Rogan Page

CS-470

Mohammed Alam

21 Dec. 2024

CS 470 Final Reflection

https://www.youtube.com/watch?v=FIYgqFFvxqc

Experiences and Strengths

This course has been instrumental in advancing my professional goals by enhancing my technical skills and deepening my understanding of full-stack development and cloud services. I have mastered key skills such as developing APIs, integrating cloud services, and implementing efficient testing strategies. These competencies make me a more marketable candidate, as they align with industry demands for developers proficient in scalable, cloud-based applications.

As a software developer, my strengths include problem-solving, adaptability, and a thorough understanding of both back-end and front-end technologies. My experience with developing, deploying, and testing web applications in the cloud has prepared me to assume roles such as full-stack developer, cloud solutions architect, or DevOps engineer. These positions leverage my ability to create robust applications while ensuring scalability and efficiency in cloud environments.

Planning for Growth

To plan for the future growth of my web application, I would leverage microservices and serverless architecture to ensure efficient management and scaling.

Scaling and Error Handling:

Microservices would allow each component of the application to scale independently, reducing resource waste and improving performance.

Serverless solutions can handle dynamic workloads, automatically scaling based on demand while providing built-in error handling through retries and failover mechanisms.

Cost Prediction:

Serverless architecture offers better cost predictability for sporadic workloads because you pay only for what you use. In contrast, containers may be more cost-effective for consistently high workloads.

Using tools like AWS Cost Explorer or Azure Cost Management, I would track usage patterns to predict costs and optimize resource allocation.

Pros and Cons for Expansion:

Serverless Pros: Automatic scaling, reduced operational complexity, cost efficiency for unpredictable workloads.

Serverless Cons: Cold starts, limited runtime control, and dependency on vendor-specific solutions.

Microservices Pros: Greater flexibility, modular development, and easier debugging.

Microservices Cons: Higher initial complexity, potential for increased latency between services.

Elasticity and Pay-for-Service:

Elasticity ensures that the application can handle spikes in demand without over-provisioning resources. This is crucial for maintaining user experience during traffic surges.

The pay-for-service model allows for predictable budgeting and cost optimization, as resources are provisioned only when needed.

In planning for future growth, I would analyze workload patterns, weigh the cost-effectiveness of microservices versus serverless options, and ensure that the architecture remains flexible to accommodate new features and evolving user needs.