Evergreen principles and auto-generating API documentation

AI can accelerate drafting, but without information architecture (IA) discipline it also accelerates disorder.

When generated content is allowed to accumulate without classification, ownership, or lifecycle rules, maintenance costs grow faster than the product itself. Teams spend time reconciling conflicting pages, answering repeat questions, and correcting downstream errors that originated in ungoverned docs. Applying IA guardrails early, before AI-generated files proliferate, keeps documentation searchable, scoped, and predictable. This turns AI from a short-term speed boost into a long-term cost control mechanism.

Consider a common failure mode. An AI-generated page describing an authentication flow is placed under a Quick Start instead of the API Specification. Later, the auth scheme changes. Engineers update the spec, but the Quick Start page is missed because it is not owned or classified as reference material. New integrators follow the outdated page, support tickets spike, and the team burns cycles explaining a problem that already has a correct answer elsewhere. Nothing was technically wrong with the API. The failure was structural. One misclassified page created duplicated effort, user confusion, and avoidable cost.

If you cannot clearly state which documentation style/type a page belongs to, who owns it, and how it will be updated to stay evergreen, that page is already an asset liability.

However, there are use cases to start a rough draft for your knowledge site, your “doc repo”. MkDocs is a static site generator designed for project documentation written in Markdown. When you combine it with the right plugins and external tools, such as `mkdocs-material`, `mkdocs-swagger-ui`, `mkdocstrings`, or Redoc integrations, it becomes a strong platform for API documentation.

Suggested plugins and external tools.

- `mkdocstrings` or `mkdocs-autoapi` plugins: These plugins for MkDocs can extract docstrings from your Python code (or other languages with appropriate

handlers) and automatically generate API reference documentation within your MkDocs site.

- `Pydoc-markdown` tool: This tool can generate API references in Markdown format, which can then be included in your MkDocs project.
- **Redoc integrations:** make MkDocs capable of publishing professional-grade, interactive API reference documentation without needing a separate portal.
    - **Redoc itself:** Converts your `openapi.yaml` or `openapi.json` file into a visually structured, developer-friendly page. *Redoc integrations* make MkDocs capable of publishing professional-grade, interactive API reference documentation without needing a separate portal.
    - **Integration with MkDocs:** You can use a plugin such as **`mkdocs-redoc-plugin`** or **`mkdocs-swagger-ui`** to display the rendered API spec within your MkDocs site.

Documenting APIs using OpenAPI/Swagger specifications:

- `mkdocs-swagger-ui-tag` or similar plugin: These plugins allow you to embed interactive Swagger UI or Redoc interfaces within your MkDocs pages, directly rendering your OpenAPI specifications for API documentation.

Serving MkDocs documentation alongside an API:

- You can host the generated static MkDocs site alongside an API built with frameworks like FastAPI.
  Example case: Build the MkDocs site into static HTML files and then configure your API framework to serve these static files at a specific endpoint.

See more about IA at www.susankraftconsulting.com