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

**(CHAPTER 31-40)**

# **DONT TOUCH THAT CODE!**

In this chapter it talks about were you need to only work where your role is befitting to you, it gives us insight of restricting someone particularly to the developers, so that the on-production system will not have a significant problem.

It was given that the developers should only have access to the development server for tweaking, fixing and testing the code system if there was an error on the production server, it should be limited to the specific roles.

By limiting the access to the staging and production server, it will not complicate the current issue, developers can tweak or fix that issue in the development server, so that the production code will not be affected.

# **ENCAPSULATE BEHAVIOR, NOT JUST STATE**

In this chapter it talks about the importance of encapsulating both state and behavior within objects in object-oriented programming. By encapsulating behavior along with state, objects become self-contained entities that can manage their own state and respond to messages or requests appropriately.

It also says that through out the years, the most common problem for developers is that constructing classes and by demonstrating this the developers have not fully understood object-oriented thinking.

By understanding encapsulation in object-oriented design/program you can improve your codebase system’s maintainability and also this promotes code reuse, reduces dependencies.

# **FLOATING-POINT NUMBERS ARENT REAL**

In this chapter it talks about what are the different between real numbers and non-real numbers, in this case floating point numbers are commonly used in programming language which some of the languages make it a real number, but in this chapter, it emphasizes that floating-point numbers are not the same as real numbers in mathematics due to their finite precision and rounding errors.

So, it serves as a reminder to programmers to be cautious when working with floating-point numbers and to understand the implications of their finite precision and rounding behavior.

# **FULLFILL YOUR AMBITIONS WITH OPEN SOURCE**

In this chapter it talks about how you can still fulfill your ambition even though the current work you had is not the ambition you want, don’t stop there It says, there are many open-source codes that are in the online webs you can browse, where you can help by giving insights or you can give a solution to that open-source code.

# **THE GOLDEN RULE OF API DESIGN**

In this chapter it talks about importance of considering testing as a crucial aspect of API design. It says that the developers may face challenges when designing APIs that will be used by a large number of users and emphasizes the need to anticipate future changes without breaking existing client code.

In some cases, you’ll code in your internal which users can override your output codes and that could be disastrous to your system.

By Applying the golden rule of APIs design you can limit the overriding’s and can improve the codes performance and maintainable.

# **THE GURU MYTH**

In this chapter it talks about the common perception of software developers as "gurus" who possess mystical powers to solve any problem without needing sufficient context or evidence. It critiques this myth by highlighting the importance of systematic analysis, logic, and continuous learning in software development.

# **HARD WORK DOES NOT PAY OFF**

In this chapter it challenges the common belief that working hard always leads to success in programming and software development. Instead, it argues that working smarter, not harder, is often more effective in achieving goals and maintaining productivity.

By encouraging programmers to shift their focus from working long hours to working efficiently and effectively. By prioritizing continuous learning, maintaining a sustainable pace, and acting like professionals, programmers can achieve greater success and avoid burnout in the dynamic field of software development.

# **HOW TO USE A BUG TRACKER**

In this chapter it emphasizes the importance of effective bug tracking in software development projects. A well-documented bug report plays a crucial role in facilitating efficient bug fixing and maintaining project momentum. It should include precise details on how to reproduce the bug, the expected outcome, and the actual observed outcome. Additionally, the quality of a bug report reflects the professionalism of the reporter, with clear and respectful communication fostering collaboration within the development team. Bug tracking is portrayed as a conversational process, where transparency and constructive engagement are key.

# **IMPROVE CODE BY REMOVING IT**

In this chapter it talks about the importance of adhering to the principle of "less is more" in software development. It discusses the author's experience of improving the codebase by removing unnecessary chunks of code that were implemented but not actually needed. The practice of simplifying the code not only enhanced product performance but also reduced code entropy, leading to a more manageable and maintainable codebase. The author emphasizes the significance of writing code that adds value rather than simply being a "fun" addition or anticipating future requirements. It also warns against inventing extra requirements or features that were neither documented nor discussed with the customer, as this can lead to unnecessary complexity and maintenance overhead.

It encourages developers to evaluate their current tasks and consider whether all aspects of their work are truly needed, advocating for a more pragmatic and minimalist approach to software development.

# **INSTALL ME**

In this chapter it talks about users’ behavior and expectations when it comes to installing and using software. It emphasizes the importance of simplicity, convenience, and transparency in the installation process. Users want to know exactly where the software will be installed and how to uninstall it if needed. They prefer straightforward actions and immediate results, without the need for complex setups or unnecessary steps. Additionally, users appreciate clear and easy-to-follow tutorials that address their specific problems and guide them through the software's features and functionalities.

By prioritizing user experience and providing a seamless installation process and helpful documentation, developers can increase user satisfaction and build trust in their software products.