**📘 Enterprise Monitoring Dashboard – Deep-Dive Documentation**

1. **High-Level System Architecture**

**A diagram of a computer process

AI-generated content may be incorrect.**

**Legend:**

* 🟢 MockGen = creates fake systems, alerts, metrics in-memory.
* 🔵 DB = optional persistence (not required in mock version).
* 🔄 UI ↔ APIClient ↔ Views = how the dashboard talks to backend.

**2. Data Flow Diagram**

A diagram with colorful text

AI-generated content may be incorrect.**3. Frontend Component Tree**

A diagram of a company

AI-generated content may be incorrect.

**4. Backend Flow (Django REST Framework)**

* **/api/systems/**
  + Calls make\_system(i) → generates random CPU, memory, disk, department, OS.
  + Returns list of 75 fake systems.
* **/api/alerts/**
  + Creates 25 alerts with random severity (critical, warning, info).
  + Includes timestamps, messages, system reference.
* **/api/systems/<id>/metrics/**
  + Loops 24 times → generates hourly CPU/memory/disk/network usage.
  + Returns timeseries for Recharts line chart.

**5. Mock Data Example**

**System**

{

"id": 1,

"name": "WS-001",

"hostname": "workstation-1.company.local",

"department": "IT",

"location": "Toronto",

"os": "Windows 11",

"version": "v2.3.14",

"last\_seen": "2025-09-04T18:31:00",

"latest": {

"cpu": 42.1,

"memory": 65.4,

"disk": 71.2,

"network": 15.3,

"healthScore": 74

}

}

**Alert**

{

"id": 5,

"systemId": 12,

"type": "High CPU Usage",

"severity": "critical",

"message": "Alert 5: System requires attention",

"ts": "2025-09-04T16:10:00",

"acknowledged": false,

"resolved": false

}

**6. Challenges & Fixes**

**⚡ CORS Blocker**

* **Problem:** Browser blocked API calls (No 'Access-Control-Allow-Origin' header).
* **Fix:** Added django-cors-headers.
* **Config:**
* INSTALLED\_APPS += ['corsheaders']
* MIDDLEWARE.insert(2, 'corsheaders.middleware.CorsMiddleware')
* CORS\_ALLOWED\_ORIGINS = ["http://localhost:5173"]

**⚡ Empty Dashboard**

* **Problem:** Backend returned JSON but frontend still showed zeros.
* **Fix:** Confirmed .env points to http://localhost:8000, ensured refresh logic (useEffect) was running.

**⚡ Pie Chart Label Wrong**

* **Problem:** Tooltip showed “Systems” instead of department.
* **Fix:** Passed the payload into the pie chart instead of using a static systems name
* <Tooltip formatter={(value, \_name, { payload }) => [value, payload.name]} />

**⚡ Random Data Instability**

* **Problem:** Every refresh = completely different data.
* **Fix:** Carried over acknowledged/resolved flags between refreshes for realism.

**7. Deployment Considerations**

* **Frontend**: Deploy via Vercel
* **Backend**: Host Django on Render
* **Persistence**: Swap SQLite → PostgreSQL when saving real data.
* **Security**: Add JWT authentication if multiple users.

**8. Future Enhancements**

1. 🔐 **User Authentication** (Admins vs View-only roles).
2. 📈 **Persistent Metrics** (save into Postgres, query trends).
3. 🔔 **Live Alerts** (WebSockets / Django Channels).
4. 🛠 **Agent.py** integration (real system metrics instead of mock).
5. 🗂 **Export Reports** (CSV / PDF).

**9. Interview Talking Points**

* **Tech Stack**: React (UI), Tailwind (styling), Recharts (data viz), Django + DRF (API).
* **Problem Solving**: Handled CORS, cleaned refresh intervals, customized tooltips.
* **Scalability**: Mock now, Postgres later. Modular component design for growth.
* **Real-World Tie-In**: Simulates enterprise IT monitoring (system health, alerts, departments).

💡 This doc + diagrams give you **both the 10,000-foot view** and **nitty-gritty fixes**. You can now explain architecture, data flow, libraries, bugs, and improvements confidently.

Would you like me to **export this whole doc with diagrams into a PDF** (so you can carry it as revision notes)?