JunAiKey #OmniKey – User Manual (Quantum Codex Edition)

Version: 5.0
Status: Perfected

1. Introduction: Your Universal System

Welcome to the perfected version of JunAiKey.

This system operates on the **Quantum Codex Architecture**, a multi-layered design that ensures security, scalability, and intelligent operation. Its purpose is to unify your digital world, automate your workflows, and transform your data into actionable knowledge.

This manual is your guide to understanding and wielding the full power of JunAiKey.

---

2. The Quantum Codex Architecture: A 5-Layer Model

Your JunAiKey instance is built on five concentric layers, each with a distinct responsibility. Understanding this structure helps in mastering the system.

- **Layer 5: Boundary Layer (Outer Shell)**
  - **What it does:** Secures and monitors the entire system.
  - **Your Interaction:** This layer works silently to protect your data. You interact with its effects through the system's overall stability and security. The `ErrorBoundary` is a visible part of this layer.

- **Layer 4: Interface Layer**
  - **What it does:** Manages all interaction with you, the user.
  - **Your Interaction:** This is the entire UI—the pages you see, the buttons you click, the themes, and the AI-generated vocabulary. The **Theming Engine** and **Oracle Chat** are key components here.

- **Layer 3: Service Layer**
  - **What it does:** Provides all the core functionalities as services.
  - **Your Interaction:** When you use the `Knowledge Chat`

(Oracle), generate content, or browse the `API Library`, you are using services from this layer. The **Knowledge Hub** (Memory Vault) resides here.

-   **Layer 2: Control Layer**
    -   **What it does:** The "brain" that controls workflows and processes.
    -   **Your Interaction:** When you create rules in `OmniFlow` or manage agents in `OmniAgents`, you are configuring the Control Layer. It decides what tasks to run and when.

-   **Layer 1: Core Layer (The Heart)**
    -   **What it does:** The system's fundamental essence and evolutionary engine.
    -   **Your Interaction:** You can interact with this layer directly through the `AI Core Terminal` (Sanctum) to run diagnostic commands. This layer is responsible for the system's ability to self-improve over time.

You can view a visual representation of this architecture on the **System Architecture** page in the application.

---

3. Core Modules & Features

Each module in JunAiKey maps to one or more architectural layers to deliver its functionality.

3.1 OmniNote (The Memory Vault)
-   **Layer:** Service Layer
-   **Purpose**: The central repository for all memories captured by the system. All chat interactions are stored here to provide context for future conversations.
-   **How to Use**: Navigate to the Memory Vault to view and search a complete log of all stored memories.

3.2 OmniAgents (Agent Management)
-   **Layer:** Control Layer
-   **Purpose**: Oversee, manage, and configure all autonomous agents operating within the system.

3.3 OmniFlow (Automation)
-   **Layer:** Control Layer
-   **Purpose**: Define the logical "if-then" rules that govern

AI behavior and system automation.

3.4 OmniLog (Dashboard)
-   **Layer:** Interface Layer
-   **Purpose**: Your central dashboard, providing a real-time, visual overview of all system activity and key metrics.

3.5 AI Core Terminal (Sanctum)
-   **Layer:** Core Layer Interface
-   **Purpose**: A powerful, terminal-like interface for issuing diagnostic and analytical commands directly to the AI Core.

---

4. The AI Oracle & Knowledge Chat

All chat interfaces in JunAiKey are now powered by the **Knowledge Hub** (Service Layer).

-   **How it Works**:
    1. When you start a conversation, the AI first searches the Memory Vault for relevant past interactions.
    2. This context is used to provide a more personalized and accurate response.
    3. At the end of your conversation, the entire interaction is saved as a new memory, allowing the system to learn from it in the future.

---

5. Appendix: System Specifications

-   **Data Persistence**: All your data (Rules, Memories, Custom Pages, Theme) is stored locally in your browser's `localStorage`. Clearing this storage will reset the platform to its initial state.
-   **API Requirements**: The platform's intelligence relies on a connection to the Google Gemini API, which requires a valid API key to be set in the environment.