Jovan Martinez-Saldana

SNHU, CS-470-H7085

December 13th, 2023

https://youtu.be/XXAnJVjpyUs

CS 470 Final Reflection

In terms of experience and strength, I had intended to continue my CS degree so that I could become a Systems Engineer in the company I work for currently. I'm not really that well versed with coding, I often struggle with remembering the simplest terminology and the entry level expectations. This contradicts my original endeavor of becoming a Software Engineer. The first two years of CS are very front-end related work. I was really interested in coding because of this. On to the more complex classes that occurred three plus years in, I realized that most of the technical work I was just really bad at since I didn't have the memory retention for it. I basically do systems engineering work now, and really like the documentation and requirements type of work it entails, but more importantly it gives me a position where I can have a general idea of software and hardware still, without needing to dive too deeply.

This class at face value would directly help accomplish what I stated above since this is the final class within the Bachelor of Science degree. However, I would like to bring up that the class has taught me a lot about file structures, how websites are designed and stored, as well as overall security and necessities needed to successfully operate a software related product. I often have to associate with a command line at work, and I must say, both the first and second iteration of this Full Stack Development course have greatly increased my capabilities. I feel more confident, and I tend to ask less questions.

A big thing that would "cripple" my self esteem and honestly probably deterred me from Software altogether was the troubleshooting process. It felt so disheartening to troubleshoot an issue for hours on end that would resolve maybe one small aspect of my work. There was a lot of troubleshooting I needed to do in the class, maybe not to the level of impact that I had experienced before, and a lot of problems I would say were on my end and my machine's doing, but I didn't feel dread troubleshooting issues with the assignments in this course. I don't think I will ever be a crazy developer, but I think I hardened the skills I will need for the path that I do want to take.

I would handle scale and error handling by considering microservices or serverless architecture. Microservices allow you to scale specific components of your application independently. You can deploy and scale only the services that require more resources, rather than scaling the entire monolith. Serverless platforms often provide built-in error handling and logging. Each function execution is isolated, enhancing fault tolerance.

For cost, it can be more predictable for individual functions, as you typically pay per execution and resource usage. However, it can be challenging to estimate costs accurately for complex applications with many interdependent functions.

Finally, I think that containers are more cost predictable than serverless. Containers can be more predictable because you have more control over the underlying infrastructure. However, it requires more manual management of scaling and resource allocation so its not always necessarily the deciding factor in development whether you should go with containers or serverless.