CS 470 Final Reflection

https://youtu.be/2iZVQWRYj1o

Garrett Strang

CS 470 – Full Stack Development II

Nizar Dajani

June 23, 2023

- **Experiences and Strengths:**

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

  - What skills have you learned, developed, or mastered in this course to help you become a more marketable candidate in your career field?
  - Describe your strengths as a software developer.
  - Identify the types of roles you are prepared to assume in a new job.

Throughout this course I learned a lot of valuable skills that will help me as I progress through my career. From older technologies from previous experiences and courses, to new technology that I had never used before, we were able to completely and successfully develop a full stack web application in a cloud environment.

From Full Stack Development I. we became accustomed to the MEAN stack development. Using MongoDB, Express.JS, Angular, and Node.JS, we built a web application. In Full Stack Development II. we used the picked up where we left Full Stack Development I. and implemented new technology, Docker Desktop and Amazon Web Services (AWS), to successfully migrate a MEAN stack web application into the AWS cloud. Although Docker and AWS were new to me as a software developer, I quickly realized how integral they are in web applications and how useful they are.

Throughout my growth as a software developer, I have learned that you can never stop learning. As everything is constantly advancing and changing, we as developers also have to advance our skills and change with the technologies we use. Early on in my Computer Science degree, I struggled. Between work, life, and school, I found it difficult to be able to truly grasp the complexities of software engineer. The turnaround from using one language to another was quick and the initial learning curve was steeper than I expected. As I have approached the end of my Computer Science degree, I have found that I retained much more of the information than I thought I did. Identifying my strengths as a developer is challenging, however, I would say that I have become very good at understanding various software technologies, programming languages, and taking a complex system and being able to break it down into simpler and more comprehensive building blocks. There are many intricacies and nuances in the software world. My biggest strength right now is that I am able to have a well-rounded understanding of many different complexities of computer science.

These strengths will certainly help me market myself to various software developing roles. At the current position I am in, I have the ability to be useful in different roles because I am not necessarily specialized yet. This will give me the chance to work through different roles like: backend developer, front end developer, data analyst, embedded systems, API engineer, cloud-based engineer, database architect/engineer, etc. I feel as though I have been prepared to be able to step into any of these positions and pick up on the specific roles within each position.

- **Planning for Growth:**

  Synthesize the knowledge you have gathered about cloud services.

  - Identify various ways that microservices or serverless may be used to produce efficiencies of management and scale in your web application in the future. Consider the following:
    - How would you handle scale and error handling?
    - How would you predict the cost?
    - What is more cost predictable, containers or serverless?
  - Explain several pros and cons that would be deciding factors in plans for expansion.
  - What roles do elasticity and pay-for-service play in decision making for planned future growth?

As this course has come to a close, I have learned much more of the intricacies of cloud development. One of the sayings I have heard more often as I have progressed through my Computer Science degree is that "you don't have to reinvent the wheel". I think that is one of the important things I applied while going through this Full Stack Development II course. Utilizing the cloud services and the myriad of tools we've learned so far and the tools available in AWS has shown me that understanding which "puzzle piece" to put where is extremely important. In AWS, there are several different ways that the AWS cloud approach can be used to produce efficiencies of management and scale a web application in the future. The power of all the AWS services is rather astounding when you think about it.

Scale and Error Handling (AWS-specific): AWS offers services like AWS Lambda for serverless computing and Amazon Elastic Container Service (ECS) or Amazon Elastic Kubernetes Service (EKS) for container orchestration. With AWS Lambda, you can easily scale individual functions independently based on demand. AWS Lambda automatically manages the scaling and provisioning of resources, allowing you to handle increased traffic efficiently. In case of errors, Lambda provides built-in mechanisms for retries and error handling.

For containers, ECS and EKS offer auto-scaling capabilities that allow you to scale the number of container instances based on predefined scaling policies or custom metrics. AWS provides features like Elastic Load Balancing to distribute traffic across containers and handle failures gracefully.

Cost Prediction: In AWS Lambda, you are billed based on the number of invocations and the duration of function execution, measured in milliseconds. AWS provides a detailed cost breakdown, including the number of invocations, compute time, and associated resources. This granularity enables more accurate cost prediction, as you can estimate costs based on the expected usage patterns and resource consumption of your functions.

With containers, AWS offers services like Amazon EC2, ECS, and EKS. While the cost of EC2 instances can be predicted, estimating container-related costs involves factoring in various aspects such as instance types, networking, storage, and scaling policies. AWS provides tools

like AWS Cost Explorer and AWS Pricing Calculator to help estimate container costs based on specific configurations.

Cost Predictability: Containers vs. Serverless: In terms of cost predictability, serverless architectures in AWS Lambda generally offer more granular control and cost visibility. You pay only for the actual compute time and resources used by each function invocation, allowing for precise cost estimation. AWS provides cost monitoring tools like AWS Cost Explorer and AWS Cost Anomaly Detection to track and analyze serverless costs.

Containers in AWS can offer cost efficiency through optimized resource allocation and auto-scaling policies. However, accurately predicting costs involves considering factors like instance types, container configurations, scaling rules, and data transfer costs. While AWS provides cost management tools, cost predictability in containerized architectures may require additional monitoring and optimization efforts.

The pros and cons of expansion can be quite extension and it is an important aspect of an organization that separates the large entities and the smaller or startup entities. With expansion, there is a lot to consider.

Pros:

- **Increased scalability**: allows you to dynamically build your organization and infrastructure to provide for growing business demands.
- **Better performance**: allows for multiple services, workloads, and leads to better user experience.
- **Reach**: expansion allows for more reach across the world, increasing your customer base and thus increasing workflow.
- **Flexibility**: expansion allows a company to adapt to different business models and various market demands.
- **Revenue**: expansion can considerably increase a company’s revenue

Cons:

- **Complexity**: expanding can have a steep learning curve to newer and bigger infrastructure. This can lead to increased complexity in the organization.
- **Costs**: Just like how expanding can generate revenue, it can also increase costs to keep a bigger organization above ground with positive market projections.
- **Maintenance**: expansion introduces an increased need in maintenance. From one local server to multiple cloud based servers in different regions, expansion in any level will need maintenance systems in place to regulate.
- **Security**: When expanding, especially to the cloud, there is going to be increased security risks. Ensuring security risks are mitigated and accounted for is important for the success of an expanding organization.

These are just a few of the pros and cons that an organization will have to consider when attempting to expand their business.

Elasticity and pay-for-service models play important roles in decision making for planned future growth:

Elasticity: Elasticity allows your infrastructure to dynamically scale resources based on demand. This ensures optimal resource utilization, cost efficiency, and the ability to handle sudden spikes in traffic or usage. Elasticity enables you to scale up or down quickly, aligning resources with actual needs, and avoiding over-provisioning or under-utilization.

Pay-for-Service: Pay-for-service models, such as cloud computing and serverless architectures, enable cost optimization by paying only for the resources or services used. It eliminates the need for upfront investments in hardware and allows you to align costs directly with usage. Pay-for-service models provide flexibility to scale resources and services as needed, which can be cost-effective during periods of growth.

When planning for expansion, evaluating elasticity and pay-for-service options helps in optimizing costs, managing scalability, and aligning resources efficiently. It allows you to strike a balance between meeting growing demands and controlling operational expenses, making future growth more sustainable and cost-effective.