Om Patel

Cambridge, MA | ompatel@mit.edu | linkedin.com/in/ompatel-oth | github.com/patelom864 | (270) 980-0315

Education

Massachusetts Institute of Technology **Expected May 2028**

Bachelor of Science, Computer Science and Engineering, Finance | GPA: 5.0/5.0 *Cambridge, MA*

- *Relevant Coursework:* 6.1220 Design & Analysis of Algorithms, 6.1810 Operating System Engineering, 6.1910 Computation Structures, 6.1800 Computer Systems Engineering, 6.1020 Software Construction, 6.1904 Programming in C & Assembly, 6.3700 Probability, 18.600 Probability & Random Variables, 18.650 Statistics, 18.06 Linear Algebra, 18.615 Stochastic Processes
- *Lab Assistant, 6.1910 Computation Structures:* Instruct students building pipelined RISC-V processors and cache hierarchies, debugging RTL and timing issues in weekly lab sessions.

Work Experience

Software Engineer (Part-Time) **Apr 2025 – Present**

Code Four *San Francisco, CA / Remote*

- Cut GPU video redaction latency 9x, from 47 to 5 minutes per video, by ROI-scoping model inference to detected regions, offloading encode to NVENC hardware, and tuning worker concurrency, also lowering per-video GPU compute cost.
- Built core search and redaction workflows for Insight, a CJIS-compliant investigations platform indexing multimodal evidence across phone extractions, bodycam video, and jail audio, merging every change through peer code review.
- Hardened model-driven redaction of faces, license plates, voices, and PII by rewriting long-video object tracking and failure-recovery logic, reducing dropped-track defects in production evidence review.
- Shipped prioritized search, redaction, and review improvements sourced from operational feedback across 15+ law enforcement agencies, building supporting full-stack workflows in React, Next.js, and Node.js.

Software Engineer Intern **May 2026 – Aug 2026**

Dell Technologies *Hopkinton, MA*

- Reduced root-cause investigation latency 2x in concurrency testing by replacing sequential agent calls with distributed parallel fan-out/fan-in execution and idempotent state reducers for concurrent-write safety.
- Built a full-stack root-cause analysis platform over seven specialist retrieval agents spanning Jira, logs, source, Confluence, and knowledge-base search, converting defect keys into evidence-cited reports with confidence-scored fixes.
- Cut repeat-query cost via ChromaDB-backed retrieval with semantic answer caching and feedback-weighted case memory, unifying search across issue, investigation, code, and documentation stores.
- Validated end-to-end accuracy against ground-truth root causes on production defects, exposing the system through secure Flask APIs and Socket.IO streaming for read-only SSH log analysis.

Projects

Low-Latency Order Book & ITCH Feed Handler | *C++20, CMake, NASDAQ ITCH 5.0* **Sept 2026 – Present**

- Building a NASDAQ ITCH 5.0 binary parser and limit order book with price-time priority, replaying full trading sessions with book invariants asserted on every update.
- Verifying the parser with a GoogleTest suite running under AddressSanitizer in CI, cross-validated against an independently written Python implementation, framing a full session byte-exact to its final record.
- Validating book state against the exchange's own execution messages as ground truth, with per-message update latency benchmarked across container designs.

RISC-V Processor with Custom SIMD Extension | *Minispec, RTL Synthesis & Timing Analysis* **Dec 2025**

- Built a 4-stage pipelined RISC-V core with two-way set-associative I/D caches, hazard handling, and branch annulment, synthesizing to a 517 ps clock and running MNIST inference in 239 microseconds to meet the top scoring band.
- Designed a packed 8-bit SIMD multiply instruction for quantized inference, pipelining partial-product reduction across Execute and Writeback to keep the adder tree off the critical path.
- Cut critical-path delay 9%, from 645 to 584 ps, by tracing synthesis timing reports to an ALU control dependency and moving immediate substitution into Decode.
- Instrumented stall counters to attribute 30% of cycles to data-cache misses, then prototyped a prefetch instruction that eliminated 99% of load-wait stalls. Measured request-port contention erasing the gain and rejected the optimization.

Quantitative Trading Strategy Simulation | *Python, Pandas, NumPy, Monte Carlo Simulation* **May 2024**

- Built a 12–1 cross-sectional momentum strategy on a static S&P 500 universe over Jan 2013 – Jan 2023, rebalanced monthly, measuring 14.8% annualized return, 1.28 Sharpe, and -13.6% maximum drawdown net of 10 bps per-side transaction costs.
- Stress-tested robustness with 10,000 block-bootstrap Monte Carlo paths over 3-month blocks to preserve serial dependence, recomputing Sharpe, drawdown, and a 1.71 Sortino ratio across resampled return orderings.

Technical Skills

Languages: C++, Python, C, Java, SQL, TypeScript, Minispec
Systems: Data structures & algorithms, Object-oriented design, Concurrency & multithreading, Distributed execution, Low-latency systems, Pipelining & hazard resolution, Cache hierarchies, RTL synthesis & timing analysis, Performance profiling
Quantitative: Probability, Statistics, Stochastic Processes, Linear Algebra, Monte Carlo Simulation, Backtesting
Tools: Git, Linux, CMake, GoogleTest, PyTorch, NumPy, Pandas