the module made it very easy to create an instance of the PDTSimpleCalendar and customize its finer details, such as the color upon selection of a date, font style, font color, and shape of the background for today's date.

B. Use of StoryBoard

As novice Xcode users, we grappled for a long time during our planning with how we could efficiently implement several seemingly simple features; making buttons lead to pages, creating tabbed pages, and stuffing our giant calendar module into a tabbed page are just three examples of issues we were sufficiently confused by entering our development process. We discussed at great length whether or not we should use Xcode's StoryBoard, an interactive visualization of the control flow of the application, to help us with these smaller issues. On the one hand, the StoryBoard would allow us implement basic functionalities quickly and reliably (that is, less prone to novice Xcoder mistakes). But on the other hand, it would obscure some of the finer details of certain functionalities. For example, creating a tabbed view controller in StoryBoard is as easy as dragging and dropping, but customizing actions that occur when certain tabs are selected becomes significantly more difficult.

In the end, we decided to use the StoryBoard, and we were, again, hugely satisfied with our decision. We quickly discovered that the StoryBoard is much smarter than we had originally thought; we could create custom classes programmatically in .swift files and then create instances of these classes in the StoryBoard with incredible ease. For example, we were unsatisfied with the default behavior of Xcode's UIViewController for our Calendar tab, so we created our own custom View Controller (in FirstViewController.swift), which would inherit the bits of the UIViewController class that we needed and would modify those we wanted a little differently. Then, we plugged an instance of this custom View Controller into the StoryBoard, and used the StoryBoard interface in order to modify its constraints, segue information, display size, etc. This communication that we created between our custom code and the StoryBoard made for incredibly efficient and clean implementation of many aspects of our application; custom View Controllers, Buttons, Web Controllers, and Views became shockingly simple when we began interacting with the StoryBoard in a constructive way.

Failures : What we would have done differently?

One large thing that we would have done differently if restarting this project would be to try to get access to a University database of events rather than scraping our events from different websites. While scraping events from the athletics website was relatively easy, scraping events from the ticketing site was incredibly difficult. Furthermore, if these websites change we could potentially have to edit our scraping script to reflect these changes. Ultimately, using a database directly would both be more reliable and less complicated.

Additionally, due to the fact that we had to first fill out a webform using PhantomJS and then scrape the resulting webpage using BeautifulSoup to scrape the University ticketing webpage, we were unable to host this scraper on a timer at this point. This is something we wish we could have completed, but are nonetheless are proud of our success timing the athletics data scraper and using PhantomJS to scrape the ticketing webpage.

Another major setback that we faced was initially using heroku rather than cPanel to host our CAS script. Heroku free apps have to sleep six hours in every 24 hour period. This is fine for certain things, but for the main login page for our app this was not sustainable. While we ultimately changed this, there were times before the switchover where we were locked out of our app and had to alter our code to bypass the sleeping CAS page.

Usability and Relevance

The inspiration for our project stemmed from the frenzy that accompanies ticket acquisitions around the time of a popular athletic event or dance show. The usability of our app exceeds just being able to provide consolidated information to students. It has the capability of eliminating this frenzy completely and would be used widely by students across our campus. At any given time on Princeton's campus, there is a mass amount of events occurring. We wanted to create this project to better serve students in getting involved with these events. A big step in our decision-making process was deciding what events to include in our calendar to make it useful yet simple enough to remain relevant for users. As a result we settled to two main categories of data - athletic games and arts events.

Further Development

All of us are extremely passionate about this app and the potential impact it could have on student life at Princeton. As a result, we created a list of further development ideas that we would like to implement further if we were to continue working on this project independently (which we probably will).

A.In-app contact and messaging feature

Our current app returns the netid of the person you matched to buy/sell tickets. We hope to eventually build an in-app feature that allows direct communication between user that have matched on a ticketed event.

B.Exchanging tickets in addition to buying/selling option

We recently discovered that there is also a demand for exchanging tickets for different times of any one show. This way a student would not have to relinquish/sell their current ticket before they were able to find a replacement ticket that they desired. This functionality would require a more complex matching process but it would be extremely useful.

C.Exploring additional sites that also have relevant campus events

We are currently scraping two popular university websites to populate our database. We hope to eventually research other useful sites that students would be interested in. This would be a relatively simple process but we want to maintain our simplicity and avoid over crowding our calendar.

(Real) Advice for future generations of future students (Reflections)

- Start early: Come up with your project idea at the beginning of the semester and talk to your teammates and friends to flush out what features are most important to the project.
- Don't be afraid to delete code: You will inevitably write code that you will not end up using in your final project. All the code that you write will, however, be part of the learning experience. Therefore, don't be afraid to experiment and ultimately scrap what didn't work out.
- Watch videos and read tutorials before starting: Understanding how the software and languages you are using work before you start your project will save you a lot of time and hassle. Realizing halfway through the project that you implementing something super inefficiently is frustrating and takes more time than doing it properly the first time.
- Learn how to use git: We went into the project not know anything about git and assumed it would just work out. This was the wrong assumption. We continuously had merge

conflicts and each of us at times had to delete and re-clone the entire project. Understanding git would have made everything about this project run smoother.

- Pick something you are interested in: You will spend a lot of time working on this project during the second half of the semester. Be sure to pick a project that you genuinely want to work on because it will ultimately make the experience enjoyable like it was for us.

Thank You

We'd like to thank our group manager, Nick, for being a rockstar throughout the whole process— we absolutely couldn't have done it without him.

We'd also like to thank Divya's parents for coming to the demo, it was really cool to have a cheering section while we demoed PrincetEvents.