**Reflection**

<https://ashudes.github.io/Secure_softdev.html>

<https://github.com/ashudes/ashudes.github.io/tree/main/SSD%20Module>

**Unit 1: Introduction to Secure Software Development**

*What?* The unit introduced contrasting management approaches to software development, highlighting the shift from traditional waterfall methods to agile methodologies. *So What?* The transition from waterfall to agile reflects the industry’s move towards more flexible and responsive development cycles, which can better accommodate the need for security. *Now What?* I recall a software development project I was managing recently. Due to a strict deadline with funding utilisation, we used agile development. The focus was mostly on meeting the deadline with a minimum viable product (MVP). Security was not given adequate priority. Striking a balance between a speedy development and integrating thorough security aspects is something that I look forward to applying in future similar projects.

**Unit 2: UML Modelling to Support Secure System Planning**

*What?* Unit-2 focused on UML’s practical application in creating flow charts essential during the SDLC design stage. *So What?* This is one of the areas that I had prior experience with without having adequate technical knowledge. Detailed flow charts are valuable for visualising and communicating process steps. *Now What?* This unit has equipped me to better understand and communicate software requirements, use cases, structures, and functions. However, I am curious to learn more to what extent UML can represent system security concerns.

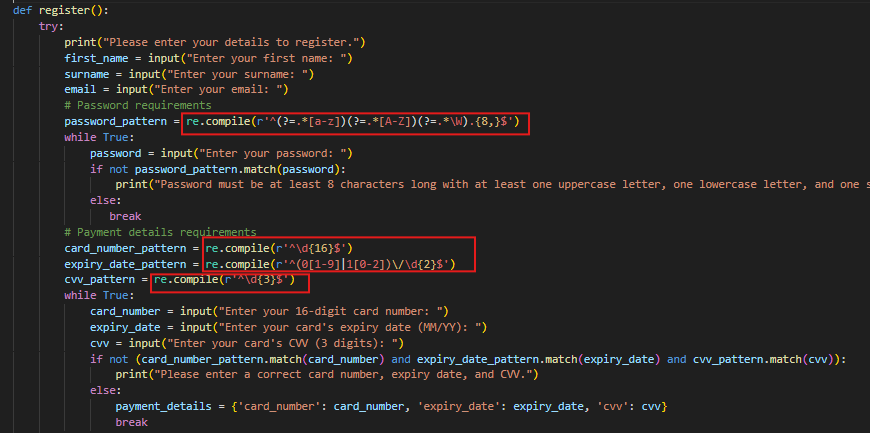
**Unit 3: Programming Languages: History, Concepts & Design**

*What?* Unit-3 delved into the evolution of programming languages, examining their history, concepts, and design, with a particular focus on Python. *So What?* Understanding the progression of programming languages offers insight into current design principles and best practices, especially regarding security challenges. *Now What?* As a beginner in coding, I like Python due to its user-friendliness. The inherent security features of different programming languages remain to be topic of interest to me.

**Unit 4: Exploring Programming Language Concepts**

*What?* Unit-4 explored the impact of programming concepts such as regular expressions and recursion on system security. *So What?* These concepts are powerful tools in a developer’s arsenal but can introduce security vulnerabilities if not used carefully according to some of my readings (Michael et al., 2019).

*Now What?* This is a topic that has had immediate practical application for the formative assessment under Unit-11. As part of my code implementation for the online shopping application, I made use of the built-in Regular Expression in Python for password and payment card validation. I am keen to appreciate some real-life examples of DoS and other vulnerabilities that Regex can introduce to systems.



**Unit 5: An Introduction to Testing**

*What?* Unit-5 introduced the art of testing software with a focus on security, covering key terms, techniques, and industry practices. *So What?* The unit emphasised the importance of a comprehensive test plan and introduced us to industry standards like OWASP and ISO/IEC/IEEE. *Now What?* From the practical exercise, I learned that the effectiveness of automated testing is tied to the quality of the test cases. As a project manager, this unit has made me eager to explore different testing approaches in future projects.

**Unit 6: Using Linters to Support Python Testing**

*What?* Unit-6 focused on using linters to develop high-quality Python code, emphasising that quality code is secure. *So What?* In the e-Portfolio activity, I experimented with pylint and flake 8. These tools are useful for code development and testing as they help catch errors and maintain code quality. *Now What?* From my observation so far, while linters are useful for analysing codes, bugs, and style errors, they should not be the only tools relied upon for ensuring code security and functionality.

**Unit 7: Introduction to Operating Systems**

*What?* Unit-7 delved into OS and their relationship with programming languages and security, discussing OS functions, common systems, and the distinction between processes, threads, and schedulers. *So What?* The unit emphasised the importance of securing operating systems and provided insights into mitigating OS vulnerabilities. *Now What?* While I cannot say that I have grasped enough about OS and security, the unit has raised my interest to read and learn more on this topic.

**Unit 8: Cryptography and Its Use in Operating Systems**

*What?* Unit-8 explored cryptography principles and their application in operating systems, including cryptographic libraries. *So What?* Cryptography’s practical application in operating systems is crucial for modern security. *Now What?* In Unit-11, I applied cryptography to my e-commerce app using the bcrypt library for password hashing and salting. Initially, I struggled with hashlib, spending significant time resolving errors before succeeding with bcrypt. This experience underscores a broader usability concern with cryptographic libraries, as noted by Acar et al (2017).

**Unit 9: Developing an API for a Distributed Environment**

*What?* Unit-9 built on Unit-7’s API exercise, providing practical Python skills for creating an API and experimenting with CRUD operations. *So What?* The unit’s hands-on approach to API development and exploration of Python’s flask library are key to understanding how back-end code developments can be accessed through a user interface. *Now What?* This was my first experience with API development. It has sparked my interest to explore more API development approaches in a distributed environment and learn about potential vulnerabilities.

**Unit 10: From Distributed Computing to Microarchitectures**

*What?* Unit-10 examined the evolution from monolithic architecture to distributed systems and microservices, and their security challenges. *So What?* The shift to distributed systems has expanded the attack surface, requiring advanced security measures such as encryption and key distribution patterns. *Now What?* This unit has prepared me to anticipate and address the emerging challenges from increased fragmentation through microarchitectures.

**Unit 11: Future trends in Secure Software Development**

*What?* Unit-11 focused on the future trends in secure software development. *So What?* Understanding these trends is crucial for staying ahead in the field of secure software development. *Now What?* Integration of security in the emerging trends is a critical aspect that I would like to learn more.

**Unit 12: The Great Tanenbaum-Torvalds Debate Revisited**

*What?* As a final unit, much of the focus was on finalising the remaining formative assessment. The Tanenbaum-Torvalds debate is something that I would have liked to participate in but did not get enough time for. *Now What?* I will continue to read on the Tanenbaum-Torvalds debate and use the insights from it to inform my approach to system design, considering the trade-offs between monolithic and microservice architectures.

**References**

Acar, Y., et al. (2017). Comparing the Usability of Cryptographic APIs. 2017 IEEE Symposium on Security and Privacy (SP). doi:https://doi.org/10.1109/sp.2017.52.

Michael, L., Tech, V., Donohue, J., Davis, J. and Lee, D. (2019). Regexes are Hard: Decision-making, Difficulties, and Risks in Programming Regular Expressions Francisco Servant. [online] Available at: <https://arxiv.org/pdf/2303.02555> [Accessed 27 May 2024].