McKenzie Jackson

CS 470 Full Stack Development II

June 29, 2025

https://www.youtube.com/watch?v=pogKKXQlPs0

CS 470 Final Reflection:

This course was extremely valuable as it provides the steeping stones for migrating a full stack application to the cloud. I learned so much, gained many life and technical skills while doing so. I have learned about deploying APIs, creating containers, and worked with AWS to create a fully functional applicational front to back. It was easy to understand everything working together in the bigger picture when all the parts were created. I felt I have gained many skills while developing the project that will be extremely helpful to become an ideal marketable candidate.

I have displayed many strengths as a software developer during this journey. I've developed skills in cloud architecture, API integration, IAM security, and data modeling using both MongoDB and DynamoDB. These tools are in high demand, making me a more marketable candidate for roles in full-stack development, cloud support, or DevOps. My strengths as a software developer include problem-solving, adaptability, and the ability to quickly learn and apply new technologies. I excel at breaking down complex technical problems and delivering clean, well-documented solutions.

Through this course, I've gained a solid understanding of cloud services, particularly how to use serverless architecture and microservices to build scalable and maintainable applications. I'm prepared to take on roles such as Software Support Analyst, Junior Cloud Developer, Backend Developer, or Cloud Application Engineer.

In the future, I can use microservices to isolate parts of my application (e.g., auth, payment, notifications), making it easier to scale and manage. Serverless functions (like AWS Lambda) are ideal for infrequent or lightweight tasks that don't require persistent infrastructure. To handle scaling, I would Use API Gateway throttling and Lambda concurrency limits for error handling. To predict costs, I would analyze historical usage patterns through CloudWatch metrics. Serverless is generally more cost-predictable at low-to-moderate traffic levels, while containers become more cost-effective with consistent, high-traffic workloads. Elasticity allows my app to automatically scale up or down based on traffic and pay-for-service ensures I only spend money when my app is being used. Both are critical in planning future growth without overcommitting resources or budget.