Reflection C

Learning Highlights

In Module 4 to 6, I got a better understanding of how to figure out which systems and infrastructure are really important for an organization. Using techniques like Business Impact Analysis (BIA) and Risk Assessment, I learned how to evaluate what keeps things stable and secure. It's pretty cool to see how these methods can help target the systems that are crucial for everything to run smoothly and spot any risks that could mess things up.

I also got to dive into Docker and Kubernetes, which are key tools in the world of Continuous Integration and Continuous Deployment/Delivery (CICD). Docker is like putting an application in a neat little box so that you can bring this box to anywhere and run the application, and Kubernetes helps manage all these containers across different environments. These tools make it way easier to automate and scale things, which is a game-changer in software development.

Performance and scalability are also important in software development, as they can determine the upper limit of a software's potential or even a company's overall success. To optimize performance and scalability, we have to find out the bottleneck first by thoroughly analyzing the system. This involves monitoring resource utilization, such as CPU, memory, and I/O, and identifying the components or factors that are slowing down the system. Once the bottleneck is identified, targeted optimization techniques—such as code refactoring, database indexing, load balancing, or implementing caching strategies—can be applied to improve system efficiency. Additionally, implementing proper scalability strategies, like vertical scaling (adding resources to existing servers, for example, upgrade gpu, cup) or horizontal scaling (adding more servers with the same standard), ensures the system can handle increased loads effectively.

To keep an app running smoothly, the first thing we need to do is check out the tech stack and all its dependencies. Basically, this means making a list of everything the app uses—like programming languages, frameworks, libraries, databases, and any third-party services. Then, you look at which versions you're using, how well they work together, and if there are any security issues. We also need to check if these tools are still being updated or if they're getting outdated. Same goes for any external APIs or libraries the application is relying on so that we can make sure they're still working and won't be glitched. By doing this upfront, you can catch any problems before they cause issues, plan for updates, and make sure the

app stays secure and ready for future changes. This step is key to avoiding surprises and making sure the app can keep evolving.

Implementation

I have run a text-to-image app on Discord for about 18 months; it used to run on my local Linux system. After I moved to Australia, the maintenance job became harder, as I have to use remote software such as TeamViewer or AnyDesk whenever I need to configure or modify something on the app. Things become more complicated when I try to migrate the application to my current desktop or to the cloud, as I have to set all configurations again and possibly justify the differences in operating systems. So, I have started learning Docker and am trying to containerize this application. It's not done yet, but I hope I can migrate the whole application to the cloud by the end of this year.

Challenges

Module 6 is quite a big module; it is hard to understand all the content at once, especially the security topic, as cybersecurity is a different field than software development. For me, it's difficult to visualize the process of conducting security audits and vulnerability assessments, and I still don't fully understand how penetration testing works in detail. However, I believe the goal of this course is to provide an overview of each topic, not knowing every detail, so I think it's fine that I am still confused at certain points.