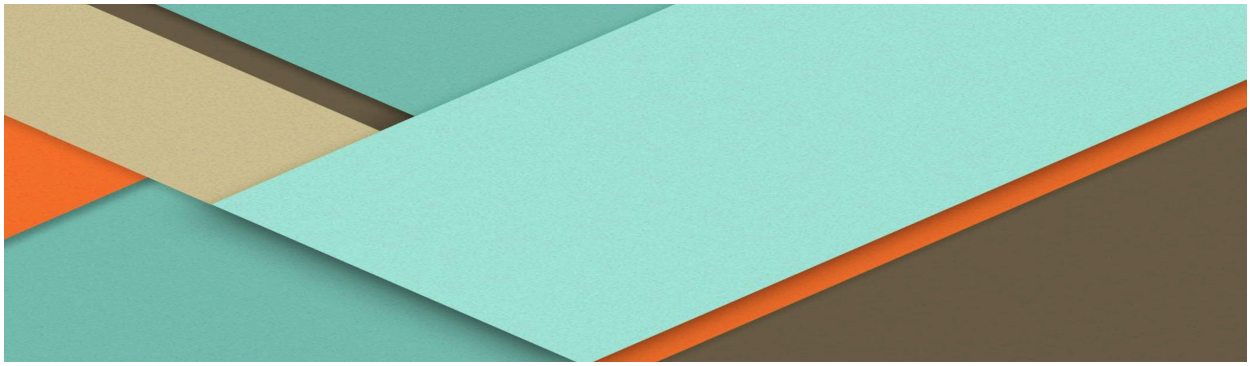

Crypto Private Strategy Website

23 June, 2025

—

Daily Edge Trading Group

Overview

We propose to build a purpose-driven web application on Next.js and Vercel, backed by Supabase, that empowers users to subscribe to custom cryptocurrency tickers, view interactive real-time charts enriched with live TradingView alerts, and access advanced Bitcoin analytics. The platform will include a public marketing site—with affiliate, pricing, testimonials, and sign-up flows, and a streamlined authenticated dashboard featuring a sidebar, top-bar metrics, an interactive Lightweight Chart with buy/sell markers, and fully customizable layouts. By combining reliable WebSocket/SSE data feeds for live pricing with Supabase's real-time database and authentication, we will deliver an engaging, performant, and secure experience tailored to both novice and professional traders.

Goals

1. Deliver a responsive Next.js frontend deployed on Vercel with SEO-friendly marketing pages (affiliate, pricing, testimonials, login/signup).
2. Implement Supabase Auth with row-level security so each user securely manages subscriptions, alerts, and personalized settings.
3. Provide REST endpoints and Supabase Edge Functions to load historical OHLC data and cache it for fast rendering in Lightweight Charts.
4. Integrate a reliable WebSocket/SSE feed (e.g., Binance WS or CoinCap SSE) to stream live ticker data and update charts in second intervals.
5. Ingest TradingView webhook alerts via a secret-authenticated endpoint and persist buy/sell signals in Supabase, overlaying them on the user's chart as markers.
6. Create two advanced Bitcoin analytics modules: (a) a 200-week heatmap showing long-term cycles and (b) a 2-year cycle indicator highlighting deviations from moving averages.
7. Develop a predictive "Cycle Forecaster" that combines halving dates and on-chain metrics (e.g., Coin Days Destroyed) to project future price trajectories.
8. Enable full user customization: user profile, alert preferences, stored and retrieved via Supabase.
9. Add an admin-only superuser section for one designated user. This page allows the admin to enable or disable available tickers, which dynamically control the symbols shown on the member charting page and associated widgets.

10. Implement a public-facing landing page with hero section and embedded chart examples that users can preview before subscribing.
11. Achieve production-ready performance, security, and scalability, including responsive design, HTTPS everywhere, CI/CD via GitHub Actions → Vercel, and thorough testing/documentation.

Specifications and Technology Stack

- **Frontend Framework**: Next.js (React + TypeScript) deployed on Vercel for server-side rendering (SSR) and incremental static regeneration (ISR).
- **Backend Platform**: Supabase (PostgreSQL, Auth, Edge Functions, Realtime) for database, authentication, and serverless logic.
- **Charting Library**: Lightweight Charts (TradingView) embedded in Next.js components for high-performance candlestick, line, and Heikin-Ashi charts.
- **Real-Time Data Feeds**:
    - **WebSocket**: Binance WebSocket API (primary) for sub-second kline updates.
    - **Fallback SSE**: CoinCap Server-Sent Events for simple last-trade price streams.
- **Alert Ingestion**: HTTP webhook endpoint secured by URL-embedded secret, implemented as a Supabase Edge Function.
- **Notifications**: SendGrid for email, Twilio for SMS, and Web Push (VAPID) for browser push notifications.
- **CI/CD**: GitHub Actions automates tests, linting, and deployment to Vercel. Supabase CLI manages database migrations.

Milestones

Below are four sequential milestones that structure development into logical phases, each building on the previous. Dependencies and suggested roles are noted for clarity.

Milestone 1: Foundation & Infrastructure Setup

Objective: Establish the foundational backend infrastructure, authentication logic, and database schema to support user specific data, admin-only controls, and real-time customization. This milestone lays the groundwork for all secure access flows, including dashboard and admin panel logic

Key Activities

- **Project Kickoff & Requirements Review**

    - Confirm scope with stakeholders
    - Finalize user stories and technical requirements
        - User vs. Admin capabilities
        - Role-based rendering logic
        - How ticker availability affects chart subscriptions
        - Customization persistence expectations

- **Supabase Project Initialization & Schema Design**
    - Provision Supabase project and configure environment variables
    - Implement primary tables (**examples only below**, TBD on implementation):
        - `users` (with `id`, `email`, `is_admin`, `created_at`)
        - `subscriptions` (user-specific ticker follows)
        - `alert_signals` (incoming alerts tied to user ID)
        - `ohlc_cache` (historical market data)
        - `user_settings` (preferences, layout metadata)
        - `webhook_secrets` (alert ingestion validation)
        - `available_tickers` **(NEW)**:

```
CREATE TABLE available_tickers (
  ticker_symbol TEXT PRIMARY KEY,
  is_enabled BOOLEAN NOT NULL DEFAULT TRUE,
  updated_by UUID REFERENCES users(id),
  updated_at TIMESTAMPTZ DEFAULT NOW()
);
```

    - Define Row-Level Security (RLS) policies for user isolation
        - Users can only read/update their own `subscriptions`, `alerts`, and `settings`.
        - Only users with `is_admin = true` can modify `available_tickers`.

- **Next.js Project Scaffold & CI/CD**
    - Initialize Next.js (TypeScript) repository
    - Configure GitHub Actions for linting, testing, and Vercel deployment
    - Establish environment management (Supabase keys, Exchange API keys)
    - Scaffold routes for:
        - `/login`, `/signup`, `/dashboard`, `/admin` (placeholder)
    - Configure GitHub Actions

- **Authentication & Authorization**
    - Integrate Supabase Auth for email/password sign-up and OAuth
    - Create basic login and signup pages

- Verify that `auth.uid()` maps correctly to protected resources
- Implement `useSession` and `useUser` hooks across the app
- Protect routes:
  - User dashboard (requires valid session)
  - Admin panel (requires `user.is_admin = true`)

Deliverables

1. **Supabase**
   - Live Supabase project with:
     - Tables and RLS policies deployed
     - Environment variables configured

2. **Next.js Repository**
   - Boilerplate with authentication flow (signup, login, protected routes)
   - CI/CD pipeline (GitHub Actions → Vercel) passing

3. **Documentation**
   - README with local setup instructions
   - Supabase migration guide (supabase CLI)
   - Basic API spec for auth and user endpoints

Milestone 2: Data Integration & Real-Time Feeds

Objective: Implement historical and live market data ingestion, TradingView alert capture, and second-level price streaming. Add support for dynamically available tickers, maintained by an admin, which constrain what data users can subscribe to.

Key Activities

- **Historical OHLC Service**
  - Implement `/api/ohlc` Edge Function:
    - Accepts `ticker_symbol`, `interval`, `range`
    - Checks `ohlc_cache` first
    - Falls back to Binance REST API if data is stale or missing
    - Normalizes response and inserts/upserts into `ohlc_cache`
    - Enforces that the `ticker_symbol` must be enabled in `available_tickers`

- **Live Price Streaming (Client Side)**
  - Client-side WebSocket integration to Binance/Coinbase/etc:
    - Connect to websocket source
    - Parse kline messages, update candlestick series
  - SSE fallback (CoinCap) for resilience:
    - Implement `EventSource` client
    - On message, update price series or buffer for bar updates
  - (Optional) Edge Function proxy for WebSocket → SSE if server-side control is required

- **Available Tickers API Endpoint**
  - **Implement** `GET /api/tickers`:
    - Returns list of `available_tickers where is_enabled = true`
    - Ticker search/autocomplete
    - Charting logic
    - Subscription creation filtering

- **TradingView Webhook Ingestion**
  - Create Supabase Edge Function:
    - POST /api/webhook/alerts
    - Secured via secret token (stored int `webhook_secrets`)
    - Body includes `ticker`, `signal_type`, `time`, and optionally user UUID
    - Stores result in `alert_signals` with timestamp

        - Return `201 Created` or `401 Unauthorized`

- **Realtime Broadcasting with Supabase**
    - Enable Supabase Realtime for `alert_signals` table
    - Broadcast changes by user_id
    - Frontend subscribes via Supabase client SDK
    - Event triggers buy/sell markers in the live chart

Deliverables

1. **Edge Functions**
    - `GET /api/ohlc` Historical OHLC retrieval + caching
    - `GET /api/tickers`: List of currently enabled tickers
    - `POST /api/webhook/alerts` Ingest and persist TradingView alert signals

2. **Client-Side Streaming Code**
    - WebSocket consumer that updates candlestick series in real time
    - SSE consumer as fallback

3. **Supabase Realtime**
    - Realtime subscription configured for `alert_signals`

4. **Documentation**
    - API documentation for `/api/ohlc`, `/api/webhook/alerts`, `/api/tickers`
    - Implementation notes
    - Integration guide: adding a new data source or altering interval frequency

Milestone 3: Core Dashboard & User Interface

Objective: Deliver a fully interactive user dashboard with real-time charting, subscription controls, and alert visibility. Introduce a protected admin-only interface for managing ticker availability. Ensure all UI components are aware of dynamic configuration stored in Supabase, including layout preferences, enabled tickers, and alert settings.

Key Activities

- **Dashboard Layout & Navigation**
  - Sidebar: Dashboard, Subscriptions, Alerts, Settings (for all users), Admin (for super user only)
  - Top Bar: Live ticker (Trading View Widget is fine), 24hour stats, quick profile access
  - Configure client-side route protection using Supabase session
    - `/admin` route only renders if `user.is_admin === true`
    - Redirect non-admins from protected pages
  - Ensure responsive design for desktop and tablet

- **Interactive Chart Component**
  - Embed Lightweight Charts within a React component
  - Load historical OHLC data via `/api/ohlc`
  - Update in real time via WebSocket/SSE (1-second throttled)
  - Overlay alert signals from Realtime subscription (`alert_signals`)
  - Respond to ticker change events and UI theme (e.g., dark mode)
  - Add controls: timeframe selector, chart-type toggle (candlestick, line, Heikin-Ashi), ticker selection

- **Subscription Management UI**
  - "Subscriptions" view:
    - Autocomplete ticker search using `GET /api/tickers` (only tickers with `is_enabled = true`)
    - Add/remove buttons for subscriptions
    - Display all user subscriptions
    - On selecting a ticker, update the chart dynamically

- **Alerts View**
  - Columns: Ticker, Signal Type, Signal Time, Notification Status
  - Paginated with timestamp sorting

    - Click a row to scroll to and highlight marker on the chart

- **User Settings: Profile & Customization**
    - Notification preferences: email / SMS / push toggles
    - Input fields for notification destinations (email, phone)
    - Web push registration for browser-based alerts
    - Profile card view (display name, avatar upload optional)

- **Authentication Flow Integration**
    - Protect all dashboard routes so only authenticated users can access
    - Redirect unauthorized requests to `/login`

- **Admin Panel UI (Superuser Only)**
    - `/admin` route:
    - Table of all available tickers (`available_tickers`)
    - Toggle switch to enable/disable individual tickers
    - Persist changes via Supabase mutation (admin-only RLS)
    - Ticker availability affects:
        - Subscription list options
        - Available charts
        - Alert validation
    - Activity log or audit trail view optional (for future milestone)

Deliverables

1. **UI Components**
    - Layout: Sidebar, Top Bar, Auth Wrapper
    - Chart With Alerts: Integrated with data endpoints and Realtime
    - Subscriptions CRUD UI: ticker autocomplete + validation
    - Alerts Table: live data view, interaction enabled
    - Settings UI:
        i. Webhook secret
        ii. Notification preferences
        iii. Layout configuration (grid)
    - Admin Panel:
        i. Ticker availability dashboard with toggle switches
        ii. Conditional rendering of the route

2. **API Integration**s
    - `/api/subscriptions` (POST, DELETE, GET)
    - `/api/alerts` (GET)

- `/api/settings` (GET, POST)
- `/api/tickers` (read-only for both dashboard and admin)
- `/api/admin/tickers` (update enabled status, RLS protected)

3. **Design & UX**
   - Tailwind CSS styling with consistent visual language
   - ARIA attributes and keyboard navigation for accessibility
   - Color modes (light/dark) if feasible at this stage

4. **Documentation**
   - UI component hierarchy with prop contracts
   - Admin feature flag logic
   - Supabase RLS policy mapping (admin vs. user)
   - Guide: How to register for push notifications in-browser
   - Design guide: spacing, typography, layout grid structure

Milestone 4: Advanced Analytics, Testing & Launch

Objective: Finalize production readiness by integrating advanced Bitcoin analytics modules, launching the marketing site with preview charts, ensuring robust testing across user/admin paths, and executing a hardened deployment. This milestone ensures that all goals are functionally covered and ready for secure, scalable use.

Key Activities

- **Advanced Analytics Modules**
  - **200-Week Heatmap**
    - Supabase Edge Function to compute weekly 200-SMA and percent deviations
    - React component to render heatmap with color scale and date tooltips (doesn't have to be Trading View LightWeight Charts)
    - See more:
  - **2-Year Cycle Indicator**
    - Edge Function highlighting deviations from moving averages.
    - React component to display deviation line chart with shaded ±10% bands
    - See more:
  - **Cycle Forecaster**
    - Lightweight model (exponential smoothing or regression)
      - Halving Dates
      - Coin Days Destroyed
    - React component to overlay forecast line (dashed) and confidence interval shading on the main chart
    - Doesn't have to be TradingView LightWeight Chart
    - See more:

- **Public Landing Page + Chart Previews**
  - Implement public `/` route with:
    - Hero section (marketing headline, video/image background, CTA)
    - Sample charts (static or simulated)
    - Read-only candlestick chart from `/api/ohlc?ticker=BTCUSDT&demo=true`
    - Example alerts visible as fixed markers
  - "Subscribe Now" CTA routing to `/signup`

- **Admin Panel Finalization**
  - Add any final admin panel controls:
    - Audit view (read-only history of ticker toggles) if feasible
    - Prevent disabling all tickers (fallback logic)
  - UX enhancements: loading states, update feedback, error handling
  - Log all admin actions to a dedicated `admin_activity_log` table (optional)
  - Edge Function hardening: reject invalid updates or malicious ticker symbols

- **Notification System & Background Jobs**
  - Implement `notification_queue` table and scheduled Edge Function to process pending notifications (email/SMS/push)
  - Ensure failed notifications log errors and allow re-trie

- **Testing & Quality Assurance**
  - **Unit Tests**:
    - Frontend: Jest + React Testing Library for critical components (chart updates, subscription CRUD, settings)
    - Backend: Jest for Edge Functions (OHLC fetch logic, webhook validation, indicator computation)
  - **Integration Tests**:
    - Postman/Newman collection to verify all REST endpoints (subscriptions, OHLC, alerts, settings)
    - Supabase CLI to run RLS and policy validations
  - **End-to-End (E2E) Tests**:
    - Cypress tests simulating: signup → subscribe → receive alert → chart marker appears → regenerate secret → update settings
  - **Load Testing & Performance Tuning**:
    - k6 scripts to simulate 500 simultaneous Realtime subscriptions
    - Monitor Supabase CPU/memory, optimize queries, add indexes as needed

- **Final Documentation & Deployment**
  - **Developer Documentation**:
    - Complete README with setup, migration, testing, deployment steps
  - **Operational Runbooks**:
    - Secret rotation procedure, backup/restore instructions, scaling guides for WebSocket proxy
  - **User Documentation**:

        - In-app Help/FAQ covering: subscription flow, alert management, chart customization, analytics interpretation
    - **Production Launch**:
        - Final deployment to Vercel (marketing site + dashboard) and Supabase production instance
        - Smoke tests: verify public pages load, signup/login, data flows, analytics modules render

Deliverables

1. **Advanced Analytics Features**
    - Heatmap component, 2-year deviation chart, cycle forecaster overlay
2. **Public Landing Site**
3. **Admin Finalization**
4. **Notification Engine**
5. **Test Suites & Reports**
6. **Documentation & Runbooks**
7. **Production Release**