CS 470 Final Reflection
Name: Kholood Alkohali
Date: 06/30/2025
Assignment: CS 470 Final Reflection

Experiences and Strengths

This course has been instrumental in helping me grow as a developer and better prepare for real-world roles in cloud-based software development. By developing a full stack web application and migrating it to the cloud using AWS, I gained not only technical skills but also insight into project planning, scalability, and secure deployment.

Throughout CS 470, I have learned, developed, and mastered several marketable skills:

- **Serverless architecture design** using AWS Lambda, API Gateway, and S3
- **Containerization and orchestration** with Docker and Docker Compose
- **Secure coding practices** including input validation, role-based access control, and Zero Trust principles
- **Full stack integration** with dynamic frontend-backend communication via cloud APIs
- **Deployment and monitoring** using tools such as AWS CloudWatch and IAM

As a software developer, I've become more disciplined in testing, validating, and documenting code. I am now confident in working across the entire stack—frontend, backend, and cloud infrastructure. I bring strengths in adaptability, security awareness, and scalable system design.

I feel well-prepared to take on roles such as:

- **Cloud Software Developer**
- **Full Stack Developer (Cloud-native)**
- **DevOps Engineer (Entry-level)**
- **Site Reliability Engineer (Junior)**

Planning for Growth

As web applications grow, planning for scale, performance, and cost becomes critical. CS 470 has helped me understand how to design cloud-native applications with future growth in mind.

Microservices and Serverless for Efficiency

To handle future scale, my application can adopt microservices by separating critical components—like user authentication, payment processing, and data analytics—into independently deployable services.

Alternatively, using serverless functions (AWS Lambda) allows me to break down logic into event-driven components that only run when needed, optimizing cost and management.

Handling Scale and Errors

To support scale:

- I would use **auto-scaling groups** for containers and configure **concurrent execution limits** for Lambda functions.
- **CloudWatch logs and metrics** would be used to monitor latency and errors.
- **Retry logic, DLQs (dead-letter queues), and graceful error messages** will ensure that system failures do not affect the user experience.

Predicting Cost

Cost can be forecasted using AWS Cost Explorer and setting budgets. For serverless apps, I can predict cost by estimating **invocation counts, duration, and memory size.** For container-based apps, I'd calculate cost based on **EC2 instance size, uptime, and storage.**

Containers vs. Serverless: Cost Predictability

- **Serverless** is more cost-effective for low-to-moderate traffic apps, as you only pay for what you use.
- **Containers** become more predictable for high and consistent traffic, since you can reserve capacity.

Pros and Cons for Expansion

Serverless Pros:

- No server management
- Built-in scaling
- Lower cost at low traffic

Serverless Cons:

- Cold start latency
- Limited runtime and execution time

Container Pros:

- Greater control over environment
- Suitable for long-running processes
- Works well for complex apps

Container Cons:

- Requires more management
- Higher base cost

Elasticity and Pay-for-Use in Decision-Making

Elasticity allows services to automatically scale up or down based on demand. This helps optimize performance during peak usage and cost during idle periods. Pay-for-use ensures that I only spend on active resources. Both principles make serverless ideal for startups or unpredictable workloads, while containers make sense for steady, high-load applications.

Conclusion

CS 470 has given me a strong foundation in full stack cloud development. I've learned how to plan, build, and scale secure applications using modern best practices. I'm confident in applying this knowledge to future projects and professional roles, and I look forward to continuing my journey as a secure, scalable, and cloud-savvy developer.