Kendal Guizado

# CS-470 Full Stack Development II Reflection

## Experiences and Strengths

This course has been instrumental in helping me reach my professional goals by providing hands-on experience with cloud technologies, specifically AWS. The skills I developed here, such as containerization with Docker, managing serverless architectures, and integrating microservices, have made me more marketable as a candidate in the field of software development. These skills are crucial in today's job market, where companies are increasingly adopting cloud-native solutions to build scalable and cost-efficient applications.

As a software developer, one of my key strengths is my ability to quickly adapt to new technologies and environments. I've developed a strong understanding of both frontend and backend development, but this course has allowed me to expand that knowledge into the cloud, making me proficient in designing and implementing scalable, cloud-based applications. I'm now well-prepared to take on roles such as Full Stack Developer, Cloud Developer, or DevOps Engineer, where I can apply my knowledge of cloud services to optimize applications for performance and scalability.

## Planning for Growth

In terms of planning for the future growth of my web application, microservices and serverless architecture offer significant advantages for managing scale and efficiency. AWS Lambda, for

example, is a key component in ensuring that my application can automatically scale based on demand without manual intervention. By using serverless, I can ensure that I only pay for the resources I use, which helps manage costs as the application grows.

To handle scale and error processing, I would use AWS CloudWatch to monitor system performance, set up automated alerts, and manage error handling in real-time. This ensures that any issues are identified and addressed before they impact users. Predicting costs would involve using AWS's pricing tools to calculate how different workloads and traffic patterns affect pricing. For large-scale applications with unpredictable spikes, serverless is often more cost-predictable, as I only pay for what I use. However, containers might offer better control for workloads that require long-running processes.

The pros and cons of serverless versus containers depend on the use case. Serverless is great for applications with varying workloads due to its automatic scaling and pay-per-use model. However, it can be less predictable for long-running tasks that may benefit from the stability and predictability of containerization. Containers allow more control over environments and are often better suited for complex, persistent tasks.

Elasticity and pay-for-service models play crucial roles in future planning. The ability of cloud services to automatically adjust resources based on demand ensures that I can scale my application seamlessly without over-provisioning. This flexibility, paired with cost savings from only paying for what's used, will be central to planning future expansions and ensuring that my application can grow efficiently while minimizing overhead costs.

Video Presentation Link: https://youtu.be/NVAhRr97f14