### **Project "Market Pulse" - Technical Specification & Changelog (V2.0)**

This document provides a complete technical overview of the Market Pulse application and a chronological history of its development.

### **1. Project Overview & Technical Specification**

* **1.1 Objective**: A multi-tenant web application that provides hotel performance metrics via a live dashboard, allowing individual hotel clients to compare their performance against an aggregated market average.
* **1.2 Technology Stack**:
  + **Frontend**: Vanilla JavaScript (ESM), HTML5, Tailwind CSS (via CDN).
  + **Backend**: Node.js with the Express.js framework.
  + **Database**: Neon Serverless PostgreSQL.
  + **Deployment**: Vercel, with continuous deployment connected to the main branch of the GitHub repository.
  + **Authentication**: Cloudbeds OAuth 2.0 (Authorization Code Grant) for user login and onboarding. User sessions are managed by  
     express-session with a persistent connect-pg-simple store.
* **1.3 Key Application Files**:
  + server.js: The main Express.js application server, handling all API logic and serving the frontend.
  + public/app/index.html & public/dashboard.js: The primary user-facing dashboard application.
  + public/admin/index.html & public/admin.js: An administrative panel for system health checks and manual job triggers.
  + api/daily-refresh.js & api/initial-sync.js: Vercel Serverless Functions that run as background jobs to sync data from the Cloudbeds API.
* **1.4 API Endpoints Summary**:
  + **Authentication**:
    - GET /api/auth/cloudbeds: Initiates the OAuth 2.0 login flow.
    - GET /api/auth/cloudbeds/callback: Handles the OAuth redirect from Cloudbeds.
    - POST /api/admin-login: Authenticates a user for the admin panel.
  + **Dashboard Data (Session-Protected)**:
    - GET /api/kpi-summary: Provides aggregated KPI values for the dashboard cards.
    - GET /api/metrics-from-db: Fetches time-series data for the logged-in user's hotel.
    - GET /api/competitor-metrics: Fetches aggregated time-series data for the market comparison.
    - GET /api/get-hotel-name & GET /api/last-refresh-time.
  + **Admin Panel (Session-Protected)**:
    - GET /api/test-cloudbeds, GET /api/test-database, GET /api/get-all-hotels.
    - Manual Triggers: GET /api/daily-refresh, GET /api/initial-sync.
* **1.5 Database Schema Highlights**:
  + users: Stores full user profiles, including encrypted tokens and the cloudbeds\_property\_id that links a user to their specific hotel data.
  + daily\_metrics\_snapshots: Stores all time-series data, partitioned by cloudbeds\_user\_id to ensure data isolation between tenants.
  + user\_sessions: Stores persistent session data, allowing users to stay logged in.

### **2. Project Development History**

* **July 5, 2025 (Morning)**: The project was migrated from a local setup to a cloud-native solution on Vercel. The database connection was configured for production, and the background refresh script was converted into a Vercel Serverless Function scheduled via  
   vercel.json.
* **July 5, 2025 (Afternoon)**: To solve the "cold start" problem, a script was created to seed the database with mock data for five competitor hotels, establishing the "market" for comparison. The backend API was updated to aggregate this data, providing a single, averaged "market" row per day for the dashboard.
* **July 8, 2025 (V2.0 Refactor)**: A major refactoring effort was completed to convert the application to a multi-tenant platform. This involved implementing the Cloudbeds OAuth 2.0 flow, updating the database schema to be user-aware, and refactoring the API server with session-based authentication to secure all endpoints.
* **July 8, 2025 (Post-Deployment Debugging)**: After deploying the V2.0 changes, the dashboard was inaccessible due to login session failures.
  + **Diagnosis**: The root cause was identified as an issue with session persistence. The default in-memory session store was not viable in a serverless environment, causing the user's login state to be lost between API calls.
  + **Corrective Actions**: A series of fixes were implemented, including configuring CORS and cookie domains, and most critically, replacing the in-memory session store with a persistent PostgreSQL-backed store (connect-pg-simple). The database logic was also refactored to use a single, shared connection pool for stability.
  + **Final Fixes**: A subsequent TypeError on the dashboard was traced to unimplemented API endpoints (/api/metrics-from-db, /api/competitor-metrics), which were then fully implemented. Finally, the non-functional Admin Panel was restored by adding its required backend API routes.
* **Current Status (As of July 8, 2025, ~8:00 PM CEST)**: The application is stable and fully functional as a multi-tenant platform. All core architectural goals of the V2.0 migration have been met.

### **3. V2.0 Future Roadmap**

This section outlines the strategic direction and planned features for the next major version of the Market Pulse application.

* **Major Feature: Advanced Reporting Module**
  + **Objective**: Build a dedicated "Reports" section to provide users with powerful, customizable, and exportable data views.
  + **Key Tasks**:
    - **Phase 1 (Report Builder UI)**: Create a new page with UI controls for building custom reports.
    - **Phase 2 (Configurable API)**: Create a new backend endpoint to dynamically generate data based on user selections.
    - **Phase 3 (Advanced Analytics)**: Implement advanced calculations, starting with "Market Revenue Adjusted for Hotel Size".
    - **Phase 4 (Exporting)**: Add functionality to export reports to CSV, followed by PDF and Excel.
    - **Phase 5 (Report Scheduler)**: Implement a system for users to schedule recurring reports to be sent via email.
* **User Experience & Onboarding**
  + **Objective**: Improve the first-time user experience to increase adoption and reduce support requests.
  + **Key Tasks**:
    - **Guided Product Tour**: Implement a "first-login" guided tour to explain key features.
    - **User Profile & Settings**: Create a page where users can manage their account settings.
* **Access Control & Permissions**
  + **Objective**: Create different permission levels within the application.
  + **Key Tasks**:
    - **Implement Role-Based Access Control (RBAC)**: Add a role column to the users table (e.g., 'admin', 'user').
    - **Superadmin View**: The "Admin Panel" and its APIs should only be accessible to users with the 'admin' role.