BookCub: Final Report

By: Patrick Boroughs, Grace Miles (Project Manager),
Mohamed El-Dirany, Alex Vogelsang

Generating an Idea

After getting a group together we came up with a fairly long list of ten ideas that we were interested in implementing. (We even brought four ideas to our initial meeting with Professor Kernighan!) From there we settled on doing a textbook exchange app because we wanted something that was both solidly within scope, but would also add value to Princeton students. It was critical to the team that we created an app that we would be able to get real users on before the end of the project. This was a motivating factor for us and helped us complete our deadlines and generally design a better app.

Milestones

Our Milestones went quite smoothly for the entire project. It was extremely helpful that we started working on the app and getting the basic database going over spring break. Starting early allowed us to be quite ahead of schedule for the Project Prototype and from there on stay ahead or on-schedule. Additionally, after each meeting we would set specific tasks for the upcoming week and assign them to each person on the team which allowed for accountability and kept us focused on small goals throughout the project. Even though we did not often code in the same room we had a group chat which we used to communicate where we were on each week's task, questions we had, or iterate over various ideas.

CAS
We knew that we wanted to limit the app to Princeton students, and so we decided to use CAS. About a week after we had first set up the initial Django application, we were able to incorporate CAS using a Django CAS application and by adapting the CAS code provided from COS333 for Python.

DEPLOYMENT
We deployed our app really early on, before we even had much (if any) functionality. This forced us to ensure that when we inserted a feature into the website, we tested it thoroughly locally, and then we tested it again after deployment. We did this to ensure that our code was always working regardless of which environment we were using it on, and it also forced us to have incremental progress rather than explosive growth only to realize after deploying that we had to put in a lot of work to make it work on the Google Cloud Platform.

POPULATING THE DATABASE

The app by itself was near useless without the courses and people information. Due to how our database was set up, we had to import all of the undergraduates, and we did this through the Tigerbook API. We then added all of the courses for this semester as well as next through the great webfeeds that OIT provides. After this initial set of information was placed into the database, we felt that the app was at a stage that we could call a prototype, and it also felt the app was coming together and could really be useful.

UI

Iterating over the UI as we built out the functionality was essential to our development. It allowed us to test out the functionality with real users faster. We used bootstrap templates and stackoverflow forums to help debug as we went along. A specific milestone was the development of the departmental gridview which all users are directed to after clicking “buy” or “sell” from the home page.

RATINGS

Near the end, the one function that we felt we absolutely had to have was ratings. It completed the functionality we felt that we needed in regards to buying and selling. Without it, we felt that the buying and selling functionality wasn’t as good as it could be because if a person in the interaction decided to not live up to his expected part of the deal (such as refusing to pay the requested amount, or providing a book that was more damaged than the buyer let on), there was no good way to punish either one. With ratings, we let future users know how any buying or selling interaction went with a specific user.

Unexpected Challenges

PHOTOS

Photos provided a definite challenge in terms of implementation. When first trying to implement photos, we simply wanted local photo upload and viewing to work. This meant having to deal with the media url and root for Django. This took an unexpected amount of time because our system was simply not the same as others. At some point, instead of actually uploading to the folder specified, it was actually uploading files to the C:// root of the computer, something that we couldn’t quite understand. It took several more hours of fiddling with about 10 lines of code to actually get the photos to upload to the correct location, a folder called media located in our project folder.

This was only the beginning of our problems, however. Upon deploying, we realized that this implementation was not going to work because Google Compute Engine has a read-only file system, meaning that we were going to have to use an online storage system and implement that directly into how our uploads worked. We decided to use Google Cloud Storage (GCS) because it was directly incorporated the Google Cloud Platform. This presented its own issue

since there was no official protocol for upload from Django to GCS. We found a package online that we were able to use as our own custom backend file storage system to work with GCS.

EMAILS
In a similar vein, we found that Google Cloud Platform made using emails difficult. After implementing emails locally using django's built in email features, we realized upon deploying that Google blocked all native SMTP ports (587, 25, 26, and 465). Most implementations otherwise required using SendGrid or some other marketing email service in order to send emails, but we were able to circumvent these by using the GMail API which was able to bypass the platform's email system and directly use the GMail servers.

SCHEDULING
One unexpected challenge was scheduling times when we all could work together. After forming the team it quickly became apparent how conflicting all of our schedules were. Even for our first meeting with Professor Kernighan we had to schedule it at 9am because that was the only time slot in which all group members were free. We made it work through our "lunch meetings" and constant communication through our group chat over text. At times we even resorted to have conference calls if we needed to coordinate a design issue within the app. We made it work by overly communicating what each of us were doing to the larger group. However, it definitely would have been helpful to have been able to physically meet up more often. Special shout out to our TA Sotiris for being extremely flexible about our weekly meeting times!

Testing

We were very focused in our testing efforts throughout the development of Bookcub. We used Github liberally to push any changes one of us made in a specific part of the app for testing by other members of the group. This was extremely helpful as it allowed us to catch small bugs early on as well as iterate over the desired user experience. After our alpha test we each got (at least) 5 friends to test out the app. We sent a link to Bookcub along with a Google form in order to quantify the feedback we received. In the form, we asked questions about the type of issue, where it was found in the app, as well as an in-depth description, and room for general feedback/suggestions.

Once we circulated the app among our friends for a week we sent it out to the larger student body and were able to receive more feedback/suggestions via the Google form. This was great because not only were functionality issues pointed out, but it was a big source of idea generation for us. A key feature in the app, what we call our "waitlist", originated from user feedback. At times, it even felt that we had more feedback than we knew what to do with! Collecting the feedback all in one form was great because it allowed us to prioritize what we wanted to implement alongside our normal weekly task lists.

Great (and Poor) Choices

One of the things that really was good for us was to have defined weekly goals, and then, if we finished those weekly goals, help others who might be struggling with their own goals. This kind of comradery ensured almost all of our goals were done on time (we had one week where we fell a little short of our expected goals, but it was an exception).

A poor choice we made was when we first launched the app to the greater student body we failed to realize that the mobile rendering of the app may not quite be as expected. A number of people had a poor first exposure to the app because they got the email advertisement on their phone and opened it up from there. Even though the UI we were building was supposed to be cross-platform, there were some necessary tweaks to be made to the code to better support the website. This experience definitely highlights the importance of user testing and that you never know what real users will try to do.

What We'd do With More Time

More time would definitely allow us to add more features. We've had a lot of input from users about what features they would like and we've compiled a list of things that we'd like to implement.

- Selling groups of books at a time
- Scrape all required books from the course registrar
- Create an "Other" department so people can sell miscellaneous books
- Allow users to import a recal schedule so that they are automatically added to waitlists for specific classes
- Differentiate between class links that have books listed and those that don't so users don't need an extra click
- Add graduate students

The exciting thing about this list is that we were able to implement all the core features we put on our product roadmap. The features on this list would hopefully allow us to have a better user experience and more complete app.

What We'd do Differently

We really think a lot of what we did worked well. Unlike many other groups, we did a lot of sustained work after spring break, and so we never had a really rushed and hurried period near the demo where we were stressing about whether or not our product would work.

Most of what we would do differently are “nice to have” aspects originating in how to gain more traction among the larger student body. It would have been great to get alpha users on the site earlier so that we would attain a critical mass of Princeton students sooner. While some of the functionality was not there most of what kept us from launching earlier was our desire to have more implemented. We definitely could have taken more risks in that area. Moreover, during our demo a question was brought up if we had talked to TigerApps at all and partnered with USG. While, this had been floating in the back of our minds we had yet to take action at that time and it was something that we *easily* could have done earlier in the semester. We also, never heard back from OIT about labeling certain textbooks as required via the registrar. In seeing a different group demo that was able to complete this it was definitely frustrating that we did not push OIT harder to get the ability to get that feature into our app.

Advice for Next Year

Heed the advice from years’ past! Before starting the project we had heard the importance of starting early from previous classmates and decided to get our database as well as basic site set up during spring break. While working on the project during the break and going through all the tutorials was tedious, it gave us a really good launching point for the rest of the project. There was no point in which we were that concerned about missing milestones or having a broken product. Spring break was also essential because we did not code that much. We instead, organized what we thought the database should look like, the type of features that we wanted and the general user experience. This planning made us have fewer late-stage changes and made our development more efficient.

The team is incredibly important. Something that we failed to account for when we first got together is that we all had differing obligations outside of 333, and therefore when it came to meeting together, there were few times available. It was rare for all four of us to ever be in the same building outside of TA meetings and class. We were able to work well by simply keeping in touch by texting, but communication was really important, especially if we were working on something that another team member had more expertise in. We only realized about two or three weeks before our demo how useful it could be to work together through a phone call if we couldn’t be in the same location. The thing is, not all groups will work well by working apart, and therefore you really need to know the individual members and what ways people work best, because if people need to be around others to get work done, there could be many problems if other members work best alone.

Closing remarks

This project has been a remarkable learning experience. It was fun to have complete ownership over a project and know that real users were on our app. We have deeply appreciated all the support from **Professor Kerninghan**, **Professor Moretti**, our **TA Sotiris** and our fellow

students have gone onto our app and offered us feedback. Never forget Bookcub “Textbook exchange made, like, really simple”.

Best,
BookCub Team