Catherine Hawley

4/18/2025

CS 470 Final Reflection

https://youtu.be/wXA_gtUAJBk

Experiences and Strengths

This course has given me a lot of experience in Amazon Web Services, across several AWS technologies. This was my first experience with cloud-based applications, and will be a good foundation for further experience. I'd like to use this first look to potentially develop my own sort of application, rather than an upload of the existing question-answer style site. Adding AWS to my toolbelt will allow me to consider microservices as a solution for future problems, and based on the pros and cons of AWS, it is a strong solution for many problems.

As a software developer, I believe my strengths are attention to detail, pattern recognition, and troubleshooting. Paying attention to detail can be a matter of efficiency. Looking for an error caused by syntax such as ‘ instead of ` can take a lot of time, so attention to those small details is a valuable skill. By pattern recognition, I mean identifying the base pattern of code, or the template of it, and knowing which parts can be changed to suit the project and which parts are the pattern, the foundation of the code. Troubleshooting is a valuable skill, as so little code works correctly the first time, and using error messages to find and fix the issue quickly is a boon.

The types of roles I'd consider myself best suited for right now are likely entry level. I'm about to be fresh out of university with a bachelor's degree, but I've put all of my effort into absorbing the information from my classes and I've gotten a lot of skill in several programming languages and, more importantly, the development lifecycle itself. I can see myself as a member of a software development team for any type of web or mobile application, as a developer, QA tester or a peer reviewer.

Planning for Growth

Microservices and serverless structures lend to a lot of efficiency for all but the largest projects. Paying Amazon on a pay-per-use basis will generally be more cost-effective than buying hardware and paying a team to setup and maintain it. It's also a much faster process. AWS handles scaling automatically, and unless you're a large company looking to host millions of computes a month, the cost is better. Error handling using microservices can be a simpler process than handling errors manually. Relying on user-generated error reporting or automated messages from a self-managed server can be done, and with some thought can be fairly effective. AWS offers CloudWatch logs and Step Functions for finding and solving errors.

Predicting the cost of an application, using microservices or not, requires estimating the user base of the application. For a small business of less than 1000 customers, I'd expect a few hundred website visits or purchases per month. Each visit or purchase will consist of multiple functions: several page loads, a few database operations, some email being sent out, etc. Each of these functions has its separate cost associated with it. Starting small and getting an idea of demand before scaling up would be the smartest method.

Containers are probably the more predictable technology, cost-wise. It is easy to set limits on containers, and they're more longer-running. Microservices however charge per request, leading to higher variability. It could be a lower cost, but a less predictable cost.

Planning for expansion depends on scale and expectations. Expanding beyond demand results in a higher cost, which may not be offset by revenue if demand does not catch up. Failing to expand to meet demand is essentially leaving money on the table, and it could cause user frustration, leading to poor satisfaction, which could damage reputation. Elasticity is an important consideration, as it requires less of a thin line between wasting resources and failing to meet user availability.