### **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.

### **Entry: Wednesday, July 9, 2025 - 12:40 PM CEST**

**Objective:** Implement, test, and debug the complete passwordless magic link authentication flow for returning users.

**Summary of Actions & Resolutions:**

1. **Initial Implementation & Scaffolding:**
   * The core backend logic was implemented, including a POST /api/auth/login endpoint to generate and email a secure token, and a GET /api/auth/magic-link-callback endpoint to validate the token and create a user session.
   * A new magic\_login\_tokens table was added to the database to store the single-use, expiring tokens.
   * The SendGrid API was integrated for transactional email delivery.
   * A new login.html page was created to provide the user interface for the email login flow.
2. **Debugging Vercel Caching & Routing Conflicts:**
   * Initial testing was blocked by a persistent, incorrect redirect from the login page (/login) to the dashboard (/app/).
   * Using browser network tools, this was diagnosed as a stale redirect served directly from Vercel's Edge Cache (x-vercel-cache: HIT).
   * **Resolution:** To bypass the stuck cache, the login route was permanently changed from /login to /signin in server.js.
   * A secondary client-side redirect was discovered and traced to an erroneous <meta http-equiv="refresh"> tag in a legacy version of login.html, which was subsequently removed.
3. **Resolving Email Delivery & Security Issues:**
   * Testing revealed that magic links were causing an HTTPS security warning in the browser. This was traced to SendGrid's "Click Tracking" feature, which wrapped the secure login URL in an insecure tracking domain.
   * **Resolution:** Click Tracking was disabled in the SendGrid account settings, ensuring the link sent to users was the direct, secure URL to the application. This also improved email deliverability.
4. **Final Data Logic Correction:**
   * After resolving all redirect and link issues, users could log in but the dashboard failed to load their properties.
   * The root cause was identified as a data mismatch in the session logic. The magic link flow was incorrectly setting the session userId to the database's internal integer ID (e.g., 2), while all property data is correctly linked to the cloudbeds\_user\_id (e.g., '534490').
   * **Resolution:** The /api/auth/magic-link-callback function in server.js was refactored. It now uses the internal ID from the token to perform a secondary lookup, retrieves the correct cloudbeds\_user\_id, and sets *that* ID in the user's session.

**Final Status:** The magic link authentication system is now functional. Outstanding item - users landing on the app dashboard page should get redirected to login screen if not currently logged in - this isn’t working for some reason.

**Entry: Wednesday, July 9, 2025 - 7:45 PM CEST**

**Objective:** Overhaul the user authentication flow, enhance the user interface, and implement comprehensive legal and support documentation to create a production-ready, professional user experience.

**Summary of Actions & Resolutions:**

This was a major effort focused on stabilizing the application's core authentication logic and dramatically improving the user-facing interface and documentation.

**1. Critical Authentication & Routing Fixes:**

* **Unauthenticated Access Vulnerability:** Resolved a critical bug where unauthenticated users could access the /app/ dashboard.
  + **Diagnosis:** The express.static() middleware in server.js was serving the static index.html file before the requirePageLogin authentication check could run.
  + **Resolution:** The middleware order in server.js was corrected, moving express.static() to execute *after* all protected page routes, ensuring authentication is always checked first.
* **Infinite Redirect Loop:** Fixed a bug causing an ERR\_TOO\_MANY\_REDIRECTS error after a successful magic link login.
  + **Diagnosis:** Redundant redirect routes for /app and /admin were creating a conflict with Vercel's routing.
  + **Resolution:** The unnecessary redirect routes were removed from server.js, stabilizing the login flow.
* **Dashboard Loading Failure:** Fixed an issue where the dashboard would get stuck on a loading spinner.
  + **Diagnosis:** A 404 Not Found error for dashboard.mjs was discovered, caused by an unreliable relative path in index.html.
  + **Resolution:** The script path was changed to an absolute path (/app/dashboard.mjs) to ensure it loads correctly in all scenarios.

**2. User Experience & Feature Enhancements:**

* **Storefront Redesign:** The /signin page was completely redesigned from a simple form into a futuristic, minimalistic "storefront" marketing page. This new design includes a hero section, feature highlights, and an improved layout to clearly separate the login flow for new and returning users.
* **Functional Logout:** A "Log Out" button was implemented in the user dropdown menu on the dashboard. This button calls a new /api/auth/logout endpoint that securely destroys the user's session and redirects them to the sign-in page.
* **In-App Support & Legal Access:** "Support" and "Legal" options were added to the user dropdown menu within the dashboard, providing users with easy access to help and documentation without leaving the application.
* **Chart Rendering Fix:** A visual bug in the dashboard chart was fixed where bars could render below the zero-axis. The chart logic was updated to enforce a minimum value of 0, ensuring correct data visualization.

**3. Legal & Documentation Implementation:**

* **Terms of Service:** A comprehensive Terms of Service was drafted and implemented. It includes key clauses regarding the free nature of the current service, the right to introduce fees with 30 days' notice, and a clear data license granting Market Pulse the right to use anonymized and aggregated hotel data for commercial purposes.
* **Privacy Policy:** A detailed Privacy Policy was created, specifying the types of data collected, the use of Google Analytics, and the data retention policies for both user and hotel data.
* **Expanded FAQs:** The FAQ section was significantly expanded to proactively answer user questions about data security, disconnecting the app, the read-only nature of the service, and how to suggest new features or PMS integrations.

**Current Status:** The application is now stable, secure, and provides a polished, professional user experience from the initial landing page through to the dashboard. The core authentication lifecycle is complete, and all critical bugs identified during the session have been resolved.

### **Entry: Thursday, July 10, 2025 - 10:57 AM CEST**

**Objective:** Overhaul the application's authentication model to resolve a critical session conflict and implement a secure, scalable, database-driven role system. This effort also includes a complete redesign of the Admin Panel UI/UX to align with the main dashboard and improve usability.

**Summary of Actions:** A foundational refactor of the authentication and authorization system was completed. The previous dual-login system (magic link for users, separate password for admin) was identified as the root cause of a session corruption bug, where an admin login would overwrite a user's session ID.

The system was migrated to a unified login flow where authorization levels are stored in the database. This eliminates the session conflict and introduces a robust, single source of truth for user roles. The Admin Panel was also completely redesigned for a more professional and seamless user experience.

### **1. Core Architectural Change: Database-Driven Roles**

The most significant change was the move from a separate, hardcoded admin password to a role-based system managed within the database.

**Database Schema Update:** An is\_admin column was added to the users table to permanently flag administrator accounts.  
ALTER TABLE users ADD COLUMN is\_admin BOOLEAN DEFAULT FALSE;

* **Unified Authentication Flow:** The separate POST /api/admin-login endpoint and its associated client-side password form were completely removed. All users, including administrators, now authenticate through the same secure magic link flow.

### **2. Backend Refactor (server.js)**

The server logic was updated to support the new role-based architecture.

* **Dynamic Role Assignment on Login:** The magic link (/api/auth/magic-link-callback) and Cloudbeds OAuth (/api/auth/cloudbeds/callback) endpoints were modified. Upon successful login, they now query the users table for the is\_admin flag and store it in the user's session (e.g., req.session.isAdmin = true).
* **Session-Info Endpoint:** A new endpoint, GET /api/auth/session-info, was created. This allows the frontend to securely check the current user's role (isAdmin: true/false) without exposing any sensitive information.
* **Role-Specific API Middleware:** The single, generic requireApiLogin middleware was replaced with two specific functions to provide granular security:
  + requireUserApi: Protects all standard user data endpoints.
  + requireAdminApi: Protects all administrative endpoints (e.g., /api/get-all-hotels, /api/test-database), ensuring they can only be accessed by a session with the isAdmin flag.
* **Session Duration:** The session cookie's maxAge was extended from 30 to **60 days** to improve user convenience.

### **3. Frontend & UX Overhaul**

Significant improvements were made to the user interface and experience for both the dashboard and the admin panel.

**Admin Panel (public/admin/):**

* **Complete Redesign:** The admin page was rebuilt from the ground up to match the visual style (fonts, colors, layout) of the main dashboard, including the addition of a shared navigation sidebar.
* **Seamless Admin Access:** The admin.mjs script now uses the /api/auth/session-info endpoint on page load. If an admin session is detected, it displays the admin tools directly, bypassing the need for a password and removing the annoying "flash" of the old login form.
* **Non-Disruptive Notifications:** The browser alert() calls for job status updates were replaced with a clean, on-page status message that appears near the trigger buttons and fades after a few seconds.

**Dashboard (public/app/):**

* **Dynamic Navigation:** The dashboard.mjs script now calls /api/auth/session-info on page load. The "Admin Panel" link in the sidebar is now dynamically shown or hidden based on the user's admin status.
* **UI Polish:** A "shield" icon was added to the "Admin Panel" link, making its appearance consistent with other navigation items.

**Current Status:** The application is stable and the session conflict bug is resolved. The new role-based system is more secure, scalable, and provides a significantly improved and more professional user experience for both regular users and administrators.

Of course. Here is a new changelog entry summarizing the creation of the API Explorer tool.

### **Entry: Thursday, July 10, 2025 - 12:30 PM CEST**

**Objective:** Implement a dynamic, in-app API exploration tool to discover and visualize the full range of data available from the Cloudbeds "Data Insights" API.

**Summary of Actions & Resolutions:**

A new "Cloudbeds API Explorer" has been successfully integrated into the Admin Panel. This tool allows administrators to dynamically query the Cloudbeds API, providing a clear view of available datasets and their detailed structures, which will inform future feature development.

The implementation was a multi-step process that involved overcoming a server deployment issue and focusing on robust data presentation.

* **1. Admin Panel UI Enhancement (admin/index.html):**
  + A new "Cloudbeds API Explorer" section was added to the admin page.
  + This UI includes controls for fetching all datasets and for inspecting the structure of a specific dataset by its ID.
* **2. Backend Proxy Implementation (in server.js):**
  + To bypass a persistent Vercel deployment issue that was preventing new, separate API files from being routed correctly, a pragmatic approach was taken by adding the required logic directly to the main server.js application.
  + Two new, admin-protected endpoints were created:
    - GET /api/explore/datasets: Acts as a proxy to securely call the Cloudbeds GET /datasets endpoint, retrieving the list of all available datasets.
    - GET /api/explore/dataset-structure: A second proxy that calls the Cloudbeds GET /datasets/{id} endpoint to get the detailed field structure for a specific dataset ID passed in the query string.
* **3. Dynamic Frontend Rendering (admin.mjs):**
  + The API explorer buttons were wired up to call the new backend proxy endpoints.
  + To handle the complex JSON responses from the API, two new helper functions were created: renderDatasetsTable and renderFieldsTable.
  + These functions parse the raw JSON and dynamically render the data into clean, user-friendly HTML tables. For detailed datasets, the fields are automatically grouped by category (e.g., "Booking", "Finance") to improve readability.

**Current Status:** The API Explorer tool is fully functional. Administrators can now view all available Cloudbeds datasets and inspect the categorized field structure of any dataset directly from the Admin Panel.