Concepcion Pendergrass

24 June 2023

CS 470 Final Reflection

https://youtu.be/aZTyYJ4NXDY

I have learned a variety of skills in CS 470 that have significantly improved my talents as a software developer and made me a more marketable prospect for the job market. I now know the fundamentals and best practices of cloud computing. Furthermore, I now know how to use cloud infrastructure and services to create and deploy applications. By using frameworks and tools made expressly for cloud-based applications, I have improved my full stack development abilities even further. This comprises databases, frontend frameworks, and backend frameworks. I have developed my ability to design and implement cloud architectures for my software stack applications, gained expertise in managing scalability and cloud application deployment, shown my capacity to communicate highly technical material to various audiences and in various formats, and sharpened my problem-solving abilities by overcoming actual development challenges for cloud-based applications. My abilities to solve problems logically and deconstruct difficult problems, my attention to detail to ensure accuracy and high-quality outputs, my adaptability, my strong analytical skills, and my good communication skills are some of my talents as a developer. Along with my programming knowledge and creativity.

Full Stack Developer: I am capable of taking on the duty of creating a web application's frontend and backend. Designing user interfaces, putting business logic into practice, and interacting with databases or APIs are all included in this. Web Developer: I may focus on creating and maintaining websites and online apps as a web development specialist. I would be

knowledgeable in backend frameworks and databases in addition to frontend technologies like HTML, CSS, and JavaScript. Cloud Developer: I can work on creating and maintaining cloud-based solutions thanks to my knowledge of cloud architecture and expertise delivering applications to the cloud. This entails managing resources, securing scalability and reliability, and establishing and optimizing cloud architecture. DevOps Engineer: Because of my familiarity with cloud-based development and best practices, I might also be qualified to work as a DevOps engineer. This entails managing deployment pipelines, automating software development processes, and fostering effective communication between the development and operations teams.

To handle scale and error handling I would use serverless. Each part of the program can be grown independently based on demand thanks to microservices. By scaling only the required services, this enables more effective resource use and the ability to handle growing demand. Furthermore, error handling can be restricted to a few selected microservices, reducing the impact on the entire system. Serverless architectures grow automatically in response to events or incoming requests. Because of this, the system can handle variable workloads without the need for human scaling. The serverless platform itself frequently handles error management, offering built-in fault tolerance and robustness. To predict the cost I would use the serverless because with serverless platforms uses a pay-per-invocation billing model, where you are charged for the actual number of function calls and the time it takes for them to complete. This can make cost prediction simpler because you can calculate costs based on anticipated traffic patterns and invocation rates. The cost of microservices, however, can be difficult to anticipate because it depends on the precise resource allocation and scaling requirements for each component. Careful monitoring and analysis of resource utilization throughout time may be necessary for accurate

cost estimation. Since containers need specialized resources and their deployment costs are largely influenced by their size and number, they offer greater cost certainty. Since you only pay for what you use, serverless architectures offer a high level of cost predictability. It is simpler to predict the cost impact of scaling the program because costs scale linearly with the quantity of invocations and execution time.

Microservices: The advantages of this approach are scalability, fault isolation, autonomous development and deployment, freedom to use various technologies for each microservice, and resource efficiency. Disadvantages are managing many services becomes more difficult, inter-service communication is more time-consuming, network latency may affect performance, and operational costs are higher. Serverless: Advantages are built-in fault tolerance, automatic scaling, decreased operational complexity, pay-per-use pricing, streamlined deployment, and the flexibility to concentrate on application logic rather than infrastructure management. Disadvantages are vendor lock-in, resource limitations, significant cost increases for workloads that often operate or for long periods of time, and heightened complexity for applications with intricate workflows.

Elasticity makes it possible to allocate resources efficiently, boosts performance, and ensures that the web application can manage variable traffic or workloads, supporting planned future growth without the need for manual intervention. By requiring no upfront infrastructure investment, pay-for-service promotes effective resource management. Additionally, it offers the option to scale various application components individually, enabling fine-grained cost control and ensuring that resources are deployed where they are most required. Elasticity and pay-for-service models aid in optimizing resource allocation, controlling costs, and adjusting to changing demands while making decisions for projected future expansion. These methods

encourage scalability, agility, and resource efficiency, enabling the web application to easily handle growth while keeping operating expenses in check.