Ben Miller

2/25/2025

Assignment name: CS 470 Final Reflection

Youtube link: **https://youtu.be/mCeXpMT6GM0**

Experiences and Strengths: Going into this course, I reflected on what I felt I was lacking in my journey to obtain a computer science degree, and I decided I didn't have a grasp on containerization. This course has provided direct insight into the value that containerization can bring to a full-stack application. The ability to drop the components of your application into buckets with all the necessary dependencies and run them individually was a gratifying process. Then going further and uploading the application to the AWS cloud, getting it all configured, and finally seeing it running from a browser was fantastic. While I don't feel I've become a master with local or cloud-based containerization, this was a critical introduction to a modern software development tool that I now have in my toolbelt, thanks to this course.

As a software developer, I now have a solid foundation in writing secure software in multiple languages. I have had a real introduction to developing AI, reverse software engineering, and containerization. I'm confident that I can succeed as a full-stack developer. I am prepared to assume a software engineering role, writing software as part of a team, designing and implementing solutions with success.

Planning and Growth: Breaking down an application into smaller components that communicate using microservices can be beneficial in many ways, but probably the most impactful is the ability to scale, since each component and service can be scaled. Therefore, if one portion of your application needs to grow to accommodate that need, the scaling doesn't disrupt the other components and services. More containers are added for scaling the components, and services can use load balancing to evenly distribute requests, allowing the application to handle the increased load.

Breaking down an application into smaller components also helps to isolate failures, preventing the entire application from crashing. It may be possible to have services that check for failures. When a failure is detected, the detection service could trigger a recovery event that could restore the failed portion of the application.

Cost prediction is likely easier to do with containers, but a serverless implementation could end up being cheaper if your service runtime is low and the service load is relatively low. This is because the serverless cost model is pay for usage. The cost model for containers is typically to pay for the number of containers, no matter how much or little they are used.

Expansion of a serverless application that was designed to be implemented into smaller components is a no-brainer because the process is so simple. This is one of the biggest advantages of a good serverless application design. The increased cost would be about the only thing to consider. If your application growth was enough that service activity was high all the time,

the cost may be prohibitive. There is more to consider with a local containerized application, like server storage. Ultimately, the most difficult to expand is an application that is not containerized.