Zackery Steck
M08707502
CS5002 - Senior Design II
4/10/2020

Final Self Assessment

Part A:

My contributions to this project include both back- and front-end development and support. In terms of backend development, I: implemented the server monitoring agent in GoLang, created/modified/migrated databases to support the real time entry and aggregation of server metrics, and implemented api routes for client submission/retrieval of information. On the frontend, I implemented the real-time charting libraries necessary for visualization in tandem with AngularJS, as well as implemented the controllers for managing the middleware service calls and data bindings.
In my initial assessment from CS5001 in Fall of 2019, I identified 5 key skills that I hoped to improve upon throughout the course: design of application pipelines, implementation client-server architectures, data aggregation, data mining, and data visualization. I applied and built upon these skills in a way that other courses at UC have never allowed.

Designing Application Pipelines: I think my understanding of the design process has been greatly improved as a result of this course and our project. Prior to designing this project, I hadn't been afforded the opportunity to work collaboratively with a team during the design phase of a project- my only professional experience was with joining teams post-design. Getting to have a voice in how the application framework is designed from the ground up allowed me to offer some of my favorite web-app technologies as suitable components in our framework. Consequentially, I then got to reflect back on how I would have designed things differently.

Implementing Client-Server Architectures: Going into this process, I'd had prior experience with client-server architectures from OS and Software Engineering. However, this was an opportunity for my to apply that knowledge outside the classroom and implement a thin-client for scraping performance metrics from host machines. This skill benefited greatly from the work I did with the GoLang monitoring agent.

Data Aggregation: As a part of our CS curriculum, we cover data aggregation as a part of the Database Design course. However, most of the course is theoretical and we are not offered any opportunities to actually write SQL- most of what we did was algebraic. This project really allowed me to evolve my DB management abilities, especially when it comes to the aggregation of large tables into something more manageable. This skill benefited greatly from the work I did with the aggregation view in PostgreSQL, that took data from second-intervals and aggregated it into minute intervals.

Data Visualization: I was exposed to many facets of data visualization using the MEAN stack during my first co-op rotation in 2016, so this aspect of the process was not new to me.

However, being able to apply my knowledge to a new project after years of not using it was an excellent exercise. This skill benefited from the exposure to AngularJS and chartJS that this project afforded, in which I was able to design a dashboard for viewing important system stats.

Overall, I feel that I am a more competent web-designer, database admin, and backend dev than I was a short time ago. Some of these improvements are the result of the co-op program, but this project offered a rare reflection opportunity.

My successes and obstacles were actually both from the same two tasks: The server monitoring agent and the data aggregation. Before implementing the server monitoring agent, I'd never written code in GoLang, so this proved to be a fundamental obstacle that I had to overcome if this project was ever going to get off the ground. Consequently, learning GoLang and implementing a fast and customizable monitoring agent was a success. Likewise, my only experience with database administration was from Database Design- I'd never had to perform any formal db admin work in a professional manner. As a result, a major obstacle that reared its head late in development was the real-time server metrics. Since the monitoring agents were transmitting data every 1-5 seconds, the database grew in size rapidly. The size of the datasets caused issues with visualization on the front-end- querying was slow, so the UI was unresponsive/sluggish. However, I was successfully able to apply some of the aggregation tactics we learned in Database Design to this project and massively reduce the query times (from the order of minutes to milliseconds). This allowed for the UI to be more responsive, overall resulting in a more polished product.

Part B:

Our group accomplished a lot over the past two semesters, namely the design of an application pipeline supporting the real-time extraction, transmission, and visualization of system metrics. Although we did not reach an end state that I'm satisfied with (I would love to see the ability to add user defined conditions, I feel that we've delivered on what we initially set out to create. Throughout the process, I learned a lot about our group dynamic and how big of a role communication plays in the development process. The members of my group have been friends for years, so communication was a non-issue. However, I had a different schedule and courseload than the rest of my groupmates, making finding a common time for all of us to meet and discuss a bit of a challenge. That was more of a personal challenge than a group challenge, as my group always kept me in the loop on any meetings that I was unable to attend. Which leads me to our greatest successes- division of labor and communication. I feel that, with us being such close friends, we all had a thorough understanding of each others strengths and weaknesses. This allowed us the ability to plan the division of labor around others strengths. However, that is not to say that every member exclusively got to work on components in which they excelled- actually the opposite. This insight allowed us to give each other learning opportunities- for example, I haven't had much real-world experience on the database side while Vivek/Kyle have, so they allowed me to work on the aggregation mechanism and some schemas in order to provide me a learning opportunity. I struggle to think of an aspect of our team dynamic that was unsuccessful, but my only complaint (which is outside all of our control),

is the COVID-19 pandemic. Due to my different schedule, meeting with my groupmates was already a struggle, but we made an effort anyways- collaboration in-person felt like a mini hackathon. However, once social-distancing orders were in-place, this was no longer viable.

I feel that my efforts were comparable to that of my teammates, and I think we all put equal amounts of effort into this project. Giving an exact division of labor would be 25/25/25/25 for Michael, Kyle, Vivek, and myself. I think all of my groupmates deserve special recognition, as the project would be DOA without them. Vivek and Kyle really stepped up on the database side of things. Michael and I had little experience with db admin work, so we would've hindered their ability to rapidly develop a functional schema. However, once things were in a functional state, they were more than happy to allow Michael and I to get hands-on experience with Postgres. Michael really excelled at UI/UX, as this is his real-world job. He brought professional experience and insight to the table that no other group member has. This allowed are UI design to be almost a non-issue (and I say almost, because he prefers C#, and we don't like that). Kyle, aside from excelling on the database side of things, also picked up a lot of slack on both the backend and the frontend. He designed various components from the disk data display/api. Vivek brought professional webapp experience to the table that none of us had seen before, and really pushed our backend development forward. He had methods for creating routes and directory structures that I hadn't seen before and was really impressed with.

Overall, my group performed wonderfully. Everyone pulled their weight, and everyone communicated extremely well- everyone knew what everyone else was working on at all times.