Documentation is essential for maintaining readable and maintainable codebases because it serves as the bridge between the developer's intent and the reader's understanding. Even well-written code can be ambiguous without clear explanations of design choices, function purposes, and expected behaviours. Documentation provides this context, allowing other developers — or the original author months later — to quickly grasp how and why a piece of code works. This reduces the learning curve for new contributors and minimizes the time wasted deciphering logic or debugging unexpected interactions.

Good documentation also enforces discipline in software design. When developers document their code, they are compelled to think critically about structure, naming conventions, and the clarity of their logic. This reflection often leads to cleaner, more modular implementations. It also creates a natural feedback loop: if a section of code is difficult to explain, it's likely too complex and needs refactoring. In this way, documentation acts as both a guide for readers and a design check for writers.

Beyond clarity, documentation is vital for collaboration and long-term sustainability. In any team environment, codebases evolve as new features are added, bugs are fixed, and technologies change. Without consistent documentation, institutional knowledge is lost as team members move on, forcing others to reverse-engineer design decisions. Well-maintained documentation — from inline comments to architecture diagrams — preserves that knowledge and keeps projects resilient against personnel changes.

Finally, thorough documentation enhances reproducibility and compliance, particularly in data-driven or regulated domains. For instance, when building financial or scientific software, auditors and collaborators need to verify that methods align with industry standards. Clear, version-controlled documentation provides a trustworthy record of functionality and rationale. In this sense, documentation is not merely a convenience but a cornerstone of professional software engineering, ensuring that readability, reliability, and accountability endure over time.