Oblack Technologies · Aurora Platform Technology Depth

Executive Overview

- Aurora is a production-grade, AI-assisted quant research lab built on Next.js 14, LangGraph agents, and a Neon/Postgres + Supabase foundation. The stack continuously generates, backtests, and publishes multi-timeframe trading playbooks while gating premium access via Stripe-powered subscriptions.
- The codebase demonstrates end-to-end ownership—from deterministic strategy generation (`lib/agents/strategy-generator.ts`) through telemetry-rich admin surfaces (`app/admin/page.tsx`)—proving we can ship sophisticated ML/SaaS systems without exposing proprietary signal formulas.
- Cross-cutting infrastructure (cost tracking, cron-driven loops, Pinecone + OpenAI assistants, secure API management) shows repeatable patterns we can reapply to other data-intensive products such as publishing, research portals, or immersive reader experiences.

Platform Pillars

1. Adaptive Strategy Builder

- **LangGraph research loop** (`lib/agents/orchestrator.ts` , `lib/agents/backtester.ts`) orchestrates GPT-5 prompts, OANDA data (`lib/market-data/oanda.ts`), and the multi-timeframe engine (`lib/backtesting/engine.ts`) to turn support/resistance templates into vetted JSON payloads.
- **Deterministic templates** in `lib/strategies/support-resistance.ts` blend ATR heuristics, candlestick pattern detection, and builder diagnostics so every generated strategy includes provenance—trend timeframe, session filters, risk envelope, and checklist logs.
- **Mini-sim & gatekeeping** enforce quality before persistence: orchestration runs quick backtests, calculates Sharpe/return thresholds (`lib/backtesting/metrics.ts`), logs agent activity (`lib/db/queries.ts`), and only stores assets that meet or can be improved by `lib/agents/improvement-orchestrator.ts` .

2. Immersive Reading & Publishing Engine

- **Strategy Vault UX** (`app/strategies/_components/StrategyVault.tsx` , `components/StrategyCard.tsx`) packages data science outputs into searchable, filter-rich cards with cost-aware metrics, market/timeframe filters, and subscription-aware gating.
- **Playbook reader** under `app/strategies/[id]/playbook/` orchestrates a multi-tab narrative (overview, entry, exits, sizing, checklist) backed by data from `lib/db/strategies.ts` . `components/EliteLock.tsx` and tier checks toggle pro/elite-only content without leaking raw rules.
- **Export & publishing surfaces**—PDF generation (`lib/pdf/strategy-report.ts` , `components/DownloadPdfButton.tsx`), MT4/PineScript code paths (`lib/export/*` , `app/api/strategies/[id]/export/route.ts`), and REST delivery (`app/api/v1/strategies/route.ts`)—turn research into assets clients can operationalize.

3. Unified Commerce & Access Governance

- **Stripe-first billing** (`app/api/stripe/checkout` , `app/api/stripe/webhook/route.ts`) normalizes tiers, billing intervals, and lifecycle events, triggers Postmark welcomes (`lib/email/postmark.ts`), and syncs status into Neon via `lib/db/subscriptions.ts` .
- **Supabase authentication middleware** (`lib/supabase/*` , `middleware.ts`) keeps session cookies aligned across ISR/SSR boundaries, while `lib/subscriptions/tiers.ts` + `lib/api/keys.ts` handle entitlements, per-feature quotas, and hashed API-key issuance.
- **Pricing intelligence** surfaces under `app/admin/pricing/page.tsx` , combining live cost data, usage projections, and plan modeling so GTM teams can adapt price architecture without touching code.

4. AI Ops, Telemetry & Growth Flywheel

- **Cost telemetry** uses `lib/cost-tracking/pricing.ts` + `lib/cost-tracking/db.ts` , Vercel cron endpoints (`app/api/admin/costs/route.ts`), and the glassmorphic admin UI (`app/admin/page.tsx`) to surface per-strategy GPT spend, model mix, and weekly/monthly forecasts.
- **Automated engagement** leverages the X Campaigner agent (`lib/agents/x-campaigner.ts`) + cron endpoint (`app/api/cron/x-campaign/route.ts`) to auto-post validated insights, record outcomes in `lib/db/x-posts.ts` , and recycle underperformers into improvement loops.
- **Operational scripts** (`scripts/run-research-loop.ts` , `scripts/test-backtest.ts` , `scripts/reset-neon.js`) and documented playbooks (`STRATEGY_SYSTEM*.md` , `ADMIN_DASHBOARD.md` , `VERCEL_DEPLOYMENT.md`) prove we can bootstrap infra, migrate schema, and rehearse DR scenarios without manual guesswork.

Reference Architectures

Research & Improvement Graph

```
flowchart LR
    Cron["Vercel Cron / CLI\n(app/api/cron/*)"]
    Orchestrator["LangGraph Orchestrator\nlib/agents/orchestrator.ts"]
    Generator["Template Builder\nlib/strategies/support-resistance.ts"]
    Backtester["Backtester Agent\nlib/agents/backtester.ts"]
    MarketData["OANDA Fetcher\nlib/market-data/oanda.ts"]
    DB["Neon Postgres\nstrategies + backtest_results"]
    Metrics["Dashboards & APIs\napp/dashboard · app/api/v1/strategies"]
    Improver["Improvement Loop\nlib/agents/improvement-orchestrator.ts"]
    Cost["Cost Tracking\nlib/cost-tracking/*"]

    Cron --> Orchestrator
    Orchestrator --> Generator
    Generator --> DB
    Orchestrator --> Backtester
    Backtester --> MarketData
    Backtester --> DB
    DB --> Metrics
    Metrics --> Improver
    Improver --> Generator
    Orchestrator --> Cost
    Improver --> Cost
```

Experience, Commerce & Publishing Flow

```
flowchart LR
    Visitor["Visitor / Member"]
    Marketing["Marketing Pages\napp/page.tsx · app/subscribe/page.tsx"]
    Auth["Supabase Auth + Session Middleware\nlib/supabase/*"]
    Dashboard["Member Dashboard & Strategy Vault\napp/dashboard · app/strategies"]
    Stripe["Stripe Checkout/Webhooks\napp/api/stripe/*"]
    Postmark["Postmark Email\nlib/email/postmark.ts"]
    Entitlements["Tier + Usage Logic\nlib/subscriptions/tiers.ts"]
    API["REST + Key Mgmt\napp/api/v1/strategies · app/api/account/api-keys"]
    Publishing["PDF/Code Export APIs\napp/api/strategies/[id]/*"]
    Telemetry["Admin + Cost Ops\napp/admin/* · lib/cost-tracking"]

    Visitor --> Marketing --> Auth --> Dashboard
    Dashboard --> Stripe --> Postmark
    Stripe --> Entitlements --> Dashboard
    Dashboard --> API
    Dashboard --> Publishing
    API --> Telemetry
    Publishing --> Telemetry
```

Capability Matrix

Capability	Representative Assets	Key Technologies	Business Outcome
Adaptive Strategy Builder	`lib/agents/orchestrator.ts`, `lib/backtesting/engine.ts`, `lib/strategies/support-resistance.ts`	LangGraph, GPT-5 via OpenAI SDK, OANDA market data, TypeScript heuristics	Rapid generation & validation of institution-grade playbooks with full provenance.
Immersive Reader & Publishing	`app/strategies/[id]/playbook/*`, `components/StrategyCard.tsx`, `lib/pdf/strategy-report.ts`, `lib/export/*`	Next.js server components, Tailwind design system, jsPDF, Pinecone assistant	Converts quant research into human-readable narratives, PDFs, and broker-ready scripts.
Commerce & Access Governance	`app/api/stripe/*`, `lib/subscriptions/tiers.ts`, `components/PricingPlans.tsx`, `lib/api/keys.ts`	Stripe, Supabase, Neon/Postgres, bcrypt, Lucide UI toolkit	Tier-aware experiences, automated onboarding, secure API distribution, and adaptive pricing controls.
	`lib/cost-tracking/*`, `app/admin/page.tsx`,	Neon analytics queries, cost	Real-time view of GPT spend, run-rate,

Capability	Representative Assets	Key Technologies	Business Outcome
AI Ops & Telemetry	`scripts/check-costs.ts`, `app/api/admin/costs/route.ts`	heuristics, serverless cron jobs, glass UI components	and ROI → keeps research velocity aligned with budget.
Engagement & Distribution	`lib/agents/x-campaigner.ts`, `app/api/cron/x-campaign/route.ts`, `lib/db/x-posts.ts`, `app/api-docs/page.tsx`	Twitter API v2, LangGraph, Pinecone assistant, REST docs	Always-on outbound flywheel plus self-serve API docs to scale distribution safely.

Highlighted Modules & Case Notes

- *Multi-timeframe Strategy Engine (lib/backtesting/ + lib/strategies/)*
  Runs ATR-adjusted zone detection, candlestick pattern scoring, and pip-aware sizing. Checklists (support-resistance-checklist.ts) capture human review data so improvements can reason about what changed.
- **Quality-Scored Strategy Vault (lib/db/strategies.ts + app/strategies/)**
  Neon queries compute quality scores (Sharpe, return, drawdown weights). `StrategyVault` renders them with tier-aware truncation, search, and UX touches like mini badges for cost drag—ideal pattern for any catalog requiring KPI storytelling.
- **Immersive Reader & Elite Controls (app/strategies/[id]/playbook, components/EliteLock.tsx)**
  Tabbed storytelling, sticky TOC, inline code blocks, and callouts translate data into action. Elite gating toggles PineScript snippets, checklists, and session filters without duplicating markup.
- **Publishing & Automation Pipelines (app/api/strategies/[id]/download-pdf, app/api/strategies/[id]/export/route.ts)**
  Button clicks invoke server actions that package JSON into jsPDF or PineScript/MT4 scripts. Usage quotas from `lib/subscriptions/tiers.ts` ensure exports respect plan limits—pattern extends to any premium download service.
- **Commerce, Pricing & Lifecycle Ops (app/api/stripe/*, app/admin/pricing/page.tsx)**
  Webhooks normalize statuses, trigger Supabase account provisioning, and send Postmark onboarding. Admin pricing UI ingests live cost signals to recommend plan deltas—demonstrates data-informed monetization.
- **Cost Intelligence & Admin Surfaces (lib/cost-tracking/*, app/admin/page.tsx)**
  Aggregates GPT token counts by operation/model, projects weekly/monthly run rates, and exposes frictionless controls for ops teams. The same pattern can back telemetry for any LLM-heavy workload.
- **Growth Flywheel: X Campaigner (lib/agents/x-campaigner.ts, app/api/cron/x-campaign/route.ts)**
  LangGraph agent selects which strategy to spotlight, generates data-backed tweets/comments, respects rate limits, and captures rotation stats in Postgres (`lib/db/x-posts.ts`). This is a blueprint for any autonomous marketing assistant.
- **Tooling & Reliability Scripts (scripts/*.ts)**
  CLI utilities reset Neon schemas, dry-run strategy generation, profile costs, and seed Stripe products. Each script references the same libraries used in production, ensuring parity during incident response.

Delivery Practice & Cross-Cutting Concerns

- **Security & Governance:** API keys are bcrypt-hashed (`lib/api/keys.ts`), cron endpoints enforce bearer secrets, and Supabase middleware ensures session refresh without leaking tokens. Feature gating centralizes in `lib/subscriptions/middleware.ts`, making entitlement audits trivial.
- **Observability & Cost Control:** Agent logs, cost tracking tables (`cost_tracking` schema in `scripts/reset-neon.js`), and admin dashboards surface per-run telemetry. `components/StrategyCard.tsx` even displays cost drag so end users understand net vs. gross returns.
- **Design System & UX:** Global styles (`app/globals.css`) combine Tailwind with custom “glass surface” utilities, Lucide icons, Framer Motion accents, and accessibility-friendly typography—illustrating our ability to craft premium brand experiences.
- **Deployment & Environment Management:** Vercel config (`vercel.json`, `VERCEL_DEPLOYMENT.md`) plus environment loaders in `lib/db/client.ts` and `lib/market-data/oanda.ts` guarantee secrets stay server-side. Scripts automate Neon schema creation, while Vitest coverage (`lib/agents/__tests__/strategy-generator.test.ts`) protects deterministic builders.
- **Documentation & Playbooks:** Internal briefs (`STRATEGY_SYSTEM.md`, `STRATEGY_SYSTEM_OVERVIEW.md`, `ADMIN_DASHBOARD.md`, `PLAYBOOK_TEMPLATE.md`) capture architecture decisions, runbooks, and UX standards—evidence of a mature delivery culture.

Call to Action

Aurora proves Oblack Technologies can fuse AI research, immersive publishing, and SaaS-grade commerce into one cohesive platform. If you need a similarly rigorous strategy builder, reader experience, or telemetry-backed SaaS launchpad, let’s co-design the roadmap. Reach out to schedule a technical working session and we’ll tailor these building blocks to your portfolio.