KAITLYN BASSFORD

New York, NY — 240-270-4606 — kaitlynbassford@gmail.com
LinkedIn — GitHub — Website — Technical Portfolio

EDUCATION

Columbia University, *The Fu Foundation School of Engineering and Applied Science* **New York, NY**
Bachelor of Science in Computer Science *Expected Graduation: May 2028*
Relevant Coursework: Artificial Intelligence, Applied Machine Learning, Fundamentals of Computer Systems, Natural Language Processing, Designing LLM Agents, C++ Deep Dive for C Programmers, Advanced Programming (C/Unix)

EXPERIENCE

AeroVironment **Germantown, MD**
Software Engineering Intern *Jun 2026 – Sep 2026*

- Diagnosed frontend memory leaks that crashed the browser under multi-site monitoring, then eliminated them by unmounting hidden components, memoizing distance caches, consolidating 100+ polling intervals, and fixing a 119,000-listener leak.
- Cut local dev server refresh times for the whole team from 10+ seconds to 0.6 seconds on Linux/WSL and from 60+ seconds to 7-8 seconds on Windows using Vite file warming and modular lazy loading, and upgraded Vite and Node.js.
- Built a Bash network diagnostics daemon that runs a four-stage connectivity, DNS, and HTTP pipeline against deployed customer sites to separate site-specific outages from broader failures, deployed to production with Ansible and Systemd.
- Remediated multiple CVEs, including a Log4j TLS hostname-verification flaw and a PMD cross-site scripting flaw.
- Integrated KICS IaC SAST scanning, Checkstyle, and PMD into GitLab CI/CD, and built Python tooling that tracks each Checkstyle and PMD finding as new, fixed, or pre-existing directly in merge requests, without relying on Jenkins.
- Built structured audit logging for the Java Spring REST server, routing Logback output through Vector pipelines into VictoriaLogs, and migrated a large Python service to structured logging with environment-controlled log levels.

PROJECTS

nibble – LLM Inference Engine | *C++20, AVX2, CMake* **Fall 2026**

- Built a from-scratch LLM inference engine in C++20 that runs Qwen2.5, Llama 3.2, and TinyLlama GGUF models on the CPU in F16 and seven quantized formats, with its own tokenizers, forward pass, and multi-turn chat and no dependencies beyond the standard library.
- Verified correctness against llama.cpp with 50 end-to-end tests: token IDs match exactly, F16 logits reach at least 0.99996 cosine similarity, and quantized dot products in all seven formats match its reference bit for bit, pinned by golden vectors.
- Optimized it with quantized AVX2 and AVX-VNNI kernels kept bit-identical to the scalar reference and picked at startup by runtime CPU dispatch, a spin-then-futex thread pool that cut dispatch latency 81–171×, and a tiled prefill GEMM, bringing Q4_0 decode within 1.19× of llama.cpp at 8 threads in paired A/B/B/A runs.
- Hardened the GGUF loader with bounds checks, overflow-checked arithmetic, and libFuzzer; 304 tests (94.8% line coverage) run in CI with ASan, UBSan, and TSan. Built it spec-first with Claude Code from design specs and test-first plans.

Search Agents | **Live Demo** | *Python, PyTorch, NumPy, FastAPI, Docker, pytest* **Fall 2026**

- Built 26 games and 9 interactive algorithm labs served through FastAPI (WebSocket streaming of live search), deployed on a shared 1 GB machine PyTorch-free by exporting models to NumPy weights, with 2,200+ pytest tests in CI.
- Trained a neural 15-puzzle heuristic that solves 40/40 puzzles with 120× fewer node expansions than optimal IDA*, and AlphaZero-style Connect Four self-play that scores 77% against a depth-4 alpha-beta opponent, trained on a laptop CPU.

Intraday Trading Engine | **Site** | *C++20, Python, PyTorch* **Fall 2026**

- Built a C++20 market-data and decision engine (pybind11, simdjson) that replays a 5.1M-quote NVDA session 86× faster than Python (1.3 s vs. 111.5 s), bit-identical to the Python reference, with 12.7 μs p50 tick-to-decision.
- Trained 108 ML day-trader variants (LightGBM, GRU, transformer, CNN) on 67M minute bars with FinBERT news sentiment, judged against buy-and-hold on unseen years with bootstrap intervals; none beat holding, so none was deployed.
- Deployed multi-stock Alpaca paper trading with risk guards and broker reconciliation, fixing bugs that live sessions surfaced.

Candlebench | **Live Demo** | *Python, C++, NumPy, pandas* **Fall 2026**

- Published a PyPI backtesting framework that tests 66 candlestick patterns on 1M+ trades against matched random entries, charging each trade its quoted NBBO spread, with a CLI that holds any user-written signal to the same test.
- Ported the trade simulator to a pybind11 C++ extension that runs 4–5× faster and stays bit-identical to Python, with every CI-built Linux, macOS, and Windows wheel reproducing a recorded fixture before release; nearly 1,500 tests in CI.

Multimodel Face Analyzer | **Live Demo** | *Python, PyTorch, ONNX, OpenCV, Streamlit, Docker* **Fall 2026**

- Built a computer-vision app with 15 face-analysis features and up to 4 models per attribute running side by side, as an installable package with 380+ tests in CI.
- Evaluated on 1,002 held-out faces with bootstrap intervals, replacing race fusion with best-model selection (63% to 77%), and fixed the public demo's misgendered faces with landmark alignment (88% to 92% gender accuracy).

SKILLS

Languages: Python, C++, C, Java, TypeScript, JavaScript, Bash, CSS, HTML
Frameworks & Libraries: PyTorch, TensorFlow, React, Node.js, Vite, NumPy, pandas
DevOps & Infrastructure: Unix/Linux, Git, GitLab CI/CD, GitHub Actions, Docker, Ansible, Systemd, CMake