JOSHUA COTY CHRISTOPHER

Senior Software Engineer — Developer Experience & AI Tooling

Kansas City, MO • joshchrisw5@gmail.com

PROFESSIONAL SUMMARY

Software engineer with sixteen years of experience, currently focused on applied AI systems for developer platforms. At Meta, expanded a developer documentation role into AI engineering: an automated screenshot generation system built on Claude, AI-augmented authoring pipelines running on Claude and Llama 4, and MCP servers that expose SDK reference content as structured, retrievable context. Most of that work is retrieval and evaluation engineering rather than model training, because the hard problem is getting an assistant to answer from the real API surface instead of inventing a method name that sounds correct. Built an evaluation harness that scores generated documentation against source-of-truth SDK signatures, on the principle that an AI content pipeline without measurement is only a faster way to publish mistakes. That tooling is Python: the screenshot pipeline, the MCP servers, the evaluation harness, and the scripts that keep versioned SDK content in sync are all written and maintained in it. Also delivered developer-facing content for more than 40 REST APIs on Wit.ai, Meta's natural language processing platform, covering intent and entity modeling and training data design. Comes to this from engineering rather than from content: the first eight years were spent at a precision metrology R&D group writing C# and .NET software that talked directly to measuring hardware, where a defect meant a machine on a factory floor reported the wrong dimension. That led into developer experience at Red Hat, leading documentation for container and virtual machine migration onto OpenShift and serving as technical lead on a next-generation modular publishing platform. Works inside the codebase rather than alongside it, since most tasks start by reading the SDK source and building the sample project. Docs-as-code by default: AsciiDoc and Markdown in Git, reviewed through pull requests, built in CI, with automated linting to catch mechanical problems before a human reviewer has to. Has owned documentation for more than eight versioned SDK releases across two years, which means working to engineering's release calendar rather than a content calendar. Pays attention to information architecture and URL hygiene, because a restructure that quietly loses search ranking costs more than it gains. Wrote the style guide that product teams across the organization now write against, and mentors both writers and engineers on applying it. Comfortable presenting in public, having spoken at conferences on open-source documentation, publishing toolchains, and upstream and downstream collaboration.

TECHNICAL SKILLS

AI & Machine Learning: Python for AI tooling and pipelines, Claude and Llama 4 in production workflows, prompt engineering and staged multi-pass generation, retrieval-augmented generation, MCP server design and implementation, semantic chunking and retrieval-oriented content modeling, vision models for automated visual verification, LLM evaluation harnesses and regression testing, hallucination measurement against source-of-truth code, structured and schema-constrained output, human-in-the-loop review design, NLP intent and entity modeling (Wit.ai)

Languages: Python, C#, JavaScript, TypeScript, SQL, Bash, HTML5, CSS3, XML, JSON, YAML

Frameworks & Runtimes: .NET Framework and .NET, ASP.NET MVC, WPF, WinForms, Node.js, React, jQuery, Entity Framework, NUnit

Developer Platforms & SDKs: Meta Quest Unity SDKs (voice interaction, hand and controller interaction), Unreal Engine, Unity, Horizon Studio, Wit.ai NLP APIs, REST API design and documentation, SDK versioning and deprecation guidance

Docs-as-Code & Publishing: AsciiDoc and Asciidoctor, Markdown, modular documentation (concept, procedure, reference), static site generators, versioned documentation sets, Vale style linting, Git-based review workflows, release-train documentation

Cloud & Infrastructure: Red Hat OpenShift, Kubernetes, Podman and Docker, Linux (RHEL), Red Hat Virtualization, container and VM migration tooling, Jenkins, GitHub Actions, GitLab CI

Data & Storage: Microsoft SQL Server, PostgreSQL, SQLite, schema design, query tuning, reporting and export pipelines

Content Operations & SEO: Information architecture, URL migration and redirect mapping, search-equity preservation, discoverability and search analytics, content audits, localization handoff, style guide authorship and governance

Hardware & Instrumentation: Instrument integration over RS-232, USB, and Ethernet, real-time data acquisition, statistical process control (Cp, Cpk, Pp, Ppk), control charting, calibration and tolerance workflows

Collaboration & Process: Git, GitHub, GitLab, Jira, Bugzilla, Confluence, Figma, Agile and Scrum, code and content review, mentoring, cross-functional partnership with Engineering, Product, QA, and Support, conference speaking

PROFESSIONAL EXPERIENCE

Meta | Menlo Park, California May 2022 – Present
Senior Software Engineer — VR Developer Platforms

Responsibilities:

- Expanded the role from technical writing into AI-powered documentation automation, building and owning the systems that generate, validate, and structure developer content across the VR SDK portfolio.
- Co-built an AI-powered automated screenshot generation system in Python on Claude, replacing manual capture entirely. The system drives the target application, captures interface state, and uses a vision model to confirm the screenshot shows what the procedure actually claims before it is committed.
- Designed and refined AI-augmented authoring pipelines on Claude and Llama 4, covering draft generation from SDK source, structural rewriting, and automated consistency checks against the organization style guide.
- Built and maintain MCP servers in Python that expose SDK reference content as structured, retrievable context, so that assistants answer from the real API surface rather than generating plausible method names that do not exist.
- Restructured content specifically for machine retrieval: chunk boundaries placed at semantic units, explicit version and platform metadata attached to every chunk, and procedures written to remain correct when retrieved out of their surrounding context. Page structure optimized for human reading turned out to retrieve poorly.
- Built an evaluation harness in Python that scores generated documentation against source-of-truth SDK signatures and a held-out set of human-written reference pages, so regressions in the prompt chain are caught before a developer sees them.
- Measured and drove down hallucinated API references, which was the failure mode that mattered most. A wrong parameter name in generated documentation costs a developer more time than having no documentation at all.
- Owned the prompt engineering and the Python orchestration around it: few-shot examples drawn from the style guide, schema-constrained output for machine-parseable sections, and decomposition of long generation tasks into staged passes instead of one oversized request.
- Evaluated fine-tuning against prompting with retrieval for the authoring tasks and kept retrieval, since the SDK surface changes every release and a fine-tuned model goes stale on the next one.
- Kept a human review gate on everything generated. The automation removed the mechanical work; it did not remove accountability for what shipped to developers.
- Developed developer-facing content covering more than 10 features and over 40 REST APIs for Wit.ai, Meta's natural language processing platform, including intent and entity modeling, training data design, and the API workflows developers use to build conversational features.
- Lead developer documentation across major VR initiatives spanning Unity, Unreal Engine, and Horizon Studio, covering three engine idioms and three developer audiences that overlap far less than people assume.
- Led a complete restructuring of the Meta Quest Unreal Engine developer documentation, delivering more than 65 technical content updates and rewrites, reorganizing the information architecture, and managing URL migrations with redirect mapping so existing search equity was preserved rather than discarded.
- Own documentation for every major release of two Unity SDKs covering voice interaction and hand and controller interaction, supporting more than eight versioned releases across two years, on the engineering release calendar rather than a content calendar.
- Authored a comprehensive developer documentation style and writing guide establishing organization-wide standards for voice, tone, formatting, structure, and technical quality. Review cycles shortened once teams had a written standard instead of a reviewer preference to guess at.
- Build the sample projects before documenting them. A procedure that has not been run in the actual engine is a guess, and in VR the gap between the intended flow and the real one is usually a device permission or a scene setup step nobody remembered to mention.
- Partner with SDK engineers through the release cycle, reviewing API changes early enough that breaking changes ship with migration guidance instead of a support thread.
- Maintain versioned documentation across concurrent SDK versions, including deprecation notices and upgrade paths for developers who cannot move to the latest release immediately.
- Track discoverability using search analytics and developer feedback, and ran the redirect audit after the Unreal restructure to confirm nothing valuable had been orphaned.
- Review incoming documentation from product teams against the style guide and mentor engineers who write their own, which is most of them.

Environment: *Claude, Llama 4, MCP servers, retrieval-augmented generation, vision models, LLM evaluation harnesses, Python, C#, JavaScript, Unity, Unreal Engine, Horizon Studio, Meta Quest SDKs, Wit.ai, REST APIs, Markdown, static site generators, Git, GitHub, CI-based documentation builds, search analytics, Figma, Jira*

Red Hat | Raleigh, North Carolina Jul 2018 – Aug 2021
Principal Technical Writer

Responsibilities:

- Led documentation for Migration Solutions, supporting container, application, and virtual machine migrations onto Red Hat OpenShift. The audience was almost always under time pressure with a production workload at stake, which shapes how a procedure has to be written.
- Managed documentation for Red Hat's next-generation open-source modular publishing platform, working directly with the engineering team building it while also being one of its users.
- Served as Technical Lead, driving content strategy, mentoring writers, and improving documentation processes across the group.
- Created technical documentation for Red Hat Virtualization and OpenShift in collaboration with Engineering, DevOps, QA, Product, and Support.
- Installed and migrated the products being documented rather than relying on an engineering walkthrough. This consistently surfaced steps missing from the official procedures, usually prerequisites that were obvious to the person who built the feature.
- Improved the onboarding process so new writers reached productivity roughly twice as fast, replacing a wiki page with a working environment setup, a sequenced first assignment, and a named review partner.
- Contributed to the Red Hat Style Guide and to the modular documentation standards separating concept, procedure, and reference content.
- Worked docs-as-code throughout: AsciiDoc source in Git, changes reviewed as merge requests, builds run in CI, and documentation defects tracked in the same system as product bugs so they could be prioritized against them.
- Set up style linting in the build pipeline, with Python scripts checking cross-references and broken procedure steps on every merge request, which kept human review time on accuracy and structure rather than on comma placement.
- Coordinated upstream community documentation against downstream supported product documentation, which diverge more than outsiders expect and serve genuinely different audiences.
- Aligned documentation delivery to Red Hat release trains, including errata and z-stream updates where the content window is short and non-negotiable.
- Worked with the Support organization to identify which documentation gaps were generating tickets, and prioritized from that list rather than only from the feature launch schedule.
- Presented at conferences on open-source documentation, publishing tools, and upstream and downstream collaboration.

Environment: *Red Hat OpenShift, Red Hat Virtualization, RHEL, Kubernetes, Podman, Python, AsciiDoc and Asciidoctor, modular documentation, Git, GitLab, Jenkins, Vale, Jira, Bugzilla, Linux command line, virtualization and migration tooling*

Mitutoyo Research & Development America | Kirkland, Washington Jul 2010 – Jun 2018
Full Stack Developer

Responsibilities:

- Built desktop and web applications for measurement data collection and quality analysis, used by manufacturing customers in automotive, aerospace, and medical device production.
- Worked primarily in C# on the .NET Framework, starting in WinForms and moving to WPF as the product line was modernized.
- Wrote the instrument integration layer for gages and measuring devices over RS-232, USB, and later Ethernet, including protocol handling for older instruments that long predated any of those interfaces.
- Built the statistical process control reporting module covering control charts and capability indices (Cp, Cpk, Pp, Ppk), along with the report exports quality engineers submitted to their own customers as evidence.
- Designed SQL Server schemas for measurement records and tuned the queries as customer databases grew into the tens of millions of measured features. Indexing decisions that were fine at pilot scale were not fine three years in.
- Built an ASP.NET MVC dashboard so plant managers could see line status in a browser without a client install, which had been the single most common support complaint.
- Wrote the import and export layer for legacy file formats, since customers held years of historical measurement data in formats nobody was willing to lose.

- Implemented real-time data acquisition with local buffering and retry, because shop floor networks drop connections and an operator will not re-measure a part that has already moved down the line.
- Localized the applications for English and Japanese and coordinated with the parent engineering organization in Japan across a sixteen-hour time difference, which is where I learned to write a specification that survives without a follow-up call.
- Built the installer and update mechanism, and supported deployments on locked-down manufacturing networks with no outbound internet access.
- Took support escalations directly and traveled to customer sites a few times a year. Watching an operator use the software on a noisy factory floor changed how the logging, diagnostics, and error messages were designed.
- Worked with hardware engineers on calibration routines and tolerance handling, where the software had to respect measurement uncertainty rather than treat a reading as an exact number.
- Introduced unit testing with NUnit and regular code review, and helped move the team from Subversion to Git.
- Wrote the internal developer documentation and integration guides for the device APIs used by OEM partners. That work is what led to the move into developer experience.

Environment: *C#, .NET Framework, WPF, WinForms, ASP.NET MVC, JavaScript, jQuery, HTML, CSS, Microsoft SQL Server, Entity Framework, XML, RS-232 and USB instrument protocols, NUnit, Subversion, Git, Visual Studio, Windows deployment and installers*

EDUCATION

Missouri State University | Springfield, Missouri — Aug 2005 – May 2009

Bachelor of Science in Computer Science