**LOUIE JAY A. CENIZA BSIT 4-1 FEBRUARY 27, 2024**

**(CHAPTER 61-70)**

# **ONE BINARY**

In this chapter it talks about the benefits and challenges of deploying a single, monolithic binary as opposed to a more modular, microservices-based architecture. the simplicity and convenience of deploying a single binary. With all components bundled together, deployment becomes easier, and there are fewer dependencies to manage. This simplicity can lead to faster development cycles and quicker time-to-market for software products.

It provides us valuable insights into the trade-offs involved in choosing between a monolithic binary and a microservices architecture, highlighting the importance of balancing simplicity with scalability and flexibility in software design.

# **ONLY THE CODE TELLS THE TRUTH**

In this chapter it talks about the importance of relying on the actual implementation of code rather than assumptions or documentation. while documentation can provide valuable context and guidance, it may not always accurately reflect the current state of the codebase.

The tendency for documentation to become outdated or misleading over time, especially in fast-paced development environments were code changes frequently. Relying solely on documentation can lead to misunderstandings, wasted effort, and bugs.

Instead of relying for documentation, practice code reading where developers prioritize understanding the code itself as the primary source of truth. By thoroughly examining the codebase, developers can gain insights into its structure, behavior, and intent.

By prioritizing code reading and writing clear, understandable code, developers can mitigate risks and build more robust software systems.

# **OWN (AND REFACTOR) THE BUILD**

In this chapter it talks about the importance of treating the build system as a critical component of the software development process, that developers should take ownership of the build process and continuously refactor it to maintain its efficiency and reliability.

By owning and refactoring the build system, developers can improve the overall development experience, reduce cycle times, and deliver higher-quality software more efficiently. This proactive approach to build management aligns with modern software development practices such as continuous integration and continuous delivery, enabling teams to deliver value to customers faster and more reliably.

# **PAIR PROGRAM AND FEEL THE FLOW**

In this chapter it talks about the importance of pair programming as a practice that not only enhances code quality but also promotes team collaboration and individual growth. The benefits of pair programming and share insights into how it can lead to a state of flow for developers.

# **PREFER DOMAIN SPECIFIC TYPES TO PRIMITIVE TYPES**

In this chapter it talks about the importance of creating meaningful abstractions that accurately represent the concepts and entities within a specific domain, rather than relying solely on built-in language primitives.

# **PREVENT ERRORS**

In this chapter it talks about strategies for reducing errors in software development by focusing on usability and user experience design. Colborne emphasizes the importance of designing systems that make it difficult for users to make mistakes in the first place, rather than relying solely on error detection and correction mechanisms.

# **THE PROFESSIONAL PROGRAMMER**

In this chapter it talks about the importance of professionalism, integrity, and a commitment to excellence in the practice of software development. It serves as a guide for aspiring programmers seeking to establish themselves as true professionals in the industry.

# **PUT EVERYTHING UNDER VERSION CONTROL**

In this chapter it talks about the importance of comprehensive use of version control systems (VCS) in software development.

By putting everything under version control, developers can improve productivity, ensure consistency, and mitigate risks throughout the lifecycle of their projects.

# **PUT THE MOUSE DOWN AND STEP AWAY FROM THE KEYBOARD**

In this chapter it talks about the importance of keyboard-centric workflows for developers, keyboard-centric workflows are a means to enhance productivity, reduce cognitive load, and improve the overall development experience.

By mastering keyboard shortcuts and commands, developers can work more efficiently and effectively in their chosen development environments.

# **READ CODE**

In this chapter it talks about the importance of reading and understanding code as a crucial skill for software developers. Code reading as a fundamental aspect of software development.

By approaching code with curiosity, diligence, and a willingness to learn, developers can enhance their understanding, contribute effectively to projects, and continually grow as professionals.