LIVEWYER

# AI Engineering Principles

AI is here, and it's here to stay, no matter your opinion, and it is fairly easy to list the things your engineering teams should not be doing when using it. Everything from drive-by PRs without understanding project context, not following-up when maintainers ask questions, making claims to fix bugs that don't exist, and general "vibe coding" just to increase your GitHub activity. However, at LiveWyer, we wanted to provide real Engineer to Engineer guidance of what can be done, and how to effectively use AI.

Remember, when used correctly, it is a tool that can be of great influence and value add when it comes to both productivity and day to day tasks.

## 1 Issues before implementation

***Never submit code as the first interaction*** with a project. Open an issue, describe the problem, and discuss potential approaches. Let maintainers validate that the problem is real and a solution is required. This single step can eliminate up to 90% of the current wasted review time.

## 2 Disclose AI usage upfront

***State which tools you used*** (Claude Code, Copilot, etc.) and the extent of AI assistance. This isn't about shaming the gap in your knowledge. Everyone has gaps somewhere, but disclosing your use of AI sets the correct expectations with the reviewer.

*Note: Many projects now require this in their contribution guidelines as a must have.*

## 3 Verify everything yourself

***Don't forget that AI hallucinates***, it can invent APIs that don't exist, it suggests patterns that don't fit the codebase, and it makes very confident assertions about requirements it doesn't understand. If you can't explain why every line of AI-generated code is correct, then please don't submit it for someone else to figure out why.

## 4 Test before submission

***Never submit code for platforms or environments you can't personally test***. If you're using AI to write macOS-specific code but you're on Linux, you have no way to verify that your code works correctly. Once again, this is putting the onus on someone else to do the work which you haven't done. Please don't use maintainers as a QA team.

## 5 Show your thinking

***Include the prompts you used***, the decisions you made, and any corrections you applied. This "show your working out" principle provides reviewers and maintainers with the context to understand your intent and approach. The logic you disclose will provide the human reasoning, which is more valuable than the final code.

## 6 Commit & proceed until Closed

***Respond within 48 hours*** if maintainers ask questions or request changes. If you don't have time to iterate on feedback, then you don't really have time to submit the PR. Abandoned pull requests are worse than no pull requests as it promotes a "throw it over the fence" mentality.

## 7 Respect the "No"

***Accept it gracefully*** if maintainers close your PR or reject your approach. They understand the project better than you do. This is particularly important when AI has convinced you that your solution is correct. We hope this list provides a safe list for engineering teams to get utilise. The common thread within all of them is about ownership.

## Get in touch and see how we can help embed AI principles within your engineering teams

LiveWyer has worked with many global enterprises to help implement robust, sustainable and elegant architectures so their Platforms can withstand the test of time.

With AI here to stay, we want to help you implement the correct engineering principles within your existing teams so that your architecture can remain resilient, secure and high performing. **Book an initial consultation here**.

**hello@livewyer.io** | **+44 (0) 20 3608 0110** | **9 Great Russell St, London**