Valerie J. Smith

August 13, 2023

CS 470 Final Reflection

YouTube Link: https://youtu.be/4JdsdYy5xLE

Experiences and Strengths:

In this course, I have learned how to create a full stack, serverless application. For the field of software engineering, this is a very important and relevant skill to have. As a Senior Software Engineer in my current place of employment, I still see a lot of other software engineers who do not have any front-end web development skills, and do not have knowledge in serverless applications.

In this course, I learned how to create and test a full stack application with the MEAN tech stack. This application was created as a containerized application by using Docker, which is also a new skill. I also got to work with Amazon Web Services and learn how to use the various components there, such as the Lambda, API Gateway, Dynamo DB, and the S3 object storage bucket. All of these skills will not only cause me to be more marketable, but they are also skills that I can take back to my organization and share with others. This course will help me reach my professional goals by adding to the skill set that I already have as well as opening some doors to becoming a Technical Expert, which is a goal of mine.

My strengths as a software developer are API's, Rest Services, Java and Spring Boot applications, Cloud computing architecture, front end client development, as well as UI/UX concepts. I have had a passion for web services and UI/UX design since the advent of a small website that I watched grow in the late 1990's, called eBay. I have never let go of that passion for creating and utilizing web application technology, and it has changed my life and inspired me

to obtain this Computer Science degree. While I do not plan to leave my current role as a Senior Software Engineer, I am hopeful to advance to the next level of Technical Expert or Systems Analyst in my current place of employment.

Planning for Growth:

The knowledge that I have gathered from this course about serverless cloud architecture I can add to my skillset as I have experience in working with private cloud technology. It was very interesting to construct and deploy a containerized application to a serverless environment. I have learned what components are needed and how to configure them to create the serverless environment. It is very different from a private cloud that have computes that still need to be maintained and configured by the application team, as well as very different from a physical server, which I also work with on a regular basis.

Microservices are services that perform a specific task and are engineered to be efficient and scalable. They are the opposite of the lengthy API's that pull back a lot of data that are still used today. Rather than having one large call that returns everything, a microservice is constructed to be more efficient and faster. A serverless architecture is one that will scale and help to manage the microservices in a more efficient manner. Handling scale and errors in microservices is important as with a lot of microservices to manage, it is important to ensure that they will scale and send appropriate errors. Scaling is something that should be determined while building and testing the microservice. If it is one that has a lot of traffic, then it should be one that should have the appropriate amount of resources allocated to it based on that priority. Tools such as Kubernetes can be utilized to manage scaling for microservices. Error handling should be incorporated into any web service, not only a microservice. As a part of the coding of the

service, proper error handling should be implemented for transient and non-transient errors. Tools such as Splunk, Dynatrace, Greylog, AWS CloudWatch, and others can be configured to assist in the monitoring and health of microservices and other web application components.

Predicting the cost of serverless microservices is difficult if there are a lot of unknown variables. If a microservice exceeds the usage of what is expected, the cost will be more than expected. Understanding how the services will be used and ensuring that the proper resources are allocated for the service will help in the long term with costs. Ensuring that the microservice is built for the serverless environment is also important to save on costs. Many serverless providers have tools that can be used to generate an approximation of what the costs will be.

Predicting costs in containers versus serverless depends upon the needs of the application. If a system is used by less users and not consistently, such as having times when it is not used at all, it can be more cost effective. Container architecture has baseline costs and there will be a charge for idle resources. This can be more predictable if the application has a low to medium, consistent workload. Services that are dynamic and change frequently can be less efficient in containers if the containers aren't managed effectively.

Some pros and cons of deciding whether to expand would ultimately depend on the use of the application and how much will be added. Some disadvantages could be that testing and debugging might become more challenging and new security issues may be introduced. If introducing a long running service to the serverless environment, it may be better to put it into an environment that is more traditional. The performance of the existing services may be affected and vendor lock-in is also a risk factor. Some pros would be that if the additional functionality is compatible and will not cause performance and security issues, the existing architecture has already been created and will not need to be reproduced.

The roles of elasticity and pay-as-you-go services are an important factor in future planned growth as they will determine whether it is beneficial to grow the existing serverless architecture, or if it is time to move the architecture into something more traditional, such as a managed compute environment. These concepts will help to determine if it is a good idea financially to add any additional implementations.