Explain how this course will help you in reaching your professional goals

Throughout my work in CS 470 Full Stack Development II, I obtained valuable skills in cloud-native application development that will support my future professional goal of becoming a software developer directly. By migrating a full-stack web application to AWS, I gained knowledge about the idea of containerization with Docker, serverless computing with AWS Lambda functions, and backend integration using API Gateway. These experiences have strengthened my ability to design and build scalable, efficient applications while also minimizing their infrastructure complexity. These technical skills make me a strong candidate for software development roles that require knowledge of modern cloud computing and architecture.

Synthesize the knowledge you have gathered about cloud services

Completing this course provided me with a deep understanding of cloud services and how they are able to be leveraged to develop cost-efficient and scalable applications. Utilizing microservices and serverless architecture provides flexibility when managing workloads and eliminates the need for local infrastructure maintenance. In particular, AWS Lambda can scale an application automatically based on demand by increasing or decreasing the number of execution environments created to handle requests. In addition to this, cloud storage solutions, such as AWS S3, offer high availability and built-in redundancy. This makes data management much more reliable. By incorporating these cloud-native solutions into future applications, I can create software that efficiently handles varying workloads, supports business growth, and has minimal overhead.

Explain several pros and cons that would be deciding factors in plans for expansion

When thinking about how to expand an application, it's essential to weigh the benefits and trade-offs of different cloud-native approaches. Serverless computing is great for its ability to automatically scale and reduce costs since you only pay for what you use. Despite this, it comes with limitations that include potential restrictions from cloud providers, like Amazon's short execution limit or cold start delays. Containerization off the cloud, on the other hand, provides much more control over performance and resource allocation, which can make it a solid choice for applications with consistent workloads. This approach does, however, require more effort and resources to manage local infrastructure. Choosing the proper approach depends on the specific needs of a project, whether flexibility, cost-effectiveness, or long-term scalability is the priority.

What roles do elasticity and pay-for-service play in decision making for planned future growth?

Elasticity and the pay-for-service model are both key factors when planning for future growth. In cloud computing, elasticity is the ability of an application to automatically adjust resource allocation based on real-time demand. This ensures when traffic increases, the system scales up to maintain performance, and when demand drops, the system releases resources to avoid

unnecessary costs. This concept is crucial because it optimizes for both efficiency and user experience. Without it, applications would suffer from slower response times or waste money on idle infrastructure. The pay-for-service model, another key advantage of cloud computing, allows businesses to only pay for the compute time that is actually used. Unlike traditional infrastructure, which requires a significant upfront investment in hardware and ongoing maintenance, cloud platforms handle infrastructure and charge for usage of their cloud. This further reduces a company's financial burden, enables efficient scaling, and lowers waste. By utilizing both of these cloud-native advantages, developers can build applications that are both cost-effective and highly scalable.

CS 470 Video Presentation: https://youtu.be/kzY8bYHEfGI