ProspectAve Final Report

BILL DONG

BEN LIANG

MICHAEL MAN

JUNE HO PARK

YANG TU

Progress

Timeline

As part of the Design Document, we outlined a timeline for our project. In it, we detailed which features and progress steps we would have accomplished by a given week. While this was extremely helpful in staying on track, we found that we may have been a little too ambitious with our goals, particularly in the early weeks. We overestimated how much work we would do both during Spring Break, as well as during the first week after Spring Break, when we still had one last assignment due for COS 333. As a result, we were a week behind our timeline. Luckily, we had enough built-in buffer time in our plan to still finish and release our project before demo days, while also hitting many of our “stretch” goals (for example, our calendar feature for future dates was an example of “stretch” goal). In addition, we were able to implement some ideas we originally did not have in the Design Document, such as the ability to filter for only dates with PUID events or Pass/List events. Despite all of this, we were able to go public by our planned date, and reach over 1,000 unique users.

To-Do List

On top of our timeline, we kept a running to-do list throughout the semester. On it, we kept an updated list of the various features and bugs that we were currently working on, sorted by the various part of the product they were associated with. This way, whenever someone had free time, they could check the to-do list and hopefully knock out a bullet or two. Furthermore, this ensured that every known bug written down so that nothing was forgotten or slipped through.

Weekly Meetings

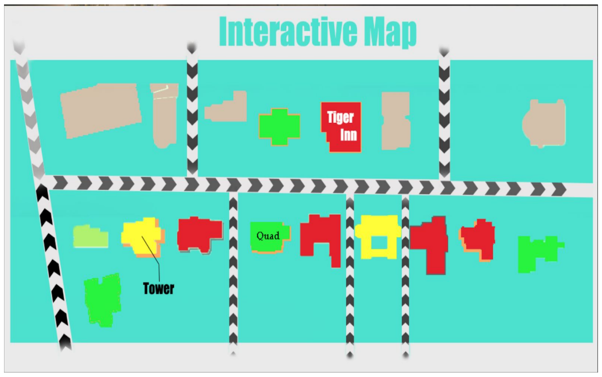
On top of working individually or in smaller groups, we made sure to meet every week as an entire group. During these sessions, which averaged three to four hours long, we would communicate and decide on design changes. Having everyone on board with design decisions reduced wasted time working on features that would be vetoed later on. Working late into the night as a group also just helped build team chemistry, which made working on the project fun.

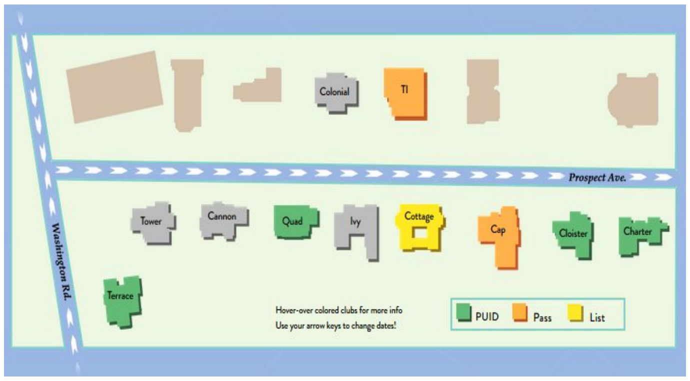
Design Decisions

UI Design - Map

UI design was our biggest priority, and we spent a lot of time going over the overall design of our app. Before we officially decided to build a webapp with information about eating clubs, we pitched the idea to some upperclassmen friends; a very common response was that the idea had been done before, but the projects always fizzled out after a few weeks. When we asked why, almost everyone told us that the reason was clunky, complicated UI for either students or officers (or both). We wanted ProspectAve to be different; as a result, we made it our top priority to have a simple, attractive UI that both students and officers would want to keep coming back to.

One of our first ideas was a map of the Street that students could interact with to see information about the eating clubs. The idea stuck, and it eventually became the focal point of our homepage. However, it took a lot of time to come up with a good design for it (one of our initial mock-ups is shown below on the left), and we went through close to a dozen different layouts before we finally settled on our current layout (shown below on the right).





Our final layout does an excellent job of highlighting the eating clubs while downplaying any extraneous features. There is just enough detail to help students recognize "The Street" without distracting from the information. Using light, faded colors for the background and closed eating clubs and bolder colors for open eating clubs helped further distinguish the important features.

UI Design - Map

Mobile support was also a big priority from the very beginning--we realized that in many cases, students would want to check which clubs were open on the go. Unfortunately, we quickly realized that there was no good way to port the map onto a mobile screen, and the ability to clearly see what events were happening on the Street at one glance was important to us. As a result, we spent even more time deciding on a good mobile layout, finally deciding on the club icons. We also wanted to take advantage of touch-screen capabilities on mobile, so we added more intuitive swiping for the dates and infobar.

UI Design - Officer Page

From our conversations with upperclassmen, we knew that the reason so many projects like ours had fizzled out before was because the officers simply lost interest, and we realized that a big reason for that was bad officer UI. We really wanted to create an elegant officer UI that was as easy as possible to use. We started with a simple table listing all of the events to a "block" design in which each event had its own rectangle, to the calendar view we have now. This allowed officers to easily see a month's worth of information without crowding the screen with details. Furthermore, we realized that many clubs, such as Charter and Colonial, have the same event on multiple days. The bulk upload form was something we added to allow officers to upload an entire semester's worth of events at once; rather than filling out the same form 10 or 12 times with identical information, they could do it with one click.

Trade Offs

Designing a product inevitably involves sacrificing some features in favor of others. One such trade off was to sacrifice Microsoft Edge compatibility in favor of applying CSS filters that Edge does not properly support. We decided that the colored clubs was too important to sacrifice for a little-used browser. This decision was later validated by our analytics which showed that just three, less than 0.3%, of our users used Edge. Another trade off was catering towards students versus eating clubs. One feature that we prototyped was a heat map indicating how popular club was. While this would have been great for students, it may have angered eating clubs that are less popular. Since we rely on eating club support for our data, we decided to shelve this feature.

Languages And Systems

Front-End

We hacked our front-end together with HTML, CSS, and JavaScript, along with a host of JavaScript third-party libraries. We wanted our main web app to be one main page, so we decided on a scrolling template that we found online. To generate our map view, we overlaid cutouts of the clubs onto a custom map of Prospect Avenue, and then dynamically added color filters with JavaScript and CSS. The main page queries the backend for club event data through AJAX requests (GET and POST), then stores the results in a local JavaScript object. Furthermore, jQuery and CSS were very helpful in animating our application. We could create slide-in, fade, and swipe effects through cleverly written animation functions.

The reason why we coded our webapp through mostly JavaScript was because of JavaScript's modular nature. Given the large developer community JS has, we could easily find third-party plugins to augment our code. For example, our calendar relied on vanilla-calendar, a small third-party calendar package. Meanwhile, jQuery was instrumental in allowing us to edit our Document Object Model (DOM) and modify HTML elements.

Back-End

While we quickly got our map drawn up and coded, setting up the back-end server was essential to the operation of ProspectAve. We eventually settled on using Node.js as our backend server. Considering how much JavaScript we were coding in our front-end, using Node.js was an easy transition for the team. To hold event data, we decided to use SQLite 3, as it is extremely lightweight and easy to modify locally. From this, we built a Node server that used Express to store event data from officers and provide the data to user clients. One challenge we faced was figuring out how to authenticate officers. After integrating CAS authentication, we used the Node package cookie-session to create session cookies for officers, remembering them so they wouldn't have to login every time they opened the page.

Choosing Node.js was a good choice because of its strong developer community. Using Node Package Manager (npm), we could easily integrated various plugins into our code to fill our needs. Furthermore, both Node and SQLite exceeded our scalability needs. We chose SQLite over options like MongoDB, MySQL, and PostgreSQL because we knew that SQLite could fulfill our performance requirements. Therefore, the lightweight nature of SQLite made it the most attractive choice.

Conclusion

Our choices of programming languages and full-stack design were ultimately shaped by the needs of ProspectAve. We aimed to keep our code as modular as possible in order to facilitate future additions to the codebase and also migrating our code between platforms. In doing so, we could easily connect the mobile page to the server by running the same AJAX code. This helped save many hours of programming. We couldn't be happier with the languages and libraries we chose to use, and we are confident that they will make our maintenance and future development of ProspectAve as simple as possible.

TESTING

SELF TESTING

We discovered nearly all of our bugs by extensively self-testing our app. We self-tested the back-end by inputting, editing, and deleting fake data for eating clubs. We did not want to risk asking officers to input real data for their eating clubs and have a major bug show up, leaving them unimpressed with our app and jeopardizing our future relationships with them. Through self-testing, we realized that we had not protected our app against HTML injections, which was something we were quickly able to fix.

We spent most of our self-testing time, however, on the front-end. Since formatting is a huge aspect of our project (with the map, infobar, calendar, etc.), we spent a lot of time ensuring that ProspectAve could handle unexpected changes to the screen width, screen height, etc., and that all of our elements adjusted properly with these changes. For example, we discovered numerous browser compatibility issues with Firefox that we fixed--in particular, one problem that went unnoticed for a while was that if a Firefox user had their browser at exactly 1084 pixels wide, the sidebar did not properly resize (which turned out to be because we used a > sign instead of >= at one point).

ALPHA & BETA TESTING

For alpha testing, we wanted to check two main things: that the app was intuitive, and that it looked fine on various screen ratios and browsers. We asked some of our close friends to use our site, and gave them little to no instruction. Thankfully, they all seemed to have no trouble navigating around the app.

We also realized at this stage that the “top bar” for tablet screen ratios looked off on some devices, allowing us to fix some formatting issues.

After we were confident with our officer page, we reached out to officers about a week and a half before lawn parties to start gathering real data. We gave a live demo in front of every officer we met, and made sure that we authorized the right NetID’s. During one of the live demos, we caught a bug where attempting to put in a duplicate event through the bulk-upload form caused undefined behavior.

We publicly advertised ProspectAve the night before lawn parties, which allowed us to test our app on a broader scale. To reach the widest audience possible, we went on Facebook, residential college Listservs, Real-Talk Princeton, and posted flyers around campus. Our built-in feedback form meant it was easy for real users to give us feedback and notify us of any bugs. We were able to catch some resizing bugs and incorrect information thanks to that.

What We Learned

Early Minimum Viable Product

While we were a week behind our timeline, our original milestones were so ambitious that we still achieved our Minimum Viable Product (MVP) very early in the semester. This was essential, as we could start obtaining real feedback as soon as possible. In addition, a key component of our product was meeting with eating club officers to gain their support and data. This required us to demo our product for officers as soon as possible. Having a MVP done early allowed us to start demo-ing for officers two weeks before our actual public release. This turned out to be essential, as scheduling meetings with clubs took longer than expected. Furthermore, demo-ing for officers was great practice for the actual project demo, and meant that we already had a decent chunk of our product guide finished in the form of an officer guide which we had distributed to officers.

Minimal Reliance On Others

Naturally, other people are less interested and less invested in your project than you are. As such, be careful with relying on others for information, and be extremely proactive if you are reliant on them. Setting up meetings with eating club officers took a lot longer than expected, delaying our public release by nearly two weeks. (People do not respond to emails in a timely manner!) While this gave us more time to find and fix bugs before release, this meant that we had limited events left in the semester before students left. Being more proactive with contacting officers and anticipating rather than waiting for completion of our MVP would have helped tremendously with an earlier release.

Check For Compatibility

Not everyone uses the same browsers or has the same screen sizes. Having developers that used different browsers, from Google Chrome to Firefox to Safari, and different sized laptops helped naturally check for compatibility issues. Many Firefox issues for example may not have been found if not for one developer who primarily uses Firefox. Having a developer with a 13” display helped debug responsiveness on smaller screen sizes.

The Future Of ProspectAve

New Features

While we are happy with ProspectAve’s release, we have gotten a lot of awesome ideas from officers that we hope to implement in the future. One is theme night support, which would involve special coloring on the map, image uploading for theme night posters, and cool titles for events.

While adding new features sounds great, each new feature comes at the cost of compicating UI for both students and officers. On the officer side, it might be difficult to add more options to the event form next to the calendar due to limited space. On the student side, we need to find a way to display more information without cluttering the map and sidebar. These all have to be taken into account as we plan expansions.

Maintenance

Having frequent, transparent communication with Eating Clubs helps ensure accurate and timely information. Having in-person meetings with every club has helped establish this connection, which we hope to build on over the summer and future semesters. In terms of actual coding, the only maintenance involved is updating NetIDs as officers change from year to year, which takes minimal time.

Advice

- **Communicate changes.** Before you go on changing code, tell your teammates what you’re up to so two people aren’t writing over the same section. This helps avoid frustrating merge conflicts and possibly hours of lost progress.
- **Push and pull frequently.** Having large, impressive commits is not worth dealing with merge conflicts.
- **Get users early.** The more real feedback from end users you can get, the more you can help improve your product.
- **Keep your code clean.** Make sure to do the basics like commenting code, following uniform formatting, and using descriptive variable names. Explaining small details or where to find something wastes time.
- **Meet regularly.** Meeting at least once a week meant everyone was up-to-date. Weekly meetings really helped us meet our deadlines.
- **Design around UI.** Adding new features is pointless if your UI cannot appealingly support them. Be meticulous with your UI design, it is often what separates projects that catch on from those that do not.