Southern New Hampshire University

8-1 Assignment: Final Reflection

Preston Burkhardt

M. Shah Alam, M.S

CS-470-T3364 Full Stack Development II

February 25, 2023

Project presentation: https://youtu.be/Vd9gMdMgJac

- **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?
        - This course helped me gain skills in cloud computing and application migration. I had not used a cloud provider before, so using AWS was new and interesting for me. I am aware that cloud hosting/computing is extremely prevalent in the software engineering field, and I feel that this course gave me a solid foundation in the basics of using cloud hosting technology.
    - Describe your strengths as a software developer.
        - One of my main strengths as a software engineer is that I'm a quick learner. Throughout my coursework at SNHU, I have demonstrated my ability to be given a new concept, technology, and/or programing language and quickly pick it up to a level where I could contribute to a team. My other strength is my ability to solve problems on my own. Despite help being available from instructors, I find it more rewarding to solve problems on my own. Often this leads to me retaining what I learned better than if someone had just given me answers to my questions.
    - Identify the types of roles you are prepared to assume in a new job.
        - The types of roles I'm prepared to assume in a new job are software engineer, front-end developer, back-end developer, and full-stack developer.

- **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?
            - Scaling microservices is more efficient because you can scale only the services that need scaling. Since each microservice is self-contained, scaling the front-end to handle more concurrent users while leaving the back end is easy. Furthermore, you can track down errors more easily as each microservice will have its own error handling. There's no more tracing an error through a file with 25+ methods/functions that handles everything from displaying the

front-end, processing user input, sending requests to the database, etc.

- How would you predict the cost?
    - To predict costs of cloud hosting, you can set up logging and test your different microservices. For example, you can send a test API call to your API endpoint that handles GET requests. From here, you can look at the time the function took, and data amount transferred. After this, you get the cost from the cloud provider's pricing breakdown to get the total cost of the API call. Finally, multiply this by however many API calls you are expecting. I would also set up some kind of regular logging after deployment so that you can track the actual amount of API calls, time, data, etc. that is being used/performed and make future predictions based on this data.
- What is more cost predictable, containers or serverless?
    - When it comes to cost predictability, containers are more predictable as you pay for a specific set of hardware that the containers run on like how you would when running your own local server. You pick how much memory, processing speed, data storage space, etc. and pay based on these factors. The downside to this level of predictability is that you pay no matter if the infrastructure is used or not.

- Explain several pros and cons that would be deciding factors in plans for expansion.
    - PROS:
        - Can accommodate more traffic. Expanding infrastructure means a business can handle more customer traffic whether that be external or internal customers. More traffic generally means more revenue or better support for internal customers.
        - Competitive advantage. Expanding more than a competitor may give a business an edge when it comes to winning over customers.
        - Expanding your market share. Expansion could lead to obtaining more customers while simultaneously servicing existing customers better to the point where they don't want to leave/look for a substitute.

        - CONS:
            - Cost. Is the cost of expansion worth it? When looking to expand, you need to do a return on investment (ROI) analysis. If expansion is going to require hiring another person and provisioning another server, will the revenue generated by said expansion make up for those costs and if so, how long will it take for it to make up those costs? If the ROI analysis ends up in the negative, it might not be worth expanding.
            - Not needed during “off seasons”. If a business experiences heavy traffic seasons and wants to expand to accommodate this spike in traffic, they may find that the expanded infrastructure goes unused during an “off season”. This can result in wasted money if the spike in traffic doesn’t generate enough revenue to cover the expansion.
    - What roles do elasticity and pay-for-service play in decision making for planned future growth?
        - Elasticity solves the issue of expanding and then not using the added infrastructure during lulls in traffic or off seasons or not having the infrastructure to support peak loads/traffic spikes. It allows a company to scale up or scale down different microservices as it’s needed rather than too late or too soon. When planning future growth, you can use a cloud provider that offers automatic provisioning and de-provisioning of resources either to avoid having to expand your infrastructure or as a sort of “trial period” before you do decide to expand your infrastructure so that you can get an idea of what you’ll need and how often you’ll need it.
        - Pay-for-service is a pricing model that ties into the elasticity of a cloud provider. It makes it so that you only pay for what you use. When your microservices need to scale up, it’s handled automatically, and you’re billed accordingly based on what was used. While this does rule out some level of predictability that comes with paying a set cost for a fixed amount of resources, it allows a company to avoid the issue of paying for resources that go unused for a period of time. This can be a more cost-efficient route when planning for future growth.