Mehdi Bayat

Senior Software Engineer · Distributed & Real-Time Systems · Cloud & AI Engineering

Vienna, Austria · +43 676 560 0882 · mehdi.byt@gmail.com · linkedin.com/in/mehdibayat · github.com/matucs · mehdi-bayat.online

PROFESSIONAL SUMMARY

Ten years in. Mostly environments where latency is measured in milliseconds and downtime costs real money. Stock exchanges, live sports betting, port logistics, banking. Not the kind of domains where you get to "move fast and break things." Currently at Sportradar in Vienna, building a betting widget embedded inside live sports streams: React and TypeScript on the frontend, Node.js and NestJS on the backend, a lot of AWS. Lately I've been deep in distributed and event-driven systems too: Kafka, Redis, RabbitMQ, WebSockets. I've also been building out AI engineering work: a production RAG pipeline on AWS Bedrock with my team at Sportradar, and outside of work, AgentForge, a governed multi-agent engineering platform where a deterministic verification gate and a risk-based policy engine, not an LLM's judgment alone, decide what actually ships. I'm at my best chasing down hard production bugs, cleaning up systems that got messy under pressure, and building side projects that stress-test an architecture idea instead of just describing it. Full portfolio at mehdi-bayat.online.

TECHNICAL SKILLS

Frontend: React, Next.js (SSR/SSG/ISR), TypeScript, JavaScript (ES6+), Zustand, Redux, Angular, TanStack Query, SCSS, Tailwind, Styled Components, Storybook, Semantic-UI, Bootstrap

Backend: Node.js, NestJS, .NET Core, Web API, GraphQL, REST, gRPC, Microservices, CQRS, Event-driven Architecture, WebSockets, SignalR, OpenAPI, BFF pattern, API versioning

AI / ML: RAG (Retrieval-Augmented Generation), LLM Integration (AWS Bedrock, Anthropic & OpenAI APIs), LangGraph & Multi-Agent Orchestration, Vector Databases, Semantic Search, Prompt Engineering, Predictive Systems

Cloud & DevOps: AWS (EC2, S3, Lambda, ECS, Bedrock, Cognito, DynamoDB, SQS, SNS, CloudWatch, API Gateway), Docker, Kubernetes (K8s), Terraform, AWS CDK, GitLab CI/CD, Feature Flags, Canary Deployments

Messaging: RabbitMQ, Apache Kafka, AWS SQS/SNS, Message Broker patterns

Databases: PostgreSQL, MongoDB, SQL Server, MySQL, Redis, DynamoDB

Security: OAuth2, JWT, AWS Cognito, OWASP best practices, Secure SDLC, Data Encryption, RBAC

Performance: Lighthouse, Core Web Vitals (LCP/CLS/FID), React Profiler, Webpack Bundle Analyzer, Chrome DevTools, k6 / load testing, Web Workers

Code Quality: ESLint, Prettier, SonarQube, Husky, lint-staged, Code Coverage Enforcement, Architecture Decision Records (ADRs)

Testing: Jest, Vitest, Cypress, Playwright, E2E Testing, Integration Testing, TDD, Contract Testing

Observability & Debugging: Grafana, New Relic, Sentry, Datadog, CloudWatch, OpenTelemetry, Distributed Tracing, Structured Logging, SLO/SLA ownership, Incident Response, Postmortem Analysis

API & Dev Tools: Postman, Insomnia, Swagger, Webpack, Vite, Babel, Turborepo, Dependency Injection

Workflow: Git, Agile/Scrum, Kanban, Jira, Trunk-based Development, GitFlow, GitHub, GitLab, Bitbucket

PROFESSIONAL EXPERIENCE

Senior Software Engineer | **Sportradar** · Vienna, Austria — Oct 2022 – Present

- emBET is Sportradar's betting widget, embedded directly into live sports streams and sportsbook partner sites. Millions of concurrent users. During something like the Champions League final, the traffic spikes are genuinely wild, and the product still has to feel instant.
- I own the frontend. React, Zustand, TanStack Query, WebSockets. For a while our bundle was bloated, and I dug into it with Webpack Bundle Analyzer and React Profiler. Turned out we were pulling in libraries we barely used. Cut bundle size 30% through code splitting, tree shaking, and ditching a few heavy dependencies.
- On the backend I work across Node.js, NestJS, GraphQL, and some Go. Kafka and RabbitMQ handle the event bus between services. CQRS was worth the overhead once odds delivery and fan engagement needed to scale on different curves.
- I manage our AWS infra with Terraform and CDK. ECS, DynamoDB, SQS, SNS, Cognito, API Gateway, CloudWatch. We ship with canary deployments and feature flags so nothing is a big-bang release anymore.
- Last year I helped build our team's first production RAG pipeline using AWS Bedrock, a vector DB, and semantic search for contextual recommendations inside the platform. More moving parts than expected, especially around retrieval quality. Learned a lot. It works.

- We had a latency problem on the hottest betting endpoints under live-event load. About 180ms, which was unacceptable. Pulled it down to 40ms with Redis cache-aside in front of DynamoDB, tighter TTLs matched to odds update frequency, and switching from JSON to a more compact serialization format.
- I'm on-call. Sentry catches errors, Grafana and New Relic for everything else. When something serious happens, I write a postmortem. A few of those have turned into real architecture changes rather than just patching the symptom.
- ESLint, Prettier, SonarQube, Husky on pre-commit. Coverage thresholds baked into CI so quality can't silently erode. Cypress and Playwright for E2E on the critical flows.
- Security gets real attention here. OAuth2 and JWT through Cognito, RBAC across the API layer, OWASP checklist on the main surfaces. Sportsbook operates across multiple jurisdictions, so compliance requirements are strict and audited.
- A couple of junior engineers joined the team, and I helped them find their feet. Mostly pairing sessions. Distributed systems are genuinely hard to debug blind, and getting the mental model right early saves everyone time.
- GitLab CI, trunk-based development. Pipeline runs lint, unit tests, E2E, security scan, then deploys. Auto-rollback triggers if health checks don't pass post-deploy.

Senior Full Stack Developer | **TSETMC** · Tehran, Iran Aug 2020 – Oct 2022

- TSETMC is the tech backbone of Iran's national stock exchange. Everything financial runs through their systems. I worked on the real-time trading and market data platforms for two years. Zero-downtime requirement. When something breaks, it's not just inconvenient.
- Frontend was React, Redux, SignalR, WebSockets. The order book component was the hardest part: thousands of price ticks per second and the UI still had to feel responsive. Took real iteration to get there.
- Backend in .NET Core, REST, gRPC. I spent a lot of time with a profiler open. Eventually got p99 latency under 60ms on the main exchange-to-client data pipelines by clearing out blocking async calls and adding targeted DB indexing where it actually mattered.
- Dashboard performance was bad during peak trading hours. React Profiler pointed straight at the order book component. Virtualized the table, memoized the expensive derived calculations. About 40% faster render times during peak sessions. Worth it.
- The worst bugs I've ever debugged were here. Race conditions in concurrent WebSocket streams, state sync issues that only showed up under real trading load. You can't reproduce that in local dev. Structured logging and distributed tracing were the only things that helped.
- ESLint, Prettier, SonarQube across the whole codebase. Jest coverage above 85% on trading logic, and we kept it there. You don't want surprises in a zero-downtime environment.
- Financial regulatory requirements: JWT, RBAC, full audit logging, encrypted comms end to end. We went through formal compliance review. Passed.

Senior Front-End Developer | **Omid Computer Service** · Tehran, Iran Mar 2019 – Aug 2020

- CRM and CMS for a large bank. Account management, customer workflows, internal tools used daily by thousands of employees. High-sensitivity environment. A UI bug isn't just annoying; it blocks someone's job.
- I built out the Angular component library and documented it in Storybook so other teams could reuse components without pulling me in every time. Mostly worked. Component libraries are only as good as their documentation, and I learned that the hard way early on.
- Nailed down OpenAPI contracts with the Java backend team before anyone wrote a line of integration code. That decision alone saved weeks of back-and-forth. The WebSocket setup especially would've been a mess without it.
- Ran Lighthouse across the main user journeys. Found the usual stuff: render-blocking scripts, images with no lazy loading, no code splitting anywhere. Fixed all of it. Load times came down about 40%, and Core Web Vitals went from red to green across the board.
- Added ESLint, Prettier, and Husky to a codebase that had none of it. PRs before that were full of formatting noise. Small change, real improvement in review quality.
- Banking production issues are high pressure. Customer data, someone senior watching, pressure to fix fast. I got good at moving quickly through structured logs and DevTools without making things worse under stress.
- Scrum on some teams, Kanban on others depending on what we were building. I prefer being close to product and QA rather than just pulling tickets off a board.

Senior Full Stack Developer | **Rahyab Rayaneh Gostar** · Tehran, Iran Aug 2015 – Mar 2019

- GCOMS is the logistics system running Iran's 13 main maritime ports. Cargo tracking, customs clearance, manifests, port authority reporting. Won the eAsia Award for Trade Facilitation. I joined to rewrite the whole thing.
- Rewrote it from 2015 to 2018 in C#, .NET Core, Entity Framework, Web API, React, Angular, SQL Server. The genuinely hard part: the old system had to stay live across all 13 ports while the new one came online port by port. No big cutover. Incremental, messy, pulled off.
- I designed the API layer with versioning baked in from day one. 500+ operators, 13 ports, teams on different release schedules. You can't just ship a breaking change and ask everyone to keep up.

- Web, mobile, and desktop all in scope since port staff isn't all at desks. More cross-platform coordination than expected, but we got it done.
- Introduced code reviews and automated testing where there had been none before. Not the most popular decision at first. By the time we shipped, the team had bought in, and release quality was noticeably better.
- 24/7 operation. Production bugs don't wait. I built a hotfix and rollback process early in the project, before we needed it. Used it more than once.

Full Stack Developer & Co-Founder | **Healthcare Startup** · Tehran, Iran Aug 2014 – Mar 2016

- Co-founded a healthcare startup in Iran. Appointment booking, patient records, clinical workflow tools for private clinics. I was the only engineer for most of it, which was both exciting and exhausting.
- .NET, React, REST APIs. We moved fast, but I didn't cut corners on security. Healthcare data is sensitive, and the handling requirements don't go away just because you're a two-person team.
- Wrote ADRs and kept API docs current the whole way through. Solo engineer means if you don't write it down, it's gone. Learned that lesson early.
- Also dealt with product decisions, talking directly to clinic staff, figuring out what to actually build. Good education in everything that happens outside the code editor.

Full Stack Developer | **PressTV** · Tehran, Iran May 2014 – Aug 2015

- Broadcasting software for PressTV: desktop apps, mobile, embedded wall displays for a live news operation. Different world from web development, but low-level performance constraints were a good thing to learn early.
- C++ for real-time video and image processing off broadcast hardware. Frames had to land on large monitoring screens without lag. No browser runtime to blame when things went wrong.
- Picked up Android and iOS development here too, Java and Objective-C. First mobile work I'd done. Steep but useful.

PROJECTS

AgentForge | Governed Autonomous Software Engineering Platform github.com/matucs/agentForge

A multi-agent system (Planner, Architect, Researcher, Developer, Reviewer, QA, Security) orchestrated with LangGraph, where no agent's approval is final: a deterministic verification gate and a risk-based policy engine decide what actually merges. Proven with failure-injection demos where a real test failure and a real detected secret both override a simulated Reviewer approval.

LivePulse | Real-Time Sports Intelligence Platform github.com/matucs/LivePulse

An event-driven platform ingesting real live football data, detecting changes, and pushing real-time updates to the browser over WebSockets, backed by Kafka, Redis, and PostgreSQL. It also has a companion Architecture Lab and a Scaling Demo that measures the WebSocket fan-out design across two real instances instead of just claiming it works.

TalentMatch | Production-Oriented Job Matching Backend github.com/matucs/TalentMatch

A modular-monolith backend for publishing jobs and matching candidates: a synchronous Fastify API plus an asynchronous BullMQ worker, MongoDB as the source of truth, OpenSearch as a rebuildable read model, deterministic candidate scoring, idempotent applications, and a full AWS ECS/Fargate deployment configuration.

Full engineering write-ups, architecture decisions, and measured results for all three at mehdi-bayat.online.

EDUCATION

M.Sc. Computer Science, Intelligent Systems | **University of Tabriz** · Tabriz, #1 Ranked in Iran 2011 – 2013

- GPA 18.03/20, top of my cohort.
- Thesis was on polygon decomposition into convex parts. I improved the time and space complexity of an existing NP-complete solution in computational geometry and built a parallel version of the algorithm. Two papers came out of it, both presented at the National CSI Conference in 2014.

B.Sc. Applied Mathematics | **University of Esfahan** · Esfahan, #1 Ranked in Iran 2005 – 2010

PUBLICATIONS, AWARDS & COURSES

- "Approximate Convex Decomposition by IFACD Algorithm", 19th National CSI Computer Conference, Mar 2014
- "A Greedy Algorithm to Decompose Polygons into Uniform Polygons in 2D", 19th National CSI Computer Conference, Mar 2014
- eAsia Award for Trade Facilitation, 2011, awarded to the GCOMS port logistics platform
- Udemy course: JavaScript Problem Solving Code Challenges (Basic to Advanced)

LANGUAGES

English: C1, Professional Working Proficiency **German:** A2, Elementary