Matthew Powers

CS 470: Full Stack Development II

Final Reflection

October 19, 2022

https://youtu.be/5IaAIVAgwAs

When I was working through this course, I learned more than I thought I would. As I was working my way through all the assignments, it seemed more like a copy and paste type of thing. What I did not expect to see was the reason behind the choices made but that is exactly what happened. I saw what AWS really does in the background and why it works the way it does. For example, the first major thing we did in this course was to containerize the package that we built in the first module. As a result, we were able to send it off to any major device which had a docker daemon. By seeing the development work on an AWS system, I was able to consider the practical options available on each service. For example, when I was working on the DynamoDB system, I was able to think about the different options that would be applied for a library or a record company. Over the course of not only this class, but the program here at Southern New Hampshire University, I have had to think about the different assignments in a creative manner to find a solution given what I had learned. Based on the creative solutions I have had to apply not only in coding but also in my existing work life, I feel best about a position as a solutions architect or tester to analyze the existing structure for the boundary conditions.

When thinking about growth, scale and error handling, the easiest way would be to utilize the systems that are based on use. For example, when using a local server storage, we would have to pay for maintenance and the power whether they were being used or not. The same could not be said for cloud-based services. DynamoDB, as part of Amazon Web Services, is the one

with which I have the most experience. When using DynamoDB, we will only be charged for the access calls made to the information. Similarly, when using a local storage system, we would be subject to the maintenance and power costs for said storage system which will stay the same regardless of the frequency of the access calls of the information stored therein. When using the simple storage system or S3 of AWS, the cost of storage will be variable based on the frequency of access. When referencing the information sporadically, the cost of storage is limited to that of S3 Glacier, like long-term storage or archives. Between those two services, we see a limitation of costs based on reduction of access rather than a static cost whether we have a high frequency or a low one. We see these cost assessments based on local systems versus serverless systems such as AWS. When comparing containers and a serverless launch, we must consider how the container will be launched. The most widely used is Docker, to be seen on a Docker daemon, which requires a local storage solution.

The primary pros and cons of launch are ease of use and cost of use. If someone is understanding of MongoDB for example, they might find the limitation of DynamoDB requests too limiting. The requests for Mongo can be handled by python requests which can be automated as a result. While DynamoDB requests can still be handled by requests, the limitation for a single filter at a time can make the final request more time consuming than it needs to be. Despite this, the ease of use can make newcomers to a database handler feel more welcomed by DynamoDB. As growth on MongoDB is limited to onboard storage, DynamoDB is fluid and will grow to meet your needs. Each company will have their own growth needs and expected costs. It is through a thorough understanding of each of these that a viable conclusion can be made regarding cloud service or a local solution.