An opinionated approach to software documentation

Souvik Sarkar
sounix000@gmail.com

The hidden disconnect

Product documentation is often treated as an afterthought—a necessary evil that gets patched together after development is complete. Companies invest heavily in sophisticated tools and comprehensive user surveys, desperately trying to understand why their documentation doesn't resonate with users. Yet they're often looking in the wrong place.

The real problem isn't in the tools or the feedback mechanisms. It's a fundamental mismatch between what different stakeholders understand about the product and what ultimately gets delivered as documentation. This creates a dangerous disconnect between product philosophy and documentation philosophy.

Misaligned mental models

Every software product involves three distinct categories of stakeholders:

- **Visionaries**: Product managers, CTOs, marketers who see the big picture and market positioning
- **Builders**: Designers, software engineers, product architects who understand the technical implementation
- **Users**: Testers/QA, solution architects, support engineers, technical writers who experience the product from the end-user perspective

When these groups operate in silos, their different mental models never align. The documentation reflects only fragments of understanding, leaving users confused and frustrated.

A systematic solution framework

Here's a suggested approach to bridge this gap—for any feature requiring documentation, gather independent responses from your product manager, engineer, tester, and technical writer on the following aspects:

Context + problem

- What problem does this feature solve?
- Why did this problem occur?
- What happens if it remains unsolved?
- What benefits emerge when it's solved?

Solution + deployment

- How do use cases differ across target personas?
- How is the feature similar to or different from offerings by competitors?
- What are the feature's use cases?
- What workflows does the feature enable?
- What outcomes should users expect?
- What are the UI/UX implications?
- What dependencies exist?
- What are the performance and security implications?
- How does it integrate with other features and industry-standard products?

Pitfalls + troubleshooting

- When should this feature NOT be used?
- What happens with inappropriate usage?
- What precautions are necessary?
- How and when to disable/re-enable the feature?
- What does this feature break? What breaks this feature?
- What rollback options exist for production systems?

Why this approach works

When you collect these responses, you might discover fascinating discrepancies. The product manager might emphasize business outcomes that the engineer hasn't considered. The tester might identify edge cases that the product manager didn't anticipate. The technical writer might spot workflow gaps that everyone else missed.

These discrepancies aren't problems—they're opportunities. By surfacing and reconciling these different perspectives, you create a shared understanding that naturally leads to better documentation.

The organic emergence of quality

When everyone truly understands the product from multiple angles, something remarkable happens. Form, clarity, flow, and fit emerge organically. You don't need to force good documentation—it becomes the natural expression of shared understanding.

The technical communication to your target audiences becomes:

- **Unified** – Consistent messaging across all touchpoints
- **Comprehensive** – Covering all necessary aspects and edge cases
- **Genuinely helpful** – Actually solving user problems

More importantly, your documentation finally aligns with your product philosophy, creating a coherent experience that users can trust and navigate successfully.

Taking action

Every feature branch should have a file where those questions are answered by anyone who is in a position to answer them. When the team huddles for a sprint review or goes back to the drawing board to rethink the product, they will have a series of timestamped reference points that's more readable than code commits, and act as a single source of high-level truth that everyone in the team understands.

If you're in a position to implement this approach, start small. Pick one feature in your next release and run this exercise. You'll be surprised by what you discover about your own product—and how much better your documentation becomes when everyone is truly on the same page. The investment in alignment pays dividends far beyond documentation. It creates better products, clearer communication, and more satisfied users.