ClassMaps

Final Report

Spring 2018

Team

Adrian Tong (PM/Front-end)
David Todd (Back-end)
Andy Lin (Front-end)
Yang Song (Front-end)
Jerry Wei (Advisor)

Inspiration and Planning

We've come a long way from the beginning of COS 333. We came into the semester barely knowing each other, completely inexperienced with any sort of web app development, and with no idea what to do for a project. It's hard to imagine a group that could have been less prepared at the start of the semester. By discussing potential options and beginning to think through our features well before midterms, though, we hit the ground running. Reflecting on previous projects, such as TigerMenus, PrincetonCourses, and TigerBook, we realized that many successful apps shared a key feature: they took data that Princeton hid or needlessly obfuscated and presented it in an accessible and visually appealing format.

To transform this general observation into a workable idea, we drew on our personal experiences. While we were from 3 different residential colleges, 4 different states, and 2 different countries, we all had experienced the difficulty of using Princeton's resources to navigate campus. Reflecting on the trouble we faced finding our classes in strangely named buildings with the limited information on the registrar's course listings or struggling to find the nearest printer, we knew we could do better. After identifying a problem, we envisioned the solution: a searchable map that would make finding the location and time of courses efficient and would centralize information on campus resources.

Design Document

With a solid idea, we created our design document and agreed on a plan to handle the workload. In forcing us to consider the utility of our project from a user's perspective, as well as the practical steps needed to create the app, the design document was invaluable moving forward. It enabled us to divide the project into manageable tasks and then prioritize which of these tasks was most important.
Even as technical problems arose or user feedback forced us to reconsider some details, this plan allowed us to remain organized and finish on schedule.

Getting off the Ground

Learning Through Experience

Going into the project, we all had very minimal experience with the Django framework, web development languages (HTML, CSS, JS, Bootstrap), and Postgres. So to start off, we took Spring Break and the majority of the first week afterwards playing around with things. We say "playing around," but it wasn't so fun to be honest. Frequent bugs and crashes involving mixing Django and Heroku along with some outdated tutorials left us frustrated and confused. We also had no idea how to use Github at the beginning, which was problematic when we all tried to edit and push the same file simultaneously. After plenty of trials, nearly as many errors, and lots of practice, we all eventually refined our skills enough to accomplish what was needed.

Technical Challenges

In addition to learning languages and frameworks, gathering and combining data from various sources was often frustrating. The OIT WebFeeds often had outdated and/or missing information — for example, they only had the locations of kitchens and laundry rooms within residential colleges. Additionally, the building outlines from OpenStreetMap were problematic because they often had different names from the registrar or, in some case, even no names at all. Another snag was the realization that the registrar did not provide the unique id numbers for newer buildings, such as Wallace Dance & Theater, which forced us to use buildings' non-standard names at times.

If getting the data was hard, putting it into a database was worse. We had originally planned to use MongoDB because we thought its NoSQL format would provide more flexibility. Hours of frustration, though, revealed that MongoDB was only easily compatible with an older version of Django, which had outdated and often unhelpful documentation. What could we do? While scraping the data required taking some time to deal with annoying corner cases, no amount of reading old tutorials or scrolling through endless stackoverflow pages seemed to get us any closer to integrating MongoDB successfully. Instead of wasting a week (or the rest of the semester at the rate we were going), we chose to switch to Postgres, which integrated much more smoothly with Django and provided all the functionality we needed. Without having contingencies to deal with unexpected hurdles, we would have been dead in the water.

Prototype

Despite our lack of initial inexperience and the plethora of issues described above, we persevered and got a working prototype up and running pretty quickly, about a week before it was due. Our bare bones version consisted of a simple map and search bar; a database with the buildings and courses; and search logic to link the front and back ends. While far from our goal for the final project, this was a sign of tangible progress and provided a surge of confidence. More importantly, it was a (sort of) working product we could take to our roommates, friends, and vague acquaintances to get feedback and gauge interest.

Adding more Features for the Alpha Version

UI Improvements

We felt pretty proud of ourselves after the prototype — none of us had ever programmed something so substantial. At the same time, though, we knew there was a lot of work still to be done, starting with improving the clunky and quite frankly ugly UI. Currently, we still had a huge blank spaces on all sides, a big title banner, and a small map. The most significant modifications from the prototype was changing the map to fullscreen and having the search bar as an overlay. That change substantially changed the UI of our project and it went from "This looks like crap" to "This actually looks pretty good." We also developed icons and graphics, making it look better in the browser, in bookmarks, and on mobile home screens.

Static Files Issue

Progressing rapidly, it was about time our inexperience caught up with us. While we were working one day, several components on our site suddenly broke, and the css files could not be read. Getting Django to recognize static files, a seemingly innocuous task, quickly became a serious roadblock. Instead of finding a quick fix, the problem took a whole weekend of frustrating debugging. Through a combination of finding helpful, pre-implemented packages, looking at online documentation, and talking with other groups in the same boat, we were eventually able to resolve the issue. The moral here was clear: any task, no matter how small, has the potential to become a mess. Who would have thought creating an interactive map (albeit relying heavily on Leaflet) would be significantly easier and less painful than getting some static files to load properly?

Time and Date Filters

As we wanted to improve the search algorithm to allow for date and time filters, there was some indecision as to what input methods we could use. Some potential candidates included a time slider, traditional typing input, dropdown menus, and more. The result was decided a mixture between design preferences and what we could actually get working. While there were some cool bootstrap forms online with this functionality built in, we struggled to get them to work with our transparent overlay and full screen map. We found individual components for the time in the form of a Bootstrap clockpicker, which was a very cool looking design, and a standard dropdown menu that was functionally sound.

Storing User Data and CAS

With our egos in check, we continued by adding resource overlays that included printers, as well as laundry locations, and implemented CAS authentication. CAS was also not the easiest. We had trouble setting up the structure of our website to redirect after logging in, so that took a while to fix. We even briefly considered not having CAS login, but decided that would limit the possibility of too many potential features. After getting login up, we managed to include a preliminary version of saving searched courses that just saved to a single account as a proof of concept. We then set up the database to store in the which users had saved each class in the classes table. We decided later that this design was unintuitive, since we had to loop through all the classes to display the favorites, so we created a new table, with entries for users, that stored only classes and buildings saved by each person (identified by netid). Having added all this new functionality with a drastically improved UI, this is where we decided it was time to start showing our app to others and getting feedback.

Initial TA Feedback

Frankly, we were a bit nervous when Jerry pulled our group directly into Professor Kernighan's office to give him a quick demo of our app. Within the first 3 minutes, he found an issue with the time and date inputs, and crashed our app. From this experience, we gained two important pieces of knowledge. First, we had a bug. Second, users don't always

do what you want them to do. While the main practical change from this meetings was to ensure valid input to the timepicker, we gained a deeper appreciation for the fact that as developers, we couldn't on our own anticipate all the things users would want or try to do with our app. After fixing this bug, then, we met with the other TAs, and did our best to implement their feedback. By the end of the week, we had a solid functioning product.

Soliciting User Feedback and the Beta Version

Official Public Launch

After we completed our Alpha version, tested it with TAs, and did some serious bug fixing, we thought our app was absolutely perfect. We shipped it out on May 2nd, hitting up all the listservs and Facebook groups with a link to our app and a feedback form. As they say, "you always get it right on your first try," right? ...Not exactly. After reviewing user feedback through a Google form, we got a lot of constructive criticism and knew immediately we had plenty to do.

We noticed a number of common trends among these critiques. Firstly, people didn't like the separate results page. Admittedly, it was rather clunky, and the page was rather ugly, with a plain gray background and a weird shade of turquoise text. We had left it there to "make the results viewable on mobile," but more honestly because we had been too lazy to design and implement a better solution. Secondly, people liked the resources overlays a lot and wanted more of them, even specifically requesting things like retail dining, and missing locations.

Implementing Feedback

We spent a long time thinking about how to display the results on the same page. There were many constraints: it had to look good on desktop, be usable on mobile, leave room for the map, and show a good amount of data. Drawing inspiration from Google's horizontal search result style, we decided to display the results in a horizontal list. This would not take too much space on the map page, and would be usable on mobile. After developing cards to display the individual search results and reworking some of the underlying Django structure, we managed to get our new results display working. In the meantime, we also scraped more resource sites, but ran into challenges scraping some of them. Getting the locations of filtered water stations was particularly tricky because the Office of Sustainability website only had a image to show locations. Because they didn't provide GeoJSON coordinates, so much of the data had to be put in manually. Soon, we expanded from three resource overlays to a healthy seven.

~🔥Heatmap~🔥

Despite all of these improvements, we still weren't completely satisfied with our project. We knew our app was well done and thought through, but we wanted to *blow the other projects out of the water*. We brainstormed and went back to our potential features and user feedback to think about how we could expand our functionality. Could we

implement a heatmap in a weekend? Only one way to find out! After deciding it was feasible, we implemented the heatmap within a single day and even added the specific number of courses and people in each building at any given time as popup text over the building outlines. When we finally pushed and deployed all these changes, we were hyped by how cool the feature was. Nothing pushes you to work faster or harder than being excited about your product.

When we checked the new responses on the Google form the following day, the positivity improved tremendously. People no longer had major complaints, and many were impressed and gave us compliments.

Final Stretch and Finishing Touches

After finalizing our features we decided to hunker down and fix the remaining bugs and finalize our product. David fixed the bugs, Adrian gave the UI an overhaul, Andy developed the about page, and Yang finished up resources and got current location working. We decided to make it work as a button instead of on login since Leaflet was rather slow at locating on PC. Once we got that done, we focused all of our effort towards putting together a strong and cohesive presentation and got hyped for our demo day. Like our actual project, we made sure to present to others beforehand and get feedback on our presentation skills.

Testing

Thorough testing was very important for us, considering the variety, size, and inconsistency of the data that we had to integrate together. Admittedly, most of our testing was done manually between the four of us. Going through the data, we looked for tricky corner cases and weird naming. We developed a list of queries that were known tricky cases, which was vaguely inspired by working with the course registrar data previously in Assignment 4. For example, we noted cross-listed classes, classes with no location or time, and canceled precepts, just to name a few. We do regret not creating a formal framework, since some time was wasted by making same manual queries over and over. After deploying any new features we had to run through the same checks every time. We also crowd-sourced testing by asking users to check if they could find their courses/printers/water stations/etc on the Google feedback form. All in all, we made sure that the quantity of features added did not degrade the quality overall quality of our product.

Looking Forward

Unfortunately, all things come to a close eventually. Toys R Us recently shut down their retail stores. Subway ended their $5 footlong deal. Even the Beatles split up eventually. However, we do believe that ClassMaps has great use to Princeton students, especially incoming freshmen trying to find their classes and people using the overlays to find resources for many years to come.

There are a number of features we would like to add, including walk times, directions, building navigation, and trending searches just to name a few. We plan to utilize Google Analytics more to figure out which features are used most and build upon them. Over the summer, we will update our database with the option to choose next semester's classes as we prep for our next promotion campaign.

As for future plans and partnerships, we want to work with USG to make ClassMaps an official TigerApp, which will bolster our credibility and usage. We have discussed the idea of working on this project as our independent work next year to be able to devote more time on it and get funding. We also talked with OIT about our app, and we know they are developing a new Princeton app with a new map, so we may be providing advice and assistance based on our experience.

Final Advice

The Spanish verb for "to work" is "trabajar," derived from the Latin word meaning "torture". While grinding out this project did indeed feel like torture at times, if you find a good group and work on a project you want to work on, working becomes a lot more enjoyable than painful. A few more tips:

1. If you feel like you don't know what you're doing at the start, that's fine. You'll get there eventually.
2. Be honest. Trust your group and bring up your concerns or approval.
3. Stay organized and don't get behind — you probably won't catch up.
4. Stuff happens, especially when you least expect it. Have contingencies for your contingencies.
5. Get user feedback early and often and have a variety of technical/non-technical people try out your app

Acknowledgements

Thank you to Professor Kernighan for teaching the course and giving us advice, as well as a broad understanding of many programming languages. A big thanks also to our project advisor, Jerry Wei, for providing support and helpful tips not only on the technical side, but also developing a successful project and presenting it well. Finally, thanks to all the TAs and users who have given us valuable feedback throughout the semester.

● ● ●