eXchange: a COS 333 Final Report

James Almeida Emanuel Castaneda Meaghan O'Neill
Sumer Parikh Danielle Pintz

9 May 2016

Background

Princeton has 11 eating clubs, where about 70% of upperclassmen eat their meals. In order to allow students to eat with other students who are not in their club, there is a meal exchange system in place, where students can "exchange" meals by eating together at both clubs in the same month. If a meal exchange is incomplete by the end of any given month, meaning only one meal was eaten by both students at one of the clubs, the student in that club will be charged some amount of money, so that the club can be compensated. The current system is extremely inefficient and inconvenient, consisting of students filling out small blue slips every time they exchange, and these slips being matched up manually at the end of each month.

We propose a new digitalized meal exchange system which is extremely efficient, convenient, and even fun to use.

Elevator Pitch (AKA ”Why the Current System Stinks & Change is Needed”)

When a Princeton student becomes a junior and starts eating at her eating club, she slowly begins to realize that she doesn’t like eating there. There will always be a club with better food or nicer people somewhere else on the street. Unfortunately, the current meal exchange system is so tedious and inefficient, that she is disincentivized to meal exchange, and will have to sit in her club with the same boring people and the same bland food. Furthermore, if she does decide to meal exchange, the card that she fills out may get lost, and her friend will be charged for no reason. Clubs have to go through tedious and error-prone paperwork, and the poor officers have more to worry about on top of Sumer’s rowdy and obnoxious behavior every Saturday night.

Thankfully, eXchange solves all of these problems. The app makes meal exchanging easy and fun, so that meals are easy to set up and complete. Sumer’s friends can also use the newsfeed in order to see who he’s eating with, and find out if he’s on the verge of finally getting a girlfriend! And since all the exchange information is organized in one streamlined interface, the club officers can go back to only worrying about Sumer’s rowdy and obnoxious behavior every Saturday.

For a dramatic portrayal of our elevator pitch, please visit: `https://www.youtube.com/watch?v=pA84UXk`

Baby steps

In mid-February, we were nervous, and with the exception of Eman, reasonably incompetent. While we had made an awesome prototype on proto.io (`https://jamesalmeida.proto.io/share/?id=21c47824-e34e-460b-82ea-9b173da86e97&v=6`), we did not have the hard skills to actually code up our vision. We knew exactly what we wanted to achieve, but were nervous about being able to pull it off. Apart from Eman (who knew Swift), none of us knew Swift, HTML, CSS, or Javascript, which we used for the two apps we wrote. Not to mention Django or Heroku/Postgres which we learned and wrote code for but didn't deploy!

Milestones (interspersed with failures)

Prototype complete (March 18 during Spring Break)

The prototype was done which provided a blueprint for the iOS app (the design of the prototype is mirrored strongly in the final product).

ICC Meeting and Group hacking session (March 21)

We met with the ICC meal exchange task force to flesh out the full functionality that would be expected of the app and go over the use cases. They seemed very enthusiastic to adopt the app and offered us whatever resources we may need. The only important thing they stressed was maintenance and the ability to pass on the code to someone before we graduate. We also had an extremely productive group hacking session and also where we went over the general framework of the app and how the components should interact with each other, so that we would all be on the same page.

First few views of iOS app created (March 25)

The iOS team met together for several hours after both James and Danielle completed the FoodTracker iOS development tutorial. With their newfound knowledge and Emanuel's past experience in iOS, they created three of the four screens in the storyboard that are still used in the app today. Additionally, they refined the data types so that they contained only necessary information and made the logic flow of the app much more modular.

iOS app with hardcoded data completed (April 2)

Throughout the week, the iOS team used Github to work remotely and met twice in order to finish the front end of the app. Since there was still no backend, the team decided to initialize arrays with hardcoded data to use in the control flow of the app. CAS login was enabled for the app during this time as well. For all intents and purposes, the app was working as a front end, but meal requests could not be sent between different users of the app yet.

Began implementing Django modules (April 2)

After researching several different possible frameworks for a backend, we decided that Django with a Postgres SQL database would be the best choice. It seemed like we could like it to both iOS and the web application. We began writing modules for Django and getting basic web pages to work.

Refined iOS app (April 4)

We decided that until the backend was more developed to the point that we could integrate it, we would try to perfect the UI. This involved adding a "like" button to the newsfeed, creating custom fonts, optimizing the text in the cells to be clear and readable, and making sure the phrases made sense.

Django modules completed with a Postgres database (April 5)

We completed Django modules that stored all the data that the app worked off in modules.py within the Django app. While he hadn't configured views.py, we had a basic working model that demonstrated out app's functionality!

Django module scrapped and Firebase picked up (April 7)

After two days of desperately and unsuccessfully trying to integrate the Django file with our iOS app so that we would not have to use hard-coded data, we finally decided to scrap Django and Postgres and use a different back-end. Luckily, we found Firebase, which was easy to set-up, and fit in seamlessly with our functionality on both platforms

Began integrating iOS app with Firebase (April 9)

Now that Firebase seemed promising, we all played around with it until we understood how to read/write data from it. We started to replace our hardcoded arrays from our original app with queries to the database. First we completely integrated the eXchange screen with Firebase and from there we had a working model to adapt to things like the Newsfeed, My Meals, and User screens. From this point on, we worked on/off on the app remotely until all of the moving parts were working simultaneously.

Web app front end template completed (April 10)

With the Django module scrapped, the Webapp had a lot of work to be done. Luckily, we had an HTML/CSS template that we had made before we started the Django app for Django, and we just had to reproduce it with a few changes for each of the pages.

Base alpha functionality complete (with several bugs) (April 15th)

Both apps worked with several glitches but all the elements of the system was fully connected and all of the core functionality was running though not perfectly

Design Decisions

Database Components

The components of our database our explained in detail in the Developer's Guide. Determining what components to have in our database was a very important design decision. Ultimately we decided to focus on a structure that would allow for a clear and logical data flow. This facilitates cleaner code and maintainability of our system. The tradeoff, however, is sacrificing more memory by allowing redundancy in data. In particular we create both an incomplete-exchange component and a complete-exchange component. This division creates clear data flow and allows for simpler queries from our web app.

Making the iOS app student only

Selecting our target audience was important yet simple. The meal exchange process naturally limits our audience to Princeton students. We decided to focus on inter-club exchanges to limit the scope of our project, and did not include dining hall to club meal exchanges. This means that currently only students in an eating club can use our app to meal exchange.

Making the webapp only accessible to eating club officers and meal checkers

We quickly realized that meal checkers unanimously decided that they would prefer to meal check on a computer as opposed to the iOS app. This was made clear to us by club officers as well as club managers at the ICC meeting. We therefore decided to add all the functionality for the clubs onto a web-based platform as opposed to a mobile platform.

Webapp as a standalone platform

Since not every student has access to an iOS device (and since students may want to participate in a meal exchange without having to set it up through the app), we realized that we had to create a mechanism for meal checkers to create meals on the spot. We did this through the "Create a Meal" tab, which lets a meal checker enter all relevant information in order to log a meal.

Verification and authentication

Since the iOS app can only be accessed by students of Princeton University, we decided that the easiest way to verify and authenticate users was by using CAS. This decision was reinforced by the fact that we could access the student's netid from the CAS login, which is the main key that we use for students in the database. For the webapp, we decided to authenticate using Firebase's email authentication. This seemed very convenient as we could pre-make the accounts for the 11 eating clubs. We did not have to set up a registration system, since we would never have more than 11 clubs in the system (unless President Eisgruber makes a massive donation to Campus Club and lifts it out of bankruptcy).

What Went Well

The UI of both apps

We made our graphics really early and then used them throughout. We had a very clear minimalistic vision for what the app should look like and the entire group is very happy with the UI.

Staying on Track

We hit milestones frequently and were never felt like we were in danger of not submitting a completed and working product.

Integration between the two apps

Because of the way our database was structured, both apps could seamlessly draw from and add to the same database, and give the illusion of communicating directly (even though they were both in fact only drawing from the same database). Both teams knew exactly what the other team was doing, and we often used the other teams product to help debug the product that we were working on!

Strong interest from the ICC for adoption

We were nervous that since creating a meal exchange system has been attempted in the past and not picked up, all our code may have gone to waste. Against all odds, this app seems to be breaking through the bureaucracy of the ICC. We have received strong written interest from the ICC that indicates that our app may be operational as the official system as early as fall 2016. We were able to reach out to the officers of every club and demo it for them, and they were "raving" about our app to the extent that the ICC and grad boards have to consider implementing it.

...And What Went Badly

Github Version Control

Not knowing how to handle merge conflicts caused some issues with the linking of our libraries and it took us a while to figure that out. (Note to self: never ever force push!!! It will never end well!!!)

Hooking up Postgres to the iOS app through Django

We tried hooking Postgres up to Django for a couple of days, but it proved to be very difficult and we realized that it was unlikely that we would finish it. So, we decided to cut our losses and switch to Firebase at Hansen's suggestion (which is so awesome and worked out perfectly).

The night before the demo in Friend 108

We'd prefer not to talk about it... Read 'Github Version Control' above if you must.

Advice to Future Students

1. Start on your project as early as possible. We know that your professor and TA will nag you to do this, but it really is something nontrivial that is going to require more than just 2 consecutive all nighters doing. You will inevitably pivot from whatever you believed to be the perfect framework or backend that you researched and will have to start from square 1.

2. Familiarize yourself with GitHub early and learn how it works. Make sure that when you're working in a group that everyone has the ability to push and pull from GitHub and that no one continues to work on a really old version of the code.

3. Don't debug late at night when you're exhausted. Don't put yourself in a situation where you have to debug late at night. Even if you're the best person at coding throughout all hours of the night, someone in your group is not and you will rapidly lose productivity as time goes on. Just do everyone a favor and get some sleep and reconvene in the morning.

4. Choose a deadline at least a day in advance before the demo when you stop writing code and make a presentation–Professor Moretti will stress this but some groups still won't (*cough*). DO THIS. We greatly regret not following such sound advice. If you spend all night perfecting your code, it won't even show in the presentation and you will be unprepared. There is even a chance that after you finish your code you accidentally screw up GitHub and undo several of your past commits so that nobody has a working version of the project anymore. Just cut your losses and make a professional looking presentation that will be respected.

Future Steps

In the future, we hope to have our meal exchange system adopted by the ICC (Inter-Club Council) and widely used by all Princeton students in eating clubs. As we mentioned, there is already serious interest from the ICC after seeing the app, and it's possible that as early as this fall, eXchange will be used to facilitate meal exchanges between eating clubs! After adoption, we hope to augment functionality so that the system can also be used by Princeton students on University dining plans when they meal exchange with eating club members; this will let the University do away with the paper system of dining hall-eating club exchange. We also hope to offer options for independent students (students not on dining plans or in eating clubs), such as allowing them to charge a meal to their student account if they want to meal exchange with someone in a club (since they can't complete the exchange).

In terms of functionality, we hope to add notifications and reminders to the iOS app, so that users can be reminded to complete an exchange, and notified before an upcoming meal. For the webapp, we hope to add a feature that lets you download lists of complete and incomplete exchanges as Excel documents for manual recordkeeping.