James Kendrick

(210) 429-6569 | kendrickj5@yahoo.com | GitHub | LinkedIn | Portfolio

EDUCATION

University of Illinois Urbana-Champaign Graduation Date: May 2027

B.S. Mathematics and Computer Science Seeking Fall 2026 Internships/Co-ops

- **GPA:** 3.87/4.00
- **Relevant Courses:** Data Structures, Algorithms, Database Systems, Computer Systems, Programming Languages

EXPERIENCE

Meta | Data Engineer Intern *(Incoming Summer 2026)* Seattle, WA

Meta New York City, NY

Data Engineer Intern May – August 2025

- Architected a configuration-driven Python ETL framework that horizontally scaled dimensional modeling across the Instagram Graph domain, enabling Data Scientists to define arbitrary event chains and deploy multi-terabyte pipelines in minutes.
- Optimized the framework's procedural SQL translator to execute a disk-backed, Breadth-First Search (BFS) data cube lattice, bypassing Presto out-of-memory limitations and slashing peak memory utilization by ~80% across all generated pipelines.
- Designed and deployed large-scale data pipelines in Python, powering interactive dashboards that analyze and monitor the user experience of discovering and connecting with friends on Instagram, partnering with PMs, DSs, and DEs.
- Improved private follow surface attribution accuracy by 4% in existing pipelines and overall surface attribution by 20% in new ones, impacting 30+ downstream metrics consumed by 100+ users and enabling more reliable product decision-making.
- Orchestrated complex dependency graphs within Dataswarm (Airflow), enforcing strict SLAs for 30+ downstream systems consumed by 100+ internal stakeholders.

PROJECTS | View All

Grimoire | Website | GitHub | Demo April 2026 – Present

- Designed a language-agnostic execution framework in Go that translates uninstrumented functions into fully typed CLI commands at runtime, dynamically generating Cobra subcommands from YAML configs; built around a pluggable runtime-adapter interface (Python and Go shipped, extensible to any language) to enable a zero-boilerplate plugin system.
- Implemented AST-based signature extraction with tree-sitter to derive typed argument schemas directly from source.
- Built an abstraction that auto-provisions and caches execution environments keyed by SHA-256 hash of dependency and source file manifests for freshness tracking – cutting warm-invocation overhead from ~4+ seconds to ~50ms (>80x speedup).

Sprite | Website | GitHub | Demo January 2025 – Present

- Architected a high-performance C++ CLI tool for Zsh that augments directory navigation, utilizing an embedded SQLite engine to achieve sub-millisecond latency for path resolution and command execution.
- Engineered a context-aware ranking algorithm that optimizes autocompletion suggestions based on historical usage patterns.
- Designed a "quick-nav" execution engine that wraps standard shell binaries, enabling users to execute commands on fuzzy-matched paths instantly, reducing keystrokes by ~40% during deep directory traversal.

Court Vision | Website | GitHub | Demo January 2024 – Present

- Architected a modular Python/FastAPI ETL data platform that ingests daily NBA stats and live box scores from multiple external APIs, writing to PostgreSQL with idempotent upserts, retry/circuit-breaker resilience, and per-pipeline audit trails.
- Engineered a proprietary Genetic Algorithm in Go to solve an NP-Hard Lineup Scheduling problem, utilizing concurrent population evolution to identify optimal streaming strategies across a search space of ~10^9 combinations.
- Implemented a custom constraint-satisfaction engine using backtracking to enforce complex domain rules (positional eligibility, 3-day waiver lockouts, acquisition limits), consistently outperforming greedy approaches by ~15% in projected fantasy points.
- Built a Go cron-runner service supporting multiple orchestration modes to coordinate nightly batch pipelines and live game stats.
- Embedded **SQLMate** (another personal project of mine; Website | GitHub | Demo) as a visual, no-code query interface for the public REST API, enabling non-technical users to explore the structured fantasy basketball dataset without writing SQL.

TECHNICAL SKILLS

Programming Languages: Python, Go, SQL, C++, JavaScript/TypeScript

Infrastructure and Tools: Docker, Git, Linux, AWS/GCP/Azure, CI/CD, Postgres, Agentic Coding Tools (e.g. Claude Code, Cursor)