Aaryan V Bhalla

+1 (942) 444 1412 | avbhalla@uwaterloo.ca | bhalla.info | github.com/aaryanved | in/aaryanved

Summary

Second-year Computer Science student at the University of Waterloo with hands-on production experience building AI and backend systems. Currently a Software Engineer at Xsolla working on payments infrastructure, after an AI engineering term at Decimal Technologies building LLM agent pipelines and RAG systems in Python. National hackathon winner, Anthropic-certified, and actively involved in the UWaterloo engineering and residence communities. Strongest in AI and ML engineering, distributed systems, and full-stack development.

Education

University of Waterloo Expected Graduation: Apr 2030
Bachelor of Computer Science (Honours, Co-Op) – President's Scholarship of Distinction *Waterloo, ON, CA*

- **Relevant Coursework:** CS135 (Designing Functional Programs) | CS136 (Elementary Algorithm Design and Data Abstraction)
- **Activities:** Member, Computer Science Club | Chassis Design Team, UW Formula Electric | Undergraduate Residence Advisor, Campus Housing

Scottish High International School Apr 2011 – May 2025
ISC - Computer Science, Physics, Chemistry, Mathematics, English, Hindi, Spanish *Delhi, India*

Skills

Languages: Python, C++, C, Go, JavaScript, TypeScript, Java, C#, Bash
AI & ML: LangGraph, LangChain, OpenAI API, Anthropic API, Google Gemini, Vertex AI, HuggingFace Transformers, TensorFlow, PyTorch, scikit-learn, reinforcement learning (PPO), MuJoCo, MediaPipe, ONNX, FAISS, RAG pipelines, pgvector, prompt engineering
Frontend: React, Angular, TypeScript, JavaScript, Next.js, Three.js, Tailwind CSS, HTML, CSS, face-api.js, Web Audio API
Backend & APIs: FastAPI, Flask, Node.js, Fastify, REST APIs, SSE, asyncio
Applied Cryptography & Security: RSA-OAEP/PSS, AES-256-GCM, WebCrypto/SubtleCrypto, JWT (HS256), hybrid encryption protocol design, forward secrecy
Compilers & Systems: C++, LLVM, CMake, lexing/parsing, static analysis, bytecode interpreters, GoogleTest
Databases: PostgreSQL, SQLite, Redis
DevOps & Cloud: Docker, AWS, Vercel, Git, GitHub Actions, Jenkins, CI/CD, Linux, macOS, Windows
AI Dev Tools: Claude Code, Cursor, Codex, Antigravity
Data & Analysis: Pandas, NumPy, data pipelines, ETL, vector indexing
Languages Spoken: English (Native), Hindi (Native), Spanish (Working)

Certifications

Claude Code 101 | Anthropic *2026*
Model Context Protocol: Advanced Topics | Anthropic *2026*

Experience

Software Engineer Sep 2026 – Present
Xsolla *Los Angeles, CA, USA*

- Built and shipped an internal Systems Registry dashboard in React 19, TypeScript and Neo4j through about 20 production releases, tracking stewards, builders, operators, risk scores and outstanding actions for 50+ company systems
- Ran a column-by-column data-accuracy audit of the dashboard against its graph sources, finding 11 issues; fixed the 4 app defects and routed the data gaps to their owners
- Designed a rule-based red/yellow/green risk score from Slack, meeting and weekly-update activity; removing a false-positive signal took the dashboard from 48 of 51 programs green to an accurate 47 green / 3 yellow / 1 red split
- Found missing Slack thread replies in the company knowledge graph and drove a pipeline fix that refreshes reply counts, now running in production
- Built OneBox Viewer, a sandbox and recipe-catalog app with a Cloud Run back end that hides secrets from every response, tested with Playwright against recorded real data
- Designed the architecture for Prototype Overlays, a 6-phase system of chained sandboxes, and wrote a Go control-plane feature that saves sandbox runs to an experiments repository for review
- Shipped every change through a gated pipeline (preview, stakeholder approval, pull request, CI/CD, production check and rollback), building alongside AI coding agents while owning the specs and review

Artificial Intelligence Engineer May 2026 – Aug 2026
Decimal Technologies *Delhi, India*

- Design and build AI-powered automation tools in Python - developing LLM agent pipelines using LangGraph and LangChain that replace manual multi-step workflows with production-grade automations with measurable efficiency gains

- Build and maintain RAG systems end-to-end: data ingestion, embedding generation, pgvector indexing, and retrieval-augmented generation grounded in live knowledge bases integrated with PostgreSQL
- Develop and optimize data pipelines and SQL queries, transforming structured and unstructured operational data into clean, queryable formats supporting data-informed decision making
- Collaborate cross-functionally with developers and stakeholders to identify automation opportunities, translate requirements into technical solutions, and communicate findings clearly to both technical and non-technical audiences

Undergraduate Residence Advisor Sep 2025 – Apr 2026
University of Waterloo *Waterloo, ON, CA*

- Supported a residential community of undergraduate students as a first point of contact for academic, personal, and community concerns - developing strong skills in conflict resolution, crisis response, and peer mentorship
- Organized and facilitated community programming and events, collaborating with housing administration and fellow advisors to maintain a safe and inclusive residence environment

Founder and Executive Director Apr 2021 – Apr 2025
SHISTECH *Delhi, India*

- Co-founded and, after the founding Executive Director exited early, took over as sole Founder and Executive Director of SHISTECH, a Delhi-NCR high school technology symposium launched online during COVID and later run in-person - grew from 1,100+ to 6,000+ annual participants across 40+ schools in India and Oceania over four years, running 10+ event categories spanning competitive coding, robotics, hackathons, esports, graphic design, and music production
- Led a 70-member organizing team across four domains (IT, Social Media, Design, Marketing) and structured a zero-participant-fee model funded by sponsorships from GitHub, Replit, Taskade, .xyz, Zebronics, Stellar, Gradusure, and Athena totaling CAD $490,000+ over three years, growing cash prize pools from CAD $490 to CAD $1,600; marketing efforts drove 100,000+ online views in a single week at points
- Built and maintained the entire technical stack as sole engineer - full-stack systems in Node.js and JavaScript, registration and participant databases, and relational database design supporting the event's scale
- When a from-scratch gaming lab suffered widespread hardware failures two days before the event's first offline esports tournament, secured approval to work six hours past facility closing and led the team overnight diagnosing and rebuilding machines so the tournament ran on schedule the next morning

Web Development Intern Jun 2023
Stellar Information Technology Pvt. Ltd. *Delhi, India*

- Developed and maintained web application features in a professional software environment, gaining early exposure to production development workflows and client-facing web delivery

ICT Infrastructure and Community Technology Contributor 2023
Bagiya NGO *Delhi, India*

- Designed and built an E-Library and established ICT Labs across two NGO branches, making educational technology accessible to underserved children
- Conducted stakeholder discovery with community members and leadership before specifying any technology, delivering tools that served real community outcomes rather than assumed ones

Projects

Countersign – AI Payments Gatekeeper for One-Person Companies | *TypeScript, Next.js, Convex, Claude API, Cartesia, ECDSA*

- Won the Moss track and placed Top 20 in the Hyphenate track at Cal Hacks 13.0, built with a team of four in about 35 hours: AI agents take payment requests by phone, web chat and email, call the real vendor back on the number already on file, and only a rules engine can release money; I owned the Policy Engine, the case pipeline, the sandbox bank and the scenario answer key
- Achieved 0 false payments across a 39-scenario answer key (26/26 scams blocked, 13/13 legitimate bills paid or escalated) and 28/28 scenarios on a live deployed pipeline by designing an 18-rule deterministic TypeScript Policy Engine as the only path to PAY, with Claude limited to parsing requests and writing replies
- Rejected 4/4 forged, tampered or replayed transfers by building a sandbox bank that moves money only with an ECDSA P-256 countersignature over canonical JSON, single-use nonces and a 2-minute expiry, plus import-guard tests that let only one module touch the signing key
- Fixed pipeline runs that died silently in a live rehearsal by building an idempotent Convex step function with a watchdog (90 s, at most 2 restarts, then fail closed) and race-based LLM timeouts, and wrote the Policy Engine's 53 tests and 27 pipeline integration tests in a 1,126-test CI suite

Deja – Real-Time Free-Trial Abuse Decision Service | *Java 21, PostgreSQL, LightGBM, Python, TypeScript, Next.js*

- A service that catches people farming free trials with multiple accounts: it links accounts that share a device, card, phone or email pattern into identity clusters as they form, scores them with a machine-learning model and rules, and returns allow, challenge, review or block in milliseconds; built solo, ∼34,000 lines across Java, Python and TypeScript with 485 passing tests
- Made every live decision reproducible exactly offline - a 100% match on 100,000 events from 32 concurrent clients, identical on x86 and ARM - by running one pure-function engine against Postgres online and in memory offline, with lock-ordered transactions that commit interacting events in log order
- Cut expected abuse cost per 1,000 sign-ups by 68% against a rules-only baseline in backtests on simulated data (PR-AUC 0.78, 84% recall at 1% false positives) by training a calibrated LightGBM model on identity-graph features that more than doubled PR-AUC over per-event features alone

- Hardened it for production before deployment, verified by 21/21 failure modes passing under fault injection, 90,800 generated API contract tests with 0 failures, and 300+ dependency vulnerabilities cut to 0, by building the checks into CI on every push

ApexTrace – F1 Safety-Warning Simulator | *TypeScript, three.js, Web Workers, Python, PyTorch*

- A Formula 1 driving simulator with a safety system on board that tells the driver when to brake, built by a team of three in 48 hours that won 1st place at Formula Tech Hacks 2026: you drive real-layout Monza and Baku in the browser, while the bench deliberately breaks the telemetry link to test whether the warning still arrives in time
- Built the core simulator and safety logic - 2026-regulation car dynamics (bicycle model, Pacejka-style tyres, active aero, hybrid power unit), the look-ahead BRAKE warning that never trusts stale data, and a seeded fault-injection test runner - leading the team with 62 of 111 commits
- Found 3.5× more safety failures than random testing on the same 50-simulation budget (47.0 vs 13.6) by training a PyTorch TCN that forecasts lap failure from the first 12 s of telemetry (ROC-AUC 0.918) and using it to screen scenarios for a Soft Actor-Critic search
- Took the app to zero infrastructure after the event, deleting 26.9k lines of backend, by porting the Python simulator line for line to TypeScript in a Web Worker and proving it tick-for-tick against recorded Python sessions

Weaver – Conversational Multi-Agent Workflow Builder | *Python, JavaScript, Google ADK, Cloud Run, Firestore, Vertex AI* Aug 2

- Built a conversational multi-agent workflow system solo in 10 days for Google's "All Things Agentic" hackathon (Collaborative Partner track): 15 Google ADK agents across 16 private Cloud Run services, ∼12,250 lines, 87 automated tests, deployed to production and used by real users
- Designed a two-agent architecture separating user interviewing from system design, with a deterministic, model-free validator between them catching dependency cycles, dangling references, and unknown step types before deployment
- Grounded agent capabilities in a live Firestore registry rather than prompt content, so adding video and music generation required a database row and one service each with zero prompt changes
- Diagnosed and fixed a silent-failure class where the orchestrator reported HTTP 200 as success while the underlying action had failed, by parsing raw tool-response objects from the event stream instead of trusting model-generated prose
- Enforced least-privilege security across 16 private services (per-call OIDC tokens audienced to individual service URLs, a 4-role service account, a separate weaker identity for the public proxy) and implemented transactional Firestore scheduling so concurrent instances cannot double-fire an automation, verified by firing 8 concurrent ticks and confirming exactly 1 execution

Marionette – Real-Time Webcam Motion Capture Driving a Simulated Humanoid | *TypeScript, Python, Go, MuJoCo, PyTorch*

- Built a 14,790-line browser app where a MuJoCo-simulated Unitree H1 humanoid copies your upper body from a webcam in real time, entirely on-device: pose estimation, 2D-to-3D lifting, retargeting and IK in a WebWorker, and 500 Hz control and physics on the main thread
- Trained a 1.1M-parameter residual MLP that lifts 2D landmarks to 3D on 1.04M CMU mocap poses, using a randomised virtual camera as augmentation; 74.8 mm MPJPE on 40,000 held-out-subject samples, measured on the exported ONNX model, at 0.3–0.5 ms per inference in the browser
- Wrote closed-form retargeting from human limb axes to H1 joints, verified against MuJoCo forward kinematics to under 10^{-6} on 200 sampled configurations, plus damped least-squares IK that cuts hand position error 28% (83.6 vs 116.7 mm) in 2.3 iterations on average
- Measured that H1 cannot stay upright on planted feet while its arms follow a person (centre of mass 12 mm from the support edge, falls in 3.2 s), then added centre-of-mass feedback inside the 500 Hz physics loop - stable at 500 Hz, unstable at 50 Hz - reaching 83% push recovery on a disclosed support rig that applies 0 N when undisturbed
- Cut knee sag 10× (0.206 to 0.021 rad) with a contact-aware gravity feed-forward, low-pass filtered after finding that the raw term formed a positive feedback loop with ground contact and made the robot fall faster than no feed-forward at all
- Trained a PPO residual balance policy over 37M CPU-only environment steps with domain randomisation; it beat PD alone 3.2× on push recovery but added only 4.4 points over the classical controller against a 5-point bar, so it ships switched off, after fixing clip-fraction, reward-scale and truncation-bootstrap bugs found in the diagnostics
- Held 19 ms p95 camera-to-screen latency and 500 Hz physics even at 8× CPU throttle, cut cold load from 4.34 s to 2.85 s by removing a duplicate 13.6 MB WASM download, and compiled a mesh-free physics model from 56 MB to 51 KB with 0.0 measured dynamics divergence
- Backed it with 110 unit and 9 Playwright end-to-end tests in CI, including an automated check that no request leaves the origin during a live camera session, and cross-language fixtures that caught a gravity-projection bug (matrix column instead of row) before it reached training

Crucible – Compute-in-the-Loop Regression Testing for Autonomous Driving | *C++20, Go, Python, TypeScript, NATS JetStrea*

- Built a regression-testing platform for self-driving planners solo (∼12,000 lines across four languages) that runs each planner as a CPU-quota'd OS process under an enforced millisecond deadline inside a deterministic simulator, then decides with paired statistics whether a code change made driving better or worse - live at crucible.bhalla.info
- Achieved bit-exact reproducibility across four platforms - 0 fingerprint mismatches over 40,000 runs on macOS/arm64, glibc, musl and emulated x86-64 - by writing an IEEE-exact math library, eliminating compiler-dependent argument-evaluation order, and quantising measured planner latency into the virtual clock
- Simulated driving at 501× real time per core (1,904× across 10 cores) with a zero-allocation tick loop enforced by a test that overrides `operator new`, and a POSIX shared-memory SPSC ring at 2μs p99 round trip holding 100 ms deadlines to 1.037 ms p99 overshoot

- Designed the statistics that call a regression: 10,000-sample paired bootstrap over 11 driving metrics with Benjamini-Hochberg false-discovery control and a sequential probability ratio test reaching the same verdict on 84% fewer scenarios, with power and false-alarm rate validated against injected regressions of known size rather than asserted
- Built the distributed execution path in Go - NATS JetStream work queue with leases and redelivery, content-addressed result cache (1,000 scenarios × 2 planner versions in 21.3 s cold, 0.6 s warm), idempotent Postgres writes, S3-compatible trace storage - and ran untrusted planner code under cgroup v2 quotas and a seccomp-locked container with an automated escape-probe test in CI
- Used the platform to find four real planner defects and one map-fidelity defect the metrics had been misattributing, each now covered by a regression test verified to fail when the bug is reintroduced

Verte – AI Sustainability Shopping Assistant | *JavaScript, React, Node.js, Express, Snowflake, OpenAI (GPT-4o-mini, Whisper), Ch*

- Placed 2nd at PivotHacks (12-hour, team of three): a Chrome Manifest V3 extension and Node/Express backend that surfaces secondhand and lower-carbon alternatives on Amazon and Best Buy product pages
- Led the architecture refactor (∼2,000 lines) that consolidated the team's parallel workstreams into the final codebase, built around four agreed data contracts so mid-event pivots changed the data producers without touching the UI
- Built an anti-hallucination verification pipeline: every GPT-4o-mini product suggestion is checked against live Amazon.ca search results via a custom HTML parser and token-based matching that rejects sponsored listings and accessories, with budgets enforced against real prices and a retry loop that tells the model what was unavailable - nothing unverified ever reaches the user
- Built hands-free voice search with OpenAI Whisper and a Web Audio silence detector that calibrates to the room's noise floor and stops recording ∼1.2s after speech ends, skipping the transcription call entirely when nothing was said
- Integrated Snowflake (key-pair JWT auth) as the source of a 24-category buy-used knowledge base with row validation, caching and a built-in fallback table, and wrote a rate-limited Amazon.ca scraper (custom HTTPS client, gzip, concurrency queue, 30-minute cache)

VFC – Virtual Fighting Championship | *TypeScript, React, three.js, MediaPipe, vitest* 2026 – Present

- Engineered a real-time computer-vision pipeline (MediaPipe pose landmarker) processing live webcam video entirely client-side to classify boxing techniques (punch type, speed, extension, rotation) into a scored combat system, holding robust detection across camera frame rates from 15-60fps through empirically-tuned adaptive (One Euro) signal filtering
- Diagnosed and fixed a chain of real accuracy bugs in the vision pipeline through direct before/after measurement - including a filtering bug silently discarding up to 65% of fast-punch velocity and a tracking bug letting occluded/lost landmarks score as high-quality punches - each fix backed by a regression test proving it against the prior, wrong behavior
- Built an adaptive performance governor that measures live model inference cost and paces camera/inference frame rate against actual device capability rather than a fixed assumption, with hysteresis to prevent visible stutter
- Designed a fixed-timestep, intent-based combat engine architected for future multiplayer without ever transmitting webcam video - only lightweight serialized game intents cross the network boundary - plus an adaptive AI opponent and an algorithmic "mirror champion" that profiles and replicates a player's own measured fighting style
- Built a full 3D rendering layer in three.js (procedural inverse-kinematics rigs, a researched arena environment, instanced crowd rendering within a strict draw-call budget) and maintained 400+ automated tests, including a seeded-RNG harness for reproducible AI behavior, enforced via CI on every change

Order Book Matching Engine | *Price-Time Priority, Distributed Systems Design, Zero-Knowledge Settlement* Aug 2026

- Designed a price-time priority matching engine architecture: a sorted structure over distinct price levels for O(log n) best-price lookup, a FIFO queue per price level for O(1) time-priority matching, and an order-ID-to-node hashmap for O(1) cancellation
- Modeled the full order lifecycle - validation, insertion, cross detection, partial and full fill handling, and cancellation - with explicit state transitions for each outcome
- Mapped the design onto a real off-chain sequencer / zero-knowledge prover / on-chain settlement architecture, reasoning through the trade-offs between off-chain matching speed and the trust model a zk-proof and L1 settlement layer remove
- Documented the full system as a technical architecture diagram to communicate design decisions and data structure trade-offs clearly

BagCheck – Computer Vision Cabin-Baggage Sizing System | *Python, PyTorch, YOLO11, OpenCV, FastAPI, SQLite, Open3D* Au

- Built a computer-vision system that measures hand-luggage dimensions from a single camera and flags oversized bags against airline cabin limits - ∼8,300 lines of production Python plus 1,500 lines of tests (134 passing) across a segmentation/pose pipeline, three interchangeable measurement backends, and a FastAPI web app
- Designed three swappable dimension-estimation backends behind one shared interface (depth, reference-object homography, and monocular 3D pose recovery) chosen by camera geometry, so adding a fourth backend requires implementing one abstract class rather than touching the pipeline
- Solved full camera-pose recovery from a single image via homography decomposition (SVD re-orthonormalization, mirror-ambiguity resolution) and fit 3D bounding boxes by maximizing silhouette IoU with a scaled, restarted Powell optimizer - validated to ±3 cm on a synthetic ground-truth harness
- Designed a three-verdict compliance model (compliant / oversized / needs manual check) that propagates measurement uncertainty into every decision, explicitly refusing to fail a bag on a number the system isn't confident about, backed by a SQLite audit log for full traceability

Clueless-ly – Real-Time AI Assistant Overlay | *TypeScript, Electron, React, OpenAI Realtime API, ScreenCaptureKit* 2026

- Built a cross-platform (macOS/Windows) Electron/TypeScript overlay running dual-channel real-time transcription - microphone and system output audio captured separately via ScreenCaptureKit and transcribed over independent OpenAI Realtime WebSocket sessions for speaker-attributed conversation history
- Designed a platform-contract abstraction (permissions, audio, screen, window, secure storage, shortcuts) that confines all OS-specific branching to a single resolver file, keeping macOS and Windows builds on one identical application core
- Solved pixel-level click-through interactivity via manual hit-testing (`elementFromPoint` plus a `data-interactive` marker convention) after React's mouse enter/leave events proved unreliable under forwarded-mouse mode
- Diagnosed a production-class failure where concurrent auto-answers exceeded the provider's tokens-per-minute limit and hung every in-flight request - fixed with a serialized, capped request queue and per-request `AbortController` cancellation that preserves partial streamed output
- Built a priority-ordered, question-aware token budgeter that decides which reference material survives a fixed context ceiling, explicitly labeling anything truncated, and implemented OS-level content protection (`NSWindowSharingNone`) so the overlay is excluded from screen recording, with secrets never logged and keys stored via Electron `safeStorage` backed by the OS keychain

KnightVision – Autonomous Chess Engine & Lichess Bot | *Python, python-chess, Stockfish (UCI), threading, REST/NDJSON strea*

- Built an autonomous chess-playing bot integrating the Lichess Board API and a local Stockfish engine over UCI - streaming live game state via NDJSON long-polling, converting FEN/UCI positions, and submitting moves through a multi-threaded seek/play/log pipeline with dedup locking to prevent race conditions
- Designed a human-latency model layering time-control-aware statistical delay distributions (uniform and right-skewed triangular) on top of engine move selection, so move timing reflects realistic human think-time rather than instant engine responses
- Built a 410-line opening book across 16 named repertoires as a flat UCI-move-history-keyed dictionary for O(1) lookup, with graceful fallback to engine analysis when a position falls outside the book
- Instrumented the system with structured, append-only logs (games, moves, results, API calls, errors) for full post-hoc traceability, and identified and documented a platform-policy compliance gap (Board API vs. the sanctioned Bot API) as part of a full architecture review

Melodex – AI-Powered Spotify Intelligence Layer | *TypeScript, Node.js, Fastify, React, PostgreSQL, OpenAI, Vercel, Supabase* 2026

- Architected a full-stack pnpm monorepo (Fastify backend, React/Vite frontend, shared type-contract package) deployed as a single serverless function on Vercel with Supabase Postgres - designing around hard function timeouts and a shared connection budget via checkpointed resumable jobs, idempotent writes, and fast-fail connection pooling
- Built a resumable Spotify sync engine as an explicit state machine: a persisted checkpoint advances through phases (liked songs, playlists, albums, artists) across multiple function invocations, with per-phase retry budgets, OAuth token lifecycle handling, and Retry-After-aware backoff
- Designed an OpenAI tool-calling agent grounded in real user data through structured tools (deterministic filters, not free-text matching) and a post-generation validator that cross-checks every claim against tool results, with all mutating writes gated behind a preview-confirm-undo workflow built on idempotency keys and recorded inverse operations
- Integrated both custom security SDKs in production - encrypting every sensitive API call end-to-end via the RSA/AES handshake protocol and securing sessions via the JWT/refresh-rotation library - and did substantial live production debugging (mis-mapped errors, silent upstream API schema drift) from logs

MiniLang – Compiler and Static Analyzer | *C++, LLVM, CMake, GoogleTest, ANTLR* Jul 2026

- Built a from-scratch compiler: hand-wrote a lexer and recursive-descent parser producing an AST, a type checker catching mismatches and undefined-variable errors at compile time, and a stack-based bytecode interpreter for execution
- Designed a static analysis pass detecting use-before-initialization, unreachable code, and integer overflow risks, modeled on tools like Clang-Tidy, and extended the project with an experimental LLVM IR backend for native code generation
- Wrote 100+ unit tests with GoogleTest and set up a CMake build with GitHub Actions CI so every commit was automatically tested; documented trade-offs (e.g., recursive-descent vs. table-driven parsing) throughout

Custom End-to-End Encryption & Auth SDKs | *WebCrypto, RSA, AES-256-GCM, JWT, Redis, Postgres* Jun 2026

- Designed and built a versioned hybrid encryption protocol layering application-level encryption on top of TLS, with a byte-identical wire format implemented across TypeScript, Python, and Java so any app in that stack can adopt it: an RSA-4096 OAEP/PSS handshake establishes an ephemeral per-session AES-256-GCM key with RSA-PSS server-signature verification for anti-MITM authentication, giving every request forward secrecy
- Implemented a browser handshake state machine with concurrent-request coalescing and a pluggable session-store strategy (In-Memory/Redis/Postgres) with typed expiry errors
- Built a companion stateless JWT auth library (HS256) with refresh-token rotation - single-use tokens cryptographically bound to their session to prevent replay - and httpOnly/secure/sameSite cookie hardening against XSS; shipped as semver-versioned npm packages with Jest/Vitest test suites

Custom Git-Backed CMS | *TypeScript, Next.js, React, GitHub REST/Git Data API* May 2026 – Present

- Built a headless CMS with no database - all content reads/writes go directly through the GitHub Contents and Git Data APIs, using the repository's git history itself as the content store
- Designed a draft/publish workflow modeled on git branching: edits land on an auto-created draft branch, a diff view lists pending commits with per-commit and bulk revert, and a publish action performs an API merge plus a Vercel deploy-hook rebuild

- Hand-wrote a LaTeX parser from scratch (no external libraries) that recursively resolves brace-nested macros and infers the source document's font package to render pixel-faithful HTML from raw LaTeX source, with append-only version snapshots on every save

Pulse AI | *Python, LangGraph, LangChain, OpenAI API, FastAPI, PostgreSQL, pgvector, Docker, AWS* May 2026

- Designed and built a production AI agent system using LangGraph - structuring multi-step reasoning as task graphs with defined triggers, routing logic, and failure handling, integrated with PostgreSQL and pgvector for structured and semantic retrieval
- Built a full RAG pipeline end-to-end: data ingestion, chunking, embedding generation, pgvector indexing, and retrieval-augmented generation that grounds LLM outputs in structured evidence
- Deployed on AWS with Docker and CI/CD automation, maintaining observability across latency, token usage, and retrieval quality to measure and improve performance in production

Grounded | *TypeScript, JavaScript, Gemini 2.5 Flash, ElevenLabs, face-api.js, Web Audio API* Mar 2026

- Built at Hack Canada 2026 in under 24 hours - a real-time multimodal AI application integrating computer vision (face-api.js), LLM reasoning (Gemini 2.5 Flash), and text-to-speech (ElevenLabs) into a single live pipeline with real-time state management

The Originality Engine | *Python, HuggingFace Transformers, FAISS, Pandas, NumPy* Feb 2025

- Designed and built a large-scale semantic analysis pipeline: data ingestion and preprocessing, embedding generation using open-source HuggingFace transformer models, FAISS high-dimensional vector indexing, and similarity-scored output with systematic benchmarking across multiple model architectures

Disease Detection | *Python, TensorFlow, Flask, scikit-learn, Pandas, Raspberry Pi, Linux* Aug 2023

- Trained a TensorFlow CNN for medical image classification end-to-end - data preprocessing and augmentation, architecture design, hyperparameter tuning, and held-out evaluation - deployed as a Flask REST API on Linux for real-time inference on a Raspberry Pi

Awards & Achievements

President's Scholarship of Distinction - University of Waterloo *2026*

SRCC FIC Fellowship Programme *2025*
Selected for the Fellowship and Incubation Centre programme at Shri Ram College of Commerce, University of Delhi - one of India's most prestigious undergraduate institutions.

National Winner - IIT Delhi Youth Ideathon *2024*
Placed first nationally at the Indian Institute of Technology Delhi's Youth Ideathon, competing against university teams across India in a technology and innovation challenge.

Certification of Distinction - FERMAT Mathematics Olympiad
Awarded a Certification of Distinction in the FERMAT Mathematics Olympiad, recognizing exceptional performance in advanced mathematics.

Hackathons & Competitions

Google "All Things Agentic" Hackathon
Participant, Collaborative Partner track - Built Weaver *Google*

PIVOTHACKS
2nd Place - Built Verte *Waterloo Tech Week*

Formula Tech Hacks
1st Place - Built ApexTrace *2026*

- Won 1st place with an F1 driving simulator with a corner-entry BRAKE warning and an AI test bench that found 3.5× more safety failures than random testing, with a team of three in 48 hours

HackTrack
2-time consecutive winner *Scottish High International School*

ShriTeq Hackathon
3-time participant (1-time winner) *Shri Ram School, Moulsari*

Hack Canada
Participant - Built Grounded *Wilfrid Laurier University*

- Designed and shipped a fully functional multimodal AI application in under 24 hours - integrating computer vision, LLM reasoning, and real-time text-to-speech into a unified live pipeline

Ctrl + Hack + Del 2.0 | CXC Data Science Hackathon
Participant *York University | University of Waterloo*