CS470 Final Reflection

Link to Presentation: https://www.youtube.com/watch?v=2h7EZeC2rI0

Austin Palmer

04/21/2024

Full Stack Development II has been an insightful class for learning about deployment and serverless architecture. I have improved my understanding of APIs and learned new skills with AWS Lambda, API Gateway, and S3. Additionally, I learned fundamentals in containerization and virtualization. With cloud services on the rise for corporate applications, understanding the architecture of a serverless application will give me an advantage at the start of my career. As a software engineer, I am a great problem solver and I utilize my business acumen to have a better understanding of the product and the business needs. I am prepared for a software engineer, web developer, or product owner role.

After exploring the capabilities of a serverless application and how easy it is to use, I can see why it is such a popular choice for architecture. One perk of web services is they automatically scale as the usage grows which could help to solve future user influx issues in organizations that are using a monolithic architecture. Web services also come with built-in testing and logging tools that can interact with the other services in your application to automate and track testing. Additionally, web services are pay-for-service so are only paying for the requests, storage, operations, etc that you use. Availability is another perk of web services due to the multiregional servers that are provisioned by the web service provider.

However, for organizations that rely heavily on legacy systems or have predictable workloads serverless is not as viable of an option. Web services are a relatively new development in the software world many organizations still rely on legacy systems that can't be used with web services. Additionally organizations with predictable workloads then it may be cheaper and easier to keep the operations on in-house servers.

For future applications, the decision to use serverless is dependent on the needs of the application and the frequency of use. For example, if starting a company such as an online retail website then it would be beneficial for you to use serverless due to the pay-for-service and scalability. The company is likely unsure what their customer base will look like that early on and they will want to be adaptable to handle a lot of traffic. With serverless you don't have to worry about the infrastructure so as programming languages and frameworks update, the dependency management is handled by the web service. This makes it efficient for working in large and small teams alike.