Product Requirements Document (PRD): NeoStrike Operator Portal (Serverless Edition)

1. Product Vision

A high-performance, stateless Command Center for iGaming Operators that provides real-time business intelligence and compliance oversight without the infrastructure overhead of managing persistent socket connections.

2. Revised Architecture: Serverless Event Flow

Instead of a continuous "listener," notifications are treated as **asynchronous events**.

1. **Trigger**: A database change (e.g., payment_failed or kyc_required) triggers a **PostgreSQL Trigger**.
2. **Process**: The trigger invokes a **Supabase Edge Function** to log the event into a dedicated operator_notifications table.
3. **Consume**: The React dashboard uses **SWR (Stale-While-Revalidate)** to poll the notifications table. This allows Vercel to scale horizontally as each request is stateless.

---

3. Core Feature Requirements

🔔 Feature: Serverless Notification Center

- **Purpose**: Centralized hub for operational alerts (failed payments, KYC, RG flags).
- **Requirements**:
  - **Notification Drawer**: A slide-out panel showing a chronological list of alerts.
  - **Status Management**: Notifications must have read_at, resolved_at, and severity (Critical, Warning, Info).
  - **Stateless Sync**: UI refreshes alerts every 30 seconds (or upon user interaction) using an optimized API fetch to minimize database load.

📊 Feature: 30-Day GGR Visualization

- **Purpose**: Track Gross Gaming Revenue trends over time to identify performance anomalies.
- **Requirements**:
  - **Interactive Line Chart**: Visualizing daily GGR (Total Bets - Total Wins).
  - **Serverless Aggregation**: A nightly cron job (Supabase Edge Function) aggregates daily totals into a daily_stats table to ensure chart loads are instantaneous.

🔍 Feature: Global Search & Indexing

- **Purpose**: Rapidly locate players or transactions during support or compliance audits.
- **Requirements**:
  - **Omni-Search**: Single input field supporting partial matches for username, email, and transaction_id.
  - **Deep Linking**: Clicking a search result navigates directly to a detailed **Player Profile** or **Transaction Audit** view.

📅 Feature: Global Date Range Selector

- **Purpose**: Allow operators to filter the entire dashboard by specific timeframes.
- **Requirements**:
  - **Contextual Filtering**: All metrics (KPIs) and the GGR Chart must react to the selected range (e.g., "Last 7 Days", "Last 30 Days").
  - **URL Persistence**: The date range should be stored in the URL (query params) to allow bookmarked views and shared reports.

---

4. Technical Requirements (The "Serverless" Stack)

Component	**Technology**	**Implementation Detail**
Logic/API	**Vercel Functions**	Stateless Node.js endpoints for data retrieval.
Data Sync	**TanStack Query / SWR**	Handles revalidation and caching on the client-side.
Background Tasks	**Supabase Edge Functions**	Deno-based functions for triggers and cron jobs.
Database	**PostgreSQL**	Managed by Supabase with Row-Level Security (RLS) for multi-tenancy.

5. Success Metrics (KPIs)

- **Notification Latency**: Under 60 seconds from database event to dashboard visibility (stateless polling).

- **Page Load Time**: Dashboard and charts must load in $<1.5s$ via optimized aggregation tables.
- **Scalability**: Support for 1,000+ simultaneous operators without managing socket connection pools.

6. Portal Layout & Visual Architecture

The interface uses a **Persistent Left Sidebar** and a **Floating Global Header** to maintain user orientation across deep workflows.

A. Global Header (Utility Layer)

- **Omni-Search Bar**: Centered, expanding on focus. Supports keyboard shortcut Ctrl + K for predictive lookup of players, transaction IDs, or specific compliance events.
- **Serverless Notification Bell**: Fixed top-right with a "Pulse" badge for unread alerts. Hovering reveals a mini-drawer of recent operational events (e.g., "MGA Affordability Flag: player123").
- **Global Date Selector**: A persistent dropdown allowing the operator to toggle the entire dashboard context between "Real-time," "Last 7 Days," "Last 30 Days," or "Custom Range".

B. Persistent Left Navigation (Action Layer)

- **Icons-First Design**: Collapsible sidebar with high-contrast icons for Dashboard, Player Management, Wallet, Games, Compliance (AI Duty of Care), and System Settings.
- **Breadcrumbs**: Dynamic labels at the top of each page (e.g., Compliance > Alerts > Detail) to prevent "navigational lostness".

6.1 Dashboard Interface Components

A. Primary Metrics (The "KPI Strip")

Four "Glass-card" widgets at the top display critical real-time data, each with a mini-sparkline showing the trend relative to the previous period.

- **Active Players**: Real-time count of concurrent sessions.
- **Total GGR**: Gross Gaming Revenue (Bets - Wins).
- **Approval Rate**: Success percentage for payment orchestrations.
- **Compliance Alerts**: Count of players currently flagged for AI intervention.

B. Performance Visualization (Main Body)

- **30-Day GGR Trend Chart**: A high-resolution line chart using **Glassmorphism layering** (Frosted transparency) to separate the trend lines from the background.
- **Hover Interactivity**: Users can hover over any date to see a breakdown of that day's most active game providers or specific payment failures.

6.2 Product Requirement Details: Layout Elements

Element	**Requirement**	**Visual Implementation**
Grid System	Responsive 12-column grid	Cards rearrange for tablet/mobile without losing data priority.
Hierarchy	Priority metrics "Above the Fold"	Critical financial and compliance data must be visible without scrolling.
Typography	Strategic spacing & weights	High-readability fonts for plain numbers to reduce cognitive load.
Micro-Interactions	Subtle animations	Hover effects on charts and buttons that guide the eye without distracting.

6.3. Implementation Logic for UI Developers

- **Stateless Component Design**: Each widget (Metric, Chart, Notification) is a standalone component that fetches its own data based on the **Global Date Context**.
- **Data Aggregation**: The 30-day chart should pull from a pre-aggregated daily_stats table (via cron job) rather than querying the raw audit_logs to ensure page load is under **1.2 seconds**.