Jalen Edusei

jalen.edusei@gmail.com | Atlanta, GA | jalenedusei.com | linkedin.com/in/jalenedusei | github.com/jke48222

EDUCATION

University of Georgia, Morehead Honors College, Athens, GA — May 2026
Bachelor of Science in Computer Systems Engineering, cum laude — GPA: 3.61 / 4.00

Relevant Coursework: Data Structures; Systems Programming; Software Development; Discrete Mathematics; Advanced Digital Design; Design of Digital Systems; Embedded Systems I & II; Sensors and Transducers; Linear Systems; Electronics I; Circuit Analysis; Human-Computer Interaction; Virtual Reality; Introduction to Microfabrication; Probability and Statistics for Engineers; Applied Linear Algebra.

TECHNICAL SKILLS

Programming Languages: Swift, C, C++, C#, Python, TypeScript, JavaScript, Java, Elixir, SQL, Verilog, ARM Assembly (ARMv7 / AArch32), MATLAB, R, MicroPython, GLSL, Pine Script, LaTeX, HTML/CSS

Web & Full-Stack: React 19, Next.js (App Router, ISR), React Native / Expo, Node.js, Phoenix / Ecto, Vite, Tailwind CSS, Framer Motion, Zustand, three.js / React Three Fiber, WebGL, WebGPU, Progressive Web Apps, REST and JSON API design, WebSockets / Phoenix Channels / Supabase Realtime, GSAP, TinaCMS, Shopify Storefront API, Resend, WordPress

Databases & Data: PostgreSQL (row-level security, triggers, generated columns, transactional outbox), Supabase, SQLite / FTS5, Redis, BM25 and hybrid vector search, data analysis in SQL, Python, and Excel

Apple Platforms & Systems: AppKit, SwiftUI, Core Animation, macOS Accessibility API (AXUIElement / AXObserver), CGWindowList, ScreenCaptureKit, Core Bluetooth, Speech and AVFoundation, Foundation Models, Keychain, WidgetKit, XCTest, Swift Package Manager, XcodeGen, code signing and DMG packaging, launchd / systemd services

AI & Machine Learning: Anthropic Claude API (Messages API, tool use, SSE streaming, JSON-schema structured outputs, prompt caching, vision), agentic tool-use loops with confirmation gates, MCP servers, Apple on-device Foundation Models, MLX (on-device embeddings and VLM inference, 4-bit quantization), Qwen3-Embedding, Holo1.5 vision grounding, hybrid retrieval (BM25 + binary vector index + reciprocal-rank fusion), evaluation harnesses and ablations, prompt-injection defenses, OpenAI API, Wit.ai NLU, speech recognition and TTS (ElevenLabs, edge-tts)

Embedded & Hardware: STM32, ESP32 (PlatformIO, NimBLE, MQTT, NVS), Raspberry Pi 4 / 5, Raspberry Pi Pico (RP2040), HUB75 LED matrices, BLE GATT (BlueZ, NimBLE, Core Bluetooth), UART / SPI / I2C, hardware PWM (pigpio), ADC, DSP (band-pass filtering, peak detection, Kalman filtering), PID control, sensors (geophones, load cells, thermistors, photodiodes, DHT22, TCS34725 colorimeter, IR tachometers), analog signal conditioning and op-amps, PCB design (KiCad, freerouting), schematic capture (Altium, KiCad), ngspice and Multisim simulation, power distribution and fusing, FPGA design (Verilog, Xilinx, VGA, DDR2, SD / SPI), CubeSat flight software (NASA F Prime, Zephyr), Fusion 360 and SolidWorks CAD, cleanroom microfabrication (photolithography, PDMS soft lithography, plasma bonding)

Games, XR & Graphics: Unity (C#, Meta XR SDK, OpenXR, XR Interaction Toolkit, hand tracking, passthrough, spatial anchors, VelNet networking), Unreal Engine 5 (C++, Lumen, Nanite, StateTree), three.js, WebGPU / TSL, GLSL shaders, procedural generation, physics and rollback netcode, entity-component systems, GOAP and utility AI, Blender, Final Cut Pro, Figma

DevOps & Tooling: Git / GitHub, GitHub Actions CI, Docker and docker-compose, Kubernetes manifests, Vercel, Cloudflare Pages / Workers, Supabase, restic and age encrypted backups, Claude Code and Cursor (AI-assisted development), Tectonic / LaTeX

Testing & Quality: XCTest, Vitest, ExUnit, JUnit, Python unittest, module-level Verilog testbenches, headless end-to-end rigs, benchmark-driven design, energy and latency budgets, WCAG AA accessibility audits

Business & Leadership: Business-case development, KPI and campaign analysis, product strategy, stakeholder management, technical writing, mentoring, project and event management

PROFESSIONAL EXPERIENCE

Capital One, *Business Analyst Intern,* McLean, VA — Jun 2025 – Aug 2025

- Built the business case for a Notifications Preferences Center on the CreditWise team, projected to centralize communication management for 60M+ customer accounts, and presented the recommendation to senior leadership.
- Analyzed email-campaign performance in SQL, Python, and Excel, creating a valuation framework that quantified engagement and retention impact and supported a proposal for a dedicated sending domain.
- Partnered with product, design, and engineering stakeholders to turn KPI findings into messaging-strategy and roadmap decisions.

FREELANCE CLIENT WORK

KUL Enterprises: Freight Carrier Marketing Website Jun 2026 – Present

Next.js 16 (App Router), React 19, TypeScript, Tailwind CSS v4, Framer Motion, TinaCMS, Resend, Vercel kulenterprises.com | Private repository

- Designed, built, and shipped kulenterprises.com for a Georgia freight carrier as sole designer and engineer: iterated 12 style directions as 20 built variants with the client, then delivered 22 statically generated pages (services, driver recruiting, safety, quoting, legal) in Next.js 16, React 19, TypeScript, and Tailwind CSS v4 on an 83-token design system.
- Architected a git-backed TinaCMS content model (17 typed collections) so no readable text ships hardcoded: business facts flow through a token system into centralized Copy and Prose renderers, and a custom build wrapper keeps a failing CMS build from ever taking the site down.
- Produced the opening brand film (Blender, then Higgsfield clips cut in Final Cut Pro) and shipped it as H.264 with a 720p/1080p ladder and a portrait mobile cut after documenting an iOS Safari VP9 failure, alongside a fullscreen documentary player, a dashcam hero loop, an Albers-projected interactive service-area map, and pure-CSS scroll reveals that honor `prefers-reduced-motion`.
- Built a dependency-free site search that scores CMS content directly (prefix, one-edit-typo, and run-together matching with a freight-term synonym sheet) and four rate-limited, honeypot-protected forms that re-validate server-side and deliver through Resend with a structured-log fallback, then brought the interface to WCAG AA with per-token measured contrast (ink 12.25:1, gold 4.57:1).

KUL Enterprises: Broker Credit-Vetting Operations Portal Aug 2026 – Present

Next.js 16 (App Router), React 19, TypeScript, Tailwind CSS v4, Supabase (Postgres, Auth), Vercel Private repository

- Shipped a role-based internal portal for broker credit vetting for the same carrier: a broker master database that refuses duplicate MC numbers, the client's 12-question screening checklist with FMCSA and SAFER deep links, a green/yellow/red risk rating whose red status sets a do-not-load block only an administrator can clear with a recorded reason, and per-broker credit-exposure checks with recorded approvals, behind invite-only authentication with four staff roles.
- Pushed the compliance rules into Postgres: row-level security by role, an audit log the application only reads, screenings that snapshot their question wording at submission, an administrator-only guard on risk and credit-limit changes whose messages surface verbatim in the UI, and atomic SQL functions for status changes and checklist submission.
- Built a sentence-case, sidebar design system benchmarked against Mercury, Stripe, and Cal.com, with semantic risk color, tabular numerals, and reduced-motion-aware skeleton states, documented in a committed design contract; middleware validates sessions with `getUser()` so a revoked account is out on the next request.

Akilah Mali: Musical Artist Website May 2026 – Present

Next.js 16 (App Router), React 19, TypeScript, Tailwind CSS v4, three.js, React Three Fiber, GSAP, Web Audio API, Vercel akilahmali.com | github.com/jke48222/akilahmali

- Built and shipped akilahmali.com for a recording artist: Next.js 16, React 19, TypeScript, and Tailwind CSS v4 with a real-time 3D "control room" scene (three.js, React Three Fiber, GLB models, CCTV-style feeds), a rotary-dial payphone route with Web Audio dialing, and embed-served music and tour content that leaves the client no backend to operate.
- Cut a roughly 46 MB payload by code-splitting three.js off the release routes and deferring the WebGL mount to idle behind an instant paper cover, kept reduced-motion and `noscript` fallbacks throughout, and shipped HSTS-preload and permissions-policy security headers with a Laylo-backed email-capture route.
- Built, then deliberately removed, a headless Shopify Storefront API cart (httpOnly cookie, server-side mutation boundary) after a security and architecture review, moving commerce to Shopify's hosted storefront in a 2,700-line deletion.

RESEARCH EXPERIENCE

UGA Small Satellite Research Lab, *Flight Software Developer (MEMESat-1),* Athens, GA Mar 2024 – Dec 2024

- Developed and debugged embedded flight software in C/C++ on STM32 microcontrollers for the MEMESat-1 CubeSat, bringing up and troubleshooting UART communication between onboard subsystems.
- Wrote unit and integration tests that hardened the flight software against edge cases and failure conditions, and reviewed teammates' code in the shared codebase.

Joyner Research Laboratory, UGA, *Research Assistant,* Athens, GA Sep 2022 – May 2023

- Ran 50+ wet-lab experiments (ELISAs, DNA/RNA extractions) supporting malaria bioinformatics projects, documenting data that contributed to two peer-reviewed manuscripts, and built the laboratory's website, which grew to 500+ monthly visitors.

TECHNICAL & ENGINEERING PROJECTS

Systems, macOS & AI Engineering

Otto: AI Assistant in the MacBook Notch Sep 2026 – Present

Swift, SwiftUI, AppKit, Anthropic Messages API (SSE, server tools), Carbon, XCTest github.com/jke48222/otto | otto-sandy.vercel.app

- Built Otto, a native macOS assistant that turns the MacBook notch into a Claude client (16,400+ lines of Swift, 212 unit tests, zero dependencies): resting the pointer on the notch or pressing Option-Space opens a borderless, non-activating `NSPanel` under the camera that takes files, images, PDFs, a screen region, or the front browser tab and streams the reply in place.
- Wrote the Anthropic client as raw HTTPS and server-sent events on `URLSession` with its own line splitter, since `AsyncBytes.lines` also breaks on U+2028 and U+2029 inside JSON strings; a stream accumulator folds text, thinking, web search and fetch activity, and citations, returns thinking signatures unchanged, and encodes requests with sorted keys so the prefix stays byte-stable for prompt caching.
- Modeled hover, click, and drag as a pure state machine with 24 unit tests, so the notch opens only after the pointer rests for 90 ms and not when it passes on the way to a menu; the panel toggles `ignoresMouseEvents` to take clicks only over its drawn shape, and a `--selftest` mode drives the real notch through a scripted session.
- Published the source under the MIT license with CI running the suite on every push to `main`, a product site on Vercel, and a promo film recorded from the real app window; the API key stays in the Keychain with no account or telemetry, the Carbon hot key needs no Accessibility permission, and a signed, notarized build is coming soon.

Exocortex: Local-First Personal Memory System Aug 2026 – Present

Swift, SQLite/FTS5, MLX, Qwen3-Embedding, Apple Foundation Models, Python, MCP, restic, age github.com/jke48222/exocortex

- Built a local-first personal memory system in Swift (8,400+ lines) that ingests 14 life-log streams (iMessage, Chrome and Safari history, Gmail, IMAP, Claude Code transcripts, and iPhone-backup calls, messages, calendars, contacts, and notes) into a 100,000-event SQLite/FTS5 store with content-hash deduplication, secret redaction, trust-class retention, and encrypted, restore-verified backups.
- Recovered 31,000+ iMessages with a parser for Apple's 2026 typedstream `attributedBody` format, and engineered hybrid retrieval fusing BM25 with a binary vector index over on-device MLX embeddings and reciprocal-rank fusion, measuring 0.95 Recall@1 against a 0.55 BM25 baseline after retracting an earlier result whose ground truth came from the system under test.
- Designed a bitemporal belief ledger and a `dream` cognition pipeline (contradiction detection, episode segmentation, spaced-repetition decay, commitment tracking, contact dossiers, a weekly narrative) driven by on-device Foundation Models with every model output citation-verified in code; measured 1.00 precision and 0.67 recall on commitment detection, with 105/105 regression checks across nine suites.
- Froze an MCP memory-bus contract (v1.0.0) exposing nine read-tier tools with four server-assigned trust tiers, per-client purpose allowlists, no mass-read primitive, egress sanitization against link-based exfiltration, canary rows, and a hash-chained audit log, verifying 8/8 security invariants against the live store; ran two measured feasibility spikes first, including a calendar harness that clocked Apple Calendar's median poll at 5 minutes.

WindowPet: Desktop Creature & Agentic Assistant for macOS Aug 2026 – Present

Swift, AppKit, Core Animation, macOS Accessibility API, Anthropic Messages API, Apple Foundation Models, XCTest github.com/jke48222/WindowPet

- Built a native Swift/AppKit desktop creature (11,000+ lines, 131 unit tests, zero dependencies) that rides real application windows, fusing permission-free `CGWindowList` geometry polled adaptively (60 Hz in motion, stepping down to 10 and 4 Hz at rest) with a read-free `AXObserver` event stream; passive tracking never reads window titles, so it never triggers a Screen Recording prompt.
- Wrote the physics and behavior as pure, unit-tested code: occluded window edges become platformer terrain with a ballistic leap solver tested to 1.5 pt, per-frame alpha masks open a click-through hole only over creature pixels for boop, grab, and throw, and a softmax-utility engine with a GOAP planner picks actions; a self-instrumenting benchmark enforces energy budgets, measuring 0.24% CPU asleep and 1.04% in motion at 48 MB.
- Developed a three-tier assistant behind a hotkey panel, wake word, and push-to-talk voice: an exact command grammar, an agentic Claude tool-use loop written directly against the Messages API (SSE streaming, JSON-schema

outputs, prompt caching, vision), and an on-device Foundation Models fallback, with every tool call routed through one confirmation-gated pipeline and a second human gate on privileged commands.

- Shipped v1.0 as a signed app with a DMG installer, first-run onboarding, Start at Login, four procedurally generated skins plus user-defined JSON skins, on-device speech recognition with a neural TTS chain, and a 93-check self-driving end-to-end rig covering window riding, reactions, hot-swapped Shimeji characters, and the confirmation gates.

Screen-Coach AI: Accessibility-Grounded On-Screen Coaching for macOS Aug 2026 – Present

Swift, AppKit, macOS Accessibility API, ScreenCaptureKit, MLX, Holo1.5-7B (4-bit), Speech, Python, XCTest github.com/jke48222/screen-coach

- Built a hotkey-summoned macOS coaching overlay in Swift (8,300+ lines, 96 headless tests) that points a cursor at the exact UI control a user names: it grounds on the accessibility tree first and falls back to a locally quantized Holo1.5-7B vision model (MLX, 4-bit) only when the tree cannot answer; on a cooperating app the lexical resolver answered 12/12 targets in 0.067 ms without ever loading the model.
- Let measurement set the architecture: cold accessibility reads ran 6–10x slower than warm and decayed within 10 s, so an `AXObserver`-driven cache with a heartbeat became load-bearing; batched attribute reads cut extraction 3.1x, a warm ScreenCaptureKit stream beat `screencapture` 7.9 ms to 204 ms at p90, and tree-aimed crops matched full-frame vision accuracy at 3x the speed.
- Engineered honest confidence rendering: only agreement between tree and vision earns a solid pointer ring, vision-only answers are always marked uncertain, and vague queries decline to point; a hot-reloading privacy gate blocks both capture and tree reads for excluded apps, added after testing showed the tree alone leaked private message content.
- Extended the coach into a teacher: multi-step lessons detect completion by diffing live accessibility trees against each step's starting state, a click-through scrim with numbered badges highlights targets, and a "watch me" recorder converts real clicks into machine-independent JSON workflows, idling at about 0.05% CPU and 45 MB resident.

Damage-Claim Evidence Verifier (HackerRank Orchestrate Hackathon) Jun 2026

Python, Anthropic Messages API (Claude Sonnet 4.6, Claude Opus 4.8), JSON-schema tool use, Pydantic, unittest Solo entry, 24-hour hackathon

- Built, solo in a 24-hour hackathon, a multimodal damage-claim verifier for cars, laptops, and packages: claim photos, chat transcripts, user history, and evidence rules flow through forced tool-call JSON-schema outputs at temperature 0, an atomic SHA-256-keyed response cache, typed retries with backoff, and a four-worker pool with per-row fallback so a 44-claim run always completes.
- Separated "the model does vision, code does bookkeeping": a deterministic post-processing layer (enum alias normalization, history-derived risk flags, cross-field consistency rules, review routing) lifted risk-flag F1 from 66.7% to 100% and exact-row agreement from 65% to 100% on the labeled sample.
- Hardened the pipeline with a 12-pattern multilingual prompt-injection detector that caught and routed all 7 adversarial claims to manual review, benchmarked Claude Opus 4.8 against Sonnet 4.6 live (75% claim-status accuracy at roughly $0.064 vs. $0.012 per claim), and shipped 13 unit tests including a machine-checked no-hardcoding guard.

Übersicht Desktop Widget Suite May 2026

React/JSX, Übersicht, WebGL, AppleScript, Python, MusicKit, Anthropic Messages API, NASA APOD API github.com/jke48222/widget-suite (12 public repos)

- Built 12 macOS desktop widgets for Übersicht in React/JSX (5,800 lines across 12 repositories), 11 sharing one design system: color and type tokens, a frosted-glass card shell, persisted drag and resize handles, and last-known-good caching with a stale indicator for offline resilience.
- Engineered the graphics-heavy pieces, including a full-screen WebGL shader wallpaper, a `model-viewer` PBR viewer that rotates through glTF models daily with an offline fallback, and a dot-matrix globe plotting visited cities with arcing flight paths.
- Fed the widgets through shell and Python helpers: Apple Music and Spotify now-playing via a MusicKit ES256 token flow, GitHub contributions, NASA's Astronomy Picture of the Day with inline video, and Claude-generated daily prompts with a bundled fallback library, with API keys held in the macOS Keychain and secret masking in the clipboard widget.
- Shipped each widget as its own repository with install and doctor scripts and distributable bundles, after first prototyping the suite as a native Swift WidgetKit app (3,000 lines, 12 SwiftUI widgets) and choosing Übersicht for the continuous animation WidgetKit's timeline model disallows.

Web & Full-Stack Engineering

Relay: Order Management & Fulfillment Console Jul 2026

Elixir 1.20, Phoenix 1.8, Ecto, PostgreSQL 16, Phoenix Channels, React 19, TypeScript, Vitest, Docker, Kubernetes github.com/jke48222/relay-oms

- Built an event-driven order-management system on Elixir/Phoenix and PostgreSQL with a React 19/TypeScript console: a 13-route JSON API feeds an asynchronous fulfillment worker that walks orders from received to delivered, streamed live over Phoenix Channels with metrics coalesced to one push per 300 ms.
- Implemented a transactional outbox (each state change and its event commit in one `Ecto.Multi`, fan-out after commit), an 8-state, 11-transition machine enforced under row locks so timer and cancel races resolve to HTTP 409, and boot-time rehydration that rescans in-flight orders so a restart strands nothing.
- Prevented oversell in two layers, `SELECT …FOR UPDATE` allocation preferring the destination's shipping zone across four facilities backstopped by a `CHECK (allocated <= on_hand)` constraint, with `Idempotency-Key` replay separating 201 creates from 200 replays.
- Wrote 39 ExUnit and 13 Vitest tests aimed at failure modes (oversell, partial reservations, illegal transitions, replay, rehydration), kept green by GitHub Actions with `--warnings-as-errors`, oxlint, and `tsc --strict`, plus Dockerfiles and Kubernetes manifests with health probes and a migration Job.

Live Election Platform for a Campus Engineering Organization Mar 2026 – Jun 2026

Next.js 14 (App Router), JavaScript, Tailwind CSS, Supabase (Postgres, Realtime Broadcast, RLS), Vercel github.com/jke48222/live-election-platform

- Designed and shipped a presenter-paced election system for a 14-role, 39-candidate chapter election, built for 60–80 concurrent phone voters: a three-state election machine with runoff restarts and floor nominations, countdowns synced to absolute UTC expiry timestamps, and 0–400 ms submission jitter to spread load.
- Built the realtime layer on Supabase WebSocket Broadcast with a single-row election state enforced by a unique index, a generation counter that fixed a broadcast race, and a REST fail-safe that refetches on `visibilitychange` and resubscribes after channel errors.
- Engineered layered anti-fraud: SHA-256 device fingerprinting behind a `UNIQUE(role_id, device_hash)` constraint with idempotent duplicate handling, row-level security with anonymous inserts revoked, a dues-roster gate at check-in, an HMAC-signed httpOnly admin session, and a zero-dependency sliding-window rate limiter.
- Ran a 29-finding security audit, fixing 16 and confirming the rest with 42 HTTP-level tests against the live stack, then began a multi-tenant rewrite in June on self-hosted Postgres with per-organization RBAC, a LISTEN/NOTIFY WebSocket gateway, and pluggable eligibility providers.

QR Worlds: Every URL Is a Place Aug 2026 – Present

TypeScript, three.js, GLSL, Vite, Vitest, Cloudflare Pages Functions and Workers Not yet published

- Built a TypeScript and three.js experience in which any typed URL becomes a place: its QR code renders module-for-module as a floating voxel ground while the address deterministically generates the planet above it (6–11 hues, hash-gated rings, moons, and clouds), rebuilding thousands of instanced voxels in three draw calls within single-digit milliseconds per keystroke.
- Guaranteed cross-machine determinism with xoshiro128** over 32-bit integer arithmetic (`Math.imul`, no transcendentals, since V8 and JavaScriptCore differ in the last bits), double FNV-1a seeding, and a forked random stream per subsystem so a new consumer cannot shift an existing world.
- Measured scannability rather than asserting it: projecting every module through the live camera reported at most 1.3% misread and 8% occluded modules against error-correction level M's roughly 15% budget, and a ground color that looked dark but measured 1.8:1 was deepened to 3.75:1, with tests enforcing the contrast floor and occlusion ceiling.
- Extracted `orbit-pack`, a zero-dependency TypeScript library (3.9 KB gzipped) that packs structured state into the shortest chat-safe URL via bit-width-per-field schemas, base64url and QR-alphanumeric alphabets, and `deflate-raw` only when it wins, with 38 tests including 10,000-planet fuzzing and a pre-publish gate that typechecks the packed tarball in a clean install.

Personal Portfolio: 3D Workstation & Scroll-Scrubbed Landing Page Jan 2026 – Present

React 18, TypeScript, Vite, three.js, React Three Fiber, Framer Motion, Zustand, Tailwind CSS, PWA, Vercel jalenedusei.com | github.com/jke48222/Edusei-Workstation

- Built and maintain jalenedusei.com (109 commits, 13,000+ lines of TypeScript): a library-free scroll-scrubbed video landing page (requestAnimationFrame easing toward scroll progress, reduced-motion fallback) that opens into an explorable 3D CRT workstation in React Three Fiber with eased camera transitions between 2D and 3D.
- Designed a terminal-style navigator (`help`, `list`, `run`, tab completion) on a Zustand store and a 24-preset theme engine shared by the 3D scene and the Tailwind UI, plus a playable canvas mini-game with its own simulation loop.

- Shipped it as a PWA with service-worker shell caching, lazy routes, GLTF preloading, a capped device-pixel ratio, and a render loop that pauses while the game is open, with Blender export scripts and a GitHub Actions job that converts WebM alpha video to HEVC for Safari.

Quantitative Paper-Trading Research Harness Jul 2026 – Present

TypeScript (strict, zero runtime dependencies), Node.js, Binance and Alpaca paper APIs, Pine Script, launchd Simulation only; no live order path

- Built a 4,200-line strict-TypeScript harness (zero runtime dependencies) that backtests and forward-tests systematic strategies on crypto and equity paper accounts: closed-candle ingestion from Binance and Alpaca, volatility-scaled sizing with daily-loss and cluster-cap kill switches, and an atomic two-file trade memory that refuses to repeat losing setups.
- Pre-registered seven trials in an append-only ledger (114 runs across 52 configurations) with in-sample/out-of-sample splits, 30 bps costs, drift benchmarks, and a 10,000-resample bootstrap under Bonferroni correction, rejecting every single-asset candidate and freezing the one surviving portfolio-level effect ($t = 2.91$, corrected $p = 0.021$) pending out-of-sample confirmation.
- Operated the system unattended for five-plus weeks through four launchd agents with a deadman health check, logging measured fill slippage on every paper order with zero heartbeat failures.

Travel Itinerary Application Dec 2023

Java 17, JavaFX 17, Gson, Maven, Google Places API, REST Countries API, API Ninjas Software Development, CSCI 1302

- Created a JavaFX desktop application that assembles a destination itinerary from live REST APIs: REST Countries and API Ninjas for country and city facts, and Google Places Text Search and Place Photos for hotels, activities, attractions, and restaurants.
- Moved the six sequential HTTP calls onto a background thread with `Platform.runLater` updates and a progress indicator so the interface stays responsive while data loads.

Embedded Systems & Hardware

Album-Art LED Matrix Wall Aug 2026 – Present

Python, C, MicroPython, Raspberry Pi 5, HUB75, KiCad 10, ngspice, systemd github.com/jke48222/album-art-matrix; hardware integration in progress

- Designed a 9-panel, 480 mm HUB75 LED wall (Raspberry Pi 5, 64x64 P2.5 panels) that displays the playing album cover as a spinning disc: a Python controller chains now-playing adapters (Music.app scripting, MusicKit account history, a staged Spotify PKCE adapter) into an artwork cache and pluggable frame sinks, exercised end to end through a macOS preview sink before any hardware arrived.
- Wrote the display path in C against the GPU-driven `rpi-gpu-hub75-matrix` library (9,600 Hz refresh on an isolated core) as a daemon fed raw RGB888 frames over a named pipe, with an art pipeline that downscales with Lanczos and unsharp masking, applies white-balance gains in linear light, and spins the disc with 4x supersampling, the gains derived from a TCS34725 colorimeter on a Pico 2 W.
- Replaced the HAT, bus bars, and fuse holders with a custom 2-layer, 2 oz Pi 5 backplane PCB in KiCad: four 74AHCT245 level shifters with 33 Ω termination feeding three HUB75 chains, three 12 A power sections with per-panel blade fuses, and a polyfuse- and TVS-protected power jumper, with the circuit expressed as code that emits the netlist and BOM, sized by ngspice simulations, and delivered as gerbers with zero clearance or track violations.
- Specified per-panel 14 AWG power injection from Mean Well supplies with an NTC inrush limiter and star ground, plus a systemd-managed Pi deployment with an rsync bootstrap, CPU isolation for the refresh thread, and a Mac-side reporter service the Pi polls over HTTP; panels arrived in August 2026 and first light is the next milestone.

AnimalDot Smart Bed (Senior Capstone) Aug 2025 – May 2026

ESP32, C++ (PlatformIO), NimBLE, MQTT, SwiftUI, React Native / Expo, Express, PostgreSQL, DSP Capstone Design I & II, CSEE 4910/4911; 6-person team, sole software author

- Led all software for the six-person senior capstone (36 of 37 commits), a contactless pet vital-signs bed for clients Dr. Wenzhan Song and Dr. Benjamin Brainard: ESP32 firmware, a native SwiftUI iOS app, a React Native/Expo app, an Express/PostgreSQL backend, and a React web app, about 21,000 lines in total.
- Wrote the firmware's real-time ballistocardiography DSP in C++: 200 Hz geophone sampling through a zero-phase Butterworth band-pass (0.67–3 Hz, hand-derived biquads), kurtosis-based motion rejection, trimmed-mean inter-beat heart rate, and amplitude-demodulated respiration, alongside dual BLE GATT services (a 9-characteristic custom service plus the standard 0x180D Heart Rate profile), captive-portal WiFi provisioning, and NVS-persisted calibration.
- Designed a BedDot-compatible binary MQTT telemetry protocol (20-byte header plus int32 frames) publishing to UGA's SensorWeb broker, with the C++ encoder and independently tested TypeScript and Swift decoders, porting one DSP pipeline across four languages (scipy reference, Swift Accelerate/vDSP with FFT autocorrelation, TypeScript on backend and app) with matching filter coefficients.

- Built the clients end to end: a JWT-authenticated Express/Postgres backend that ingests MQTT, re-runs the DSP server-side, and broadcasts per-user over WebSockets; a React Native app with automatic BLE-to-MQTT-to-cloud failover; and a SwiftUI iOS app with Swift Charts, covered by about 53 tests and two iOS CI pipelines.

PrimeForge: FPGA Prime-Finding Engine Jan 2026 – May 2026

Verilog, Digilent Nexys A7-100T (Artix-7), Vivado, DDR2 (MIG), SD over SPI, VGA, Icarus Verilog Advanced Digital Design, CSEE 4280; 2-person team

- Architected PrimeForge, a Verilog prime-finding system on a Digilent Nexys A7-100T (Artix-7): 33 modules and 5,500 lines of RTL across four clock domains, integrating dual compute engines, a DDR2 results pipeline, an SD-card interface, and a VGA interface under one top level with an FSM-driven four-mode menu (range, timed, single check, self-test).
- Implemented a segmented Sieve of Eratosthenes (odd-only BRAM bitmaps, 2^{20}-number segments) and a 6k$\pm$1 trial-division core; the sieve found all 5,761,455 primes below 10^8 on hardware with the exact reference count displayed on screen.
- Built a two-client DDR2 arbiter on the MIG 7-series native interface with in-order tag FIFOs, packing four 32-bit primes into 128-bit bursts through gray-coded async FIFOs across the 100-to-83 MHz clock crossing, plus an SD-card SPI controller written from scratch that feeds a lock-step DDR2-versus-SD self-test comparator.
- Rendered a 640x480 @ 60 Hz VGA interface (an 80x30 text grid from an 8x16 font ROM, a rate-scaled sprite, live count, last-20 primes, and mm:ss.fff runtime) kept tear-free by frame-stable snapshot clock-domain crossing, and verified the design with 12 self-checking Icarus Verilog testbenches over behavioral DDR2 and SD-card models with randomized back-pressure.

PARMCO: Bluetooth Motor Control System Jan 2026 – May 2026

C, BlueZ (GDBus), pigpio, ARMv7 assembly, Raspberry Pi 4, L293D, systemd, SwiftUI, Core Bluetooth Embedded Systems II, ECSE 4235; 2-person team; github.com/jke48222/parmco

- Built a 1,900-line BLE GATT server in C on a Raspberry Pi 4 over BlueZ's D-Bus API: a custom service parses UTF-8 motor commands, streams RPM telemetry, and registers a pairing agent that fixed an iOS connection hang; a systemd unit starts it at boot and recovers the pairing within about 45 s of a power cycle.
- Drove an L293D motor driver with pigpio DMA-timed PWM and ARMv7 assembly GPIO primitives over direct BCM2711 register mapping, and closed the loop with Maintain and Match modes that count IR-tachometer pulses through a 74HC374 latch and retune duty cycle every 250 ms, settling to a 4,500 RPM target in about one second without visible overshoot.
- Designed the SwiftUI iOS client over Core Bluetooth (1,600 lines) with manual and automatic control modes, a backward-compatible telemetry parser, and reconnect by stored peripheral UUID.

Smart Plant Assistant Aug 2025 – Nov 2025

Raspberry Pi Pico (RP2040), MicroPython, analog signal conditioning, op-amps, Kalman filtering Sensors and Transducers, ELEE 4230

- Built a battery-powered plant-environment monitor on a Raspberry Pi Pico that reads an NTC thermistor, a capacitive soil-moisture sensor, and a photodiode light sensor through an op-amp stage, multiplexed by a 74HC4051 into one ADC input and powered by an 18650 cell with a TP4056 charger and MT3608 boost converter.
- Applied a one-dimensional Kalman filter per channel in MicroPython to suppress sensor noise, calibrated each sensor (soil moisture spanning 2.58 V dry to 1.43 V saturated, tested in local red-clay soil), and drove an OLED display with live readings, threshold alerts, and a plant "mood" icon.

Audio Tracking Car Jan 2025 – Apr 2025

Python (gpiozero), Raspberry Pi 4, analog filter design, comparators, MOSFET H-bridge, Altium, SolidWorks ECSE Design Methodology, ECSE 2920; github.com/jke48222/Audio-Tracking-Car

- Built a two-microphone sound-seeking car in which frequency selection happens in hardware: electret microphones feed a TL084 preamplifier, analog band-pass filters tuned to target tones, envelope detectors, and LM339 comparator ladders the Raspberry Pi reads as 4-bit levels over GPIO.
- Wrote the Python control loop that rotates while sampling at 20 Hz, aligns to the peak direction, and drives until a bump switch trips, regulating wheel speed proportionally from 20-slot optical encoders through a discrete MOSFET H-bridge; documented the design with Altium schematics and SolidWorks CAD.

LED Frequency Filter Aug 2024 – Dec 2024

Analog filter design, op-amps, Multisim, breadboard prototyping Linear Systems, ELEE 6210

- Designed and built an analog circuit that classifies an input signal into five frequency bands (below 100 Hz through above 400 Hz) and lights one LED per band, using RC high-pass and low-pass stages with op-amp gain of 3, simulated in Multisim and implemented on a breadboard.
- Iterated through three designs to resolve 33% post-filter voltage attenuation, op-amp signal inversion, heat from chained stages, and premature LED activation in transition bands caused by component tolerances.

PDMS Microfluidic Mixer Jan 2026 – May 2026

Photolithography, PDMS soft lithography, cleanroom microfabrication, syringe-pump characterization Introduction to Microfabrication, ELEE 4310; 4-person team

- Designed and fabricated passive microfluidic mixers in PDMS to overcome diffusion-limited mixing in laminar, low-Reynolds-number flow, comparing Y-junction, sharp-bend, and smooth-bend serpentine geometries that trade interface generation against residence time.
- Produced photoresist molds by photolithography (1000 RPM spin coat, 95 °C soft and post-exposure bakes, mask-aligner UV exposure) and cast devices with trichlorosilane surface treatment, plasma bonding to glass, and an 80 °C cure.
- Characterized the devices on a Chemyx syringe pump at 0.001–0.1 mL/min per inlet, imaging junctions and bends under a microscope to compare mixing and diagnose dust-defect and channel-leakage failure modes.

Games & XR

Ashfall: The Last Hours of Pompeii (Vertical-Slice Prototype) Jun 2026

Unreal Engine 5.8, C++, Lumen, Nanite, StateTree, Python, three.js, Git LFS github.com/jke48222/ashfall

- Built the C++ gameplay framework for a photoreal time-travel puzzle prototype in which a Pompeii city block toggles between its intact "Zenith" and eruption "Fall" states: a world subsystem as the single source of truth, per-actor temporal components that morph transform, material, and physics with tick-on-demand blending, a lighting director that cross-fades scene moods, and causal flags that persist interventions across toggles (1,333 lines of new C++).
- Wrote a license-gated Python asset pipeline that pulls only CC0/CC-BY assets from Poly Haven, Sketchfab, and Freesound with provenance manifests, imports 9 PBR texture sets and 87 Nanite meshes with correct color spaces, and builds the entire level procedurally from a script.
- Verified the slice headlessly on macOS with UE-Python assertion suites (32 checks, all passing), tagged five build-verified milestones, and ported the scene to a three.js web build with HDRI lighting and a 6,000-particle ashfall.

Kitchen Chaos VR Oct 2025 – Dec 2025

Unity 6, C#, Meta XR SDK 81 (Interaction SDK), OpenXR, VelNet, OpenAI API Virtual Reality, CSCI 6830; 2-person team, equal split; github.com/jke48222/VR-Final-Project

- Built an Overcooked-style multiplayer VR cooking game for Meta Quest 3 in Unity: physics-driven grabbing, cutting, cooking, whisking, and plating across 133 food prefabs and 8 ScriptableObject recipes, with a four-event chaos system, an in-VR recipe book, and avatar customization in about 50 C# scripts.
- Integrated VelNet networking to synchronize player avatars, spawn assignment, and 147 physics props across clients and to gate the kitchen scene on both players joining, with editor tooling that auto-wires network sync components onto prefabs.
- Developed a recipe-scoring pipeline with trigger-based ingredient tracking and readable score breakdowns, an AI dish judge calling the OpenAI API with a fixed JSON response schema, and text-to-speech narration of round themes, chaos events, and results.

XR Portfolio 2: VR Mini Museum & Mixed-Reality Room Oct 2025 – Nov 2025

Unity 6, C#, Meta XR SDK 81, OpenXR, XR Hands, Meta Voice SDK (Wit.ai), Ready Player Me Virtual Reality, CSCI 6830; github.com/jke48222/VR-Portfolio-2

- Developed two Quest 3 experiences in 30 hand-written C# scripts: a VR mini museum with parabolic-arc teleportation, snap turning, physics grabbing with a throw-velocity estimator, velocity-scaled haptics, NavMesh NPC visitors, and trigger-zone spatial narration; and a mixed-reality room with passthrough, depth-based occlusion, hand-tracked poke and ray UI, and catalog-driven object spawning.
- Built a voice-driven NPC assistant on Wit.ai intents (panels, music, volume, spawning) with Meta Voice SDK text-to-speech and amplitude-driven lip sync on a Ready Player Me avatar.

XR Portfolio 1: Four Unity Demos Aug 2025 – Oct 2025

Unity 6, C#, OpenXR, Input System Virtual Reality, CSCI 6830; github.com/jke48222/VR-Portfolio-1

- Built four Unity demos (a solar-system transformation scene, a Rube Goldberg physics chain, a VR exergame, and a VR potion puzzle) in 18 C# scripts covering smooth locomotion on a CharacterController, collision-driven gameplay, a 399-line puzzle state machine, and reflection-based controller haptics, each documented with a demo video.

Human-Computer Interaction & Design

BreakBuddy Aug 2025 – Dec 2025

User research, affinity mapping, experience prototyping, heuristic evaluation, Figma Human-Computer Interaction, CSCI 4800

- Designed a stress-management application for educators that turns 2–5 minute pauses into guided micro-break sessions with social accountability and low cognitive load.

- Ran semi-structured interviews, affinity mapping, and assumption mapping, then validated the concept through paper and high-fidelity prototypes with defensive error states and a streak-tracking reports dashboard, refined by iterative heuristic evaluation.

TEACHING & MENTORING EXPERIENCE

UGA STEM Academic Success Program, *Student Advisor,* Athens, GA — Aug 2023 – May 2025

- Planned and ran 8+ academic-success workshops per semester for 30+ STEM students, and owned the program's digital and social-media strategy, growing online engagement by 50%.

University of Georgia Housing, *Resident Assistant,* Athens, GA — Aug 2023 – May 2025

- Built community for 45 residents through 10+ events per semester, mediated 30+ conflicts and safety concerns, and partnered with staff on academic-success and mental-health programming.

UGA Office of Engagement, Leadership, and Service, *ELS Peer Leader,* Athens, GA — Jan 2024 – May 2024

- Selected for a leadership-coaching program; mentored first- and second-year students 1:1 and facilitated workshops on values-based leadership.

LEADERSHIP & UNIVERSITY SERVICE

National Society of Black Engineers (NSBE), UGA Chapter, *Vice President* — May 2025 – May 2026

- Led strategic planning for a 100+ member chapter, coordinated convention travel and registration for 50+ members, and launched a centralized digital resource hub for internship pipelines and alumni contacts.

National Society of Black Engineers (NSBE), UGA Chapter, *Senator* — May 2024 – May 2025

- Represented the chapter in regional and national legislation votes and managed conference logistics and budgets with the Treasurer.

National Society of Black Engineers (NSBE), UGA Chapter, *Telecommunications & Vice PR Chair* — May 2023 – May 2024

- Grew chapter social-media engagement by 40% with data-driven content, and designed and launched a new chapter website.

Tau Beta Pi Engineering Honor Society, *Member* — Oct 2024 – Present

- Inducted for academic distinction and leadership; participate in professional-development and STEM-education service events.

Theta Tau Professional Engineering Fraternity, Iota Epsilon Chapter, *Brother* — Jan 2024 – Present

- Co-organize technical workshops, speaker panels, and community outreach for a 120+ member chapter.

HONORS & AWARDS

Extraordinary Engineer, UGA College of Engineering – leadership, academics, and service — Feb 2024
Presidential Scholar (2x) – 4.0 GPA in consecutive full-time semesters — Fall 2022, Spring 2023
Dean's List, University of Georgia — Spring 2024
Dire Needs Project Fund – competitive $1,500 grant for a student-led engineering project — Aug 2023 – May 2024

CERTIFICATIONS & TRAINING

Emerging Engineering Leadership Development (EELD) Program, UGA College of Engineering — May 2026

- Capstone leadership program with the J.W. Fanning Institute: conflict resolution, stakeholder engagement, and cross-generational collaboration.

CITI Program: SBE Foundations, Human Subjects Research — Sep 2025 (valid through Sep 2030)