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), Machine Learning (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 **11 LLMs** on hierarchical RAG pipeline for AI sustainability research; identified scoring bug in boolean outputs, boosting Claude 3.7 Sonnet from 0.570 to **0.721 accuracy** (+26%).
- Built **AWS Bedrock integration** with retry logic, concurrency limits, and throttling handling; optimized indexing from CPU to CUDA, reducing build time from hours to minutes.
- Designed evaluation framework analyzing cost/performance tradeoffs across models; discovered Claude 3.7's edge comes from **citation quality** (0.850 ref overlap vs 0.73 baseline), not raw accuracy.
- Architecting production deployment: Streamlit UI on AWS with S3-hosted SQLite vector indices, comparing against on-prem GB10 GPU cluster for cost/latency analysis.

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 without operational transforms or central locking.
- Compiled rendering to **WebAssembly** achieving 60 FPS with 10K+ vector objects; architected horizontal scaling via Redis Pub/Sub with session affinity.