Federico Cinelli

September 7, 2025

CS-530: Module 2

**2-1 Journal: Part Two – Reflection**

During this exercise, I worked with both ChatGPT and GitHub Copilot to explore and apply prompt engineering frameworks in a software engineering context. Each tool brought its own strengths and limitations to the table. ChatGPT excelled in high-level reasoning and structured conversations. It allowed for deep exploration of concepts like the SCREAM and SIMPLE frameworks and was particularly effective at interpreting and applying them in detailed, nuanced ways. Its ability to retain context across multiple prompts made it ideal for more complex tasks such as prompt refinement, role-based scenarios, and audience targeting. ChatGPT occasionally over-explained simple ideas or generated slightly verbose outputs, and while it's generally accurate, its responses still require user verification, especially in critical or ethical domains.

GitHub Copilot performed best in fast-paced, code-oriented workflows. It seamlessly integrated into the development environment and provided real-time code suggestions and completions based on minimal input. When the SIMPLE framework was applied via Copilot, it effectively responded to code comments and structured prompts, making it a powerful assistant for refactoring, writing tests, and generating boilerplate code. That said, Copilot lacks the conversational depth and explanatory abilities of ChatGPT. It cannot sustain multi-turn dialogue or offer nuanced reasoning, making it less suitable for complex or abstract prompt engineering discussions. Additionally, there are ongoing ethical considerations surrounding Copilot, particularly its use of publicly available code that may not always respect licensing requirements.

When considering the most effective prompt framework for each tool in the context of software engineering, SCREAM proved to be best suited for ChatGPT. This framework’s emphasis on structure, context, role, examples, audience, and medium aligned well with ChatGPT’s strengths in generating thoughtful and goal-oriented outputs. It was especially useful for tasks such as drafting documentation, analyzing code architecture, or designing prompts tailored to specific audiences. On the other hand, the SIMPLE framework was more appropriate for GitHub Copilot. Given Copilot’s strength in responding to short, structured comments, SIMPLE’s focus on specificity, intent, modularity, persona, language, and evaluation enabled better control over prompt effectiveness within the IDE.

Both frameworks had their unique advantages and limitations. SIMPLE was straightforward and ideal for technical environments that require clarity and concise instruction, but it offered less flexibility in broader conversational contexts. SCREAM, while more comprehensive and powerful for designing nuanced prompts, required more time and effort to apply, which made it less practical in quick development settings. Overall, this exercise highlighted how understanding and applying prompt frameworks can significantly enhance collaboration with AI tools, enabling more intentional and productive outcomes in software engineering tasks.