Aaron Hickey

21 April 2025

CS 470 – Final Reflection

Presentation: https://youtu.be/5_TXT1gOOA0

Experiences and Strengths

I think two major skills that I've been able to develop and master in this course are critical thinking and problem solving. There were a lot of challenges that I had to overcome early in this course due to older code being used and deprecated tools. However, these skills were key in overcoming these challenges and ultimately leading to success in this course, and would also make me more marketable in my career and as a professional in IT. As for my strengths as a software developer, I think it is my ability to analyze problems, to find the best possible solution through thorough research, development, and testing. I also believe I do well in maintaining industry standards when developing code and applications, ensuring that comments are clear and communicative, and that code is well structured and organized, using easy to understand naming conventions, and maintaining consistency throughout each component. Therefore, I think that roles I am prepared to take on are web and cloud application development, as well as mobile application development. I love analyzing problems, as well as modernizing and improving older applications, making them more secure, effective, and efficient for product owners and their customers.

Planning and Growth

As my web application could continue to grow, it's important to consider scale and error handling as more users provide for more variation in how they would attempt to use an application, where issues that weren't thought of originally, may become apparent in the future with new users. Therefore, with error handling, I would ensure that I have dedicated functions or Lambdas to handle errors using custom error pages or ensuring that the application stays operational where an invalid input would be "discarded" but logged and reported. By logging and reporting errors, it helps me identify and eliminate common occurrences for specific errors, to ensure that I find a quick and efficient solution and keep users satisfied. As for scaling, using serverless architecture is one thing, but also ensuring that my code is robust and secure, to allow it to handle high volumes of requests, and process these requests at optimal speeds, regardless of traffic volume, or geographical reach is key to ensuring that my application can scale appropriately.

When it comes to expansion, there are several pros and cons to consider. Some pros are the ability to reach more customers, provide additional services, increase "side"

products – or products that derive from the original, but also server their own purpose – for various types of user bases, increase geographical and cultural reach, and provide opportunities for partnerships. When you expand, so will your reach, providing rich opportunities to help those around the world obtain a service or product that you provide as well as find new ways to expand the application's potential. As for cons, more resources are required, higher costs, higher availability and accessibility needs for geographic and culturally diverse locations, more requirements – or "additions" – to the application, regulatory requirements, data security and user confidentiality. Expansion does come at a cost, a high cost with more resource demand, where initial costs are reasonable, but can grow exponentially when using serverless, in addition to paying employees or fulfilling partnership obligations. Additionally, geographic expansion raises needs to meet local laws and regulations such as GDPR in European Union countries, making it more challenging to ensure data protection rights are adhered to everywhere.

This is where elasticity and pay-for-service play a major role in decision making for future growth as scaling is important when considering pay-for-service, as higher scaling means higher costs, but saving on costs at lower scaling. And planning for that variation can be challenging as it can almost be unpredictable, where too few users and now your profit margins drop, or too many users and you scale higher, raising the potential to exceed profit margins or still fall short. This makes planning for using these forms of serverless architecture important, and that finding the right solution isn't just about the current needs, but future needs as well, as it does save on IT resources upfront, as well as a bit in the long run with less need for IT staff but can be costly if solely relying on cloud services. In most cases, it can be just as cost effective to have a hybrid environment, with more control, but also ensure scalability and availability for your application if the need arises.