Create Technical Documentation

September 10, 2026

Create Technical Documentation

You can create technical documentation for a project to explain its architecture, behavior, and implementation details. Keeping this documentation current helps engineering teams understand how the project is organized and how its parts work together.

Overview

Technical documentation is intended for readers who need a deeper understanding of a project than a user guide provides. Use it to describe technical areas such as application architecture, component relationships, data handling, APIs, configuration, and infrastructure.

From your project's documentation area, you can work with **Technical Docs** alongside other documentation types, including **User Docs** and **User Manuals**. You can also use **Architecture Diagram** when a visual representation will make the technical guide easier to understand.

Create a technical guide when you need to record how a feature works, explain why a design decision was made, or provide implementation guidance for future maintenance. For example, you might document how frontend components interact with backend services, how database entities are related, or how an API schema supports a workflow.

> 📷 *Screenshot pending: The project documentation area showing the Technical Docs option and related documentation sections.*

Prerequisites

Before you start, make sure you have:

- Access to the project you want to document.

- Permission to create or update project documentation.
- A clear topic for the guide, such as architecture, behavior, configuration, or implementation.
- Relevant project information, diagrams, or examples to include in the guide.
- An understanding of whether the content belongs in **Technical Docs**, **User Docs**, or 📋 **User Manuals**.

Step-by-Step Instructions

1. Open the project that contains the feature, service, or workflow you want to document.
2. Select **Technical Docs** from the project documentation options.
3. Review the available technical documentation areas before creating content.
4. Select **Create Manually** to begin a new technical guide.

> 📷 *Screenshot pending: The Technical Docs area with the Create Manually action available.*

5. Enter a clear title that identifies the technical topic you are documenting.
6. Write an overview that explains the purpose of the feature or system.
7. Describe the architecture by explaining the major areas involved and how they relate to one another.
8. Select **Architecture Diagram** if you need to review or include a visual view of the project architecture.
9. Document the feature’s behavior by explaining what triggers it, what happens during processing, and what outcome users or systems should expect.
10. Describe implementation considerations that a technical reader needs to know, such as dependencies, configuration requirements, supported integrations, or data relationships.
11. Organize the guide into sections so readers can distinguish architecture details from behavior and implementation guidance.
12. Add practical examples when they help explain a technical workflow or relationship.

13. Select **Rewrite** if you want to revise the wording of existing content.
14. Select **AI: Rewrite** when you want assistance improving clarity, structure, or readability.
15. Review the rewritten content to ensure technical terms, project-specific names, and intended meaning remain accurate.
16. Add related documentation references, such as **Architecture**, **Guides**, or **User Guides**, when those resources provide useful context.
17. Select **Export Documentation** if you need a copy of the completed documentation for sharing, review, or offline use.

Tips and Best Practices

- Create one focused guide for each technical topic. For example, document architecture separately from deployment configuration or API behavior. This makes guides easier to maintain and easier for readers to find.
- Start with the reader's goal. Explain what the feature or area is responsible for before describing its technical details.
- Use **Architecture Diagram** when text alone does not clearly show relationships between parts of the project. Diagrams are especially useful for explaining how frontend, backend, database, API schema, configuration, documentation, and infrastructure areas connect.
- Describe relationships in plain language. For example, explain which component renders another component, which component uses a hook, or which service provides information to another area.
- Keep behavior descriptions chronological. Explain what starts the process, what intermediate actions occur, and what result is produced.
- Include implementation details only when they help readers maintain, troubleshoot, extend, or safely change the project. Avoid adding background information that does not affect technical decisions.
- Use **Rewrite** or **AI: Rewrite** to improve a draft, but always confirm that the revised text still reflects the project's actual behavior.
- Keep technical guides distinct from end-user instructions. Use **User Docs** or 📋 **User Manuals** for tasks that explain what business users see and do in the application.

Related Pages

- Architecture Diagram — Use visual diagrams to explain project structure and relationships.
- User Docs — Create documentation for end-user tasks and workflows.
- 📋 User Manuals — Organize procedural manuals for application users.
- Export Documentation — Share or retain a copy of completed documentation.
- Guides — Review other available project guidance.

Troubleshooting

You cannot find the option to create a guide

Open the project documentation area and select **Technical Docs**. If **Create Manually** is not available, verify that you have permission to create or edit documentation for the project.

The guide contains information that belongs in a user manual

Move user-facing steps, screen instructions, and business-process guidance to **User Docs** or 📋 **User Manuals**. Keep **Technical Docs** focused on architecture, behavior, relationships, configuration, and implementation details.

A rewritten draft changes the intended technical meaning

Use **Rewrite** or **AI: Rewrite** as a starting point only. Review the revised content, restore project-specific terminology where needed, and verify that descriptions of architecture and behavior are still correct before sharing or exporting the documentation.