Ross Todd

Senior full-stack software engineer

Location	Munich, Germany	Work authorization	Unrestricted
Languages	English native; German A0	Email	rssmtdd@gmail.com

Senior software engineer with ~9 years building web applications, backend services, and shared platforms at scale. At Alaska Airlines, built a component platform adopted across five host frameworks; versioned props and events, explicit contracts, and a deprecation policy made that reuse durable. At Nike, built a serverless consent platform handling 30,000 requests per second across four AWS regions, including China. Led a separate migration of more than 20 million records. The work centers on frontend and platform architecture, event-driven systems, performance grounded in measurement, and the contracts and documentation that let teams ship independently. Core stack: TypeScript, JavaScript, React, Next.js, Node.js, Python, Rust, and AWS.

Based in Munich. My residence permit carries unrestricted work authorization, so no sponsorship and no work permit are required.

Experience

Independent engineering — Munich, Germany

Jan 2026 – present · Self-directed

Systems work in Rust, TypeScript, and PostgreSQL since relocating to Munich. This is self-directed work rather than client engagements, and it is listed as such. Each system below has a page at rsstdd.com/projects stating what was read in the source, what was measured, and what was not.

- Built an aircraft management engine in Rust 2024 and PostgreSQL as a hexagonal Cargo workspace, where the dependency direction is enforced by a workspace

automation task rather than by convention. Imported values stay non-canonical until a curator accepts them, because a parser proving a value is well-formed is a different claim from a curator proving it is correct. Every HTTP refusal is an RFC 9457 problem document, including the body-limit and timeout cases, which are written by hand because the `tower-http` layers answer with bodyless responses.

- Built the concurrent Rust scraping pipeline that produces its source data, with bounded backpressure through capacity-scaled channels, barrier-coordinated shutdown, and rate-limit-aware retry with capped jittered backoff.
- Built a robot telemetry console that treats silence as an event, normalizing three vendor dialects into one canonical envelope and degrading freshness on a server-side sweep rather than in the browser.
- Built a local-first Rust and Python text-to-speech pipeline engineered so a stochastic generator renders reproducibly, with every input that can change the audio named in the cache key.
- Built this site: statically generated Next.js 16 with content in git, zero client components, and a Zod content schema that fails the build rather than rendering a blank card.

Alaska Airlines — Senior software engineer, content engineering

Feb 2024 – Jan 2026 · Contract · Remote, USA

Shared frontend and content platform for alaskaair.com through the Alaska–Hawaiian integration, spanning Borealis (a component platform), RetroStack (the server-rendered content application on Kubernetes in Azure), and the Sitecore-to-Contentstack migration.

Platform architecture

- Architected and delivered Borealis, a framework-agnostic Lit and Web Components platform adopted by four to five product teams across Next.js, Svelte, React, Vue, and .NET host applications. It held across five frameworks because props and events were contract-first, versioned with semver and a deprecation policy, so consuming teams upgraded on their own schedule rather than mine.
- Pulled CMS-to-props mapping out of the components, so a Contentstack schema change could not break every consumer at once, and built the deterministic SSR-safe registration and hydration utilities the platform depended on. It ran mixed-mode through the transition: true Lit server rendering on some surfaces, client-only mounts inside server-rendered pages on others.

- Built the dual-CMS abstraction for the Next.js application behind the homepage and content pages, normalizing two content systems onto one set of render models: Sitecore through GraphQL queries and a library of page-layout models, Contentstack through SDK helpers with field mappers. Either could serve a page without the rendering layer knowing which, which is what let the migration run page by page instead of as a rewrite.
- Designed the offer-orchestration integration: one backend call per server-rendered page, roughly 30 ranked offers deduplicated on canonical keys and partitioned into CMS-defined zones with capacities and priorities, embedded as a read-only snapshot that islands hydrate from with zero client-to-API calls afterward.
- Authored the Component Personalization Service architecture, including GDPR and CCPA consent gating for personalization, against a stated 99.9% monthly availability target. Architecture and proposal stage rather than shipped.

Reliability and performance

- Led root-cause analysis and authored the incident report for a Sev2 outage in which an infinite request loop saturated backend infrastructure and degraded homepage flight search. Proved the two rolled-back commits were not the cause, isolated a pre-existing race condition exposed by a new skeleton loader, and defined remediation aimed at the class of defect rather than the single instance.
- Ran a six-scenario controlled Lighthouse matrix isolating each third-party script against a fixed baseline. Removing Optimizely alone moved the performance score from 60.33 to 70.67, cut total blocking time 65%, and cut time to interactive 1.13 seconds; AppDynamics was second at 60% of total blocking time. The per-script deferral strategy was adopted into planning.

Delivery and standards

- Led the Flight Search retro after a core feature pull request sat blocked for over two months behind chained upstream dependency fixes, analyzing the failure across roughly 20 cited issues and pull requests. The outcome fed org-wide standardization of web-component events and APIs plus readiness gates, and drove recurring cross-team architecture meetings.
- Led Sitecore-to-Contentstack migration planning with EPAM, defining integration contracts and documentation governance and writing the Contentstack onboarding cookbook internal teams used instead of depending on the vendor.

- Built the platform's CI/CD and infrastructure on Azure Static Web Apps with Terraform-provisioned Front Door, Key Vault, and a WAF policy, plus changesets-based releases, Storybook, and the deployment and visual QA runbooks that turned tribal knowledge into repeatable process.
- Won the 2025 organization-wide AI hackathon on a team that built an LLM chatbot grounded in internal FAQ content.

TypeScript, Lit, Web Components, React, Next.js, Svelte, Vue, .NET/C#, Node.js, Vite, Vitest, Playwright, Storybook, Azure (AKS, Front Door, Static Web Apps, Key Vault, APIM), Terraform, Sitecore, Contentstack, Optimizely, GraphQL

Nike — Senior software engineer, privacy and compliance engineering

May 2022 – Jul 2023 · Contract · Remote, USA

Backend and platform engineering on User Permission Services, the serverless consent platform behind Nike's digital experiences globally and in China. Concurrent with the Volume engagement from May to Sep 2022.

- Built and operated the consent service layer: the publish service wrote events to Kinesis, fanning out through SQS queues with dead-letter queues into three ingest Lambdas for purpose-limitation state, the data lake, and the audit trail, with DynamoDB persistence, Firehose delivery to S3, and X-Ray tracing across the service graph.
- Implemented the consent and permission REST APIs on a domain model separating implicit consents, meaning legal defaults per jurisdiction, from explicit user choices, with every consent event referencing a versioned context frame recording the exact interface the user saw, so the audit trail could prove what was consented to.
- Deployed and integration-tested across four AWS regions behind Akamai globally and AliCloud in China, and owned the China side: replicated global routing via AliCloud and root-caused behavioral divergence between the two CDNs.
- Led the push-notification consent migration off a third-party vendor. Root-caused the first pass's failures to DynamoDB conditional-write collisions during bulk ingest and fixed them, authored the plan and cutover runbook for the second, validated Glue jobs against production-scale mock data in test, and wrote the data loading in Rust after Node.js hit memory limits on the full dataset. The entire relevant dataset, roughly 20 million records, migrated and the vendor dependency was eliminated, with a 93% opt-in rate retained.

- Performance-validated the platform at launch scale: p95 latency of 174 to 201ms across runs of roughly 31.6 million and 1.85 million invocations with 5xx rates below 0.001%, then authored the AWS limit-increase intake for a 30,000 requests-per-second launch scenario after load-testing at that rate. Shipped and validated the API Gateway caching layer behind it.
- Enabled Codecov across the pipeline with 90% coverage thresholds enforced in CI and an integration-test matrix parameterized across all four regions, and onboarded two engineers with documentation and technical demos for distributed teams.

Node.js, JavaScript, Python, Rust, AWS (Lambda, API Gateway, Kinesis, Firehose, SQS, S3, DynamoDB, Glue, Athena, X-Ray, CloudFormation), Databricks, Jenkins, Gatling, Akamai, AliCloud

Volume Media — Senior software engineer

May 2021 – Sep 2022 · Contract · Remote, USA

Full-stack engineer on a browser-based live video and audio streaming platform for musicians, Dockerized on Google Cloud. Concurrent with the Nike engagement from May to Sep 2022.

- Audited the inherited Django monolith, 1,073 directories and 4,057 files of server-rendered templates, and made the case for migration; then co-led the incremental move to React and Next.js one page at a time, highest-traffic first, with Django continuing to serve everything not yet migrated so there was no cutover risk.
- Solved the Django-to-Next.js authentication bridge, the technical crux of the migration: CSRF token flow and HttpOnly session cookies against Django REST Framework, with the CORS changes that let credentials survive cross-origin requests. It became the pattern the rest of the migration followed.
- Built revenue and creator features end to end, including a Stripe-integrated purchase flow and artist profile panels spanning Django models and views through to React components.
- Introduced a design system on atomic design with Storybook and rebuilt the Webpack configuration for code splitting and correct production bundling. Time to first paint and time to interactive both improved.

Python, TypeScript, React, Next.js, Django, Django REST Framework, Docker, Google Cloud Platform, PostgreSQL, Webpack, Storybook, Stripe, GraphQL

Lululemon — Lead software engineer, content and education

Apr 2020 – May 2021 · Full-time · Remote and Seattle, WA

Promoted from software engineer after 11 months, leading the team owning product detail pages on lululemon.com, the highest-traffic and highest-revenue pages on the site, through the 2020 e-commerce surge and heavy cross-team dependencies.

- Restructured the application to consolidate duplicated business logic into shared services and extracted presentational UI into Storybook, and drove automated coverage above 90%.
- Ran sprint planning and refinement on a quarterly cadence coordinated against the A/B testing calendar, so feature work and live experiments shared one schedule rather than competing for the same surface.
- Built engineering culture structurally rather than ad hoc: a Community of Practice, a design-system champions group, a documented team goals framework, and the onboarding documentation new engineers started from.
- Ran 1:1s and contributed input to performance reviews; formal people management remained with the engineering manager.
- Flagged security and privacy defects through security review, including logout failing to invalidate an access token held in duplicate, a third-party script blocking some logout calls, and an analytics vendor collecting identity data.

JavaScript, React, Next.js, Storybook, React Testing Library, Jest, Webpack, SCSS, GraphQL/Apollo, Splunk

Lululemon — Software engineer, content and education

May 2019 – Apr 2020 · Full-time · Remote and Seattle, WA

Frontend engineering on the product detail page and content experiences, delivered with UX, product, brand, and engineering partners across the Web App, Adobe AEM, and Endeca/ATG teams.

- Led the GraphQL migration plan for the product detail page, the company's riskiest surface: authored a component-by-component audit mapping every component to the exact fields it pulled from the legacy gateway, flagging over-fetching where a component pulled an entire product response for two values, then sequenced the epic as a strangler so a GraphQL query could first mirror the legacy response shape and let

consumption switch without component rewrites. Rollout was planned behind an experiment flag so it could serve a rising share of real traffic with instant rollback, and the client was chosen against the company's GraphQL serving architecture rather than by preference.

- Designed and built the generic size-guide platform, replacing one-off pages with statically generated Next.js dynamic routes and a table design system consuming per-category datasets. The work spanned source data mapping from the legacy commerce system, an SRE epic for the infrastructure, a per-section heading and markup specification developed with the SEO team, a redirect strategy, localization onto the French storefront, and an audit normalizing inconsistent legacy conversion charts before templating them. Adding a garment category became a content task rather than an engineering one.
- Drove the code-splitting and lazy-loading architecture for the page, splitting eagerly rendered above-the-fold components from everything below the fold or interaction-gated, including the image carousel, recommendations, video player, and store-pickup and zoom modals, which moved behind dynamic imports.
- Delivered image optimization, script cleanup, code splitting, and lazy loading as part of a portfolio of performance work credited internally with contributing to nine-figure annual gains. Changes shipped inside measured A/B experiments rather than alongside them, which is the honest basis for that attribution; the credit belongs to the team's combined work rather than to any one change or engineer.
- Shipped conversion-facing features across the page with cross-functional partners, including the product image lightbox, "Why we made this," the Bra Quiz, ratings and reviews, social content, and store-pickup mapping. Dependency choices were made against a scored evaluation matrix rather than by preference.

JavaScript, React, Next.js, Storybook, Jest, Webpack, SCSS, GraphQL/Apollo, Adobe Target

Migo, Inc. — Software engineer

Feb 2018 – Apr 2019 · Seattle, WA

Transportation aggregation platform with large-scale static page generation.

- Built the static-generation pipeline that shipped millions of pages for search indexing, and isolated and parallelized the build to work around Node.js memory limits during compilation, which was what made generation at that volume possible at all.

- Wrote large-scale web scrapers and established the database layer behind the catalog.
- Refactored SCSS to ITCSS, cutting rule declarations by roughly two thirds and media queries by roughly half while reducing specificity.

JavaScript, Python, React, Gatsby, Node.js, DynamoDB, AWS S3, Webpack, SCSS

Earlier roles

Humming — Web application developer · Jul 2017 – Aug 2018 · Seattle, WA Early-stage localized advertising platform built by a two-person team. Owned the Node.js and Express server, the database schema and REST endpoints, and deployment on Google Cloud Platform. Early advertising-platform exposure rather than mature ad-serving experience.

LaunchCode — Teaching fellow · Jul 2017 – Aug 2018 · Seattle, WA Taught and mentored students in LC101, a frontend JavaScript and Java program. The cohort recorded the highest retention rate of any to that point.

IUNU — Software developer, contract · Apr 2017 – Jun 2017 · Seattle, WA Greenhouse environmental and inventory monitoring. Built a React, Redux, Node.js, and MongoDB application with MongoDB and Redis in production, wrote the Python agents collecting sensor and camera data on edge devices, and used Ansible to push configuration to field IoT devices.

Additional experience

University of Washington — Substitute instructor, continuing education · Jan 2020 – Jan 2026 Taught the full-stack web development curriculum part time across six years alongside other roles.

RMT Dev LLC — Owner · Apr 2022 – Dec 2025 · Remote Software engineering consultancy run alongside other roles. Next.js and Shopify storefronts, plus analytics and SEO advisory.

Skills

- **Languages:** TypeScript, JavaScript, Python, Rust, C#
- **Frontend:** React, Next.js, Lit, Web Components, Storybook, server-side rendering, hydration, islands architecture, design systems, SCSS and CSS Modules
- **Backend and APIs:** Node.js, Express, Django, Django REST Framework, Flask, REST, GraphQL, event-driven and serverless service design, backend-for-frontend patterns
- **Data:** PostgreSQL, DynamoDB, MongoDB, Redis, Kinesis, Kafka, Glue, Databricks
- **Cloud and infrastructure:** AWS (Lambda, API Gateway, S3, IAM, CloudFormation, X-Ray, SQS, SNS), Azure (AKS, Front Door, Static Web Apps, APIM), Google Cloud Platform, Terraform, Docker
- **Build, test, and CI/CD:** Vite, Webpack, changesets, Jenkins, GitHub Actions, GitLab CI, Vitest, Jest, Playwright, React Testing Library, Gatling
- **Practices:** Platform and API contract design, semver and deprecation policy, migration planning, incident response and root-cause analysis, performance measurement, technical documentation, mentoring, GDPR and CCPA compliance engineering

Education

Galvanize Inc. — Certificate, full-stack web development · 2016–2017 · Seattle, WA **Louisiana Tech University** — M.A., English language and literature · 2013–2015 · Ruston, LA **Louisiana Tech University** — B.A., history · 2010–2013 · Ruston, LA

Updated 2026-09-10