NISHANT RAJ

Comprehensive Professional Profile & Technical Portfolio

Indian Institute of Technology Kharagpur

Final Year | Class of 2026

+91 9709294505 | nishant2004@kgpian.iitkgp.ac.in

LinkedIn | GitHub

TABLE OF CONTENTS

1. Personal Profile & Education

1.1 About Me

I am Nishant Raj, a final-year B.Tech student at the Indian Institute of Technology Kharagpur, one of India's premier engineering institutions. I am pursuing a 4-year degree in Physics with a CGPA of 8.6/10. Over the course of my undergraduate journey, I have developed deep expertise in full-stack software development, cloud infrastructure, AI/ML integration, and system design. I have completed three professional internships (all remote, Canada-based companies) where I single-handedly architected, developed, and deployed production-grade platforms from the ground up. I have also completed an advanced B.Tech Project (BTP) on deep learning-based flood detection under Prof. Bhabagrahi Sahoo at IIT Kharagpur.

1.2 Education

Year	Degree / Examination	Institution	CGPA / Marks
2022-2026	Physics (4-Year)	Indian Institute of Technology Kharagpur	8.6/10
2022	Class XII (CBSE)	Guru Gobind Singh Public School	86.6%
2020	Class X (CBSE)	DAV Public School	96.6%

1.3 Contact Information

Phone: +91 9709294505

Email (IIT KGP): nishant2004@kgpian.iitkgp.ac.in

Email (Personal): nkmysign1990@gmail.com

Location: IIT Kharagpur, India

LinkedIn: linkedin.com/in/nishantraj

GitHub: github.com/nishantraj

2. Technical Skills & Competencies

2.1 Programming Languages

Language	Proficiency	Key Usage Areas
TypeScript	Expert	Primary language across all 3 internships; 126K+ lines at TasklyApp alone
JavaScript	Expert	Full-stack development, Node.js backends, React frontends
Python	Advanced	Deep learning (PyTorch), FastAPI, data processing, BTP project
C++	Advanced	DSA (700+ problems solved), competitive programming
C	Intermediate	Systems programming, coursework
SQL	Advanced	PostgreSQL, complex queries, migrations, RLS policies
HTML/CSS	Expert	Semantic HTML, Tailwind CSS, responsive design

2.2 Frontend Technologies

Technology	Version	Experience
React	18/19	Server Components, Suspense, hooks, context API, 131+ components built
Next.js	14/15/16	App Router, SSR, SSG, ISR, API Routes, middleware, Turbopack
Tailwind CSS	4	Custom themes, responsive design, dark/light mode
Radix UI / shadcn/ui	Latest	25+ accessible UI primitives
Framer Motion	Latest	Page transitions, scroll animations, micro-interactions
WebSockets / WebRTC	-	Real-time communication, live updates, pub/sub
TanStack Query	5	Server state management, caching, optimistic updates
React Hook Form + Zod	Latest	Form management with schema validation

2.3 Backend Technologies

Technology	Version	Experience
Node.js	20+	Production backends, microservices, REST APIs
NestJS	11	18-module enterprise app with 100+ services at TasklyApp
Express	5	REST APIs, middleware chains,

		WebSocket servers
FastAPI	-	Python backend for flood detection platform
BunJS	-	Modern JS runtime usage
GraphQL	-	Query language for APIs

2.4 Databases & ORMs

Technology	Experience
PostgreSQL	16-table schemas with RLS, migrations, triggers, indexes, Supabase
MongoDB	64+ schemas with Mongoose ODM, aggregation pipelines
Redis	Distributed caching, rate limiting, LRU caching (10K entries)
Supabase	Auth, Realtime (WAL-based CDC), Storage, Row Level Security
Drizzle ORM	Type-safe SQL with migrations via Drizzle Kit
Prisma	Type-safe ORM
Mongoose	MongoDB ODM with TypeScript decorators

2.5 DevOps & Cloud Infrastructure

Technology	Experience
Docker	Multi-stage builds, containerization, Docker Compose
Kubernetes	Container orchestration
GitHub Actions	CI/CD pipelines, automated builds and deployments
Google Cloud Platform	Cloud Run, Cloud Build, GCS, Secret Manager, Container Registry
AWS	S3 for file storage, general AWS services
Vercel	Auto-deploy, edge functions, analytics
Linux	Server administration, shell scripting
Nginx	Reverse proxy, load balancing
Terraform (IaC)	Infrastructure as Code
Sentry	Error monitoring with source maps and performance tracing

2.6 AI/ML & Data Science

Technology	Experience
PyTorch	Deep learning models (ResNet-50 UNet, Swin Transformer, MaxViT)
OpenAI GPT-4o	CV analysis with vision, document parsing, text generation
Google Gemini	Job analysis, interview questions, image generation
Groq (LLaMA)	Fast LLM inference, real-time processing
Ensemble Models	Multi-model prediction with uncertainty quantification
Computer Vision	Satellite image segmentation, flood detection,

	OpenCV, GDAL
Google Earth Engine	Satellite data retrieval and processing
Streamlit	Interactive ML web applications

2.7 Authentication & Security

- JWT authentication (access + refresh tokens) with role-based access control
- OAuth 2.0 (Google) via Passport.js, NextAuth.js, and Supabase Auth
- PKCE OAuth 2.0 flow with Zod validation
- Row Level Security (RLS) on PostgreSQL with JWT role claims
- Two-Factor Authentication (TOTP via Speakeasy)
- OWASP security headers (HSTS, X-Frame-Options, XSS-Protection, CSP)
- Stripe webhook signature verification
- Input sanitization, NoSQL injection prevention, DOMPurify
- Rate limiting (IP-based and Redis-backed distributed)
- Bcrypt password hashing with salt rounds

2.8 Payment Systems

- Stripe Payment Intents with manual capture (pre-authorization model)
- Stripe Connect Express for automated washer payouts
- Stripe Subscriptions with 3-tier plans and credit-based access control
- Stripe Checkout Sessions (embedded mode)
- Webhook-based payment confirmation and reconciliation
- Dynamic server-side pricing engine (6 vehicle types x 10 dirt multipliers)

2.9 Tools & Utilities

Git, GitHub, Figma, VS Code, Leaflet.js, Jupyter Notebook, Swagger/OpenAPI, Postman, Google Maps API, Twilio, Resend, Nodemailer

3. Internship 1: TasklyApp.ai

AI-Powered Job Intelligence & Interview Preparation Platform

Detail	Information
Role	Full-Stack Developer Intern (Software Development Engineer)
Company	TasklyApp.ai
Location	Remote, Canada
Duration	May 2025 - July 2025
Tech Stack	NestJS 11 + Next.js 14 + MongoDB + Redis + AI/ML
Deployment	Google Cloud Run (Docker)
Codebase Scale	126,000+ lines of TypeScript, 707 files

3.1 Executive Summary

During my internship at TasklyApp.ai, I designed, developed, and deployed a full-stack AI-powered job intelligence platform from the ground up as a solo developer. The platform helps job seekers by analyzing job postings with AI, providing deep company insights through real-time web scraping, generating personalized interview preparation content, matching resumes against job descriptions, and offering a complete monetization system. This was a greenfield project where I was responsible for the entire technical implementation.

3.2 Scale & Metrics

Metric	Value
Total Lines of Code	126,000+
Total Files	707
Backend Modules	18
API Endpoints	100+
Database Schemas	64+
React Components	131
Frontend Pages	30+
AI Providers Integrated	4 (Gemini, OpenAI GPT-4o, Groq, Tavily)
Backend Services	100+
Custom Hooks	8
Context Providers	8

3.3 System Architecture

The platform follows a decoupled monorepo architecture with independently deployable frontend and backend services. Both applications are containerized with Docker and deployed to Google Cloud Run with auto-scaling capabilities (1-10 instances).

- Frontend: Next.js 14 with App Router, React 18, Tailwind CSS 4, Radix UI, NextAuth.js
- Backend: NestJS 11 with 18 feature modules, dependency injection, decorators, guards, interceptors
- Database: MongoDB (64+ schemas via Mongoose) with Redis caching layer

- AI Layer: Multi-LLM pipeline with Google Gemini, OpenAI GPT-4o, Groq, and Tavily
- Payments: Stripe with subscriptions, credits, and feature gating
- Infrastructure: Docker multi-stage builds + GitHub Actions CI/CD + Google Cloud Run

3.4 Backend Development (NestJS 11)

Built a modular NestJS 11 application comprising 18 feature modules, 30+ controllers, and 100+ specialized services. The application bootstraps with extensive configuration including CORS (13 origins), Helmet security headers, Stripe webhook raw body handling, global validation pipes, timeout interceptors, Swagger/OpenAPI documentation, graceful shutdown handlers, and MongoDB connection pooling (200 max pool).

Job Analyzer: 79 services, 14 schemas. Core engine: job analysis, company intelligence, interview prep, AI hallucination validation, circuit breaker, distributed rate limiting

CV Analyzer: 10 services, 4 schemas. Multi-format resume parsing (PDF, DOCX, images via OCR), GPT-4o vision analysis, ATS scoring, job matching

Billing: 5 services, 3 schemas. Stripe payments, subscriptions (STARTER/PRO/MAX tiers), credit-based access control, webhook processing

Auth: 2 services. JWT strategy (access + refresh), Google OAuth 2.0 via Passport.js, admin JWT strategy

Admin: 6 services, 3 schemas. User management, audit logging, analytics, API metrics, role-based access

User Profiles: 2 services, 3 schemas. Profile CRUD, auto-populate from CV, avatar/banner upload to GCS

Companies: 4 services, 3 schemas. Company data enrichment, refresh, AI-powered disambiguation

Notification: 4 services. Email via Nodemailer + Resend, queue processing for bulk emails

Real Jobs: 3 services, 2 schemas. Multi-provider job scraping, semantic web search via Exa API

LeetCode Questions: 2 services. AI-generated coding questions matched to job/company

Common Module: 20+ shared services: Gemini API, Groq API, Tavily API, Redis cache, GCS, circuit breaker, distributed rate limiter, language detection

3.5 Frontend Development (Next.js 14)

26 page routes with 131 reusable React components, 8 context providers, and 8 custom hooks. Styled with Tailwind CSS 4 and Radix UI primitives, with Framer Motion for animations. The frontend includes public pages (landing, pricing, blog), authenticated user dashboard (job

analysis results, resume upload/analysis, billing), and a full admin panel (user management, analytics, audit logs, billing stats).

3.6 AI/ML Integration

- Multi-LLM fallback chain: Groq (primary) -> Gemini -> OpenAI GPT-4o
- Job description analysis: structured data extraction, tech stack detection, salary parsing
- Company intelligence: real web scraping via Tavily (NOT AI hallucinations) for funding, employees, revenue
- Interview preparation: personalized HR/behavioral + technical questions, LeetCode recommendations
- Resume analysis: GPT-4o vision for PDF/DOCX/image processing, ATS scoring, skill gap analysis
- AI Hallucination Validator: cross-checks all AI-generated data against real web sources
- Circuit breaker protection on all AI provider calls with automatic recovery

3.7 Key Technical Achievements

- Cut job-analysis latency by 60% via multi-layer LRU + Redis caching (10,000 entries, 1-hr TTL)
- Designed fault-tolerant microservices architecture with circuit-breaker, request-deduplication, and retry logic across 8+ modules, sustaining 99.9% uptime
- Containerized frontend + backend with Docker multi-stage builds + GitHub Actions CI/CD
- Authored full Swagger/OpenAPI 3.0 documentation for 100+ endpoints
- Deployed on GCP Cloud Run with health checks and graceful shutdown
- Built comprehensive admin dashboard with user management, analytics, and audit logging

4. Internship 2: MindGuruYoga (MGY)

Life-Intelligence Platform

Detail	Information
Role	Full-Stack Developer Intern
Company	MindGuruYoga (MGY)
Location	Remote, Canada
Duration	September 2025 - November 2025
Tech Stack	Next.js 15, React 19, TypeScript, Tailwind CSS, PostgreSQL, Supabase
Deployment	Google Cloud Platform with CI/CD

4.1 What I Built

Developed a full-stack life-intelligence platform that provides users with real-time metrics and AI-powered conversational guidance across four life domains: Career, Health, Love, and Money.

4.2 Key Contributions

- Achieved 40% faster page load via SSR + lazy loading; scaled PostgreSQL/Supabase backend for sub-200ms refresh on 28 real-time metrics
- Architected multi-domain conversational AI system with context-aware session storage, JWT-protected RESTful API routes, and role-based access control
- Automated monthly/yearly report-generation pipeline via server-side data aggregation, eliminating 100% of manual reporting effort
- Configured NextAuth.js/JWT auth with protected API routes; deployed on GCP with CI/CD, reducing release cycle time by 50%
- Maintained modular full-stack architecture across 100+ frontend components

5. Internship 3: Driveo.ca

On-Demand Mobile Car Washing Marketplace

Detail	Information
Role	Full-Stack Developer Intern
Company	Driveo.ca (Taskly Technologies Inc.)
Location	Remote, Canada
Duration	January 2026 - April 2026 (Present)
Tech Stack	Next.js 15, React 19, TypeScript, Supabase PostgreSQL, Stripe, Google Maps
Deployment	Vercel with CI/CD
Platform	driveo.ca

5.1 Executive Summary

Driveo is a production-ready, two-sided marketplace platform for on-demand mobile car washing -- an "Uber for car washing" -- targeting the Greater Toronto Area (GTA). I designed, architected, and built the entire platform from scratch as a single full-stack developer. The platform consists of 4 integrated applications in a Next.js 15 monorepo: Marketing Site, Customer PWA, Washer PWA, and Admin Dashboard.

5.2 Scale & Metrics

Metric	Value
Applications Built	4 (Marketing, Customer PWA, Washer PWA, Admin Dashboard)
Source Files	153
API Endpoints	25+
Third-Party Integrations	9
Database Tables	16 (PostgreSQL with RLS)
Migration Files	8
UI Components	25+ shadcn/ui + 20+ custom components

5.3 System Architecture

Monorepo architecture built on Next.js 15 App Router. All four applications share the same codebase, database, and authentication layer. Route groups provide logical separation while sharing components, utilities, and types.

- Database: Supabase PostgreSQL with 16 tables, foreign keys, indexes, triggers, RLS policies
- Auth: Supabase Auth (Email OTP magic links + Google OAuth), JWT cookies, 30-day sessions
- Realtime: Supabase Realtime for GPS tracking, job alerts, notification streaming
- Payments: Stripe PaymentIntents with manual capture (pre-authorization model, like Uber)
- Payouts: Stripe Connect Express for direct washer bank transfers
- Notifications: Tri-channel -- SMS (Twilio), Email (Resend), In-App (Supabase Realtime)
- Maps: Google Maps API for address autocomplete, geocoding, live washer GPS tracking

- Vehicle Images: Imagin Studio CDN for photo-realistic car images by make/model/year
- Error Monitoring: Sentry with source maps + performance tracing
- Analytics: GA4 conversion events + UTM attribution tracking

5.4 Database Design (16 Tables with RLS)

Designed a 16-table PostgreSQL schema with a 9-state booking FSM enforced via CHECK constraints and optimistic concurrency. Row Level Security (RLS) policies enforce data isolation per role.

Table	Purpose
profiles	All users (extends auth.users)
customer_profiles	Customer-specific data, referral codes, UTM tracking
washer_profiles	Washer status, GPS location, ratings, Stripe Connect
vehicles	Customer vehicles with make/model/year/type
subscription_plans	Plan definitions (Regular, I&E, Detailing)
subscriptions	Active subscriptions with Stripe IDs
subscription_usage	Monthly wash tracking (allocated vs used)
bookings	9-state FSM, pricing fields, all timestamps, Stripe PaymentIntent
booking_photos	Before/after wash photos in Supabase Storage
booking_messages	In-booking chat messages
reviews	Customer ratings (1-5) and comments
washer_availability	Weekly schedule by day of week
washer_blocks	Blocked time periods
notifications	In-app notifications with JSONB data
service_zones	GTA service areas by postal prefix
saved_locations	Customer saved addresses

5.5 Signature Feature: The Driveo Slide

A custom-built interactive dirt level slider (0-10) that renders a real-time dirt overlay via HTML Canvas on the customer's actual car image (fetched by make/model/year). The dirt images are pre-generated using Google Gemini AI and cached. This solves the industry problem where customers don't want to share photos of their dirty car, while giving washers accurate dirt estimates for dynamic pricing.

5.6 Business Logic

Dynamic Pricing Engine (148 lines):

Final Price = Base Price x Vehicle Multiplier x Dirt Multiplier + 13% HST

6 vehicle type multipliers (1.00x - 1.40x) and 10 dirt level multipliers (1.00x - 2.00x) with server-side calculation to prevent client-side tampering.

Washer Assignment Algorithm (Haversine geospatial matching):

- Query all approved, online washers with GPS coordinates
- Calculate Haversine distance from customer to each washer
- Filter within 30km radius, sort by nearest
- Auto-assign nearest washer; if none found, broadcast to all washers
- Race-condition safe claiming via atomic UPDATE

9-State Booking Lifecycle:

pending -> assigned -> en_route -> arrived -> washing -> completed -> paid | cancelled | disputed

5.7 Key Technical Achievements

- Built a two-sided marketplace with 3 sub-apps in a monorepo; designed 14-table schema with 9-state booking FSM
- Implemented Stripe Payment Intents, Connect Express for automated payouts, and Subscriptions for 3 tiers
- Engineered Haversine geospatial matching; delivered sub-second tracking via Supabase Realtime on Google Maps
- Enforced Row Level Security on JWT role claims with PKCE OAuth 2.0 and Zod validation
- Achieved 97% image compression (111MB to 3.4MB) for performance
- Implemented tri-channel notifications (SMS via Twilio, email via Resend, in-app via Supabase)
- Built UTM attribution tracking for marketing analytics across signup and booking flows
- Full SEO: sitemap, robots, JSON-LD structured data, OG/Twitter cards, security headers
- 100+ E2E SQL tests covering FSM transitions, pricing, and RLS policies

6. B.Tech Project: Flood Detection System

Deep Learning-Based Flood Inundation Mapping Using Satellite Data

Detail	Information
Institution	Indian Institute of Technology Kharagpur
Department	Civil Engineering / Remote Sensing
Supervisor	Prof. Bhabagrahi Sahoo
Duration	August 2025 - April 2026
Tech Stack	Python, PyTorch, Streamlit, Google Earth Engine, OpenCV, GDAL
Dataset	Sen1Floods11 (Sentinel-1 SAR + Sentinel-2 Optical)

6.1 Executive Summary

This project presents a comprehensive, end-to-end flood detection system leveraging deep learning on satellite imagery for rapid and accurate flood inundation mapping. The system fuses Sentinel-1 Synthetic Aperture Radar (SAR) data with Sentinel-2 optical satellite imagery, enabling all-weather flood monitoring even under heavy cloud cover. Three state-of-the-art deep learning architectures -- ResNet-50 UNet, Swin Transformer, and MaxViT -- are employed in an ensemble configuration with uncertainty quantification, achieving an IoU of 0.74 on the Sen1Floods11 benchmark (exceeding the target of 0.72).

6.2 Model Architectures

Model	Parameters	IoU	F1	Precision	Recall	Inference Time
ResNet-50 UNet	~35M	0.71	0.83	0.85	0.81	45 ms
Swin Transformer	~88M	0.69	0.81	0.87	0.76	120 ms
MaxViT	~31M	0.70	0.82	0.84	0.80	85 ms
Ensemble	~154M	0.74	0.85	0.87	0.83	250 ms

6.3 System Components

- Data Pipeline: Google Earth Engine integration for automated Sentinel-1/2 retrieval, preprocessing (band stacking, normalization, tiling), PyTorch Dataset with augmentation
- Model Training: Config-driven training with AdamW optimizer, CosineAnnealingLR scheduler, early stopping, Dice+BCE combined loss, gradient clipping, NaN/Inf guards
- Ensemble Inference: Mean/median/weighted aggregation with epistemic uncertainty (model variance) and aleatoric uncertainty (Monte Carlo Dropout, 20 forward passes)
- Web Application: Full Streamlit app with file upload and GEE modes, interactive Folium maps, uncertainty heatmaps, flood area statistics
- Google Colab Integration: 5 specialized scripts for training, evaluation, diagnosis, and debugging on free GPU

6.4 Key Technical Details

- Input: 8-channel imagery (S1: VV, VH + S2: B2, B3, B4, B8, B11, B12) at 512x512 pixels
- Loss Functions: Dice Loss, Focal Loss (gamma=2.0, alpha=0.25), Combined DiceBCE
- Uncertainty Quantification: Epistemic (model disagreement) + Aleatoric (data noise) -> combined confidence map
- Colab-First Strategy: All GPU training on Google Colab to overcome local Windows PyTorch DLL issues
- Multi-sensor fusion: S1-only achieves ~0.65 IoU; adding S2 optical data improves by 5-8%
- Ensemble benefit: Consistently outperforms individual models by 3-5% IoU

7. Independent Project: Real-Time Sports Dashboard

Detail	Information
Duration	October 2025 - November 2025
Tech Stack	Node.js, Express 5, PostgreSQL, WebSockets, Next.js 16

7.1 What I Built

- Eliminated HTTP polling via WebSocket pub/sub server (ws v8) with in-memory Map(matchId, Set(WebSocket)); delivered live scores and commentary in under 100ms latency with ping/pong heartbeat and automatic dead-connection cleanup
- Created exponential-backoff reconnection with subscription restoration and hybrid data strategy (REST + WS push + 5-second poll fallback) for zero data loss
- Secured all paths via Arcjet rate-limiting, bot detection, and shield
- Validated all inputs with Zod v4 schemas; managed migrations with Drizzle ORM + Drizzle Kit
- Modeled 10 WebSocket message types as TypeScript discriminated unions for full end-to-end type safety

8. CS Fundamentals & Coursework

8.1 Data Structures & Algorithms

700+ problems solved across competitive programming platforms.

8.2 Core CS Knowledge

- Object-Oriented Programming (OOP)
- System Design (scalable architectures, microservices, load balancing)
- Database Management Systems (DBMS)
- Computer Networks

8.3 Relevant Coursework at IIT Kharagpur

Category	Courses
Programming & CS	Programming and Data Structures, Algorithms, Machine Learning, Computational Physics
Mathematics	Probability and Statistics, Linear Algebra, Numerical Analysis, Advanced Calculus

9. Summary of Achievements & Impact Metrics

9.1 Overall Impact Numbers

Metric	Value
Total Production Code Written	250,000+ lines of TypeScript/Python
Total Backend Modules Built	18+ (NestJS) + Next.js API routes
Total API Endpoints Designed	125+
Total Database Schemas Designed	80+ (64 MongoDB + 16 PostgreSQL)
Total React Components Built	250+
Total Frontend Pages Built	56+
AI/LLM Providers Integrated	6 (Gemini, GPT-4o, Groq, Tavily, SerpAPI, Exa)
Third-Party Service Integrations	15+ (Stripe, Supabase, Google Maps, Twilio, Sentry, etc.)
Deep Learning Models Trained	3 architectures + ensemble (154M total parameters)
DSA Problems Solved	700+
Professional Internships	3 (all remote, Canada-based)
B.Tech Project	1 (under Prof. Bhabagrahi Sahoo, IIT KGP)

9.2 Key Performance Achievements

- 60% reduction in job-analysis latency via multi-layer caching (TasklyApp)
- 99.9% uptime via fault-tolerant microservices with circuit breakers (TasklyApp)
- 40% faster page loads via SSR + lazy loading (MindGuruYoga)
- 50% reduction in release cycle time via CI/CD pipeline (MindGuruYoga)
- Sub-200ms database refresh on 28 real-time metrics (MindGuruYoga)
- Sub-second GPS tracking via Supabase Realtime WAL-based CDC (Driveo)
- 97% image compression (111MB to 3.4MB) (Driveo)
- IoU of 0.74 exceeding 0.72 target on Sen1Floods11 benchmark (BTP)
- Under 100ms WebSocket latency for live sports scores (Sports Dashboard)
- 100+ E2E SQL tests covering FSM transitions, pricing, and RLS policies (Driveo)

9.3 Technology Breadth

Across my internships and projects, I have demonstrated proficiency across the full spectrum of modern software engineering:

- Frontend: React 18/19, Next.js 14/15/16, Tailwind CSS, Radix UI, shadcn/ui, Framer Motion, WebSockets
- Backend: Node.js, NestJS 11, Express 5, FastAPI, BunJS, GraphQL, microservices
- Databases: PostgreSQL, MongoDB, Redis, Supabase, Drizzle ORM, Mongoose, Prisma
- AI/ML: PyTorch, OpenAI GPT-4o, Google Gemini, Groq LLaMA, ensemble deep learning, uncertainty quantification
- DevOps: Docker, Kubernetes, GitHub Actions, GCP, AWS, Vercel, Terraform, Nginx, Sentry

- Security: JWT, OAuth 2.0, RLS, OWASP headers, rate limiting, input sanitization, Stripe webhooks
- Payments: Stripe (PaymentIntents, Connect, Subscriptions, Checkout Sessions)
- Real-time: WebSockets, Supabase Realtime, Server-Sent Events, GPS tracking
- Geospatial: Google Maps API, Haversine algorithm, Leaflet.js, Google Earth Engine, GDAL