Nils Matteson

linkedin.com/in/nilsmatteson nilsmatteson.com nilsmatteson@icloud.com Madison, WI

Data Science & CS Senior building **end-to-end ML systems**—from custom streaming infrastructure in Go/Rust to autonomous training pipelines. Experience deploying production RAG systems on AWS. Seeking ML Infrastructure, Backend, or Data Science roles.

Education

University of Wisconsin–Madison Madison, WI

B.S. Data Science, Minor in Computer Science *Expected May 2026*

- **Systems & AI**: Big Data Systems (CS 544), Causal Inference (STAT 479), Artificial Intelligence (CS 540), Machine Organization (CS 354), Programming III (CS 400), Intro to Computer Engineering (CS 252).
- **Data Science & Math**: Data Science Modeling I & II (STAT 240/340), DS Programming II (CS 320), Linear Algebra (MATH 340), Discrete Math (MATH 240).

Technical Skills

Languages: Python, Rust, Go, C++, SQL, TypeScript/JavaScript
ML & Data: XGBoost, Scikit-learn, PyTorch, Pandas, Feature Engineering, A/B Testing, LLMs, RAG, Hugging Face
Systems & Cloud: AWS, GCP, K8s, Docker, gRPC, Kafka, Redis, Distributed Systems, Postgres
DevOps & Web: CI/CD, Git, Linux, GitHub Actions, React, Next.js, WebSockets, WebGL

Experience

Research Cyberinfrastructure, UW–Madison DoIT Madison, WI

AI Workflows Research Collaborator *Jan 2026 – Present*

- Benchmarked **10 LLMs on AWS Bedrock** (Claude, Llama, DeepSeek R1, GPT-OSS) across 282-question sustainability Q&A dataset; designed **weighted scoring framework** with Jaccard ref-overlap and NA recall metrics to expose real model differentiation.
- Built full **Bedrock integration layer**—async inference with retry/backoff, per-model inference profiles for cost attribution, and HTTP-header token extraction to fix vendor-specific tracking gaps (DeepSeek R1).
- Performed **Pareto cost-efficiency analysis** across all models; discovered GPT-OSS 120B matches 97% of top-model accuracy at **95% lower cost** ($0.51 vs $10.91), directly informing deployment model selection.
- Architecting production deployment: Streamlit UI on AWS with S3-hosted vector indices, comparing cloud vs. on-prem GPU cluster (2× RTX 6000) for cost/latency tradeoffs on 12+ open-source models.

Selected Projects

Madison Metro ML: Autonomous Bus Arrival Prediction *Python, XGBoost, Sentinel, PostgreSQL, React*

16K+ LOC end-to-end ML system with ground truth generation, autonomous retraining, and live inference.

- Designed **geospatial ground truth pipeline**: Haversine-based arrival detection matches GPS coordinates to stops (30m threshold), then joins to predictions to compute actual vs. predicted error—solving the "how do we validate transit predictions?" problem.
- Built streaming data pipeline using **Sentinel** (custom message queue) with gRPC producers, dual-threaded consumers with offset tracking, and PostgreSQL persistence for 7K+ daily observations.
- Implemented **autonomous nightly retraining** via GitHub Actions: XGBoost regression on rolling 7-day window with metric-gated deployment (MAE improvement threshold) and Git-versioned model registry.
- Deployed full stack on Railway (collector, consumer, API) and Vercel (React dashboard with MapLibre live tracking); engineered temporal and route-aggregated features with proper train/test split to prevent data leakage.

Sentinel: Distributed Log Streaming Engine *Go, gRPC, Protobuf, LSM Trees, Raft*

Kafka-inspired message queue (5,600+ LOC) powering Madison Metro ML's real-time data pipeline.

- Engineered custom **LSM-tree storage engine** with skip list memtable achieving 1.7M writes/sec and 3.9M reads/sec; implemented CRC32 checksums, bloom filters, and crash-safe write-ahead log.
- Built **Raft consensus layer** for fault-tolerant leader election and log replication; designed gRPC streaming API with topic/partition semantics, consumer groups, and offset tracking.

Synapse: Real-time Collaborative Whiteboard *Rust, WASM, WebSockets, CRDTs (Yjs)*

Lock-free distributed canvas (Rust + WebAssembly) supporting 50+ concurrent editors.

- Built Rust WebSocket server (Actix) with **CRDT state sync** (Yjs) for conflict-free concurrent editing; compiled rendering to **WebAssembly** achieving 60 FPS with 10K+ vector objects.
- Architected horizontal scaling via **Redis Pub/Sub** with session affinity for 50+ concurrent editors.