? Week 02: The Centralized Event Hub

Welcome to **Week 02**! You've already knocked out your personal dashboard; now we're stepping into the world of **Dynamic Orchestration**. This week, you aren't just building a display—you are building the "Nervous System" for every application you will ever write.

? The Mission: Observation & Architecture

The goal is to build a **Centralized Event Dashboard**. This application acts as a private "Log Sink" or "Command Center." It provides a secure API that listens for "Events" from your other apps—whether it's a successful user signup from a web app, a cron job failure in a Python script, or a simple `cURL` message from your terminal.

Build a two-part system:

1. **The Receiver (Remote API):** A cloud-hosted endpoint that is "always on," waiting to catch events from your other apps, scripts, or servers.
2. **The Viewer (Local Dashboard):** A high-performance real-time feed where you can search, filter, and visualize the data flowing through your API.

?? The Architecture

- **Project-Based Isolation:** Manage multiple apps from one hub.
- **API Key Authentication:** Secure your endpoints so only your authorized apps can post data.
- **Persistent Cloud Storage:** Use a cloud database so your data is safe and accessible even when your local machine is off.

? Step 0: Choose Your Stack

Before you run your first prompt, decide on your **Tech Stack**. You will need a database (like Supabase or PostgreSQL) to store your projects and historical event data.

Component	Option A: Modern Serverless (Recommended)	Option B: Robust Python
Backend API	Next.js API Routes (Vercel) or Hono (Cloudflare)	FastAPI (Render or Railway)
Database	**Supabase** (PostgreSQL + Realtime)	**MongoDB Atlas** or Supabase
Frontend	Next.js + Tailwind + Shadcn UI	React (Vite) + Tailwind
Live Updates	Native Supabase Realtime	Pusher or Socket.io

> **Student Note:** Are you a **Next.js + Tailwind** fan? Or do you prefer **Python (FastAPI) + React**? Specify this in your initial prompt so the AI builds the API routes correctly.

? The Starter Prompt

Copy and paste this into your AI chat to generate the foundation.

```
Build me an events dashboard. Other applications send events to it
through an API, and I see them in a real-time feed.

The app has two parts:
1. A REST API that accepts events via POST request (with API key
authentication). This needs to run on a remote server so it's always
available, even when my computer is off.
```

```
2. A dashboard that displays events in a feed, with search, filtering,
and charts. This can run locally.

Each event has: a channel (category like "orders", "signups",
"deploys"), a title, an optional description, an optional emoji icon,
and optional tags.

Features I need:
- POST /api/events endpoint that accepts JSON and stores events in the
database
- API key authentication (generate a key when creating a project)
- A feed page showing events in reverse chronological order
- Filter events by channel
- Search events by title, description, or tags
- At least one chart showing event activity over time
- The dashboard should update in real-time when new events arrive
- Use a cloud database that's always available (Supabase, Convex, or
similar)

Make it clean and functional. I want to actually use this to monitor my
own projects.

Before making any decisions on the stack. Make a stack proposal and ask
me which I want to use.
```

Once you have the initial dashboard frontend, api and database running to your liking, you can continue with the next prompts, to make it better or add additional features to it.

? Evolution: 7 Prompts to Pro Power

Once your API can catch a message, use these iterative prompts to turn a "basic table" into a professional monitoring tool.

The Roadmap:

- **The Real-Time Subscription:** Implement a real-time listener (e.g., Supabase Realtime) to push new events to the feed instantly with a highlight animation.
- **Smart Emoji & Auto-Parsing:** Add logic to automatically assign emojis based on channel names (e.g., ? for orders, ? for deploys) if one isn't provided.
- **Multi-Project Management:** Build a settings page to manage multiple projects, each with its own name and unique API key validation.

- **Advanced Time-Series Analytics:** Integrate Recharts to visualize events per hour and top channels using bar and pie charts across various time ranges.
- **Desktop & Critical Alerts:** Add native browser notifications triggered by specific tags like #error or #urgent, even when the dashboard is in the background.
- **The "Deep Dive" Inspector:** Create a clickable side-drawer for every event to display the raw JSON payload and include a "Copy as cURL" button for debugging.
- **Key Rotation & Security:** Implement a security feature to regenerate API keys, instantly voiding old credentials to protect against leaks.

?? The Detailed Prompt List

1. The Real-Time Subscription

> "Since our database is in the cloud, implement a **Real-time Listener** (e.g., Supabase Realtime). Ensure that when the remote API inserts a new event, the local dashboard pushes it to the top of the feed automatically with a subtle 'new item' highlight animation."

2. Smart Emoji & Channel Parsing

> "Enhance the API logic: if an incoming event doesn't specify an emoji icon, automatically assign one based on the `channel` name (e.g., 'orders' gets ?, 'deploys' gets ?, 'errors' gets ?). Display these icons prominently next to the event title in the feed."

3. Multi-Project API Key Management

> "Build a 'Project Settings' page in the dashboard. Allow me to create multiple projects, each with its own name and unique generated API key. The API should now validate the key against the database and tag the incoming event to the correct project automatically."

4. Advanced Time-Series Analytics

> "Add a 'Metrics' tab. Use **Recharts** to create a bar chart showing 'Events per Hour' and a pie chart showing 'Top Channels by Volume.' Allow me to toggle the time range between the last 24 hours, 7 days, or 30 days."

5. Desktop & Push Notifications

> "Add a toggle in the dashboard for 'Critical Alerts.' If an event is received with a specific tag (like #error or #urgent) or a 'High' priority status, trigger a browser-native desktop notification so I see the alert even if the dashboard tab is hidden."

6. The "Deep Dive" JSON Inspector

> "Make each event card clickable. When clicked, open a side-drawer (Slide-over) that shows the full raw JSON payload received by the API formatted for readability. Include a 'Copy as cURL' button so I can easily replicate the exact request for debugging."

7. API Key Security & Rotation

> "Implement 'Key Rotation' logic. In the Project Settings, add a button to 'Regenerate API Key.' This should instantly void the old key in the database and provide a new 32-character secret to the user, ensuring security if a key is ever accidentally leaked."

Revision #2

Created 2026-03-13 09:25:33 UTC by Carsten

Updated 2026-03-13 11:41:29 UTC by Carsten