

PRATEEK BISHT

prateekbisht04@gmail.com | +91 7303202633 | github.com/prateekbisht23
linkedin.com/in/prateek-bisht-7a062a258 | prateekbisht.vercel.app

Software engineer who builds agentic systems that ship code. I build platforms where AI coding agents turn a requirement into a working, reviewed pull request, inside isolated sandboxes, on production-grade Go and TypeScript backends.

WORK EXPERIENCE

Software Engineer, Fyscal Technologies, Delhi, India

Nov 2025 – Present

- Built core parts of OneXO, an internal platform where AI coding agents (Claude Code / Codex / OpenCode) build prototypes from a written requirement in isolated sandboxes, using TypeScript/Bun microservices behind Kong. Every request ends as a reviewable pull request instead of a manual handoff.
- Built an in-app AI agent and a PlateJS editor using Mastra, AI SDK streaming and a read-only MCP server. Each AI suggestion is a change the user accepts or rejects, which cut manual edits by 80% with zero unreviewed writes.
- Led a POC automating customer support using AWS Bedrock LLMs, which cut manual query volume by 25%.
- Built a Go connector framework with dynamic provider onboarding and routing rules (weighted split, fallback, volume cap), which kept requests flowing when a provider failed.
- Built 20+ Go REST APIs for a microservices financial platform handling ~8M requests a month, with monitoring through Grafana, CloudWatch and Jaeger.
- Replaced REST with gRPC between 5+ core services, which cut cross-service latency from over 20 ms to under 2 ms (about 10x).

PROJECTS

Claude Code in microVMs (*Bun, TypeScript, React, AWS EC2, KVM, S3*)

[GitHub](#)

- Ran each chat conversation's Claude Code in its own KVM microVM using self-hosted CubeSandbox on AWS EC2. VMs start in about 0.3 s, so the agent can run any tool safely, with any damage limited to one throwaway VM.
- Kept each conversation's files and session on its own S3 volume, which let idle VMs be destroyed after 5 minutes without losing work, so 3–4 sessions could run at once on one box.
- Added per-user MCP setup with one-click OAuth (verified against 10 providers) and live streaming to the browser over an SSH tunnel, so teammates could run the full stack from their laptops.

CloudPad (*Django, Supabase, PostgreSQL*)

[GitHub](#)

- Built a real-time file and text sharing app with URL-based pads, using Supabase and Yjs for multi-user sync, which kept edits in sync across users in under 50 ms.
- Added per-pad access control and automatic expiry in PostgreSQL, which blocks unauthorized access and clears out stale data without manual cleanup.

plan-out (*Claude Code plugin, Python*)

[GitHub](#)

- Built a Claude skill that plans large features as small spec modules with a decision log, which stopped AI planning across multiple sessions from producing docs that contradict each other.
- Wrote a Python checker (standard library only) for broken links, stale decisions and duplicate definitions, plus a cleanup mode that checks no content was lost when old docs are migrated. Consistency no longer depends on the model's judgment.

Code Focus (*TypeScript, VS Code extension*)

[GitHub](#) | [Marketplace](#)

- Built and published a VS Code extension that shows a file's dependencies and dependents in one panel, which cut back-and-forth navigation in TypeScript and JavaScript codebases and has 40+ downloads on the Marketplace.
- Added right-click and command-palette entry points for .ts, .tsx, .js and .jsx files, so it works from wherever the developer already is.

OPEN SOURCE

- Supabase**: updated the import-data guide to match a dashboard UI change ([#38675](#))
- freeCodeCamp**: corrected role terminology in a curriculum user story ([#62312](#))
- Airbnb JavaScript Style Guide**: documented the ESLint --ext requirement for .jsx files ([#3153](#))
- Invoicely**: fixed unreadable optional-badge text in light mode ([#50](#)) and a wrong default status in the invoice status modal ([#56](#))

SKILLS

AI: LLM orchestration (Bifrost), AI coding agents (Claude Code, Codex, OpenCode), Mastra, AI SDK, MCP, AWS Bedrock

Languages: Go, TypeScript, Python, JavaScript

Backend: Microservices, REST, gRPC, Bun, Node.js, Kong, Django, Redis, PostgreSQL, ClickHouse

Infra and Observability: AWS (EC2, S3, Bedrock, CloudWatch), KVM microVMs, Docker, Grafana, OpenTelemetry, Jaeger

Frontend: React, TailwindCSS

EDUCATION

B.Tech, Computer Science and Engineering

2022 – 2026

Bhagwan Parshuram Institute of Technology, GGSIPU