CS470 Final Reflection

Gary Clark

August 18th, 2022

Southern New Hampshire University

Presentation Link:

https://www.youtube.com/watch?v=nAY-qKgH2Is

CS470 Final Reflection

Throughout the course of this class, we have gained a great deal of experience integrating a variety of development skills to create a full stack web application. These techniques were further integrated into two modern development architectures: Docker and Amazon Web Services. The experiences within this course will aid in furthering both my personal and professional goals while also providing new insights into my self-awareness regarding my strengths as a developer.

In the beginning of this course, our first assignment involved the migration of a MEAN stack application into containers. This introduction to Docker, a containerization engine, provided valuable experience with ensuring a development environment is properly configured to allow for services to maintain functionality across a variety of platforms. The following modules would extend the containerization process to REST API NodeJS backend applications and to an Angular frontend application. These services would require the use of Docker Compose to provide a fully functioning web application with locally hosted, containerized systems. While this development architecture is an essential tool for modern software developers, there is another system architecture that would need to be introduced for our transition into well-rounded full stack developers. Cloud-based systems are rapidly increasing in market share and formed another essential lesson provided by this course. The remainder of the course saw us learn the process of migrating a locally hosted application to a cloud system. Not only is knowledge of cloud system essential in modern development, but the lessons in this course also provided valuable experience with one of the largest providers of cloud computing services: Amazon Web Services. For the successful migration of our locally hosted application to AWS, we utilized many of the services integrated within Amazon’s ecosystem. The first step involved uploading the application layout, styles, and framework to an S3 bucket. Then, it was necessary to replace our MongoDB database by generating a database within DynamoDB. We then created several Lambda functions with test cases which can be called upon to provide the desired functionality within the application. Our knowledge of REST APIs was further reinforced with the creation of an API within API Gateway that linked the front end and back end of the application. To secure this application, we learned how to implement policies and assign permissions to specific roles using AWS IAM. When combined with our existing experience with MEAN stack and REST API development, these experiences result in a greater preparedness to entire into a role within our desired career.

As a software developer, I have gained a greater understanding of my strengths through the material in this course. There were several instances throughout this course in which I encountered unexpected errors or failures and had to troubleshoot to complete the project. An error which is particularly memorable appeared during the creation of test cases within the Lambda function. The queries to the database were being denied access because the resource could not be identified, but I soon realized that the code was parsing the name incorrectly and it wasn’t reading the last character. By adding a character to the string, I was able to get around this issue and all test cases functioned as expected. As a result of these struggles, I now have a greater confidence in my ability to problem solve and logically process the flow of data within a system. Another strength of mine which was useful in the development of this cloud-hosted web application was my creativity. The ability to think outside the box is incredibly important in the fast-paced world of iterative development, and unique solutions to challenging problems drives technological progress within all industries. Finally, I must recognize that my obligations both personal and professional created a very stressful situation in which to create this application.

However, after completion of the final project, I now have greater confidence in my ability to work under pressure. All these traits become strengths when entering a field that is at the forefront of the modern, technological world.

When considering both my experiences and strengths, there are several roles that I now feel prepared to enter. Both in previous classes and within this class, I have had experiences working with a variety of locally hosted and cloud hosted development environments. Therefore, my skills would be well suited for a role refactoring and migrating systems to cloud platforms. In previous courses, I have utilized various machine learning infrastructure and when combined with a level of comfort with mathematical operations and algorithms, my preparedness for a role as a machine learning specialist is quite high. Now with many aspects of life it can be hard to predict the future and there may be opportunities in other roles within software development. While non-specific, I do feel the broad experiences and knowledge in my possession would provide confidence that I am prepared for a variety of software engineering roles. Whatever my future career entails, my lessons in this class and others will prove essential.

To further elaborate on my knowledge of cloud-based services, the following section will explore the hypothetical future growth of the application created for this course. There is a guiding principle within many development frameworks that emphasizes the importance of modular code. Within this web application, many of the different components are separated into either different AWS services or multiple objects within a given AWS service. Modularization provides several benefits, especially in feature rich modern applications. As the application grows, we will need to consider its ability to scale and handle errors. Within a modular application, it becomes possible to better isolate services to prevent a cascading effect of an unexpected error. There is no such thing as a perfectly error free and secure application, of which it is our job as developers to minimize risk to the best of our ability. AWS IAM will make the task of securing a growing web application much easier as it allows for the management of each modularized component within a single dashboard. Also, within Lambda, it critical that each function is thoroughly tested using test cases which address every desired result of the system. In a worst-case scenario where a breach or critical error occurs, modular applications are far more resilient much like how ships that are compartmentalized can avoid sinking during a hull breach. As the application scales, modular code can be more easily migrated to systems better equipped to handle the traffic or refactored within an existing system. Services like AWS provide highly scalable, modularized platforms to allow applications to grow. However, there is a cost associated with these platforms and development teams must anticipate expenditures to budget accordingly. To accurately predict the development cost, it is important to first gather field research. Without hard data, one cannot make educated predictions about the future and developers can use this data to further refine predictions using modeling techniques. These models will help anticipate future growth to better allocate funds and resources by anticipating future industry trends. Many cloud platforms, including AWS, have a dynamic fee structure owing to their automatic resource allocation. While this payment structure is more flexible to the needs of the business, it can also be harder to predict. We can contrast this to a containerized system which is far more predictable because the compute resources utilized will be somewhat predefined. However, despite the greater predictability, containerized systems will present developers with a choice to either overpay to prepare for future growth or pay for the current needs at risk of compromising the scalability of the application. Therefore, this is far less cost efficient than serverless platforms and the cost predictability of using services like AWS can be greatly increased with accurate growth prediction models.

The decision to grow an application is not a simple one and there are several pros and cons that must be considered. When planning for expansion, developers explore the benefits of expansion such as bring their services to a wider user base. Through making the application available to a broader audience, it can snowball into further growth and adoption. The expansion of an application allows systems to better meet the predictions made by traffic models, reducing the chance of dissatisfaction among users. In the tech industry, the most successful applications are those that are first to the market. When considering the expansion of this application, we must acknowledge the benefits of building for the future today. Despite being often overlooked, expansion can also grant greater reliability of the application under the current traffic levels. By building out the application to handle larger numbers of users, it results in the current level of traffic resulting in a much smaller stress on the system. However, there are several cons to expansion that must be weighed against these benefits. Any time applications expand and grow, they become more complex. More complexity creates more points of failure and testing often takes far longer. As a result, expansion also creates the need to allocate large amounts of resources which takes a great deal of time to accomplish. Finally, all these resources create a high cost associated with expansion. Developers will have to carefully consider the cost/benefit analysis associated with plans for future expansion of the application.

Cloud platforms have two attributes that make them especially appealing for developers looking to plan for future growth. One of these attributes stems from a concept called elasticity. Because AWS dynamically allocates hardware resources based on demand, this allows applications to be far more adaptable and efficient than more traditional development platforms. This flexibility leads to a far greater uptime percentage through its resiliency to spikes in demand and user satisfaction will remain high. The elasticity of cloud platforms creates a near ideal environment to adapt and facilitate future growth. With the ability to dynamically allocate these resources, services like AWS have transitioned to a pay-for-service model. In these payment structures, developers only must pay for the resources that are used. This can reduce cost in instances where applications may have to be overbuilt to handle expansion. While this payment model provides greater efficiency and cost saving, it does necessitate for the team's budget to be flexible as cost may vary month to month. Before flocking to pay-for-service models, it is also important to consider that the organization will be relinquishing control over portions of its assets. Therefore, a team looking to retain control over the hardware, or the application data may want to avoid these platforms. When budgeting for expansion and the decision of whether to adopt a pay-for-service provider, one must also consider profitability. Specifically, the concept of profitability at scale which refers to how the profit margin increases or decreases as adoption of the application grows. If the application is only profitable above a certain size, it may not be worth planning and paying for low levels of traffic while being able to dynamically scale up. Instead, it would be a better use of resources to invest directly into the adoption of the application. However, if the application is profitable at all scales, then a pay-for-service model would make a great deal of sense. Developers would not need to worry about the adoption of the application and could instead work on the functionality of the service without concern that sudden growth would overwhelm their systems. Decision making for planned future growth is not easy, but through consideration of the many variables present within app development and deployment, it is possible to determine which solution best fits the needs of the application.