Mohamed Azahrioui

Software Developer at CodeHive | Backend Developer at Kojac | Computer Science Student, Leiden University

mohamedazahrioui2006@gmail.com | github.com/MoAz06 | LinkedIn | moaz06-dev.vercel.app | The Hague, NL

Profile

Early in a CS degree at Leiden, already working as a software developer at CodeHive, backend developer at Kojac, and freelancing for production clients on the side. My focus: deterministic, auditable guardrails for AI agents. I built two at hackathons. ReachGate asks whether a security vulnerability can actually be reached in the code. TrustGate decides whether an AI agent is allowed to take a risky action. Both let a transparent rule engine make the call and leave the AI to only explain it. Strong in Python, Node, APIs, React, Azure, and cloud.

Projects

ReachGate: vulnerability-reachability triage on GitLab Orbit GitLab Transcend Hackathon 2026

Python | GitLab Orbit graph API | BFS | GitLab CI/CD | OpenVEX | SARIF | Ed25519 | MCP | pytest

- Built a 14,000-line, 422-test tool that answers what scanners can't: can this vulnerability actually be reached from the app's entry points? It walks GitLab's code graph with a bounded BFS to find a real path. Around 80% of "critical" findings turn out to be noise.
- Made it refuse to lie. Fixed rules decide the verdict, the AI only writes the explanation. If a search runs out of budget it says UNKNOWN instead of faking a "safe", and a built-in test tries to sneak fake results past it and fails on purpose.
- Every verdict ships as proof you can check yourself: OpenVEX and SARIF exports, an sha256 manifest, an optional Ed25519 signature, and an offline verifier that needs no token and no network.
- One engine, three ways to run it: a CLI, a GitLab CI bot that comments on merge requests without spamming duplicates, and a GitLab Duo AI agent over the Orbit MCP server.

TrustGate: runtime authorization gate for AI agents Google Cloud Rapid Agent Hackathon 2026

Node.js | Google Cloud Run | Vertex AI/Gemini | BigQuery | Fivetran | MCP | React

- Built a firewall that sits between an AI agent and risky actions like issuing a refund. Before the action runs, it checks live business evidence and answers ALLOW, APPROVAL_REQUIRED, or BLOCK.
- Same idea as ReachGate, different domain: a deterministic rule engine makes the call, the model only proposes. Every decision comes with an auditable receipt instead of a black-box score.
- Pulled live evidence from Fivetran and BigQuery, checked data freshness against SLAs and schema compliance, and exposed it over an MCP endpoint on Google Cloud Run with a small React control-room dashboard.

Procurement Risk API 2025 – 2026

FastAPI | PostgreSQL | SQLAlchemy | JWT | OAuth2 | bcrypt | Swagger/OpenAPI

- REST API for invoice risk: upload CSVs, flag risky invoices, pull analytics, read auto-generated docs.
- PostgreSQL and SQLAlchemy, with every route locked down by JWT, OAuth2 Bearer tokens, and bcrypt.
- Risk rules for duplicates, threshold breaches, weekend transactions, and severity summaries.

Flask real-time chat 2025 – 2026

Flask | Flask-SocketIO | WebSockets

- Real-time chat over WebSockets with session login and an MVC layout.

Experience

Software Developer (part-time) June 2026 – present

CodeHive

- Worked on the MHPoly AI knowledge base, a RAG application where users can ask questions over uploaded documents.
- Integrated Azure AI Search, Azure Blob Storage, Azure OpenAI, and App Service configuration to get the RAG pipeline working end-to-end.
- Added generated answers with source references, relevant passages, and highlighted supporting text in the frontend.
- Validated the flow on the live Azure environment using the Noord-Holland report as a test document.

Backend Developer (part-time) April 2026 – present

Kojac

- Build backend and some frontend on live client products with a senior team. Real production code, daily reviews, while studying full-time.
- Wire data between services, APIs, and the frontend so it lines up on every device.

Backend Developer (freelance) 2025 – present

External clients

- Built a Python tool that pulls bol.com orders into SQLite and generates DPD shipping labels and Excel exports. Killed the duplicate exports and manual rework the client was doing by hand.
- Kept live and test data strictly apart so mock orders never leaked into real shipments. Zero contamination across the whole pipeline.
- Built the auth for an invoice platform: JWT and OAuth2 on FastAPI and PostgreSQL, with Swagger docs so the frontend dev never had to ask me how an endpoint worked.
- Wrote the risk rules for a fraud-detection platform: duplicate invoices, odd thresholds, weekend transactions. Turned a vague "flag the sketchy stuff" into real SQL.
- Shipped Python data pipelines and some C++ work across projects.

Education

BSc Computer Science 2025 – present

Leiden University

Coursework: algorithms and data structures, system programming (C++), databases, Unix/Linux tools.

Built a C++ scheduler using backtracking with pruning for a graded project, cutting the search space down hard. The same instinct came up later in ReachGate's graph walk.

Skills

Languages: Python, JavaScript/Node.js, C++, SQL, TypeScript

AI & security: AI-agent guardrails, RAG systems, MCP, Vertex AI/Gemini, deterministic policy engines, SAST triage, reachability analysis, OpenVEX, SARIF, Ed25519 signing

Backend & APIs: FastAPI, Flask, Node.js, REST, WebSockets, JWT, OAuth2, bcrypt, OpenAPI/Swagger

Frontend: React, HTML, CSS, responsive layouts

Cloud & data: Azure App Service, Azure AI Search, Azure Blob Storage, Azure OpenAI, Google Cloud (Cloud Run, BigQuery), Docker, PostgreSQL, SQLite, SQLAlchemy, Fivetran, CSV/Excel pipelines

Dev & infra: Git, GitLab CI/CD, Railway, Linux/WSL, Bash, pytest, respx, Python packaging

Spoken: Dutch (native), English (native)

Certifications

Software Engineer, HackerRank, 2026. Problem Solving, SQL, REST API.
Verify: https://www.hackerrank.com/certificates/aa7ff827587a

REST API (Intermediate), HackerRank, 2026.
Verify: https://www.hackerrank.com/certificates/cbd881b1a866