Unit 10 Reflection

For this unit, our focus continued on software quality with reading material pointing to software quality assurance and software quality planning. One article touched on developing secure software, highlighting principles such as least privilege, failing securely, securing the weakest link, in-depth defence, separation of privileges and complete mediation (Firdaus et al., 2014). Considering these security-related principles, I think there is a valid case for each of these in delivering quality software. Insecure software will be perceived poorly, and customers' trust may be violated due to leaked data.

This week look at using linters running against Python code. Linters analyse software source code for common errors, bugs, style issues and questionable constructs and provide one or more recommendations for developers to improve their code. Linters such as pylint, pyflakes and pydocstyle were interesting because they made me consider the output of linters and compare their usefulness with each other. While linters may provide information that, on the surface, can be regarded as pedantic—missing a space after an operator—other output messages may advise performance improvements like not comparing "false" with itself. Therefore, I think static code analysis must be regularly performed to maintain technical debt and high code quality.

The team continued to work on the project implementation and presentation content.

References

Firdaus, A., Ghani, I. & Jeong, S.R., (2014). Secure feature driven development (SFDD) model for secure software development. *Procedia-Social and Behavioral Sciences*, *129*:546-553.