Technical Proficiency

Professional Proficiency

**Sprint 1 – Getting Started**

By design, this sprint was not especially demanding, whether you look at it from a technical or professional perspective. We only had five stories to complete between us (which we could split evenly between our group of five), and the only technologies we used were HTML and CSS. My story for this sprint was perhaps the least demanding of all: on paper, all I had to do was get our site hosted on Github Pages, which was a matter of reading a short tutorial and clicking like three links.

Because of this I extended some extra effort towards writing some basic CSS early on in the sprint, so that the site was more presentable when the time came to have our work reviewed. As the principles of Agile Development tell us, our highest priority should be the "early and continuous delivery of valuable software", and form is an (often underappreciated) element of function-- it's much harder to use a site that is hard to read or navigate.

Michael also added the Bootstrap library to our repertoire, which influenced the HTML tags and classes that the other three members used. It was this CSS file that I modified, mostly to fix some padding. I also worked on the footer, following an <a href="<https://matthewjamestaylor.com/bottom-footer>">online guide</a> I found to keep it presentable. Finally, I did delve into the HTML of my other group mates' pages on the last day, to fix some minor positioning issues caused by slight differences in div tag nesting. In the industry, something similar to this process would happen during pull requests, *before* any changes were approved.

**Sprint 2 – Mobility**

For this sprint, I was kept on CSS duty: this time, I added code for responsive design, so that the site could be used on mobile devices. As far as the principles of Agile development are concerned, I think this was a fairly effective deployment choice: since I was the one who started the CSS on the previous sprint, I had a firmer idea of where to take things than another developer would have, which translated directly into faster and higher-quality work. This was also an early expression of the 11th principle: "The best architectures, requirements, and designs emerge from self-organising teams." Since the people calling the shots were the same as the people "on the ground", as it were, we were able to make intelligent ticket assignments based on our previous work.

From a learning standpoint, we had Johnny (who had basically no experience with CSS) take over on the base CSS, to create a more "final" look than what I'd written during the last sprint. This was a learning experience for me, too, since I was the main person he consulted for advice during this sprint. (See <a href="insights2.html">Insights II</a> for a more in-depth reflection on this.)</p>

**Sprint 3 – Learning Laravel**

This sprint was more technically demanding than the last two, since it involved learning an entirely new "technology"-- the Laravel framework for PHP, which makes a pretty significant departure from the PHP that I'd used up until this point in Web 2. On the other hand, the increase in technical demand did not reflect an increase in output, with only one major feature needing completion: implementing an "Edit Semesters" page, one of the "load-bearing" aspects of the finished application on which many of our client's user stories would rest.

Professionally, I think this sprint marked the beginning of a sort of "competence split" between members of our group. Michael and I took on the responsibility for writing the version of the Semester tool that we eventually implemented on the live site, which inherently gave us experience working with Laravel itself, and meant we could maintain our pace on the next few sprints (all of which used heavy amounts of Laravel). The other three members of our group had mixed results in the next three sprints-- Joe had no trouble with even quite technically challenging features, but both Josh and Johnny struggled to make much headway in Laravel on their own, which I think has its roots in their lack of contribution during this sprint.

If I were to take a leadership role on a similar project in the future, I think I would focus more on making sure that *every* member of my team was at least comfortable with the language/framework we were using, especially early on in the project. "Mark twice; cut once," as the old proverb goes.

**Sprint 4 – Transferring To Laravel**

This sprint started off with a thorough exercise in efficiently assigning user stories to developers. The overarching plan was for this sprint to be the one where we ported all of our existing functionality over to Laravel. As a secondary objective, we also wanted to implement the backbone for the features that we needed Laravel for.

Since I was the one member of the group without an “original page” from Sprint 1 to port over, I took on the lion’s share of the new features in Laravel. Specifically, adding Assignment and Course functionality on top of the Semester functions that we’d added last sprint. On a technical level, this was mostly a matter of repeating the work we had done on the Semester page, modulo variable names and the various fields not shared between the three tables. I also worked on the .blade.php file for the Creator Information, since that page’s original creator (Joe) had been assigned to work on a login system with admin privileges.

Overall this was reasonably simple: I did run into a few issues when it came to splitting the information boxes on the Creator Information page into two columns, because it transpired that Joe had formatted them as CSS tables to begin with and I was trying to use CSS tables to put them in a 2x3 grid. I managed to solve the problem by refactoring Joe’s CSS to remove the nesting table problem, while at the same time preserving the look of the boxes (which I thought was good UX design). The mobile formatting was another problem entirely, one that I eventually decided to add as a new story for a future sprint.

Maximising Work Not Done

As far as professional proficiency goes, the main attraction was definitely at the beginning of the sprint, when we were organizing which stories we wanted to complete. Part of maximising the amount of work not done-- a core tenet of the Agile philosophy-- is recognizing when multiple stories can be completed simultaneously.

<p>For example, in this sprint we wanted to <strong>a)</strong> port all of our pages over to Laravel, and <strong>b)</strong> update the formatting for some of these pages. If we had left the tickets under <strong>b)</strong> for another sprint, then all of our formatting work for this sprint would have been wasted: so, Michael and I made sure to integrate those tickets into this sprint, giving them to the people already working on porting those pages over. (For example, we assigned Michael the "port Course Information to Laravel" and "split Course Information into two columns" tickets, since they applied to the same page.)

**Sprint 5 – Integrating The Tables**

**Sprint 6 – Final Pass**

In this sprint, we focused more on polishing existing features than adding entirely new ones. For the sake of staying Agile, we'd left these tickets for the last sprint, since they didn't add any pressing functionality on their own. Michael took the task of fully automating the core date table, Joe wrote seeders to make testing and displaying our other features less of a headache, and I finally got to finish off the main data-related pages, formatting the information they provided so that they were far more readable than the simple HTML table of before.

We had a bit of trouble assigning the tickets for this sprint: this point is where we ran out of "simpler" Laravel to write, and all five of the major tickets we had left to assign had pretty high "velocity" scores. This made it hard to find a distribution of ticket assignments that suited each individual group member's ideal velocity. We eventually settled on giving our less confident group members a somewhat higher velocity than usual, with the proviso that they should ask the rest of us if they ran into a problem. Somewhat unfortunately, this ended with neither of those two tickets being completed at all.

I'm not sure what the "fix" for this issue would be in a professional setting. I suppose a professional Agile process would never run out of tickets to work on, or be in a position where leaving tickets for the "end" of the project would make sense: the sixteen-week limit that this assignment had to be structured around would just not exist in a professional setting, since the whole Agile structure chafes against the very idea of a discrete "ending" to any project. In Agile development, no product is ever completed; it's just made more complete than it was before.

**https://github.com/SoftEnOP/semester-tracker-dunedin-thunder-fingers/milestone/5?closed=1**

**https://github.com/SoftEnOP/semester-tracker-dunedin-thunder-fingers/milestone/6**