

Andrew D. Gundersen

+1 (414) 949-0219 | andrew@crimata.com | andrewgundersen.net | [LinkedIn](#) | [Git](#)

SUMMARY

Software engineer with 6 years of experience across web, full stack, and low-level systems. At Accenture, built enterprise front-end applications in Angular and TypeScript. As co-founder of Crimata, built a voice-assistant platform end to end: native C audio and desktop clients, an Electron/Vue application, an asyncio Python backend, and a TensorFlow audio super-resolution model. Also designs small systems from first principles, including a C HTTP server, a Python agent framework, and a self-hosted, hardened Linux server running git, mail, and VPN services.

EXPERIENCE

Advanced App Engineering Sr. Analyst (Software Engineer)

Jun. 2021 – Feb. 2025

Accenture

Columbus, OH

- Built [pico-http](#), a minimal C HTTP server on raw BSD sockets (~750 LOC across 11 files, gcc -Wall -Wextra, no external deps): Express-style routing API (use("/path", handler), void (*Handler)(Req*, Res*) handler shape) backed by a hand-rolled generic hashtable of function pointers (128 buckets, polynomial string hash) chained on a hand-rolled generic linked list (callback-based search / delete / loop for polymorphic reuse) — each data-structure library in its own subdirectory with its own Makefile and standalone unit test.
- Built an Angular 13 multi-step form application (~800 LOC TypeScript, 10 components) for asset check-in / check-out: declarative JSON form schema driving a StepComponent that *ngSwitches on field.type to render custom widgets — including a live camera capture component (navigator.mediaDevices.getUserMedia() + canvas.toDataURL()) and an HTML-canvas signature pad with mouse-tracked drawing and configurable stroke styles — all coordinated by a per-field ok EventEmitter aggregator that gates step progression.
- Built agent-framework, an experimental Python backend (~1.5 K LOC, aiohttp + Peewee, Vue 3 vendored no-build frontend) for exposing arbitrary Python functions and data models as agent-callable primitives: developers tag functions / models / singletons with Unicode-prefixed docstrings plus @route / @handler / @runner / @tool labels, and the framework auto-collects them via an AST-based DependencyChecker that re-queues files with unresolved names until they resolve, then binds them into a central lib namespace. Dynamic Peewee model creation via type(name, (Base,), fields) from docstrings, a custom sync pub/sub Emitter that turns CRUD calls into {method: create|update|remove, data} events, and a backend-driven UI where Vue 3 components call watch(table) over a /watch WebSocket to bind a reactive ref to live SQL state — the entire surface multiplexed onto one aiohttp process by subdomain (api. / ws. / ..).

Co-Founder and CEO

May 2019 – Jun. 2021

Crimata

Milwaukee, WI

- Built an Electron/Vue 3 desktop client (~1.6K LOC JS, 17 components, packaged via Electron Forge for macOS/Windows/Linux): frameless custom window chrome, tray state machine, context-isolated preload IPC (11-channel allowlist), WebSocket messaging with auto-reconnect, global-hotkey audio capture.
- Wrote [nodeaudio](#), a pure-C Node.js native add-on (raw N-API, 680 LOC, 10 exports) wrapping PortAudio for 16 kHz mono capture/playback: non-blocking napi_threadsafe_function marshaling of PortAudio's real-time callbacks into the JS event loop, zero-copy Int16Array handoff via external ArrayBuffers, lock-free input ring buffer, and a mutex-guarded output ring buffer with a spawn-on-full playback thread so the Electron main process never blocked on audio.
- Built the Crimata voice-assistant backend in Python: a pure asyncio WebSocket server (no REST framework — websockets 8.1), Peewee ORM over MySQL, self-hosted on Ubuntu Server; 1.2K LOC across 25 files handling multi-modal client input (text, audio, image, file) through a unified message pipeline with token-based auth on the WS handshake.
- Designed a plugin-style app framework in which each service (Contacts, Messenger, Settings) ships its own YAML-defined intents and gets dispatched via Google Cloud Dialogflow intent classification (with a BERT next-sentence-prediction fallback path); paired with Google Cloud Speech-to-Text for server-side transcription (LINEAR16, 16 kHz mono base64 in, transcript out) and an asyncio.Queue-based outbox/inbox pattern that lets an in-flight service RPC back to the user for missing parameters mid-turn.
- Built the first macOS client for Crimata (voice-assistant platform) in 2.1K lines of C: GTK3 window/event loop, Cairo-rendered custom UI composed entirely from vector primitives, 60 Hz double-buffered render pipeline coordinating three background threads (TCP, PortAudio + libfvd VAD, animation) with mutex-guarded surfaces, plus a spring-damped oscillator and eight easing functions driving voice-reactive animations.

- Built the auth + downloads service in Node.js/Express with Sequelize on MySQL (~200 LOC): single POST /auth endpoint branching register/login on payload shape, bcryptjs-hashed passwords behind a server-side policy regex, opaque uuid.v4() session tokens, GET /download serving the desktop-app installer, a PROD=1 interactive-confirmation guard against accidental prod runs, and an axios integration test covering register / login / wrong-password / duplicate-email / invalid-email.
- Built [ML Audio Enhancer](#), a 40 M-parameter time-domain audio super-resolution model in TensorFlow/Keras (~840 LOC) that 4×-upsamples 16 kHz speech, after [Kuleshov et al. \(2017\)](#): U-Net-style Conv1D encoder/bottleneck/decoder with SubPixel1D upsampling via tf.batch_to_space (instead of transposed convolutions), skip concatenations, and residual output; trained on VCTK through an HDF5 patch pipeline (50 ms chunks) with MSE waveform loss, Adam + optional SGDR cosine-annealing, orthogonal init, and dropout regularization, alongside an offline chunked inference script emitting target / cubic-spline-baseline / predicted audio and 4-panel STFT plots.
- Shipped [crimata.com](#) (plain HTML + vendored Vue 3, no build step, ~2.6 K LOC) as an in-browser demo of the Crimata desktop app: reused 947 lines of Vue components byte-for-byte from the Electron repo by writing a 50-line Emitter shim that impersonates the preload's window.mainApi (on/emit/send/invoke) and routes every channel (message, account, ws, record, playback, messenger, nav, logout) into a 594-line scripted timeline — a 10-step guided tour with simulated typing, fake WebSocket status, mock “recording” indicators, and base64-inlined demo assets in lieu of real IPC, native audio, or backend calls.

PROJECTS

[andrewgundersen.net](#) | *Linode, Ubuntu, nginx, Forgejo, WireGuard, Samba*

- Operate my personal live server on a Linode Ubuntu 24.04 VPS (g6-nanode-1, 1 vCPU / 1 GB / 25 GB) running a self-hosted Forgejo git forge (v16.0.2, served at [andrewgundersen.net/git/](#) via nginx subpath), a chasquid + dovecot mail stack (SMTP/IMAP/POP3, PAM auth, LMTP-over-unix-socket delivery, SpamAssassin), a WireGuard split-tunnel VPN, and a Samba SMB3 share bound only to the WireGuard interface — fronted by nginx + Let's Encrypt (certbot post-renewal hooks sync certs into chasquid and restart it) behind a two-layer firewall (Linode cloud firewall + on-host ufw).

TECHNICAL SKILLS

Languages: Python, JavaScript, HTML/CSS, C, SQL

Tools & Frameworks: React, Vue, Node, Next, Express, FastAPI, HTTP, Websockets, PortAudio, GTK, TensorFlow

Ecosystems: Python (AI/ML), Fullstack, Embedded

Cloud: Akamai, Google Cloud

EDUCATION

Xavier University

Bachelor of Science in Engineering Physics