Coding & Development / Documentation

API Reference Generator

Convert code signatures + comments into complete, human-readable API docs with working examples.
Difficulty: Intermediate
Model: GPT-4 / Claude / Gemini
Use Case: API Documentation, Library Docs, Internal SDKs
Updated: May 2026
Why This Prompt Exists
Most API documentation is either auto-generated (unreadable) or handwritten (outdated).

You get:

- Javadoc/Doxygen noise with no real examples
- parameter names without explanations
- missing edge cases and error responses
- docs that drift from actual behavior
- developers reading source code instead of docs

But good API docs follow a pattern:

- what the function does (one sentence)
- parameters with types and constraints
- return values and error conditions
- working code example
- see-also related functions

Without structure, docs are incomplete.

This prompt transforms code + comments into professional reference documentation.

The Prompt

```
Assume the role of a technical writer who specializes in API
documentation.

Your task is to generate complete reference documentation from code.

Generate:

1. FUNCTION/METHOD SIGNATURE
   - Formatted with type hints

2. DESCRIPTION
   - One-sentence summary
   - What it does, not how

3. PARAMETERS TABLE
   - Name, type, required/optional, default, description
   - Constraints (e.g., "must be positive integer")

4. RETURNS
   - Type and description
   - Success vs error scenarios

5. ERRORS / EXCEPTIONS
   - What conditions trigger each error
```

```
6. EXAMPLE
   - Minimal working code snippet
   - Example output if relevant

7. SEE ALSO
   - Related functions or endpoints

INPUTS:

Code Signature:
[PASTE FUNCTION/METHOD/ENDPOINT SIGNATURE]

Existing Comments (if any):
[PASTE DOCBLOCK OR COMMENTS]

Language:
[JAVASCRIPT / PYTHON / TYPESCRIPT / GO / RUST / OTHER]

Audience:
[BEGINNER / INTERMEDIATE / EXPERT]

RULES:
- If no comments exist, infer from parameter names and code structure
- Never leave a parameter undocumented
- Examples must be runnable (not pseudocode)
- Flag breaking changes or deprecations prominently
```

How To Use It

- Include type hints in your code signature — the prompt uses them heavily.

- Specify audience level: beginners need more explanation, experts want edge cases.
- For REST APIs, paste the route + expected JSON shape instead of a function signature.
- Review generated examples — they should compile/run in the target language.
- Use output directly in READMEs, Sphinx, JSDoc, or MkDocs.

Example Input

Code Signature:
def fetch_users(db_session: Session, limit: int = 50, include_inactive: bool = False) -> List[User]:
“””Retrieve users from database with pagination.”””
implementation omitted

Existing Comments (if any):
“””Retrieve users from database with pagination.”””

Language:
PYTHON

Audience:
INTERMEDIATE

Why It Works
Most doc generators produce signatures without meaning.

This framework improves outcomes by forcing:

- parameter documentation (no blind spots)
- error conditions (what actually breaks)
- working examples (not placeholders)
- audience awareness (right level of detail)

- related functions (discoverability)

Great API docs don’t list what code does — they help developers succeed on first use.

Build Better AI Systems

Subscribe for advanced prompt engineering, AI coding tools, debugging frameworks, and practical strategies for developers and engineers.

Carefully engineered prompts for people doing real work.

Share this:

- Share on Facebook (Opens in new window) Facebook
- Share on X (Opens in new window) X

See also README First Responder