CS 470 Final Reflection

Beth Campbell

04/28/2024

https://youtu.be/ECXvAUupIyI

Full Stack Development II has been instrumental in equipping me with the necessary skills and knowledge to advance towards my professional goals in the field of software development. I have learned a complete understanding of various aspects of computer science, with a specific focus on software engineering.

One of the primary skills I have learned, developed, and mastered in this course is proficiency in cloud development, particularly in transitioning from traditional full-stack architectures to a cloud-native applications leveraging service, AWS microservices. This expertise is highly sought-after in today's tech industry, where companies are increasingly migrating their infrastructure to the cloud for scalability, flexibility, and cost-effectiveness.

I have honed my abilities in containerization using Docker, orchestration with Docker Compose, and serverless computing with AWS Lambda and API Gateway. These skills not only demonstrate my competency in modern development practices but also highlight my adaptability to evolving technologies and methodologies.

As a software developer, my strengths lie in my analytical mindset, problem-solving abilities, and attention to detail. I thrive in environments that require critical thinking and creative solutions to complex challenges. I am adept at collaborating with cross-functional teams, communicating technical concepts effectively, and adapting to rapidly changing project requirements. I have also picked up multiple skills while working as a Reporting Coordinator that allows me to understand the data within a project better and to combine this with software development to discover the root of problems.

With the skills and strengths I have developed through this course, I am prepared to assume various roles in a new job within the software development industry. A few jobs within the industry I have considered are Software Engineer, DevOps Engineer, Cloud Architect, and Full-Stack Developer. Leveraging my knowledge in cloud development and software engineering principles of design, develop, and deploy will be vital as a Software Engineer. In the role of a DevOps Engineer, my knowledge of containerization, orchestration, and automation to streamline the SDLC and improve deployment processes will play a major role. Applying my understanding of cloud computing principles and best practices to design and implement scalable and cost-effective cloud solutions for organizations will help as a Cloud Architect. My proficiency in both front-end and back-end development will help me build end-to-end solutions to meet business requirements and user needs as a Full-Stack Developer.

Microservices allow applications to be broken down into smaller, independent services, enabling individual components to scale based on demand. Error handling in microservices can be decentralized, allowing for granular management of failures within specific services. However, managing the orchestration and monitoring of multiple services can introduce complexity.

On the other hand, serverless architectures provide automatic scaling and built-in error handling mechanisms. Functions are executed independently, so failures in one function do not impact others. Cloud providers offer monitoring tools for easier troubleshooting. While serverless architectures offer simplicity in scaling and error handling, they may suffer from cold start latency and vendor lock-in.

Cost prediction is another crucial consideration. Microservices' cost prediction can be challenging due to factors like resource allocation and maintenance overhead. In contrast, serverless architectures offer more predictable costs as users are charged based on actual usage. However, unexpected spikes in traffic can lead to higher costs if not properly monitored and managed.

When considering expansion plans, both microservices and serverless architectures have their pros and cons. Microservices offer scalability, flexibility, and independent deployment but may introduce complexity in orchestration and monitoring. Serverless architectures provide automatic scaling, reduced operational overhead, and pay-per-use pricing, but may suffer from cold start latency and limited control over underlying infrastructure.

Elasticity and pay-for-service play significant roles in decision-making for planned future growth. Both microservices and serverless architectures offer elasticity, allowing resources to scale dynamically based on demand. Pay-per-use pricing in serverless computing aligns costs directly with usage, promoting cost efficiency and scalability. However, careful monitoring is required to prevent unexpected cost spikes. Microservices offer more control over resource allocation but may require more upfront planning and management.

In conclusion, choosing between microservices and serverless architectures depends on factors such as anticipated workload, development complexity, resource management preferences, and budget constraints. Evaluating trade-offs and aligning architectural choices with specific business needs and growth objectives is crucial for long-term success.