

ASAD AMAD SHEIKH

Full Stack Engineer · C# / ASP.NET Core · TypeScript / React · PostgreSQL · Payments Integrations

asadamad91@gmail.com | +92 339 011 5848 | Islamabad, Pakistan (UTC+5) | Remote, available immediately

[linkedin.com/in/asadamadsheikh](https://www.linkedin.com/in/asadamadsheikh) | github.com/asadsheikh91 | asadamadsh.me | parchivisa.app

SUMMARY

Full stack engineer who ships products end to end: API backends in C#/ASP.NET Core, FastAPI and Express, PostgreSQL schemas, React and TypeScript frontends, and the infrastructure underneath. Recently built a payments backend in .NET 10 around the failure cases that decide whether money moves correctly: idempotency, signed webhook ingestion and ambiguous PSP outcomes. Also runs a live AI SaaS designed, built and deployed alone in six weeks. Earlier internships in government fleet telemetry and production telecom data. Looking for remote contract or full-time work with startups building payments, integrations or data-heavy backends; working hours that cover the full US East Coast day and afternoons in Europe.

FEATURED PROJECT

Cascade: Payments Backend Migration (Laravel to .NET)

2026, personal project

github.com/asadsheikh91/cascade-payments | C# · ASP.NET Core (.NET 10) · EF Core · Npgsql · PostgreSQL 17 · Stripe.net · Shopify APIs · xUnit · Testcontainers · Docker · GitHub Actions

- Built a payment state machine (charges, partial and full refunds) with PSP adapters behind a common interface, including a Stripe test-mode integration.
- **Ambiguous PSP outcomes:** a timeout moves a payment to a requires-verification state instead of being treated as a decline. Each attempt is saved before the PSP call and its id is sent as the PSP idempotency key, so a retry or later reroute cannot double-charge.
- **Shopify webhook ingestion** that acknowledges within Shopify's 5-second limit: HMAC-SHA256 checked over the raw request body; deduplication on webhook id (retries) and event id (across subscriptions); a transactional inbox drained with FOR UPDATE SKIP LOCKED, with retries and a dead-letter state; an order projection that ignores out-of-order events; and a scheduled GraphQL Admin API fetch that recovers missed webhooks.
- **Correctness at the database level:** idempotency keys enforced by unique indexes (tested with concurrent requests), optimistic concurrency on Postgres xmin, and CHECK constraints that make over-refunds impossible even if application code is bypassed.
- 45 xUnit tests, including integration tests against a real PostgreSQL started by Testcontainers. CI runs the tests, starts the full Docker Compose stack and smoke-tests it. Design choices recorded in Architecture Decision Records.
- **In progress:** Adyen adapter with soft-decline cascade routing, double-entry ledger with settlement reconciliation, React + TypeScript dashboard, and YARP shadow traffic comparing the legacy Laravel system with the new one.

EXPERIENCE

Founder & Full Stack Engineer | ParchiVisa

2026 to Present

Next.js · TypeScript · FastAPI · PostgreSQL · Redis · Clerk · Cloudflare R2 · Playwright · AWS Lightsail · Sentry | parchivisa.app

- Sole engineer on a live visa-assessment product: FastAPI service, Postgres schema, Next.js frontend, and the infrastructure it runs on. Validated the problem through user conversations and a scoring prototype, then shipped to production in six weeks.
- Built four country-specific eligibility engines (UK, Australia, Canada, USA). Replaced proportional scoring with tiered band-capping after finding that proportional weights let a strong financial profile mask a disqualifying academic gap; any single hard blocker now fails a candidate regardless of aggregate score.
- Report pipeline runs retrieval over a structured corpus of official immigration requirements into LLM narrative synthesis, with eligibility computed deterministically so a model failure cannot change a verdict.
- One React report template rendered to both web and PDF via headless Playwright, so the two surfaces cannot diverge as the report changes.
- Operates the service on AWS Lightsail behind Cloudflare. Intake, four-engine scoring, PDF report and payment are live end to end on a single deploy with no manual steps.

Backend Engineering Intern | Frontier Works Organization

Jun to Aug 2025

Python · PostgreSQL · REST APIs · IoT telemetry · Flotilla fleet platform

- Built the data exchange layer between PostgreSQL and the Flotilla fleet platform, moving real-time telemetry across government fleet systems covering over 300 vehicles.
- Replaced manual batch reporting with automated aggregation pipelines, cutting reporting latency from 30 minutes to under 1 minute.
- Traced production timeouts under peak concurrent load to unbounded queries and connection pool exhaustion; rewrote the queries and reconfigured pooling, eliminating the timeouts at the same peak load.

- Optimised relational schemas across production telecom infrastructure handling high-volume SME datasets, covering 80 tables.
- Found and fixed schema inefficiencies causing silent data loss in multi-departmental compliance pipelines, across a compliance dataset of roughly 670 GB.

OTHER PROJECTS

InfluencePay | ERC-20 escrow marketplace (client project)

github.com/asadsheikh91/influence

Solidity 0.8.24 · Hardhat · OpenZeppelin · ethers.js v6 · Next.js 14 · Express · Prisma · PostgreSQL · Clerk

- Modelled escrow as an on-chain state machine (fund, lock against milestone, release on acceptance, explicit dispute path) with a tier-based fee schedule applied on release. Built end to end for a client: contracts, Express API and Next.js frontend, delivered and accepted.

Veriloom | Skills-first hiring platform

github.com/asadsheikh91/VeriLoom

Next.js · TypeScript · Express.js · Prisma · PostgreSQL · Clerk · Cloudflare R2 · LLM API

- Candidates submit real deliverables scored by an LLM pass, then a follow-up interview keyed to their own submission, so outsourced work fails the second gate. Submissions are isolated per organisation on Cloudflare R2 behind Clerk auth.

VehicleWatch | Fleet telemetry & predictive maintenance

github.com/asadsheikh91/vehicle-watch

FastAPI · PostgreSQL · Redis · scikit-learn · Docker · Render

- Ingests sensor readings every 2 seconds and flags faults with an Isolation Forest and Local Outlier Factor ensemble, holding sub-100ms scoring latency on a single container.

TECHNICAL SKILLS

Languages: C#, TypeScript / JavaScript, Python, SQL, Solidity, Java

Backend: ASP.NET Core (Minimal APIs), EF Core, Npgsql, Node.js / Express, FastAPI, Prisma, SQLAlchemy, Pydantic v2, background workers, REST / OpenAPI design

Frontend: React, Next.js 14, TypeScript, PDF rendering via headless Playwright

Data: PostgreSQL (unique indexes, CHECK constraints, optimistic concurrency, SKIP LOCKED queues), Redis, Alembic, schema design, query optimisation

Payments & integrations: Stripe (Stripe.net), Shopify Webhooks and GraphQL Admin API, idempotency keys, webhook signature verification, transactional inbox with dead-lettering

AI / ML: LLM integration (Claude, Gemini), retrieval over structured corpora, structured output, evaluation sets, scikit-learn

Infrastructure: Docker, Docker Compose, GitHub Actions, Nginx, Gunicorn, DigitalOcean, AWS (EC2 / RDS / S3), AWS Lightsail, Cloudflare (R2, DNS), Render, Linux

Security & Auth: Clerk, JWT, RBAC, session handling, rate limiting, runtime secret injection, structured audit logging

Testing: xUnit, Testcontainers, pytest, httpx, schemathesis contract tests, Locust load testing, ruff and mypy in CI

EDUCATION & ACHIEVEMENTS

B.S. Computer Science | FAST NUCES, Islamabad

2022 to 2026 (graduated June 2026)

- Certificate of Appreciation, Frontier Works Organization (Aug 2025): sole recipient among a cohort of 45 interns.
- Selected for League of Launchers S2, National Incubation Centre Islamabad (ParchiVisa).
- FastAPI & Backend Development, Packt Publishing.