Maria Strasky, Sammie Jiang, Reid Oliveira, Erik Dolinsky

Pique

Reflection Statement

While the project was able to come together in the end, the greater takeaways from the project are the lessons we learned along the way. If we were to do the project again with the following lessons learned, the project would have come together not only in better quality, but with more features as we would've gotten off the ground sooner.

The first thing we learned is about estimating work for a certain size team. We were overly ambitious about our project in the beginning, completely overestimated the amount of work 7 people could do in the time constraint of the project, which is really just a few short weeks if it was full time (40 hours/week) development. Furthermore, when we discovered our team was 4 people instead of 7, we arbitrarily dropped only the mobile app part of our project. This was a mistake. We should have sat down with our 4 person team and redesigned our scope to fit. Had we sat down formally to refit our project scope we would have realized that the amount of work dropped was not equal to the 43% reduction in anticipated team size. Our failure to redesign scope as well as our inexperience with time estimates led us to overload ourselves and resulted in a poorer product.

A second lesson learned is to spend more time just learning your tools and their limitations, both through research and practicing on code that is not for your project. While this seemed like a waste of code (and time) originally, this would have actually saved us time down the road. Two examples of this are the wiring together of our back end components and passing data between back end and front end. After we created all the modules that formed our back end, we had to wire them all together in the Play framework. Without going into too much detail, 2-3 days were lost trying to get everything to talk to each other and set up correctly, all which would have been avoided by spending a bit more time with the Play example project and adding some pieces to it. As for data passing: we chose Google protobufs to serialize our data to binary for storage, and decided to then use it to serialize and data to the front end as well. Again, many hours were lost trying to make this work until realizing we were unable to use the generated JavaScript with our project. Additionally a little bit of extra research showed us that while there is an open source third party library for parsing the protobuf data on the front end, the data savings when compared to regular JSON wasn't remotely worth the effort. We then switched to serializing data using GSON (Google's JSON serializer) and implementing that took just a few minutes.

The final and biggest lesson learned is about design and communication, especially relating to working in parallel. Right from the beginning of the project our goal was to start working and make a bare bones product as quickly as possible. We immediately split off and started working in parallel on different pieces, intending them to come together at the end of the first or second sprint and have our product ready. From that point it would just be adding features. Unfortunately, we did not design the interactions between modules well enough for this strategy to be successful, leading to lots of wasted time redoing work because the completed modules didn’t work together. This design and communication issue perpetuated through the project as we felt the time constraints setting in. We continued splitting off work in parallel without communicating expectations module dependencies and ended up with duplicated work, or work that needed to be redone. This part of the project is where we learned the hardest that design is a wicked problem: designing all the interactions without actually writing code was extremely difficult but writing the code without designing the interactions caused a wealth of problems.