

Stepan Kotok

AI Engineer, Full Stack Developer, Solution Architect

React · Next.js · Node.js · TypeScript · PostgreSQL/Supabase · Claude API, OpenAI API, MCP, RAG · Playwright, Vitest · GitHub Actions
Russian, Ukrainian – native; English – B2, conversational.

Chişinău, Moldova · [+373 78 722 382](tel:+37378722382) · wa.me/37378722382 · stkotok@gmail.com ·
linkedin.com/in/stkotok · github.com/StKotok · stepankotok.com



Profile

In software development since 2014; since 2020 led projects as a tech lead at my own company, [Neatapps](#), for clients in the US and Europe. Since 2024 I have been systematically strengthening the whole engineering cycle with AI-driven development.

Team role

- Primary role: Senior developer responsible for technical reliability. I choose technologies for the task, design the system architecture, write code and control quality at every stage.
- When needed, I take the team lead role: at [Neatapps](#) I coordinated work with clients and the team, and ran daily stand-ups with reports on completed work and a review of difficulties.

How I write code with AI

Tools

Claude Code – main tool, used daily, on the top Max plan. Subagents, skills, hooks, custom MCP servers, parallel work in isolated git worktrees. I choose the model for the task: a stronger one plans and accepts the work, a faster one writes code from a brief with acceptance criteria.

Cursor – proficient; used on a paid subscription; my main tool is now Claude Code.

ChatGPT/Codex, Gemini, Grok – on the paid Plus, Pro and Super plans: independent review of code and plans. Several models in turn, with fixes between rounds; a second opinion before irreversible steps: a migration, a deployment, a change of access rights.

Browser and phone testing (Playwright, Mobile-MCP) – agents verify the result themselves in a real browser and on a mobile phone: end-to-end scenarios, accessibility, screenshots.

How this speeds up work without lowering quality

- **Faster code.** In four days I built, with AI agents, an online course platform with offline mode, sign-in and access rights, and an admin panel, covered by end-to-end automated tests.
- **Faster bug finding.** I hand log and error analysis to agents: they find the point of failure and propose a fix, and I check the diagnosis and accept the result.
- **Faster, more thorough testing.** Agents write automated tests in code and scenarios for manual testing. On the [AI assistant project for a tourism service](#) (below), I wrote 1,310 tests in six layers in seven days. The AI also tests like a live tester: it runs scenarios in a real browser and on a mobile phone, takes screenshots and reports what to fix.
- **Faster, safer releases.** A blocking check before the main branch, every change linked to a task, CI in GitHub Actions.
- **Routine goes to agents.** Repetitive edits across dozens of files, issue triage and documentation search are done by agents. I handle architecture, difficult parts and verification of the result.
- **I verify rather than trust.** When the AI makes a mistake, I record a rule for it so the mistake does not repeat. Any generated code is first verified with typing, tests and architecture review, and only then adopted.

Selected projects

AI assistant for a tourism service: hotels, routes, booking

2026 · Next.js, TypeScript, Supabase/PostgreSQL, Claude API and OpenAI API, Vitest, Playwright, GitHub Actions

Willy Agency project, Saas-Fee, Switzerland · [Tourism service](#)

- The owner built the product with AI agents in 71 days; it was in operation and close to its first sales. In one week in August 2026, directing AI agents, I:
- Set up reliable operation of the conversational assistant with the tourism service's knowledge base (hotels, routes, guests' service questions, slot booking).
 - Built tests from scratch: 1,310 tests in six layers, with no network, database or model; a full run takes 3.64 seconds and needs no secrets.
 - Audited the whole surface: route authorization, row-level data isolation (Supabase RLS to protect private data from prompt injection), incoming webhooks, secrets, environment separation; closed the critical issues and locked them in with tests that fail the build if a problem returns.

Device condition assessment app

Neatapps · NDA project for an international consumer electronics company · Flutter (iOS, Android), native iOS, Node.js, computer vision

Turnkey: interface design, a Flutter mobile app with native iOS modules, and a Node.js backend. Real-time computer vision detected the device's position in front of the camera and took photos itself, and an API returned the condition assessment from them.

Work experience

- Willy Agency – technical partner (contract)**
August 2026 – present · part-time · Saas-Fee, Switzerland, remote
- Full Stack / AI Developer (contract)**
April 2026 – present · Chişinău
 - Take AI-assisted prototypes to production: architecture, data, authorization, payments, tests of critical scenarios, release.
- Neatapps – Co-founder, Technical Project Lead**
July 2020 – March 2026 · clients in the US and Europe
 - Turned client goals into a technical scope of work: research, requirements, estimates, delivery plan, client communication.
 - Coordinated developers and designers across the whole cycle: architecture, development, testing, release, post-release support.
- Neatapps – Co-founder, Mobile Developer**
July 2015 – July 2020
 - Native Android (Java, Kotlin), then iOS (Swift) and cross-platform development in Flutter.

Skills

Frontend:	React, Next.js, TypeScript, Vite, Tailwind CSS, PWA (Service Worker, IndexedDB), TanStack Query.
Backend and data:	Node.js, NestJS, RESTful API, PostgreSQL, Supabase (RLS, RPC, migrations, pgvector), pgTAP.
AI and LLM:	Claude API (Anthropic SDK), OpenAI API, Gemini API, MCP (custom servers), RAG, Function Calling, structured outputs.
Architecture:	Solution Architecture, System Design, data security.
Cloud and DevOps:	Docker, Cloudflare Workers/Pages, Vercel, Netlify, GitHub Actions (CI/CD), Git, Linux.
Mobile development:	Flutter, Android (Java, Kotlin), iOS (Swift), BLE/IoT.
Quality and reliability:	Vitest, Testing Library, Playwright, pgTAP, axe, Sentry.
Used previously / familiar:	Java (Core, Spring), Python (FastAPI, SQLAlchemy), Redis, MongoDB, AWS, GCP, Nginx.