Carmen Kingery

Full Stack Development

Kaan Esendemir

CS 470 Final Reflection

06/24/2024

Presentation link: https://youtu.be/7xSQfGlz6PY

Experiences and Strengths

Throughout this course, I expanded on the knowledge gained in Full Stack Development I, where we built a full-stack application using the MEAN stack. This time, we focused on containerizing applications with Docker Compose. Before this class, I was unfamiliar with Docker, but now I can confidently navigate and use Docker Compose for any future projects.

Additionally, I learned how to migrate an application to a cloud-based platform using AWS. This was entirely new to me, but by the end of the course, I could efficiently create and manage Lambda functions and test scripts within minutes.

While I acknowledge that I still need to enhance my back-end development skills, my front-end abilities are strong and continuously improving with each project. As I look for new job opportunities, I am interested in roles within System Operations and Development. I am prepared to handle the background work of referencing and integrating systems into a cloud environment, given a system and operational plan.

Plan for Growth

When planning to grow a web application, using microservices and serverless architectures can make managing and scaling easier. Both systems can automatically adjust resources based on how much the application is being used. To handle errors, AWS Step Functions and methods like circuit breakers and retries in microservices ensure the system runs smoothly.

Predicting costs is easier with tools like AWS Cost Explorer. Serverless systems are straightforward because you only pay for the time your code runs. This makes costs predictable,

but it can get pricey if there's a lot of usage. Containers have fixed resources, which can simplify budgeting, but they might cost more if the resources aren't fully used.

Microservices allow different parts of the application to scale independently and provide better fault isolation, but they are more complex to manage. Serverless systems simplify operations and scale automatically, but they can have delays when starting and have limits on execution time. Elasticity helps applications handle sudden increases in demand, and the pay-for-service model means you only pay for what you use, making budget management and scaling easier. These principles are key for growing cloud-based applications efficiently and cost-effectively.