**Phase 0 – Scaffolding & Connectivity**

**Purpose**

Phase 0 focused on establishing the **project foundation**. The goal was to verify that the three core components — **backend (FastAPI)**, **frontend overlay (Tauri)**, and **browser extension (Chrome)** — can communicate with each other. Instead of diving into analytics or AI too early, we first ensured the data pipeline is alive end-to-end.

**Thought Process**

* The project is meant to be an **Agentic AI–driven productivity assistant**.
* For the assistant to reason about user behavior, it first needs a **reliable stream of signals**.
* Phase 0 was about creating a **minimal vertical slice**: capture one type of event (tab switch), send it across layers, and confirm everything connects correctly.
* This avoids building features on shaky foundations and provides confidence in the stack before moving forward.

**Components Built**

1. **Backend (FastAPI Service)**
   * Runs locally on 127.0.0.1:8765.
   * Endpoints:
     + /event (POST): Receives events from extension/UI.
     + /health (GET): Used by Tauri to check connectivity.
   * Purpose: **central hub** for all incoming activity data.
2. **Frontend Overlay (Tauri App)**
   * Simple UI window that checks /health.
   * Confirms bridge is alive and ready for real-time updates.
   * Purpose: **user-facing interface** that will eventually display insights, alerts, and messages.
3. **Browser Extension (Chrome, MV3)**
   * Minimal setup with manifest.json and background.js.
   * Captures tab switch events.
   * Sends them as JSON to FastAPI via fetch POST requests.
   * CORS enabled on backend to accept these requests.
   * Purpose: **first data source**, simulating user activity monitoring.

**Technical Details**

* **Languages & Frameworks**:
  + Backend → Python 3.10 + FastAPI + Uvicorn
  + Frontend → Rust + Tauri (with PNPM/Node for scaffolding)
  + Extension → Chrome MV3 (JavaScript background script)
* **Data Format**:
  + Events structured as JSON, e.g.:
  + {
  + "ts": 1724325982,
  + "event": "tab\_switch",
  + "domain": "example.com"
  + }
  + Purpose: establish a consistent schema early for all signals.
* **Connectivity Tests**:
  + Curl/PowerShell POSTs verified /event.
  + Switching tabs in Chrome produced TabEvent logs in Python console.
  + Tauri displayed “online” when /health responded.

**Why This Matters**

* Without Phase 0, we’d risk building an agent system on **uncertain connectivity**.
* Now we know:
  1. The backend can receive and parse events.
  2. The extension can push events in real time.
  3. The frontend overlay can pull health status and later display insights.
* This gives us a **solid foundation** to scale into Phase 1 (data collection and storage).

✅ **Phase 0 Outcome:**  
We now have a working **event pipeline** from browser → backend → UI.  
This proves the architecture is sound and sets the stage for real analytics, agent logic, and productivity insights.