NEELESH KAKARAPARTHI

Senior Software Engineer · Java 21 / Spring Boot 3 · Next.js / React / TypeScript · PostgreSQL / Kafka · Distributed Tracing

845-309-0645 | neelesh1206@gmail.com | LinkedIn | Redmond, WA

8+	150x	38	+6.8%	>92%
Years @ Walmart	Homepage Scale (20 → 3,500+)	Multi-Tenant Storefronts	ATF Content CTR (mWeb)	W3C Trace Propagation

⚡ CORE COMPETENCIES

Java 21 · **Spring Boot 3** · **Hibernate JPA / QueryDSL** · **PostgreSQL (Partitioning)** · **Kafka / Transactional Outbox** · **Next.js 15 / App Router** · **React 19** · **TypeScript 5** · **Node.js / Fastify BFF** · **GraphQL & REST** · **Multi-Tenant Architecture** · **W3C Distributed Tracing** · **Kubernetes (WCNP / GKE)** · **CI/CD (Concord / LooperPro)** · **Production Observability**

👤 PROFESSIONAL SUMMARY

Full-Stack Senior Software Engineer with **8+ years** at Walmart Global Tech, shipping production systems across **38 multi-tenant storefronts** (Walmart US, Canada, Sams Club, ASDA, Mexico, Health, B2B). Comfortable across the entire stack — currently leading the **Tempo V3 UI** rewrite on **Next.js 15 / React 19 / TypeScript** with a Fastify BFF serving 15 downstream services, and contributing to **PRISM's `cxt-msg-asset-service`** backend on **Java 21 / Spring Boot 3 / PostgreSQL / Kafka** that scaled Walmart homepages **150x (20 → 3,500+)**. Deep expertise in REST + GraphQL APIs, transactional outbox / CDC patterns, multi-tenant data access, distributed tracing, and WCNP / Kubernetes / GCP production operations.

🛠️ TECHNICAL SKILLS

Languages:	Java 21, TypeScript, JavaScript (ES2024), Python, SQL
Backend:	Spring Boot 3, Hibernate JPA, QueryDSL 5, MapStruct, Walmart Strati (`io.strati.persistence`), Forklift (transactional outbox), Node.js 20, Fastify, hapi, NestJS, REST, GraphQL, gRPC
Frontend:	Next.js 15 (App Router), React 19, React Hook Form + Zod, TanStack Query, Redux Toolkit, Apollo Client, Tailwind CSS, Walmart Living Design, Angular
Data:	PostgreSQL (list partitioning, deferred-cascade FKs, Liquibase), MS SQL Server, MongoDB, Cassandra, Cosmos DB, Apache Kafka, Memcached (Meghacache)
Cloud & Infra:	GCP, Azure, Kubernetes (WCNP / GKE), Docker, Istio, KITT, Concord, LooperPro, Jenkins, Akeyless
Observability:	OpenTelemetry, W3C Trace Context, OpenObserve, Grafana, Prometheus, Quantum Metrics
Testing:	JUnit 5, Mockito, JaCoCo, SonarQube, Jest, React Testing Library, MSW, Playwright, Enzyme
Practices:	Agile / Scrum, TDD, CI/CD, code review, on-call rotation, SLO design

💼 PROFESSIONAL EXPERIENCE

Walmart Global Tech CURRENT

Oct 2017 – Present

Software Engineer — Tempo CMS & PRISM (Full-Stack)

Redmond, WA

Tempo V3 UI (current) — Lead frontend / BFF engineer on the rewrite of Walmart's merchandising CMS authoring tool, used by hundreds of site merchants to publish content across **38 production storefronts**.

- Migrated the authoring UI from legacy **Electrode V1 + GraphQL + React 16** onto **Next.js 15 (App Router) + React 19 + TypeScript 5** on a Fastify custom-server (`@walmart/wml-server-fastify`) — cut cold-start build time by **~60%** and shrank the client bundle via React Server Components + TanStack Query streaming Suspense.
- Designed and shipped a unified **/api/proxy Fastify BFF gateway** fanning out to **15 downstream Walmart services** — collapsed 15 duplicated GraphQL resolver layers from V2 into a **~50-line typed-client template**.
- Implemented **W3C Trace Context propagation** across the BFF — validated **>92% propagation** in stage (1,284 of 1,387 inbound requests over 2 hours), unblocking first-time end-to-end distributed tracing for Tempo.
- Replaced the **IAM + RMA auth chain with RMA-only authorization** through the Role Manager Auth Engine — enforcing access at **tenant × pageType** granularity and cutting outage modes from two systems to one.
- Built tenant-aware **Next.js Edge middleware** handling `/{tenant}/` URL rewrites across all 38 tenants, cookie sync, feature-flag-driven legacy fallback, and SSRF / open-redirect protection.
- Deployed **multi-region active-active** across three production clusters under the KITT `nextjs-electrode-v1` profile with Concord + LooperPro pipelines and Akeyless-managed signature secrets.

Stack: *Next.js 15, React 19, TypeScript 5, Fastify, TanStack Query, Zod, OpenTelemetry, OpenObserve, Living Design, Playwright, WCNP, KITT, Concord, Akeyless.*

Message Asset Service — PRISM Backend — Backend engineer on `cxt-msg-asset-service`, the system of record for PRISM: the **Java 21 / Spring Boot 3** platform that decouples customer-facing copy and creative from Tempo modules into reusable, personalizable Messages and Assets — powering Tempo V2, Tempo V3, MMUI, and downstream P13N personalization.

- Co-owned the PRISM backend that **scaled Walmart homepages 150x** (from 20 to **3,500+ unique homepages**), drove **+16% efficiency** on CP/HP page-build workflows and **+6.8% ATF Content CTR (mWeb)**, and now holds **12k+ messages and 24k+ assets** in production at ~2 TPS with p95 ≈ 250ms.
- Modeled the PRISM domain — `message ↔ asset (1:N)`, `asset ↔ asset_config (1:N)`, `message_pages`, `message_targeting`, `message_group`, and a `message_hierarchy` rooted at an M0 parent — plus the `Draft → In Progress → Published → Ended / Archived / Unpublished` state machine consumed by P13N for per-customer surface decisions.
- Designed a **PostgreSQL list-partitioned schema** with composite primary key `(asset_id, status)` and a **deferred-cascade FK** from `asset_config` — enabling lifecycle-driven partition row-movement (`ACTIVE → ARCHIVED`) across `assets_active / assets_inactive / assets_default` without losing referential integrity.
- Architected a **four-datasource layout** (`postgres`, `postgres-no-cdc`, `postgres-wtp`, `postgres-asset-ingestion`) over Walmart's Strati framework with **CCM-driven role injection**, toggling Kafka emission per write-path and isolating translation + ingestion workloads from core OLTP load.
- Hardened DB → Kafka reliability with the **Forklift transactional-outbox / tracker-table pattern**, writing the outbox row inside the same DB transaction as the entity mutation so downstream consumers (**Pronto, IronBank, asset-discovery, P13N**) never see a dual-write inconsistency.
- Delivered the auto-ingestion pipeline (**Figma → Digital Eyes → SmartCreative → Pronto → Asset**) with `unique_ingested_figma_asset_constraint` for idempotency and a partial unique index (`WHERE status = 'START'`) enforcing single-flight ingestion per tenant per metadata key.
- Shipped the **Dynamic Asset Update (DAU) pipeline** that propagates CTA / attribute changes from the canonical asset out to every PRISM module's deep-copied default — keeping non-personalized renders consistent with the system of record.
- Built a **multi-tenant data-access layer** that transparently injects a `tenantId` predicate on every query via a request-scoped `WcpHeaders` bean and `GenericBaseDAO.getDefaultPredicates` — eliminating per-DAO tenancy plumbing across 25+ repositories.

- Implemented **optimistic concurrency control** with Hibernate `@Version` on a `BaseDO` superclass and a structured `LockException` unwrap pipeline (Strati → Found → Persistence → PSQL) that surfaces deeply-nested errors as actionable **HTTP 409**s on bulk operations.
- Integrated **Digital Eyes** asset-image analysis and the **Walmart Translation Platform** — persisting translation requests to a dedicated `postgres-wtp` datasource and triggering them on every asset insert and default-locale change.
- Deployed to GKE behind GSLB `api.message-asset.experiencetools.prod.walmart.com` with Istio sidecar, Akeyless-managed secrets, and a SonarQube quality gate at 80% — running for ~$11/day in combined cloud spend.

Stack: *Java 21, Spring Boot 3.5, Hibernate JPA, QueryDSL 5, MapStruct, Walmart Strati, Forklift, PostgreSQL (Liquibase), Kafka, OpenTelemetry, JUnit 5, JaCoCo, SonarQube, WCNP / GKE, KITT, Concord, LooperPro, Akeyless.*

Tempo V2 UI (Tempo-CMS) — Frontend engineer on the legacy Tempo authoring tool while it served production traffic in parallel with the V3 rollout.

- Extended module-editor UIs and lifecycle (`draft → staged → published → unpublished → archived`) on a **React 17 + TypeScript + Redux Toolkit + Apollo Client** Nx monorepo with a Koa + apollo-server-koa BFF schema-stitching 26 GraphQL providers.
- Wired role-driven access via a config-driven **`roleFeatureMap`** (Redux + `authorizationUtil`) so merchants only see modules for the tenants and page-types they own.
- Integrated **Mixpanel** for authoring telemetry that directly shaped V3 design decisions; raised Jest + React Testing Library coverage on critical paths.

Stack: *React 17, TypeScript, Redux Toolkit, Apollo Client, Styled Components, Node.js, Koa, Jest, Docker, Looper / Concord / Jenkins, Nx monorepo, Mixpanel.*

Tango (Taxonomy Management) — Full-stack engineer on the self-service tool that automates Walmart's site browse hierarchy for merchandising teams.

- Designed and shipped front-end (React) and back-end (Node.js / GraphQL) features so merchandisers can visually edit and publish category hierarchies, replacing ticket-driven workflows for tenants like **walmart-usgm, brandshop, R2D2, SamsClub**.
- Extended GraphQL schemas + resolvers to support bulk category moves, the CCM-configurable per-tenant state machine (`Draft → Staged → Published → Unpublished → Archived`), and audit history with robust validation.
- Contributed to the **cursor-pagination redesign** for large category trees, replacing parallel-fetch-all-children with Tango Service v2's `nextCursor` API — protecting CosmosDB from recursive-query timeouts on folders with hundreds of children.

Stack: *React 16.14, Node.js 16.20, GraphQL (Apollo), Java 21 / Spring 6 service, CosmosDB, Kafka, Jest, Enzyme, Docker, Looper, Concord, Yarn, Nx monorepo.*

Tempo A/B Testing — Sole engineer on the Tempo + Expo integration that lets merchants run A/B experiments on individual modules without leaving Tempo.

- Designed and shipped an end-to-end React + GraphQL workflow that captures merchant inputs and calls Walmart's Expo platform APIs to create, configure, and monitor module-level experiments — packaged as an npm module and published to the Walmart registry.
- Owned the full product surface from architecture review through implementation, code review, and production deployment.

Stack: *React, Saber UI, Redux, GraphQL, Jest, Yarn, Looper, OneOps.*

Earlier Walmart Projects

- **Modular Draw** — Full-stack engineer on a planogram-automation tool. Angular + Angular Material UI as an npm package, NestJS microservices, MongoDB, Kafka pub/sub on Azure (Blob, Cosmos), serverless on WCNP.
- **Workbench** — Full-stack engineer on an analytics platform for buyers and planners with historical sales and market-trend dashboards (AmChart, AG Grid) on Angular + NestJS + SQL Server + Cosmos.

- **Assortment Review / Line Review** — 5-step Angular 5 + NgRx + AG Grid front-end for assortment planning, backed by a C# .NET Web API 2.0 and RabbitMQ; team of 7 engineers + 3 QE.
- **Apparel Planning and Reporting** — Front-end engineer on an Angular 1.5 + C# tool letting apparel planners view last-year sales / profit patterns and order by color / size per store across the US.

CVS Health
Sep 2016 – Aug 2017

UI / MEAN Stack Developer
Richardson, TX

- Built responsive web UIs (HTML5 / CSS3 / AngularJS / Bootstrap) and RESTful services (Node.js / Express / MongoDB), including the Splunk-backed REST API powering big-data dashboards.
- Implemented client-side authentication with Passport.js and integrated with existing Java / Spring / Hibernate services.

EDUCATION

Master of Science — Computer Science *2016*
Southern Arkansas University · Magnolia, AR

Bachelor of Technology — Electrical & Electronics Engineering *2014*
Kakatiya University · Warangal, India