Steve Howell

showell285@gmail.com · 202-213-7553
Chambersburg, PA · US-based, willing to relocate
lynrummy.com · lynrummy.com/blog · github.com/showell

Forward-thinking software engineer with 35+ years across systems programming in C, web platforms, and AI-agent development.

Languages: C, C++, CoffeeScript, Elm, Go, Java, JavaScript, Odin, Perl, Python, Ruby, TypeScript, Zig

Education

Duke University — Durham, NC, Class of 1989
Phi Beta Kappa. BS with a double major (Computer Science / Electrical Engineering) completed in three years.

Experience

Personal Projects — 2026

- **January** — Mentored a colleague on contributing to Zulip; co-authored commits for third-party integrations and personalization.
- **February** — Developed the card game Lyn Rummy in TypeScript.
- **March** — Learned Go, Odin, and Zig.
- **April** — Taught an agent (Claude) to play an optimal version of Lyn Rummy.
- **May** — Steered Claude to write a Go-based Chat/Docs system.
- **June** — Had Claude write a 3D safari driving game, and ported the Chat/Docs system to Zig.

Zulip / Kandra Labs — 2016–2025

Remote · open-source office chat application

I was one of the most prolific contributors to Zulip as it grew into a premier open-source office chat application. Zulip is a large chat system with a wide surface area — messaging, search, notifications, presence, permissions, and administration — built on Python, Django, and PostgreSQL on the backend; jQuery/JavaScript (and eventually TypeScript) on the frontend; and a real-time event system connecting the two. I made over 5,700 atomic commits across the codebase. My role was to improve the code along four axes: readability, reliability, usability, and performance.

Much of my work involved difficult structural changes to a large legacy codebase. I took sprawling, tightly coupled code and extracted the key concepts into cleaner, modular components — for example, consolidating dozens of ad hoc event-validation checks into a single shared schema, or carving a hard-to-test piece of UI logic into its own module so the surrounding code could be tested thoroughly. These changes had to respect legacy constraints and existing dependents, which meant understanding the system deeply before changing anything.

I also focused on performance, tuning Django ORM queries, indexes, and Python-side ORM code to do less work.

I built Zulip's unit-testing framework on Node.js in 2014; it remains in use today.

Software Engineer — Dropbox

Apr 2014 – Oct 2014 · San Francisco, CA

Zulip was acquired by Dropbox in March 2014. There I added groups support to Dropbox for Business and sped up their continuous-integration pipeline. I worked almost exclusively in Python, against proprietary Dropbox code, building web interfaces and managing distributed metadata for customers' uploaded and synced files.

Software Engineer — Zulip, Inc.

Apr 2013 – Mar 2014 · Cambridge, MA

I was one of the original ten developers at the Cambridge-based Zulip startup, founded by four MIT graduates who had exited another startup and saw a niche for topic-based chat. We grew a beta program to thirty organizations on the standard stack of the era — jQuery on the front end, Django on the back. Everyone worked full-stack, so I split my time evenly between JavaScript and Python.

Software Engineer — DomainTools LLC

Feb 2012 – Apr 2013 · Seattle, WA

DomainTools built services around its extensive in-house database of domain records. I maintained their PHP web tools and Python backend systems, and wrote C. Several daily map/reduce jobs that analyzed large domain-data sets ran in Python; I ported them to C, making them 30x faster. That work included a lightweight, home-grown JSON parser and an arena-based allocation system to prevent memory leaks.

Software Developer — AmazonFresh

Sep 2007 – Jan 2009 · Seattle, WA

I developed software for the AmazonFresh grocery-delivery service. Our team split time between the Java-based customer-facing website and the Ruby-on-Rails-based warehouse management system (WMS). I focused on the WMS, writing browser-based software for the warehouse scanner devices so associates could receive, stow, pick, pack, and load groceries. We used a MySQL database to track inventory throughout its life cycle — from the receiving truck to the delivery truck. I also built the reporting systems that let managers track inventory, associate productivity, and delivery status.

Because we operated in a single exploratory market (Seattle), we were not writing software at tremendous scale, and we used rudimentary but

effective Rails concepts to build the entire warehouse-facing system. The challenge was instead the pace: we iterated quickly on new features and new processes in the warehouse. The whole software team worked closely with product and warehouse managers to evaluate, improve, and automate processes for picking, stowing, and packing. Amazon already had deep experience delivering books and other consumer products, but it had never faced the challenges of groceries — perishables, separate picking zones for frozen and cold items, and delivery trucks owned and operated by Amazon.

We optimized processes enough to turn a marginal profit against operating costs, though not against the full cost structure of product teams and engineering. The AmazonFresh experiment eventually gave way to Amazon's acquisition of Whole Foods. More importantly, Amazon learned an enormous amount about last-mile delivery in a tricky market, and the software my teammates and I built was integral to that exploration.

Software Developer — Merchant Link

Apr 2004 – Sep 2007 · Silver Spring, MD

I developed credit-card gateway applications in Python, along with supporting tools for accounting and performance monitoring (some web-based). I also handled production support: troubleshooting and deploying major software releases.

Software Developer — NXT

1997 – Jul 1999 · Bethesda, MD

This company later became Merchant Link / Chase Paymentech through acquisitions (where I worked again in 2004–2007). During my original tenure at NXT, I primarily wrote asynchronous credit-card gateway software in straight C on Sun/Solaris. Our small team of five wrote to multiple credit-card formats and protocols, processing nearly a million transactions per day in the 1990s on two large Unix boxes.

I was also the primary architect of our monitoring and error-tracking systems, written in C/C++. We had a small suite of automated tests, but relied on extensive code review to ensure the software's quality and reliability. The format translations were mostly mechanical, but the protocol translations required a deep understanding of the async nature of legacy dial-up credit-card protocols and the careful construction of state machines in C to react to unexpected events and line noise.

Software Consultant — Watson Wyatt Worldwide

Jun 1992 – Nov 1996 · Washington, DC

Developed applications for cafeteria-style corporate benefits. C/C++, SQL Server.

Software Developer — Oracle

Sep 1989 – Jun 1991 · Redwood City, CA

Automated QA for Oracle mainframe products using Rexx. Installed new database releases and reviewed documentation.

Internships

Maryland

- **Summer 1989** — Computer Sciences Corporation
- **Summer 1988** — Noise Cancellation Technologies
- **Summer 1987** — Computer Sciences Corporation
- **Summer 1986** — Computer Sciences Corporation