CS-470 Final Reflection Michael Linsenbigler

Final Presentation Link: https://youtu.be/_2EQaiHVWqI 10/14/21

This course will help me reach my professional goals by providing me hands-on experience with current technologies and services. I have been exposed to and now have familiarity with concepts related to containerization and cloud computing. I have a much better understanding of cloud computing and its benefits. I also understand how various cloud services integrate with each other to accomplish a full stack environment.

My strengths as a developer is my ability to take a larger task and break it down into smaller and more manageable components. This way, I can focus on individual goals and then expand them into the larger project scope. I have also become more proficient in building a basic framework of an application to ensure the major aspects function together. Once this is done, I move onto adding in customization and more details of the application's needs such as ease of use and visual interface improvements.

I am prepared to assume any entry-level position related to computer science or technology-related fields. I would also perform well as a junior software developer, systems analyst, web developer, or data scientist.

A serverless environment, specifically the AWS one that we used in this course, has multiple features to help increase efficiency of management and scaling. Using the S3 bucket storage will allow the storage needs to scale automatically based on how much storage is requested through the application. This can also be monitored and adjusted manually. Error handling can be taken care of using AWS Step Functions combined with the existing Lambda functions. The AWS Step Function allows for the creation of a separate error-handling workflow that can be coded and tested to account for almost any type of runtime error that may occur. Using this, we can set the response we need for individual errors to keep our application operating efficiently.

The cost of the application can be predicted based on the current usage numbers that we have gathered. AWS pricing is based on how many function calls take place and the storage pricing is according both how much storage is needed and how often it is accessed. Keeping an eye on the application's usage of memory, function actions, API calls, and storage needs is important to

accurately predict what kinds of costs are being incurred by its operation. For example, if the storage needs continually exceed the allotted size of the current pricing tier, we can elect to move our service into the next tier instead of paying the higher overage charges month after month. Running the application in the container model is more cost predictable because the resources allotted are configured within the container and we can code the application not to exceed the limits that we set. The serverless model will incur costs as long as the services are being activated and utilized even if we did not plan for the amount of executions and calls ahead of time.

When considering expansion plans, we would want to look at a number of factors related to our applications. Looking at how many API calls and functions activations would be very important when deciding on our platform. Resource needs such as memory, storage, processing power, and amount of backend queries could factor into dictating future plans as well. If we need to operate with a database or proprietary software that might not integrate into a cloud environment easily would also be a factor. Our ability to manage every aspect of developing and maintain the full stack application in-house or the need to offload some of that management to a cloud service would also be a key point to discuss in planning.

Elasticity and pay-for-service options should be considered a very high priority when planning for future growth. If done well, a platform that offers the right amount of elasticity for our application can ensure a high level of operating efficiency and help to eliminate unnecessary costs. Keeping in mind that if our application does not need a high degree of elasticity, using a service that offers it can also be a wasted expense because that feature is certainly built into the cost structure. The same is true for any pay-for-service options. You can have a basic structure that you pay a flat rate for each activation of your service. However, this can be expensive if the number of activations reaches a high enough threshold. In this case, you can opt for a different model that basically guarantees a certain number of activations each month but in return the cost-per is lower than the basic format. Again, if properly researched and planned for these features can have a heavy impact positively or negatively to the operating costs of a future application.