CourseComb

Final Report

HJ Suh
DoWon Kim
Ada Nguyen
Matt Yi

Planning and Milestones

1. TIMELINE

We tried to be as specific as possible when writing up the timeline of our project in the Design Document. Our initial idea for the final product and the deadlines for a prototype, alpha and beta test largely determined our milestones. Since the product is a course selection website, we also took into account when Fall 2018-19 courses were released on the registrar and the timings of each year group's course selection period. We set very clear deadlines for each week, with the plan to implement all of our stretch goals.

2. EXECUTION

The timeline was very helpful as a reference to check whether we were making healthy progress. We mostly stuck to the deadlines we set for ourselves, with the exception of cases where we made decisions to include or exclude certain features. The fact that we were only making small style changes and implementing user feedback in the final week is testament to how our execution of the plan we set out at the start was free of any severe problems.

We were disappointed that we could not implement all of our stretch goals, such as exporting class schedules to Google Calendar and TigerHub. In the final weeks, we had to choose between taking user feedback into account and implementing our stretch goals. Throughout the process of developing the website, we realised more and more that our initial design decisions were often wrong or unnoticed by the user. Therefore, we turned our focus to exposing the website more to users and to incorporate as much of the feedback as possible.

Gathering user feedback was not difficult, since the development process overlapped with students' course selection period. However, we had very little time between the start of the project and the start of course selection period. By the later stages of our development process, we had to rely on reaching out to friends who had already decided on their course schedule. Faster implementation would have helped with gathering a wider and more diverse user base.

Product Design

1. USER INTERFACE

From the very start of the project, we were very conscious of the importance of deciding on the best way to display information to the user. It was imperative to choose a display where different variations of course schedules were easy to differentiate in one look. We had particularly long discussions in the designing

process about how to optimise the way in which the calendar interacts with the user, and what information to display or hide from the user.

We were set on making the application a single page website. Although the course selection process is split largely into three steps on our website (choosing courses of interest, choosing a course combination, choosing precept times), it did not make sense to make separate pages for each of these steps. We wanted the course selection process to feel as fluid as possible, and therefore decided on rendering most components of our website dynamically rather than having separate pages for each component.

Making a single page website meant that we had to be very careful not to overload the page with too much information. Making decisions on which parts of the website should be dynamically rendered was difficult and we struggled with these decisions throughout the development process. We decided from the start that the filter should only appear at the press of a button, but we made decisions on how to render other components mostly based on user feedback.

The most significant departure from our initial design was the calendar display of different course combinations. We quickly realized that our initial sketch greatly restricted the size of the calendar display, since we used a lot of plain text to convey information. We decided that a visual representation of a course schedule was much more effective than a textual representation. Increasing the size of the calendar meant that there was less space in other parts of the website, such as the course queue column. This forced us to opt for dynamically sliding buttons for links to the registrar page, course evaluation page and deletion of course from the course queue. This sliding button feature, which was motivated by practical reasons, turned out in the end to be one that users often specifically singled out as particularly aesthetically pleasing.

The user interface component that we take the most pride in is the interactive calendar. The biggest problem we identified with existing course selection websites was that they show all classes of all courses on the calendar display when users add courses to their course queue. Our goal was to allow users to choose precept/lab/drill times one course at a time to make the job more manageable. We believe we achieved this by creating buttons below the calendar for each course that can be clicked/unclicked to display/hide precept times for that course. In terms of functionality, this was the feature that users were most enthusiastic about and was clearly an improvement on existing course selection websites.

2. WEBSITE NAVIGATION

When we first tested our website with new users, we were shocked to find that navigating through the webpage was not intuitive at all. Users were often confused as to what the core functionality of the website was. It became clear that we had fallen into the trap of becoming too immersed in the project. What seemed

obvious to our team was incredibly unclear to users. There was an urgent need to significantly change the business logic of the website and offer clearer guidance to the user.

Until the alpha test, we had two separate buttons labelled as 'Create My Schedule' and 'Update Filters'. Alpha test user feedback showed us that it is confusing and misleading to the user to have two different buttons that were very similar in their functionality. We incorporated this feedback by getting rid of the 'Update Filters' button and by making the filters pop up when the user clicks on 'Create My Schedule'.

A key feature of the website is its ranking of course combinations by the preferences set by the user. However, when we tested our product with users, many of them were not aware of this feature until we mentioned it to them. Users suggested that we change the label of the second column from 'Possible Combinations' to 'Possible Combinations Ranked By Preference'. We also decided to add number labels to course combinations to better convey the fact that the list is in ranked order.

Furthermore, we decided to write a set of instructions to help the user get started. Instead of adding the instructions to a separate page, we display a step-by-step guide in the space where the calendar appears. The calendar only replaces this text when the user clicks on a course combination. Therefore, new users can read these instructions carefully as they navigate through the initial few steps.

3. DATABASE DESIGN

The database can be split into course and user data. For course data, we adopted the database design of previous course selection projects and did not encounter any major problems. We decided that we should store user data so that we can restore what users were working on when they return to the site. For each user, we store their courses of interest, filter inputs, favorited schedules and course combinations. This is why when users return to the site, the main page has the course queue, course combinations and filters in the state they last left them. Also, users' favorited schedules are saved and can be accessed on the favorites page.

Product Implementation

1. FRONT-END: BOOTSTRAP, FULLCALENDAR, JQUERY

In our Design Document, we proposed using React for our frontend as it was recommended from previous groups such as ReCourse. However, after consulting Lance, our project TA, we realized that using React would be unnecessary for our purpose and might hinder the development timeline because of its steep learning

curve. Thus, we decided to only use Bootstrap (with some customized CSS), jQuery, and vanilla JavaScript for our front-end.

Looking back, it was a good decision as we were able to fully design the front-end for our need with the aforementioned stack alone. In particular, Bootstrap was good for making the website responsive and nice-looking, and the combination of jQuery and vanilla JavaScript was powerful for our application's interactive and dynamic nature.

In addition, we also used FullCalendar to display the course schedule. FullCalendar APIs offer a comprehensive set of tools, from saving events to dynamically changing visuals from user's input, which was suitable for our needs. Furthermore, the events rendering could easily be changed through vanilla JavaScript, which allowed us to customize things such as coloring each course with different colors, displaying un-selected courses with different opacity, hide/show options for non-primary meetings, etc.

2. BACK-END: DJANGO, BEAUTIFUL SOUP

In the back-end, we used Django (Python) because of its simplicity and the fact that a group member already had experience with this framework. In addition, since previous projects in the category of course selection such as ReCourse and ReCal also used Django, it would be efficient to leverage some of the open-sourced code for our project. Furthermore, the nature of our application requires several database queries, so Django's object-relational mapper (ORM) layer is suitable for the project as it provides flexible functionalities for making queries and has built-in protection from SQL injections.

Beautiful Soup is a Python library that is often used to parse data from web pages and was used in our scraper. It collects data from the registrar page and returns all the information in JSON format. Scrapers from previous projects used this library, and we simply decided to make use of previous works.

3. DEPLOYMENT: HEROKU

It was very simple and easy to work with Heroku, and there were useful features built into Heroku that we took good advantage of. There was extensive documentation available online on how to deploy Django applications using Heroku, so it was natural to choose Heroku to host our website. Heroku can also be integrated with Github, which made it incredibly easy to deploy code. We also use the Heroku Scheduler for scraping purposes.

4. VERSION CONTROL: GITHUB

We used Github for collaboration and version control. Since the task for each member was divided clearly, there was no need to create separate branches and we all pushed to the master branch. There were very few merge conflicts, so our

approach worked and helped maintain the simplicity for collaboration logistics. We kept a separate folder for our Heroku version, which helped ensure that new features would be tested thoroughly before being deployed to production.

Testing

1. SELF TESTING

We extensively self tested each stage of our development. This was most important in ensuring the correctness of our filtering algorithm. We would input several courses and set the filter to see if the returned combinations made sense. As a sanity check, if the results indicated that there were no possible course combinations, we would try to see if it was possible to create a schedule with the restrictions through other tools like ReCal. This allowed us to catch any edge cases not previously included in the implementation.

Similarly, we also self tested the display of the calendar to see if its interactivity made sense. We would use a variety of courses: courses with several primary meetings (e.g. CHI 101, PHY 103), courses with different types of non-primary meetings (e.g. CHM 201 has both labs and precepts), courses without any meetings (e.g. COS 497), courses without any non-primary meetings (e.g. LIN 302), etc. This helped us expand the implementation of the show/hide button and selection options on the calendar to accommodate a variety of courses.

2. USER TESTING

Our user testing was done primarily through asking friends to try out our live website and sending emails to listservs with an anonymous feedback form. This was most helpful in refining the UI/UX design of CourseComb, particularly the color scheme, ranking of course combinations, and instructions for new users. These changes are detailed in the Website Navigation of the Product Design section above.

In addition, user testing also helped us discover some backend bugs. The most important bug was that our filter was not saved properly for a new user who logged in through CAS for the very first time. Since we were only able to self test with our own Princeton credentials, which had been logged in with CourseComb before, we would not have been able to discover this bug without feedback from users.

3. AUTOMATED TESTING

We automated the test for our filtering algorithm by using Python's unittest module. Since the algorithm's main job is to create combinations by taking into account courses' time conflicts and user's preferences from the filter, we created a main automated test TestConflict to check if the algorithm's conflict-resolving was

correct. We did not create a test for the correctness of the filters because they were mostly implemented through Django's get/filter functionalities.

TestConflict would create a test user and choose 7 random courses to put in the course queue to create a 4-course schedule. We chose the aforementioned numbers since user testing suggested that on average people were interested in 7 courses and would like to create a 4-course schedule. Besides the number of courses, the filter is set in the default state. After course combinations are generated, the test goes through each combination and checks if there are any conflicts in primary meetings between courses.

The automated test was done in addition to self testing and ensured that there was no bias since the chosen courses were random. We would also have liked to automate checking whether it is a false alarm when no combinations are generated; however, we could not think of a way to automate such a test besides self testing and manual comparison with ReCal.

Final Reflections

Our team is satisfied with the product we have created, although we acknowledge the fact that there is still plenty of room for improvement. There are 3 areas in particular that we would like to continue working on before the start of the Fall 2018-19 semester.

1. FASTER ALGORITHM

The current algorithm for calculating course combinations is very much a brute force method. While the processing time currently never exceeds more than a couple of seconds, we worry that if we have more users using the website, the slow speed of computations can be detrimental to the user experience.

2. EXPORT TO GOOGLE CALENDAR/TIGERHUB

In the future, we would like our website to enable users to export their final course schedule to both Google Calendar and Tigerhub. This would be a natural extension to our current application since it would complete the full course selection process from finding interested courses to ultimately enrolling.

3. SHOPPING PERIOD MODE

This is an idea we had from the very start of this project. Most students 'shop' for courses during course shopping period, and it is often difficult to plan out the optimal shopping schedule to try out as many courses as possible. With 'shopping period mode' implemented, students would be able to input all the courses they want to try out, and the website will display a schedule that maximizes the number of different classes the student can attend. Since during shopping period mode students are not obligated to attend every lecture, this problem is similar yet different from the problem we solve with our current product.

Advice for Future COS 333 Students

We recommend future COS 333 students to be brave in seeking user feedback. It is important to get a minimum viable product running as quickly as possible and to show it to new users. Our team was late to deploy our website to the university student body and struggled to get willing participants in our alpha and beta tests. It is also important to thoroughly test code before moving onto other parts. Some code that we had written at the early stages of development had a major bug that we did not realize existed until the final few weeks.