▼ Title Of Book and Chapter

## (Chapter 4) Automate Your Coding Standard

▼ What are Three Things I learned today

1. **Before** - recognizing the significance of coding standards and how they affect the creation and upkeep of software.

**After** - I’ve learned about this examples on automate and enforced such codes:

* Make sure the build process includes code formatting such that it is executed automatically each time someone compiles code.
* Search the code for undesirable anti-patterns using static code analysis tools. Break the build if any are discovered.
* Learn how to set up those tools so you can search for anti-patterns that are unique to your own projects.
* Measure test coverage as well as automatically reviewing the outcomes. Once more, if test coverage is too low, break the build.

2. **Before** - learning the best practices and frequently used coding conventions can be helpful to the standard on automation your coding

**After**  - putting in place automated code quality-checking technologies like code analysis and linting tools.

* using a version control system, like Git, to keep track of code changes and work with other developers.
* defining standards for brief and unambiguous code documentation, such as external documentation and inline comments.
* putting automated testing frameworks into place to guarantee the consistency and dependability of code.

3. **Before** - Code review procedures are incorporated into the development process to enhance cooperation and code quality.

**After** - Become familiar with commonly used project management tools, IDEs, and text editors for software development.

▼ Title Of Book and Chapter

## (Chapter 5) Beauty Is in Simplicity

▼ What are Three Things I learned today

1**. Before** – I don’t quite like simple codes because I kind like the complete detail of the specific code.

**After** - but, I’ve learned that there is beauty on simplicity of the code.

2. **Before** – I don’t know before that it is nicer to code to be simple

**After** - I’ve learned these In our programming, we aim for a number of objectives, including:

* + Readability
  + Maintainability
  + Rapidity of development
  + the elusive characteristic of beauty

3. **Before** – I practiced myself to put all the name id of a certain button of a simple project, like for example this code <button id= “button\_submit” /> instead like this <button id=”btn\_submit” />. Although it has the same logic but the last one is simplified.

**After**  - I’ve learned this statement based on the chapter “With clean, simple, tested code and a fast development speed throughout the system's lifespan, we can keep our systems manageable throughout time.”

▼ Title Of Book and Chapter

## (Chapter 6) **Before You Refactor**

▼ What are Three Things I learned today

1. **Before** - I experienced this many times and little did I know it consumes my time and energy on doing it. I always restructure my code and refactor if there is an error or bugs even without backing up the project and as what I have stated it consumes my time and energy t do it

**After** - I’ve learned this, *Avoid the temptation to rewrite everything.* As what I have stated that I experienced this scenario and I will learn from this. I have regretted it before saying that “I hope I didn't change the code”.

2. **Before** – I experienced and witnessed a lot of CRUD systems and saying their systems as old and not quite bit updated and there it is on my mind to help them and change it to new framework.

**After** - but, I’ve learned this *New technology is insufficient reason to refactor.* although new technology is rising, we cannot forced them to change into new frameworks on their systems and I have witnessed old systems and they leave it as is as long as it is working in their own company or organization.

3. **Before** – I experienced one of my classmate’s project and he wanted a help to me and so I helped him on his CRUD system and I tweaked all of his codes and then suddenly when we run it the program now has lots of errors that is need to be bug although the code is right but his project would not open. So, we decided to make new and copy all of the code on the last project.

**After** - I’ve learned this *Remember that humans make mistakes.* Based on the statement I realized that it “The new code may not always be better than or even close to the previous attempt after restructuring.” Because I witnessed and experienced it and regretted and also, I’ve learned from it. We humans commit mistakes and earned a lesson for it.

▼ Title Of Book and Chapter

## (Chapter 7)Beware the Share

▼ What are Three Things I learned today

1. **Before** – I have an experience that I copied a code that is not aligned on my context and later on I discover an error and I solve it.

**After** - Based on what I’ve read I learned also from his experience that in order for you to copy a code you must know and understand the logic of the context because as he stated that “Each could change on its own. Each might modify its logic to meet the requirements of the system's shifting commercial environment.” So, I will learn from it.

2. **Before** – I also experienced that I outnumbered the dependencies of my projects and it is needed for the project I made but also, erased some lines of code on it.

**After**  - I’ve learned from the statements I’ve read It's important to consider the context of these dependencies; if they had been localized, they could have been acceptable and had some benefit. Even though the code itself appears to be in order, when these dependencies aren't kept in check, their tentacles entangle the system's more significant issues.

3. **Before** – although the code I have copied is running but still without knowing the logic of the context is for what then later on I will encounter outnumbered errors.

**After** - I’ve learned that they may appear to be wise decisions and these methods are beneficial when used appropriately. They raise expense instead of value when used improperly. So, doing such copying another context must be well studied its logic and well tested before copying it.

▼ Title Of Book and Chapter

## (Chapter 8) **The Boy Scout Rule**

▼ What are Three Things I learned today

1. **Before** – I’ve experienced a scenario when I got became selfish on my project and then later on one of my classmates came and she has no pair and I assigned her tasks then, problems were popping out and I found out that it is her code that found to be error so got it troubleshooted and simply solving her mistakes of the code.

**After** – So, I’ve learned that in a team we must help each other and must teach ourselves to not be boastful on ourselves but instead helping each other’s codes.

2. **Before** – We also have this project that was a CRUD system and me and my partner was confused of our project because the codes weren’t coinciding, like we have different logic of our own so as a team we helped each other and we undergone meetings after meetings just to come up a solution to it. We’ve made the modules simplified than before.

**After** – As the context I’ve read on a team even though it is a small part of help you contribute still it is counted.

3. **Before** – As far I’ve experienced although it can’t be project related but it is a teamwork of a team is tested. I experienced an event of ours and as a team we have specific designated tasks and for me I am a Technical Director of our Organization and being a Technical Director I myself will be a role model to my subordinate’s and after we have setup the sound systems and etc. I discover that there is a certain part of the multimedia was failed to do so. So I helped them and teaching them what to do next time.

**After** – From the statements I’ve read there lots of example that is relevant to it can be applied in all aspects.

As a team if and only if you found trash (not simplified code or functions) of your team’s code then “The Boy Scout Rule” says “Always leave the campground cleaner than you found it”. Then we must help it to clean it out because we all have perspectives and point of view between you and your teammates and they didn’t see what you saw and you also see what they can see.