Brylene Patrick

08/05/2023

CS 470 Final Reflection

https://youtu.be/C4PrE78B7HM

This course has been instrumental in preparing me for the professional goals I have set for myself. Through CS 470, I have learned valuable skills that make me a more marketable candidate in my career field. Some of these skills include:

1. Cloud Computing: I have gained a solid understanding of cloud service concepts, containerization, and orchestration. This knowledge enables me to design and deploy applications in the cloud efficiently.
2. Serverless Architecture: I have learned how to leverage serverless computing, particularly with AWS Lambda, to build scalable and cost-effective applications without the need to manage servers. This expertise allows me to focus on application logic and user experience.
3. Data Modeling and Databases: I have explored different data modeling scenarios, comparing MongoDB and DynamoDB. This experience equips me to make informed decisions about data storage and retrieval for my future applications.
4. Security in the Cloud: I have learned how to implement security measures to protect cloud applications, including IAM roles, access controls, and encryption. This knowledge ensures the safety and privacy of sensitive data.

As a software developer, my strengths lie in my ability to quickly grasp complex concepts and apply them practically. I am skilled at designing efficient and maintainable code architectures. Moreover, I have a keen eye for problem-solving and debugging, which enables me to tackle challenges effectively. I am also a good team player, and I enjoy collaborating with others to deliver high-quality solutions.

With the skills and knowledge gained from this course, I feel prepared to assume various roles in a new job. I am well-equipped to work as a Cloud Application Developer, DevOps

Engineer, or Cloud Solutions Architect. These roles align with my interests and allow me to leverage my expertise in cloud-based development, containerization, and serverless architecture.

When planning for the future growth of my web application, I can leverage microservices and serverless architecture to produce efficiencies in management and scale. Here's how I would handle different aspects:

1. Scale and Error Handling: I would break down my monolithic application into smaller microservices, each responsible for a specific function. By doing this, I can independently scale the components that experience higher demand without affecting the entire application. Additionally, I would implement robust error handling and monitoring mechanisms to detect issues early and respond quickly.
2. Cost Prediction: To predict the cost, I would closely monitor the resource usage of my microservices and serverless functions. By setting up billing alerts and analyzing usage patterns, I can estimate future costs accurately and optimize resource allocation accordingly.
3. Cost Predictability: Serverless architecture offers more cost predictability since I only pay for the actual usage of functions, without the need to maintain idle resources. Containers might incur higher costs, especially when resources are underutilized.
4. Pros and Cons for Expansion: Several factors would influence my expansion plans:

   Pros of Expansion with Microservices:

   - Improved Scalability: Microservices allow independent scaling, ensuring better performance during spikes in demand.
   - Modularity: Smaller services are easier to maintain, update, and debug.

- Team Autonomy: Different teams can work on separate microservices, enabling faster development.

Cons of Expansion with Microservices:

- Complexity: Managing multiple microservices can introduce complexity and require robust orchestration tools.
- Inter-service Communication: Proper communication between microservices is crucial but can be challenging to implement.

Pros of Expansion with Serverless:

- Automatic Scaling: Serverless platforms handle scaling automatically based on demand.
- Cost-Efficiency: With serverless, you only pay for the actual usage, reducing idle resource costs.

Cons of Expansion with Serverless:

- Cold Start Latency: Serverless functions may experience latency on initial requests due to cold starts.
- Resource Limits: Serverless functions have resource constraints, which may impact certain workloads.

5. Role of Elasticity and Pay-for-Service: Elasticity allows my application to adapt to varying workloads, ensuring optimal performance and cost efficiency. Pay-for-service ensures that I only pay for the resources I consume, aligning costs with actual usage. Together, these factors play a significant role in making informed decisions for planned future growth, allowing me to scale my application efficiently and cost-effectively.