### **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.

### **Entry: Tuesday, July 8, 2025 - 9:08 PM CEST**

**Objective:** Establish a safe and isolated local development environment to prepare for the multi-property and competitive set refactor.

**Summary of Actions:**

* **Code & Data Isolation:**
  + Created a new Git branch named feature/multi-property-and-comps to isolate all upcoming code changes from the stable main branch.
  + Created a new Neon database branch named dev-multi-property to serve as a complete, isolated checkpoint of the database schema and data, protecting the production database from any development changes.
* **Local Development Environment Configuration (server.js):**
  + **Development Login Endpoint:** Added a new POST /api/dev-login route. This endpoint is wrapped in a !isProduction check, ensuring it only exists in the local environment. It allows developers to create an authenticated session by sending a userId, bypassing the need for the live Cloudbeds OAuth flow which is tied to the production URI.
  + **CORS Policy Update:** Modified the corsOptions to conditionally add "http://localhost:3000" to the allowedOrigins array when the application is not running in a production environment. This permits API requests from the local frontend to the local server.
  + **Session Cookie Policy Update:** Modified the express-session cookie configuration to be environment-aware. It now sets sameSite: "lax" for local development (allowing cookies over HTTP) and sameSite: "none" for production (required for cross-domain OAuth).

**Current Status:** The local development environment is fully configured and operational. We are now ready to begin the database schema modifications for the multi-property and competitive set features on the isolated development branches.

**Entry: Tuesday, July 8, 2025 - 10:44 PM CEST**

**Objective:** Refactor the application to support multi-property user accounts and implement an intelligent competitive set based on a manually assigned hotel quality tier.

**Summary of Changes:**

* **Database Schema Rework:**
  + Added a  
     star\_rating column to the hotels table to store a manually assigned quality tier for each property.
  + Removed the cloudbeds\_property\_id column from the users table to de-couple users from a single property.
  + Created a new user\_properties linking table to establish a many-to-many relationship between users and their properties.
* **Backend API Refactor (server.js):**
  + The /api/auth/cloudbeds/callback endpoint was updated to query all of a user's properties from the Cloudbeds API and create a link for each one in the user\_properties table.
  + All data-fetching API endpoints (e.g., /api/kpi-summary, /api/metrics-from-db) were refactored to accept a propertyId as a query parameter and include a security check to verify the logged-in user has access to the requested property.
  + The /api/competitor-metrics endpoint was significantly enhanced. It now first determines the star\_rating of the user's selected property and then filters the market data to only include competitor hotels with a matching rating, providing a true "like-for-like" comparison.
  + A new /api/my-properties endpoint was created to provide the frontend with a list of properties a user has connected.
* **Frontend UI Development (dashboard.js & index.html):**
  + The property switcher dropdown in the dashboard header was made dynamic. It now populates by fetching data from the new /api/my-properties endpoint.
  + All data-loading functions were updated to pass the propertyId of the currently selected property from the new dropdown to the backend API, ensuring the correct data is displayed.
* **User Experience & Authentication Flow:**
  + A new /login page was created to serve as a dedicated entry point for new and returning users.
  + The authentication flow was corrected by implementing Vercel Middleware (middleware.js) to protect page routes. Logged-out users attempting to access /app/ or /admin/ are meant to be redirected to the login page - this is not working for unknown reason. Additionally, in the future logged out users shouldn’t be redirected to the oAuth stage again - this should happen only once with Cloudbeds when connecting the app for the first time. Instead after this is done we should have our own login page - either with email and password or magic link.  
      
    Also on the <https://www.market-pulse.io/admin/> page - Run Job Daily Forecast Refresh doesn’t work, resulting in a 404 error in the console.

Of course. Here is a new changelog entry summarizing the issues and the successful recovery process. You can add this to your document.

**Entry: Wednesday, July 9, 2025 - 9:30 AM CEST**

**Objective:** Diagnose and resolve a critical production failure that rendered the entire application non-functional following a branch revert.

**Summary of Actions:**

1. **Initial Diagnosis & Triage:**
   * Following a revert to the main branch, the application became unstable. The primary symptom was a total failure of the Cloudbeds OAuth login process, which crashed with a 500: FUNCTION\_INVOCATION\_FAILED error.
   * Initial troubleshooting steps, including verifying environment variables and forcing a redeployment of the local codebase to Vercel, were undertaken to rule out configuration or code synchronization issues. These steps did not resolve the problem.
2. **Root Cause Analysis:**
   * To get a definitive error message, enhanced debug logging was temporarily added to the GET /api/auth/cloudbeds endpoint in server.js.
   * A subsequent deployment and test revealed the true root cause for all failures: a ReferenceError: requirePageLogin is not defined was being thrown the moment the server tried to initialize.
   * This error was traced to two lines of code in server.js that were attempting to use a page-protection middleware function that did not exist. This remnant from a previous development effort was preventing the Express application from starting, causing all incoming requests to fail.
3. **Corrective Actions & Resolution:**
   * The two calls to the undefined requirePageLogin function were removed from the page-serving routes (/app/ and /admin/) in server.js.
   * Upon deploying this fix, all server crashes ceased. The Cloudbeds OAuth flow was restored to full functionality.
   * Subsequent testing confirmed that the previously reported 404 error on the "Daily Refresh" job trigger was also resolved by the code synchronization.

**Current Status:** The application is stable and fully functional. All issues stemming from the branch revert have been resolved. The main branches on both Git and Vercel now reflect the correct, working version of the application. Core functionalities, including user authentication, dashboard data rendering, and admin panel job triggers, are operating as expected.

**Entry: Wednesday, July 9, 2025 - 11:00 AM CEST**

**Objective:** Implement a secure, passwordless magic link login system to create a seamless user experience for returning users and enhance overall application security.

**Summary of Actions:**

1. **Email Service Integration:**
   * The SendGrid transactional email service was integrated into the application to handle the delivery of secure login links.
   * The project's domain (market-pulse.io) was authenticated by configuring the necessary CNAME and TXT DNS records, ensuring high email deliverability.
2. **Environment Configuration:**
   * A new SENDGRID\_API\_KEY environment variable was added to the Vercel project to securely connect to the email service.
3. **Database Schema Extension:**
   * A new magic\_login\_tokens table was created in the PostgreSQL database. This table is designed to store single-use, expiring tokens, ensuring each login link is secure and time-sensitive.
4. **Backend API Development:**
   * A POST /api/auth/login endpoint was created in server.js. This endpoint accepts a user's email, verifies if the user exists, and if so, triggers the email service to send the login link.
   * A GET /api/auth/magic-link-callback endpoint was created to handle the verification of the token from the user's email, create a persistent user session upon success, and grant access to the application.
5. **Frontend Rework:**
   * The login.html page was completely redesigned with a new UI and client-side JavaScript to support the email-based login flow.
   * The page now provides clear user feedback, either confirming that a link has been sent or instructing new users on how to connect via the Cloudbeds Marketplace.

**Final Status:** The magic link authentication system is yet to be tested including user flow, redirects etc.

### **Entry: Wednesday, July 9, 2025 - 11:30 AM CEST**

**Objective:** Diagnose and resolve a critical Vercel build issue that was causing frontend application scripts to fail.

**Summary of Actions:**

* **Initial Symptoms & Diagnosis:**
  + Despite a clean deployment, the application's frontend was non-functional. The browser console showed a persistent Uncaught ReferenceError: require is not defined on the first line of admin.js and dashboard.js.
  + Investigation confirmed that the source files on Vercel were correct and used modern import syntax, but the files being served to the browser were being incorrectly transpiled to use require().
* **Root Cause Analysis:**
  + The root cause was identified as an ambiguity in the project's configuration that confused Vercel's build system.
  + The catch-all route { "src": "/(.\*)", "dest": "server.js" } in vercel.json, combined with the increasing complexity of server.js, led the build system to misidentify frontend ES Module assets as backend CommonJS dependencies.
* **Corrective Actions & Resolution:**
  + To resolve this build conflict, frontend JavaScript files that use ES Module syntax were renamed to use the .mjs extension. This explicitly signals their module type to the Vercel build system, preventing incorrect transpilation.
  + The following files were renamed:
    - public/constants.js was renamed to public/constants.mjs.
    - public/admin/admin.js was renamed to public/admin/admin.mjs.
    - public/app/dashboard.js was renamed to public/app/dashboard.mjs.
  + References to these files were updated accordingly:
    - The <script> tag in public/admin/index.html was updated to point to admin.mjs.
    - The <script> tag in public/app/index.html was updated to point to dashboard.mjs.
    - The import statement in public/admin/admin.mjs was updated to import from ../constants.mjs.
* **Current Status:** The application is stable and fully functional. The build-time module conflict has been resolved, and all frontend assets are now being served correctly to the browser.