Benjamin Aepli

+1 (307) 413-7191 | Wilson, WY | benaepli@princeton.edu | github.com/benaepli | linkedin.com/in/benaepli

Education

Princeton University Princeton, NJ

A.B. in Computer Science, Minors in Mathematics and Quantitative Economics *Sep 2024 — May 2028*

- **Cumulative GPA:** 3.96.
- **Awards:** Shapiro Prize for Academic Excellence (2025, top 3% of class), COSCON 2nd Place (2024, Princeton CS competition)
- **CS Coursework:** Distributed Systems (A+), Algorithms & Data Structures (A+), Computer Networks (A+), Operating Systems
- **Activities:** Cryptography Club, COS 226 (Algorithms) UCA, TigerApps, Climbing Team

Experience

Incoming Software Engineering Intern May 2027 — Aug 2027

Jane Street *New York, NY*

Software Engineering Intern May 2026 — Aug 2026

Google *Sunnyvale, CA*

- Extended Knowledge Catalog's query generation and result aggregation logic for **nested BigQuery fields**, supporting quality rules and statistical profiling across the **full stack (Go, Java, Angular)** to improve agentic context quality and traditional governance
- Implemented **hierarchical filtering and validation logic** to support incremental scans and filtering over nested columns, handling partial tree reconstruction and appropriate semantics at negligible overhead relative to query time
- **Authored 3 approved design documents**, added support for profiling of time-typed fields, and benchmarked/documented future pruning approaches to JSON schema discovery across large, nested datasets (1TB+)

Software Engineering Intern (STEP) May 2025 — Aug 2025

Google *Mountain View, CA*

- Implemented LRO infrastructure and use cases on Orion WiFi in Go, launching a new, **supplier-facing bulk update feature** over Spanner with an acyclic dependency graph, scaling to **100k+ updates/RPC**
- Engineered an **LRO milestone tracking system** across parallel tasks using an asynchronous RPC service to provide progress to network suppliers, ensuring sub-second accuracy without compromising scalability
- Co-authored a **27-page design document**, a **9-page developer guide**, and delivered a **technical deep-dive and live demo** on architecture and APIs to ensure knowledge transfer and guide development

Skills

- **Languages:** C++ (20/23), Go, Rust, SQL, TypeScript, OCaml, Java, HTML/CSS
- **Technologies:** Distributed Systems, Concurrency, Databases (Spanner, BigQuery, Postgres), WASM, Vulkan, Linux/Unix, CMake
- **Personal Interests:** Aviation (FAA-certified Private Pilot), Outdoor Climbing

Projects

Spur: Distributed Systems Simulator and Language | *Rust, OCaml, Go* Oct 2025 — Present

- In research advised by Prof. Wyatt Lloyd & Zack Kincaid, designed a DSL in Rust to model distributed protocols, implementing a compiler and a parallelized simulator capable of 5,000+ executions/second to identify consistency violations
- Specified a collection of consensus protocols (Viewstamped Replication, EPaxos, Raft, SDPaxos, etc), discovered a novel bug in Mencius, and am currently working on tuning heuristics to improve bug-finding at longer trace depth

alpmap: An Optimized HashMap Implementation | *C++ 23, GBench* Dec 2025 — Jan 2026

- Re-implemented Google's SwissTable in C++ 23 (with modules) to explore and apply cache-locality optimizations
- Utilized and benchmarked three backends — native SSE, C++ 26's std::simd, and EVE — to parallelize metadata accesses
- Achieved comparable performance to abseil, and a **17x speedup in iteration performance** compared to the standard library

alpbook: A Fast NASDAQ Order Book and Trading System | *C++ 23, Rust* Jan 2026 — Present

- Engineered a **zero-allocation** hot path order book using flat index-based object pools and intrusive linked lists
- Implemented an ITCH event loop harness (**alpdaq**) with robust state machine handling for gap recovery and trading halts
- Achieved **110ns median / 701ns P99** userspace latency by utilizing HugePages-backed pools and thread pinning
- Designed a C++ Concepts system source interface supporting **kernel-bypassed `AF_XDP` network ingestion** and historical replay, validated via a **custom Rust simulator** (`avalanche`) handling GLIMPSE snapshots and ITCH recovery