SRIDHAR MAHALINGAM

Software Developer · Python / Django · React & Next.js · TypeScript · Flutter

+974 3145 2430 | sridharansridhar22@gmail.com | LinkedIn | GitHub | Portfolio

Target Role: Software Developer · Currently in Doha, Qatar · Open to relocation to the United Kingdom · Requires Skilled Worker sponsorship

PROFESSIONAL SUMMARY

Software Developer with 3+ years of experience building and shipping production web and mobile applications end to end, currently serving as Senior Backend Developer at Holora Performance in Doha — from database schema and REST API design through to responsive frontends, Linux deployment, security hardening and post-launch support. Specialises in Python/Django REST Framework backends paired with React and Next.js frontends, with hands-on delivery of Flutter, React Native and native Android applications. Has independently owned six live client platforms across UK and Qatar markets, covering Stripe payment integration, AWS cloud services, real-time features via WebSockets and Celery, JWT and role-based access control, and VPS administration. Comfortable as the sole technical owner of a product and experienced in direct English-language requirements gathering with international stakeholders. Additionally leads in-house machine learning development — training OCR and computer-vision models internally on self-hosted infrastructure without third-party AI services — and has architected an offline-first Tauri desktop ERP that runs without a hosted server. Seeking a Software Developer role in the United Kingdom.

TECHNICAL SKILLS

Languages Python, TypeScript, JavaScript (ES6+), Dart, Kotlin, C#, Rust (Tauri), SQL, HTML5, CSS3

Backend Django, Django REST Framework, Django Channels, Celery, Flask, RESTful API design, JWT & OAuth authentication, WebSockets, Role-Based Access Control, Redis task queues

Frontend React.js, Next.js 14 (App Router, SSR/ISR), React Router, Redux, Tailwind CSS, Bootstrap, responsive & mobile-first UI, Core Web Vitals optimisation

Mobile Flutter (Riverpod, Freezed, Dio, GoRouter, Hive, Secure Storage), React Native, Native Android (Kotlin 2.0, Jetpack Compose, Material 3, Hilt, Room, DataStore, WorkManager, VpnService)

Architecture Clean Architecture, ports and adapters, modular monolith, metadata-driven system design, event sourcing, multi-tenancy with row-level security and query-layer isolation, business rules engines, double-entry ledger design, offline-first / local-first systems, deterministic and explainable pipeline design.

Machine Learning PyTorch, LangGraph multi-agent orchestration, RAG (pgvector, FAISS, hybrid BM25 retrieval), Ollama local LLM inference, FCOS object detection, ONNX Runtime, computer vision, custom model architecture and training, dataset preparation and annotation, evaluation and benchmarking (IoU precision/recall, mAP), OCR model development, Tesseract, self-hosted GPU inference — built without third-party AI services or external inference APIs.

Desktop Tauri (Rust shell + web frontend), C# / .NET 10 with WPF, offline-first / local-first architecture, on-device persistence, peer synchronisation, Windows packaging and installer builds (Inno Setup).

Databases PostgreSQL, MySQL, Redis, SQLite — schema design, indexing, query optimisation, migrations, backups

Cloud & DevOps AWS (S3, SES), Docker, Nginx, Ubuntu Linux VPS administration, Contabo, Vercel, Render, HTTPS/TLS configuration, Git & GitHub

Integrations Stripe Payment Gateway, Firebase Cloud Messaging, Google Maps Platform, AWS Simple Email Service, third-party REST APIs

Testing & Tooling Postman, JUnit, MockK, LeakCanary, Timber, microbenchmarking, Django test framework, code review

Systems & Networking DNS protocol and resolution, DNS-over-HTTPS / DNS-over-TLS, Android VpnService and TUN packet processing, local DNS server implementation, Bloom filters, Patricia tries, LFU caching, Manifest V3 browser extension development.

Practices Agile delivery, requirements analysis, technical documentation, performance profiling, debugging, security-first implementation

SEO & Web Performance (Supporting Skill Set)

- On-page search engine optimisation — meta tags, structured metadata, semantic markup, dynamic sitemap generation and robots.txt configuration.
- Google Analytics and Google Search Console setup, indexing diagnostics and crawl-efficiency improvements.

- Website performance and page-speed optimisation, including server-side rendering strategy and asset delivery via CDN/object storage.
- Content structure optimisation for organic search ranking, plus social media integration support.

PROFESSIONAL EXPERIENCE

Senior Backend Developer — Holora Performance, Doha, Qatar | *Feb 2026 – Present*

Lead backend engineering across a concurrent portfolio of seven products spanning e-commerce, offline-first desktop ERP, a no-code metadata-driven enterprise platform, in-house document intelligence and fitness technology — owning API architecture, data modelling, model development, security and deployment.

- **Own backend architecture across seven concurrent products** — designing Django REST Framework services, database schemas, authentication and authorisation layers, and integration contracts consumed by web, mobile and desktop clients.
- **Lead in-house machine learning development** for OCR and document intelligence — all models trained internally on self-hosted infrastructure, with no third-party AI provider or external inference API anywhere in the pipeline.
- **Architected an offline-first desktop ERP** using Tauri, replacing hosted-server dependency with local device storage and peer synchronisation on reconnection.
- **Design and optimise PostgreSQL data models** for multi-module enterprise systems, covering schema design, indexing strategy, migration management and query performance tuning under production load.
- **Build mobile-facing API layers** consumed by Flutter and React Native applications, with JWT authentication, token refresh, role-based access control and versioned REST contracts.
- **Implement asynchronous and background processing** using Celery and Redis for scheduled jobs, notification dispatch, document processing pipelines and long-running tasks.
- **Own deployment and production operations** — Linux server administration, Nginx configuration, HTTPS/TLS, environment separation, database backups and post-release support.

Projects

E-Commerce Web Platform

Tools: Python (Django REST Framework), PostgreSQL, Redis, React.js / Next.js, Celery

Backend platform powering a full online retail experience — catalogue, cart, checkout, order lifecycle and administrative operations — built as a REST API consumed by the web storefront and an internal admin interface.

Key Features

- **Catalogue & Inventory Services —** product, category, variant and stock modelling with API endpoints supporting search, filtering and pagination at scale.
- **Order Lifecycle Management —** cart, checkout, order state transitions, fulfilment tracking and administrative override workflows exposed through secured REST endpoints.
- **Authentication & Access Control —** JWT-based authentication with role separation between customers, staff and administrators.
- **Asynchronous Processing —** Celery and Redis handle order confirmation emails, notification dispatch and scheduled reporting jobs.

E-Commerce Mobile Application

Tools: Flutter (Dart), Python (Django REST Framework), PostgreSQL, Firebase Cloud Messaging

Mobile commerce application delivering the shopping experience natively on Android and iOS, backed by a shared REST API layer and push-notification infrastructure.

Key Features

- **Mobile-Optimised API Layer —** endpoints designed for mobile constraints — payload minimisation, pagination and offline-tolerant synchronisation patterns.
- **Cross-Platform Delivery —** single Flutter codebase producing consistent Android and iOS builds with shared state management and routing.
- **Push Notification Infrastructure —** Firebase Cloud Messaging integration for order updates, promotional campaigns and transactional alerts.

- **Secure Session Handling —** token storage, automatic refresh and session invalidation aligned with the backend authentication model.

Construction ERP — Offline-First Windows Desktop Application

Tools: Tauri (Rust shell + web frontend), REST framework, local on-device storage, peer synchronisation

Platform: Native Windows desktop application — no hosted server dependency

Enterprise resource planning system for construction operations, delivered as a native Windows desktop application rather than a hosted web service. All records persist to local device storage, allowing the application to operate fully offline; when connectivity becomes available, data synchronises to other authorised users' installations — removing the need for an always-on central server along with its hosting cost and single point of failure.

Key Features

- **Local-First Data Persistence —** complete application functionality without network access; all records written to and read from local device storage, so site and field users remain fully productive with no connectivity.
- **Peer Synchronisation on Reconnection —** on regaining connectivity, changes propagate to other users' installations, keeping distributed copies consistent without a permanently hosted backend.
- **Native Desktop Packaging with Tauri —** Rust-based application shell paired with a web-technology frontend, producing a lightweight Windows binary with a substantially smaller footprint and memory profile than Electron-based equivalents.
- **REST Interface Layer —** internal REST contracts decouple the interface from the data layer, so the same API surface serves both local operation and synchronised multi-user operation.
- **Modular ERP Domains —** discrete modules covering project, resource, procurement and reporting workflows, sharing a common permissions model.
- **Conflict Reconciliation —** handling of concurrent edits made offline by multiple users, resolving divergent local state on synchronisation.

S POS — Multi-Platform Retail Point of Sale (Web, Desktop, Mobile)

Role: Independent product — sole architect and developer

Tools: Python, Django 5.2, Django REST Framework, Celery, Redis, PostgreSQL, SQLite, React 19, React Native (Expo), PyInstaller + WebView2, waitress, Docker, Nginx

Status: Complete — full retail suite operational across web, desktop and mobile clients

A complete, multi-region, multi-currency point-of-sale and back-office platform for shops of any size — one Django 5.2 + DRF backend exposing a single /api/v1 contract (JWT auth, idempotent writes) that three clients speak identically: a React 19 back-office SPA, an Expo/React Native mobile app and a native Windows desktop application. The desktop build packages the entire backend into one SPOS.exe with SQLite inside — one PC becomes the shop's own server, with tablets, phones and extra registers connecting over Wi-Fi — while the mobile app can run the entire store on a phone with no server at all. Designed and built solo, from the transaction engine and double-entry ledger through to the deployment stack.

Key Features

- **Server-authoritative checkout** — every client computes cart totals locally for instant UI, but the server recomputes the full invoice with identical rounding, tax and discount rules and refuses the sale on any mismatch; sale creation, row-locked stock decrement, per-item COGS capture and double-entry ledger posting happen in one database transaction.
- **Idempotent, auditable transactions** — checkout is idempotent at HTTP and business-transaction levels (the same Idempotency-Key can never produce a duplicate charge), invoice numbers come from a locked per-store sequence, failed checkouts persist a FAILED record, and void/refund reverse both inventory and accounting.
- **Full register lifecycle** — open with float, paid-in/paid-out cash movements, X report mid-shift, Z report and drawer reconciliation against the ledger at close, with cash refunds reducing register cash expectations and closing restricted to the cashier or a manager.
- **Deep retail back-office** — product variants and modifiers, server-side promotions and customer price lists, gift cards and store credit with server-enforced balances, loyalty, suppliers and purchase orders with partial receiving, inter-store transfers, stock counts and an expense approve/pay/void workflow across multi-store, multi-company setups, each company with its own currency.
- **Three deployment shapes from one codebase** — a hosted SaaS on Docker (PostgreSQL, Redis, Celery behind Nginx/TLS), a self-contained SPOS.exe (PyInstaller) that migrates SQLite, picks a free port and serves the API and web UI to the whole LAN via waitress inside a native WebView2 window, and a mobile offline mode with a complete on-phone store engine.

- **Security throughout** — HMAC-SHA256-signed payment webhooks with idempotency, tenant scoping on money paths, refresh-token revocation, discount caps, admin-gated account creation and a first-run setup wizard with no default password.

AI Research Agent — Multi-Agent Research System with Verified Citations

Role: Independent project — sole architect and developer

Tools: Python, LangGraph, FastAPI, Pydantic v2, Ollama (local LLM), PostgreSQL + pgvector, FAISS, sentence-transformers, Next.js 16, TypeScript, Docker Compose

Status: Complete — 38 tests passing, benchmarked end to end on real research runs

A local-first, multi-agent AI research workstation. LangGraph-orchestrated agents plan the research, search the real web, ground every finding in retrieved evidence, criticise their own analysis and write cited research reports on a fully local LLM — with every citation verified against the session's own sources before publishing. Everything runs locally except web search: local LLM via Ollama, local embeddings, local database. Designed and built solo, from the agent state machine and RAG pipeline through to the FastAPI backend and the Next.js research-workstation UI.

Key Features

- **Multi-agent verification loop** — a Critic agent checks the analysis against the retrieved evidence and loops the workflow back to research with refined queries when evidence is insufficient (hard-capped at two iterations); the report cannot be written until the Critic approves.
- **Structural hallucination reduction** — findings citing pages that were never fetched are discarded deterministically, schema-constrained JSON decoding means the model cannot emit malformed output, and honest errors replace fabricated results when sources are unavailable.
- **Claim-level citation verification** — every factual sentence in the report is verified as supported, partially supported or unsupported against the session's own evidence; invalid citations are stripped and weak ones disclosed — measured citation coverage rose from 0% to 60–86%.
- **Hybrid RAG retrieval** — a per-session vector store (pgvector → FAISS → numpy auto-selection) fuses dense embeddings with a BM25 keyword index via reciprocal rank fusion, with full provenance on every chunk.
- **Live research workstation** — the FastAPI backend streams agent progress over Server-Sent Events to a Next.js UI with source cards, evidence quotes, confidence bars, clickable citations, agent traces and per-stage timing.
- **Honest evaluation built in** — a benchmark harness runs real research sessions and scores citation coverage, groundedness, completeness, retrieval relevance and latency per model and mode — 100% completion and 78–79% overall quality measured, with quick mode optimised from ~120s to 64s.

TrafficVision — AI Traffic Intelligence Platform

Role: Independent research project — sole architect and developer

Tools: Python, PyTorch, FCOS / ONNX Runtime, Django REST Framework, Celery, Redis, PostgreSQL, Next.js

Status: Active development — detection model in training

A computer-vision platform that ingests traffic video and turns it into structured, real-time intelligence about vehicles and road conditions. It models the full traffic domain — from a city down to individual lanes, cameras, regions-of-interest and counting lines — and runs a GPU-managed processing engine that detects vehicles frame-by-frame and feeds the results into downstream analytics. Designed and built solo as a research project, from the video-ingestion and detection pipeline through to the operations backend and web dashboard.

Key Features

- **Traffic-scene modelling —** represents the physical network (city → intersection → road → lane), camera placements, ROIs and virtual counting lines, giving every detection real spatial meaning rather than raw pixel boxes.
- **Vehicle detection engine —** a self-trained FCOS detector, exported to ONNX for fast inference, identifies and localises vehicles per frame, with train-equals-serve preprocessing so results stay consistent from training to production.
- **GPU-managed processing sessions —** a Celery-driven engine runs video through the model off the request path, with GPU allocation, heartbeats and automatic crash recovery, so long-running analysis never blocks the API.
- **End-to-end training pipeline —** dataset preparation, leakage-safe train/validation splits, multi-day interruptible training with checkpoint and resume, and honest evaluation (IoU precision/recall, mAP) on held-out data — all on self-owned infrastructure.

- **Structured, validated output —** detections become typed, confidence-scored records in a canonical contract, ready for the counting, tracking, congestion and signal-optimisation modules on the roadmap.
- **Built for scale and rigour —** modular-monolith architecture with clean interfaces for future RTSP/CCTV ingestion and multi-GPU support.

CommerceOS — Business-Agnostic Commerce Platform

Role: Independent project — sole architect and developer

Tools: Python 3.12, Django 5.2, Django REST Framework, Django Channels, Celery, Redis, PostgreSQL, Next.js 15, TypeScript, TanStack Query, Docker

Status: Active development — multi-tenant foundation, catalogue, storefront and admin complete; global-market pricing engine in progress

A multi-tenant, headless commerce platform whose backend is deliberately vertical-neutral — the same engine powers a fashion boutique, a grocery, a pharmacy or a B2B wholesaler, with each store's identity expressed as data and configuration rather than hardcoded logic. It models the full commerce domain — from tenants and catalogues down to products, variants, dynamic attributes and global-market pricing — behind a single versioned API that any frontend can consume. Designed and built solo, from the Clean Architecture backend and adapter subsystem through to the Next.js storefront and admin dashboard.

Key Features

- **Business-agnostic domain modelling —** represents the commerce network (tenant → catalogue → category → product → variant) with dynamic attributes and pluggable product-type strategies, so a new vertical is a configuration change rather than a code fork — never a business-type conditional branch anywhere in the backend.
- **Multi-tenant isolation engine —** every tenant-scoped row carries a tenant identifier and isolation is enforced at the query layer, so independent stores run on one deployment without ever seeing each other's data.
- **Catalogue engine owned end to end —** products, a full variant engine (options → values with an enforced generation ceiling), dynamic attributes, brands with verification, aliases and merge, collections, tags, media references, and SEO / JSON-LD / sitemap generation.
- **Global-market pricing engine —** models market as distinct from currency, with explicit locked prices, FX reference-currency fallback and a deterministic resolver that never borrows between markets, fed by a batched, N+1-safe read path folding price into product detail and variants.
- **Ports and adapters from day one —** every external integration — payment, email, storage, search, notification — sits behind an abstract base service with swappable adapters (Stripe/PayPal, SMTP/SES, Local/S3, Postgres-FTS/OpenSearch) resolved per tenant, so the core depends only on interfaces.
- **API-first, replaceable UI —** a standard response envelope, OpenAPI schema and versioned REST contract mean the Next.js storefront and admin can be redesigned or replaced with zero backend change.
- **Built for scale and rigour —** Clean Architecture with CI-enforced dependency boundaries via import-linter, strict typing under mypy with zero suppressions, UUIDv7 keys, feature flags and formal per-phase governance through a constitution, ADRs and validation reports.

Airsume — Fully-Offline Resume Intelligence & ATS Platform

Role: Independent research project — sole architect and developer

Tools: Python, Django 5.2, Django REST Framework, Celery, Redis, PostgreSQL, PyMuPDF / pdfplumber / python-docx, Tesseract OCR, RapidFuzz, Next.js 16, React 19, TypeScript, Tailwind v4, TanStack Query, Zustand

Status: Active development

A resume-analysis and ATS platform that turns unstructured resumes and job descriptions into structured, explainable hiring intelligence — without any LLM or cloud AI. Every result is produced by deterministic, auditable logic, so each parsed field and score traces back to the exact text and rule that produced it. It models the full document domain — resumes, job descriptions, skills and their relationships — and runs an asynchronous parsing-and-scoring engine that extracts typed, confidence-scored data and grades resume-to-job fit against a versioned rubric. Designed and built solo end to end: the extraction pipeline, the skill taxonomy, the scoring engine, the benchmark harness and the web dashboard.

Key Features

- **Deterministic, explainable parsing —** a fully offline engine (PyMuPDF, pdfplumber, python-docx) handles two-column layouts, tables and scanned or image PDFs via Tesseract OCR, emitting every field as a value, confidence, method and source-span envelope with a strict no-guess threshold — it never fabricates data.

- **Versioned skill taxonomy —** 432 skills across 33 categories and 10 industries with typed aliases and weighted relationships, matched by a staged offline matcher (exact → alias → normalised → fuzzy, never fuzzy-first) that records exactly how each match was made.
- **Document-agnostic framework —** resumes and job descriptions run through one shared plugin-based parsing pipeline, with industry derived deterministically from the taxonomy rather than guessed.
- **Explainable ATS scoring engine —** grades resume-to-job fit across six weighted categories with budgets summing to 100, using taxonomy-grounded skill matching, a neutralise-and-redistribute strategy for non-applicable criteria and a reconciliation invariant — each score ships with the human-readable reasons behind it.
- **Asynchronous processing off the request path —** Celery and Redis run parsing and scoring as background jobs (uploaded → parsing → parsed or failed) with live status polling in the interface, so long documents never block the API.
- **Honest evaluation built in —** a golden-dataset benchmark harness with a synthetic document generator, tolerant per-field comparison and regression-versus-baseline checks keeps accuracy measurable and regressions visible.
- **Production-grade foundation —** JWT authentication with a custom email user model, versioned documents, SHA-256 upload deduplication, audit logging and activity feeds, OpenAPI documentation and a full Next.js dashboard.

MeetingMind AI — Fully-Offline, Local-First Meeting Intelligence Platform

Role: Independent research project — sole architect and developer

Tools: Python, Django 5, Django REST Framework, Celery, Redis, PostgreSQL, Faster-Whisper, Ollama (Llama 3.2), SpeechBrain ECAPA, FFmpeg, sentence / Nomic embeddings, PostgreSQL full-text and vector search, yt-dlp / feedparser, Next.js 16, React 19, TypeScript, Tailwind v4, TanStack Query, Zustand

Status: Active development

A from-scratch meeting-assistant SaaS that turns raw audio and video into transcripts, grounded summaries, tracked work and organisational knowledge — running entirely locally with no paid APIs, cloud AI or commercial SDKs. Every AI capability (speech-to-text, summarisation, chat, embeddings, diarisation) sits behind a provider interface defaulting to a local engine, so the product works end to end after a clone and setup, yet can swap in optional cloud providers by configuration alone. Designed and built solo across 15 phases — the Clean Architecture backend, the background processing engine, the local AI stack, the knowledge and executive-intelligence layers, a multi-agent platform and the full Next.js dashboard.

Key Features

- **Local speech-to-text and media pipeline —** an asynchronous Celery/Redis pipeline (validate → media-inspect → extract → normalise → transcribe → diarise → summarise → index) transcribes audio and video offline via Faster-Whisper and FFmpeg, emitting timestamped, editable transcripts in TXT, MD, SRT, VTT and JSON with graceful degradation, so a failed AI step never discards a good transcript.
- **Grounded, cited local AI —** Ollama running Llama 3.2 produces executive and detailed summaries, action items, decisions, risks and keywords in a single versioned-prompt inference; a RAG chat answers questions over one meeting or the whole organisation using hybrid retrieval (local embeddings, Postgres full-text and recency), returning clickable-timestamp citations and refusing to answer beyond the source material — no hallucinated facts.
- **Reusable background-job engine —** a domain-agnostic job and pipeline platform with self-registering stages, dependency-graph topological ordering, per-stage retry with exponential backoff, idempotent resume, cancellation, live progress and an in-process event bus, running all long work off the request path and powering transcription, AI, imports and exports alike.
- **Human-in-the-loop workspace —** AI never auto-creates work; it emits confidence-scored suggestions with full explainability — source meeting, segment, speaker, quote and reason — that a user approves, edits or rejects to materialise real tasks (Kanban), issues, decisions and risks, each carrying an immutable audit trail back to its evidence.
- **Bitemporal organisational knowledge hub —** an append-only, event-sourced knowledge index — versioned, with time-travel "what did we know as of…" queries, decision-evolution history, consensus tracking and conflict detection — powers cross-meeting search, insights, recommendations and an executive dashboard whose health, score and analytics are materialised incrementally (around 14 ms dashboard reads) rather than recomputed per request.
- **Multi-agent intelligence platform —** 12 declarative, owner-scoped agents access data only through a permissioned tool registry, coordinated by a planner (intent → agent selection by reputation → parallel execution → merge → conflict resolution) and a collaboration engine offering review, vote, debate and hand-off workflow templates, with sandbox previews and human approval gates throughout.
- **Speaker diarisation and cross-meeting voice identity —** segments are attributed to first-class speaker entities via SpeechBrain ECAPA embeddings and clustering, with an AI name-suggestion step that is suggest-only and never auto-

applied, plus a cross-meeting voice registry recognising recurring speakers through tiered similarity matching — reusing embeddings persisted at processing time so nothing is ever reprocessed.

- **Universal media import —** a provider-based ingestion layer imports YouTube and Vimeo links, direct media URLs and podcast/RSS feeds with episode pickers and batch fan-out straight into the existing pipeline via yt-dlp and feedparser, guarded by SSRF protection and three-layer deduplication, with zero duplicated AI logic.
- **Reproducible evaluation built in —** a diarisation benchmarking framework with public and user-verified datasets, a settings-free re-clustering harness sweeping tuning thresholds against stored embeddings without re-transcribing, honest ground-truth labelling distinguishing verified from approximate, and provenance-stamped runs for regression tracking.

No-Code Enterprise ERP Platform — Metadata-Driven Application Builder

Tools: Python (Django REST Framework), Next.js 14 (App Router), TypeScript, PostgreSQL with Row-Level Security, Redis, Celery

Positioning: Configurable enterprise platform in the category addressed by Odoo and Salesforce

A no-code, metadata-driven enterprise operating system that allows organisations to define their own data models, workflows and business logic through configuration rather than code. Entities, fields, relationships and rules are stored as metadata and interpreted at runtime, so a new business application can be assembled and changed without a code deployment.

Key Features

- **Metadata-Driven Model Layer —** entities, fields, relationships and validation rules defined as metadata and interpreted at runtime; new modules are created through configuration rather than schema migration and redeployment.
- **Event-Sourced Architecture —** state changes captured as an immutable event stream, providing complete audit history and point-in-time reconstruction of any record.
- **Business Rules Engine —** a core architectural pillar: conditional logic, triggers, computed fields and process automation authored by administrators without developer involvement.
- **Multi-Tenant Isolation via PostgreSQL Row-Level Security —** tenant separation enforced at the database layer rather than in application code, removing an entire class of cross-tenant data-leak risk.
- **Approval Matrix & SLA Engine —** configurable multi-stage approval chains and service-level tracking with automated escalation on breach.
- **Double-Entry Accounting Ledger —** a genuine double-entry ledger underpinning the financial modules, maintaining balanced books across all transactional activity.
- **Webhook & API Builders —** administrators expose custom REST endpoints and configure outbound webhooks through the interface, enabling third-party integration without engineering work.
- **External Portal Builder & White-Labelling —** customer- and vendor-facing portals plus per-tenant branding configured entirely without code.

OCR & Document Intelligence — In-House Model Development

Tools: Python, PyTorch, self-trained recognition models, Django REST Framework, Celery, Redis, PostgreSQL

Status: Active research and development

Document intelligence capability built entirely on internally trained models, with no dependency on third-party AI services or external inference APIs at any stage of the pipeline. Scope covers dataset preparation, model training and evaluation, and the serving layer that exposes structured extraction results to downstream business systems.

Key Features

- **Fully In-House Model Stack —** all recognition and extraction models trained internally; no external AI provider or hosted inference API sits anywhere in the pipeline, keeping the capability under owned infrastructure and free of per-call vendor cost.
- **Data Sovereignty & Confidentiality —** source documents never leave controlled infrastructure, addressing privacy and regulatory requirements that rule out third-party document processing.
- **Training & Evaluation Pipeline —** dataset preparation, annotation workflow, training runs and accuracy benchmarking against held-out evaluation sets.
- **Asynchronous Serving Layer —** Celery workers execute inference off the request path, keeping API latency independent of model processing time.
- **Structured Extraction & Validation —** parsing of recognised text into typed, validated fields, with confidence thresholds, retry logic and a human review queue for low-confidence results.

Trainer & Fitness Platform (Web)

Tools: Python (Django REST Framework), PostgreSQL, Redis, React.js, Celery

Dual-sided web platform connecting fitness trainers and clients, covering trainer profiles, programme delivery, scheduling and client progress tracking through a unified backend API.

Key Features

- **Dual-Role Domain Model —** separate trainer and client roles with distinct permissions, dashboards and data visibility rules.
- **Programme & Scheduling Services —** backend support for training programme creation, assignment, session scheduling and calendar management.
- **Progress Tracking —** structured capture and retrieval of client metrics and session history for longitudinal reporting.
- **Media Handling —** storage and delivery pipeline for training content, with access control tied to programme entitlement.

Confidential Mobile Application

Tools: Flutter (Dart), Python (Django REST Framework), PostgreSQL

Note: Client and product details withheld under non-disclosure agreement

Cross-platform mobile application delivered under NDA. Responsible for backend architecture, API design, data modelling and integration with the Flutter client. Further detail available in interview to the extent permitted by the agreement.

Previous Role

Full Stack Developer — Techynova, Chennai, India | *Jan 2023 – Feb 2026*

Technology consultancy delivering web and mobile products for clients in the United Kingdom, Qatar and India. Owned the full delivery lifecycle — frontend, backend, deployment, security and direct client communication in English — across five production platforms.

- **Delivered five production platforms end to end** — architecture, backend APIs, frontend, deployment and security — as the primary or sole engineer on each, working within teams of three to four.
- **Built and secured payment and transactional flows** using Stripe, achieving PCI-compliant encrypted checkout, automated invoice generation and order-confirmation email pipelines via AWS SES.
- **Administered production Linux infrastructure** on Contabo VPS and Render — Nginx reverse proxying, HTTPS/TLS certificates, JWT-protected API routes, environment isolation and database backups.
- **Improved organic reach for client sites** through dynamic sitemap generation, structured metadata, robots.txt configuration and Next.js server-side rendering, alongside Google Analytics and Search Console setup.
- **Ran client-facing delivery in English** with UK and Qatar stakeholders — requirements gathering, scope negotiation, demos and post-launch support — without an intermediary account manager.

Client Deliveries

Plugged In Scents — E-Commerce Platform

Client: UK-based fragrance brand

Tools: React.js (admin & user frontend), Python (Django backend), MySQL, Stripe Payment Gateway, AWS S3

Operating System: Linux

Team Size: 3

Developed a secure e-commerce platform for a UK-based fragrance brand, featuring a custom-built admin panel on the frontend. The admin dashboard allows complete control over products, orders, users and website data, while customers interact through a separate user-facing storefront.

Key Features

- **Frontend Admin Dashboard —** a dedicated React-based admin panel where administrators can add, edit, delete and manage products, categories, pricing, inventory, users and orders, including full order status tracking and data verification.
- **Order & Customer Management —** administrators view complete order details, customer information and payment status, and update order workflows directly from the frontend dashboard.
- **Secure Checkout & Payments —** integrated Stripe payment gateway ensuring encrypted, PCI-compliant online transactions.
- **Image Storage with AWS S3 —** product images securely stored and served via AWS S3, enabling scalable and fast media delivery while reducing origin server load.

- **Backend & API Integration —** Python Django backend exposing secure REST APIs handling authentication, role-based access, cart logic, orders and admin operations.
- **Database & Hosting —** MySQL database hosted on Contabo VPS, providing reliable and optimised data storage.
- **Modern Responsive UI —** built using React.js to deliver fast, responsive and user-friendly interfaces for both administrators and customers.
- **Linux-Based Deployment & Security —** deployed on a Linux server with JWT authentication, protected admin routes, secured APIs and HTTPS configuration.

Domain: www.pluggedinscents.co.uk

My Role: *Full Stack Developer — responsible for React admin dashboard development, Django REST API design, Stripe integration, AWS S3 configuration, Linux server deployment, security implementation and direct coordination with the UK client in English.*

Seven Stars Stationery — E-Commerce Platform

Client: Doha, Qatar

Tools: Next.js, Python (Django backend), MySQL

Operating System: Linux

Developed an e-commerce platform for Seven Stars Stationery, a Doha-based store specialising in stationery, office supplies and tech gadgets for students, professionals and businesses. The platform extends the store's reach beyond physical limitations, enabling customers to browse and purchase 24/7 and significantly improving visibility, convenience and revenue potential.

Key Features

- **Product Management System —** administrators efficiently add, update and manage products with images, detailed descriptions, category classification and pricing through a user-friendly Django Admin interface.
- **Cart & Secure Checkout Process —** users add products to the cart, update quantities and complete purchases through a streamlined checkout flow supporting both Cash on Delivery and online payment.
- **Automated Invoice & Order Confirmation —** on successful order placement the system generates a downloadable invoice and sends confirmation emails containing order details to both customer and administrator.
- **REST API Integration —** the Next.js frontend communicates securely with the Django backend through RESTful APIs managing product listings, cart operations and order processing workflows.
- **High-Performance Responsive UI —** developed using Next.js to deliver fast-loading, SEO-optimised and fully responsive product pages across desktop and mobile devices.
- **SEO Optimisation & Sitemap Configuration —** dynamic sitemap generation and properly configured robots.txt enhance search engine crawling, indexing efficiency and overall online visibility.
- **Linux-Based VPS Deployment —** deployed on a Contabo Linux virtual private server, ensuring reliable performance, server-level control and scalability for production traffic.

Domain: www.sevenstars.qa

My Role: *Full Stack Developer — responsible for Next.js frontend development, Django REST API design, payment and invoice integration, SEO and sitemap configuration, Linux VPS deployment and direct coordination with the Qatar-based client in English.*

Indiguard Security — Service Company Platform

Client: UK-based security services provider

Tools: React.js, Python (Django backend), MySQL, Tailwind CSS

Operating System: Windows / Linux

Engineered a scalable web solution for Indiguard Security, a UK-based provider of manned guarding, CCTV monitoring and keyholding services. Built using React.js and Django, the platform features dynamic service modules, secure contact workflows and SEO-optimised content.

Key Features

- **Modern Full-Stack Architecture —** developed using React.js for the frontend and Django for the backend, ensuring a modular, scalable and service-oriented structure.
- **Service Management Modules —** security services such as CCTV monitoring, mobile patrol, key holding and sector-specific solutions are structured as dynamic Django models and displayed through reusable React components, letting the client add services without a code change.

- **Efficient Routing & Navigation —** frontend navigation handled with React Router for smooth transitions, while Django manages secure REST API endpoints, admin operations and form processing.
- **Custom Admin Dashboard —** an enhanced Django Admin panel enables administrators to manage service listings, client inquiries and operational content efficiently.
- **Responsive User Interface —** mobile-first UI developed using Tailwind CSS ensures fast loading, consistent design and a seamless user experience across all devices.
- **Contact & Quotation System —** secure inquiry and quotation request forms with backend validation and automated email notifications using Django's mailing functionality.
- **Cloud Deployment & Performance Optimisation —** React frontend deployed on Vercel (serverless infrastructure, edge caching), while the Django backend is hosted on Render with automated build pipelines and persistent storage.

Domain: www.indiguardsecurity.co.uk

My Role: *Full Stack Developer — contributed to frontend development, backend API integration and deployment, coordinating in English with stakeholders to align the platform with business goals.*

NH LiveSpace — Construction & Interior Business Platform

Tools: React.js, Python (Django backend), JavaScript, HTML, CSS

Operating System: Linux

Developed NH LiveSpace, a professional business website for a licensed construction and interior solutions company recognised for quality craftsmanship and client satisfaction. The platform establishes a strong digital presence to showcase construction services, architectural expertise and completed projects while enabling smooth client engagement.

Key Features

- **Service & Portfolio Showcase —** dedicated sections highlighting construction services, interior solutions, architectural capabilities and completed projects to build trust and credibility.
- **Client Inquiry & Contact System —** integrated secure contact forms allowing customers to submit project requirements and inquiries, managed through the Django backend.
- **Modern Responsive UI —** developed using React.js to ensure fast-loading, mobile-friendly and visually appealing interfaces across all devices.
- **Backend & API Integration —** Python Django backend handles form submissions, validations and business data securely through REST APIs.
- **SEO-Optimised Structure —** optimised page content, metadata and structure to improve online visibility and reach potential construction clients organically.
- **Linux-Based Deployment —** deployed on a Linux server, ensuring high performance, reliability and scalability for business growth.

Domain: www.nhlivespace.com

My Role: *Full Stack Developer — contributed to frontend development, backend API integration and deployment, coordinating in English with stakeholders to align the platform with business goals.*

Techynova — Business Website & Application Development Platform

Tools: React.js, Python (Django), JavaScript, HTML, CSS

Operating System: Linux

Developed Techynova, a business-focused technology website offering website development, web application development and digital solutions for clients across industries. The platform represents the company's own startup initiative, providing a professional online presence to showcase services, attract clients and manage business inquiries.

Key Features

- **Service Showcase —** clearly structured service pages for website development, web applications and custom software solutions.
- **Client Inquiry System —** secure contact and inquiry forms allowing potential clients to submit project requirements, processed through the Python backend.
- **Modern Responsive UI —** built using React.js to deliver fast, responsive and user-friendly interfaces for desktop and mobile.

- **Backend Integration —** Python (Django) backend handles form submissions, validations and business data management through secure APIs.
- **SEO-Friendly Structure —** optimised page structure and metadata to improve search engine visibility.
- **Linux-Based Deployment —** hosted on a Linux server, ensuring stable performance, security and scalability.

Domain: www.techynova.tech

My Role: *Full Stack Developer — contributed to frontend development, backend integration and deployment, communicating directly in English with clients to understand requirements and deliver solutions.*

KEY ENGINEERING PROJECTS

Medical ERP System (SridharERP)

Tools: Next.js 14, TypeScript, React.js, Tailwind CSS, Python (Django REST Framework), PostgreSQL, Redis, WebSockets, JWT Authentication

A comprehensive full-stack healthcare management platform developed to digitise and streamline hospital operations through a centralised system. It integrates patient management, electronic medical records (EMR), pharmacy, laboratory, nursing, ward management, billing, staff administration and analytics into a single enterprise solution, delivering secure, scalable, real-time healthcare workflows for modern hospitals and clinics.

Key Features

- **Patient Management —** centralised patient registration, demographic information, medical history, visit tracking and profile management.
- **Electronic Medical Records (EMR) —** maintains clinical notes, diagnoses, prescriptions, patient vitals and complete treatment history in digital format.
- **Pharmacy Management —** drug inventory management, medication dispensing, stock monitoring and automated low-stock alerts.
- **Laboratory Management —** supports lab test orders, sample tracking, result generation and real-time delivery of laboratory reports.
- **Nursing & Ward Management —** handles nursing tasks, shift assignments, patient monitoring, bed allocation, ward occupancy tracking and patient transfers.
- **Billing & Finance —** generates invoices, manages payments, tracks insurance claims and automates financial record management.
- **Staff Management —** maintains employee records, scheduling, role assignments and workforce administration.
- **Analytics Dashboard —** real-time KPIs, operational reports, revenue analysis, occupancy statistics and hospital performance insights.
- **JWT Authentication & RBAC —** secure JWT-based authentication with role-based access control across eight distinct roles: doctors, nurses, pharmacists, lab technicians, accountants, receptionists and administrators.
- **Real-Time Notifications —** WebSockets deliver instant updates including lab results, pharmacy stock alerts, emergency notifications and bed occupancy changes.
- **Database Integration —** PostgreSQL and Redis ensure secure, reliable and scalable healthcare data management, with Redis caching for read-heavy dashboards.

Repository: github.com/Sridhar08-glitch/Medical--ERP

ShieldDNS v2.0 — Cross-Platform Ad, Tracker & Malware Blocking Ecosystem

Scope: Three independent applications — Android VPN filter, Windows desktop DNS server and browser extension — designed, built and released end to end as sole developer

Android: Kotlin 2.0.21, Jetpack Compose, Material 3, MVVM + Clean Architecture, Hilt, Room, DataStore, WorkManager, Android VpnService + AIDL, OkHttp/Retrofit, Coroutines + Flow, kotlinx.serialization

Windows: C# / .NET 10, WPF, local DNS server on 127.0.0.1:53, DNS caching, system resolver redirection, Inno Setup installer

Browser Extension: JavaScript, Manifest V3, declarativeNetRequest, content scripts, webNavigation — Chrome, Edge, Firefox and Brave

License: MIT — open source

A self-hosted, privacy-first blocking ecosystem that filters ads, trackers, malware and redirect scams at three complementary layers: inside the browser, across every Windows application, and system-wide on Android without root. No accounts, no

telemetry and no data leaving the device — all filtering happens locally. The unifying principle is that an ad cannot load if its domain never resolves.

Key Features

- **Five-Layer Blocking Pipeline —** domains resolve through an ordered, fastest-first pipeline: session whitelist, LFU hot cache, session blacklist, Bloom-filter prefilter, then an authoritative Patricia trie match. Delivers microsecond lookups against millions of blocklist entries by making the common case nearly free.
- **CNAME Uncloaking —** inspects every hop of the CNAME chain rather than only the queried hostname, catching first-party-disguised trackers that domain-only blockers miss entirely.
- **Behavioural Reputation Scoring —** assigns each domain a live 0–100 score derived from subdomain entropy, risky TLDs, homoglyph look-alike detection and visit frequency. Scores decay over time so a single anomalous spike never permanently blocks a legitimate site; domains are quarantined or auto-blocked against configurable thresholds.
- **Redirect-Chain Reconstruction —** rebuilds multi-hop redirect chains from a rolling DNS buffer, scoring them by hop count, /24 IP-block diversity and inter-hop timing to distinguish genuine ad-redirect chains from ordinary page loads.
- **Tiered Shield Levels —** Normal, Aggressive and Nuclear presets trade false-positive rate against detection aggressiveness, each enabling a different combination of static lists, chain auto-blocking, reputation thresholds and burst detection.
- **Encrypted Upstream Resolution —** clean queries forwarded over DNS-over-HTTPS and DNS-over-TLS to Cloudflare, Quad9, AdGuard, Mullvad or Google, with bootstrap IPs compiled into the build to avoid a chicken-and-egg resolution loop before the tunnel is established.
- **Per-App Firewall —** individual Android applications blocked independently across Wi-Fi, mobile data and roaming, enforced at the DNS layer.
- **Root-Free System-Wide Coverage —** Android VpnService loopback tunnel intercepts DNS queries only; user traffic is never routed off-device or through any third party.
- **Windows Local DNS Server —** .NET 10 WPF application running a resolver on 127.0.0.1:53, automatically redirecting the system DNS on start and restoring it on stop, filtering games, telemetry and every browser simultaneously.
- **Manifest V3 Browser Extension —** declarativeNetRequest network blocking, CSS element hiding, popup-overlay removal via keyword and size heuristics, and redirect-tab closure that distinguishes user-initiated navigation from script-opened tabs.
- **Automated Blocklist Maintenance —** eight community blocklists integrated (EasyList, EasyPrivacy, uBlock Filters, OISD, StevenBlack, Hagezi Threat Intel, URLhaus, AdGuard Popups), refreshed in the background via WorkManager with boot-receiver persistence and gzip-compressed offline bundling.
- **Release Engineering —** R8 code and resource shrinking, certificate pinning enabled in release builds, separate debug and release application IDs and signing configuration, a Gradle task for blocklist asset bundling, and unit testing with JUnit and MockK alongside LeakCanary leak detection.

Repository: github.com/Sridhar08-glitch/Sheild-DNS

CarWash Booking Application

Tools: Flutter, Dart, Riverpod, Freezed, Dio, Go Router, Hive, Flutter Secure Storage, Stripe, Google Maps, Firebase Cloud Messaging, WebSockets, Django REST Framework, PostgreSQL, Redis, Celery, Django Channels, JWT Authentication

A full-stack mobile platform developed to streamline vehicle service scheduling, booking management, customer engagement and operational workflows for car wash businesses. Customers can book services, track appointments in real time, manage vehicles, purchase memberships, earn loyalty rewards and make secure online payments. Built with Flutter and Django REST Framework for scalable, secure, real-time service management.

Key Features

- **Online Service Booking —** customers browse wash packages, select service locations, choose preferred time slots and schedule appointments seamlessly.
- **Real-Time Booking Tracking —** live service tracking using WebSockets and Google Maps integration lets customers monitor appointment progress and status in real time.
- **OTP Authentication & JWT Security —** passwordless OTP login with JWT-based authentication and automatic token refresh for secure access.
- **Vehicle Management —** users register, update and manage multiple vehicles along with service history and booking records.

- **Membership & Subscription Plans —** supports premium memberships, subscription packages, recurring services and exclusive customer benefits.
- **Loyalty & Referral Programme —** customers earn reward points, loyalty tiers and referral bonuses redeemable for discounts and promotions.
- **Secure Payment Processing —** integrated Stripe Payment Gateway for secure payments, subscription billing and transaction management.
- **Address & Location Management —** saved addresses, geolocation services, reverse geocoding and route-based service delivery.
- **Product Store & Shopping Cart —** built-in e-commerce functionality for purchasing vehicle care products and accessories.
- **Push Notifications & Alerts —** Firebase Cloud Messaging provides booking confirmations, reminders, promotional offers and status updates.
- **Staff & Operations Management —** administrators manage bookings, employees, attendance, schedules and daily operations via a centralised dashboard.
- **Analytics & Reporting —** revenue reports, booking statistics, customer insights, service utilisation metrics and operational performance tracking.
- **Background Task Processing —** Celery and Redis handle scheduled jobs, notifications, payment processing and automated system tasks, keeping request latency independent of background workload.
- **Database Integration —** customer, booking, vehicle, payment and membership data securely stored and managed using PostgreSQL.

Repository: github.com/Sridhar08-glitch/Carwash-Booking-App

Student–Teacher Management Mobile Application

Tools: React Native, JavaScript, Python (Django REST Framework), MySQL, REST API

A cross-platform mobile solution developed to digitalise academic administration and improve communication between teachers and students. It provides secure role-based access with separate dashboards for teachers and students, enabling efficient management of attendance, academic records, class information and notifications, with real-time data synchronisation.

Key Features

- **Secure Role-Based Authentication —** students and teachers log in securely with separate access permissions and personalised dashboards.
- **Teacher Management Dashboard —** teachers add and manage student records, create classes and subjects, record attendance, update marks and send notifications.
- **Student Dashboard —** students view attendance status, academic performance and subject details, and receive notifications in real time.
- **Attendance & Marks Management —** digital attendance tracking and marks entry reduces manual workload and ensures accurate records.
- **Notification System —** teachers send instant updates and academic alerts to students through in-app notifications.
- **Real-Time Backend Integration —** the React Native app communicates with Django REST APIs to fetch and update academic data efficiently.
- **Cross-Platform Mobile Experience —** built with React Native for smooth performance and consistent UI across Android and iOS.
- **Database & Data Security —** academic data securely stored and managed using MySQL for reliability and scalability.

Frontend Repository: github.com/Sridhar08-glitch/Student-management-Mobile-APP-frontend

Backend Repository: github.com/Sridhar08-glitch/Student-management-Mobile-APP-backend

Inventory Management System

Tools: React.js, HTML5, CSS3, JavaScript, Python (Django), MySQL

A full-stack web application developed to manage product inventory, monitor stock movement, track seller activities and analyse sales performance. It provides real-time inventory updates, secure authentication and analytics dashboards that help organisations streamline operations and make data-driven decisions.

Key Features

- **Product & Stock Management —** administrators add, update, delete and monitor product details including quantity, pricing and categories, with stock levels auto-updated from sales and transactions.
- **Seller Management —** maintains comprehensive records of sellers, including listed products, performance metrics and transaction history.
- **Sales Management —** tracks daily, monthly and overall sales with detailed order information for better revenue monitoring.
- **Analytics Dashboard —** displays revenue trends, top-selling products, stock availability and seller performance through graphical reports.
- **JWT-Based Authentication —** secure login and authorisation using JSON Web Tokens to protect sessions and API access.
- **Low Stock Alerts —** automatic notifications generated when inventory reaches minimum threshold levels.
- **Advanced Search & Filtering —** quick searching and filtering of products, sellers and sales records.
- **Database Integration —** all inventory and sales data securely stored and managed using MySQL.

Frontend Repository: github.com/Sridhar08-glitch/Inventory-Frontend

Backend Repository: github.com/Sridhar08-glitch/Inventory-Backend

People Safety & Crime Reporting System

Tools: HTML5, CSS3, JavaScript, Python (Django), MySQL, WebSockets

Team Size: 4

A web application designed to enhance public safety by providing a platform for users to report crimes, receive alerts and access important security features. The system integrates real-time crime reporting, communication and emergency assistance tools to ensure a safer environment for communities.

Key Features

- **Crime Reporting —** users report crimes by selecting their location on an interactive map, with both auto-detection and manual selection.
- **Crime Reports & Alerts —** users view reported crimes in their area and receive notifications about nearby incidents.
- **Live Chat & Notifications —** a WebSocket-based chat system enables real-time communication between users and authorities, plus crime-related and general alerts.
- **SOS Feature —** users send an emergency alert to their saved contacts in case of danger.
- **Danger Zones —** administrators define high-crime areas on the map, categorised into risk levels, helping users avoid unsafe locations.
- **Safe Places & Nearby Police —** provides information about nearby police stations and designated safe places for users in distress.
- **Interactive Map Integration —** a map API displays crime reports, danger zones and essential safety locations.

Repository: github.com/Sridhar08-glitch/Feel-safe

Smart Cafeteria Management System

Tools: HTML, CSS, Bootstrap, JavaScript, Python (Flask), SQLite

A full-stack web application built using Flask to streamline cafeteria operations for both administrators and users, covering menu management, order processing and nutritional awareness.

Key Features

- **User Authentication —** secure login system for both users and administrators.
- **Menu Management —** administrators add, update or remove food items with details such as name, price and nutrition.
- **Order Processing —** users place orders and receive real-time feedback with automated email notifications.
- **Nutritional Tracking —** integrated nutrition awareness by tracking calories and food types.
- **Responsive UI —** designed using Bootstrap with user-friendly navigation and functionality.

Repository: github.com/Sridhar08-glitch/SMART-CAFETERIA

EDUCATION

B.Tech, Petrochemical Technology — Anna University, India | 2022 | CGPA 7.61 / 10

CERTIFICATIONS

- Full Stack Development with Python — IBM Certification, SLA Institute, Chennai
- Codeathon Event Project — IBM Certification, SLA Institute, Chennai

PROFESSIONAL SKILLS

- Client communication and requirement analysis
- Full-stack web and mobile application development
- Independent project execution and end-to-end delivery
- Problem solving, debugging and performance optimisation
- REST API design, integration and backend development
- Responsive UI development with a user-experience focus
- Agile development and team collaboration
- Technical documentation and knowledge transfer

ADDITIONAL INFORMATION

Target Role Software Developer (full-stack, backend or frontend focus) — permanent, UK-based.

Work Authorisation Requires Skilled Worker visa sponsorship; degree certificate and transcripts available on request.

Availability Currently employed — notice period to be confirmed on offer.

Languages English (professional working proficiency — primary language of all client delivery), Tamil (native).

Strengths Sole-owner product delivery, direct stakeholder communication, rapid onboarding to unfamiliar stacks, debugging and performance optimisation, security-first implementation.

References Available on request.