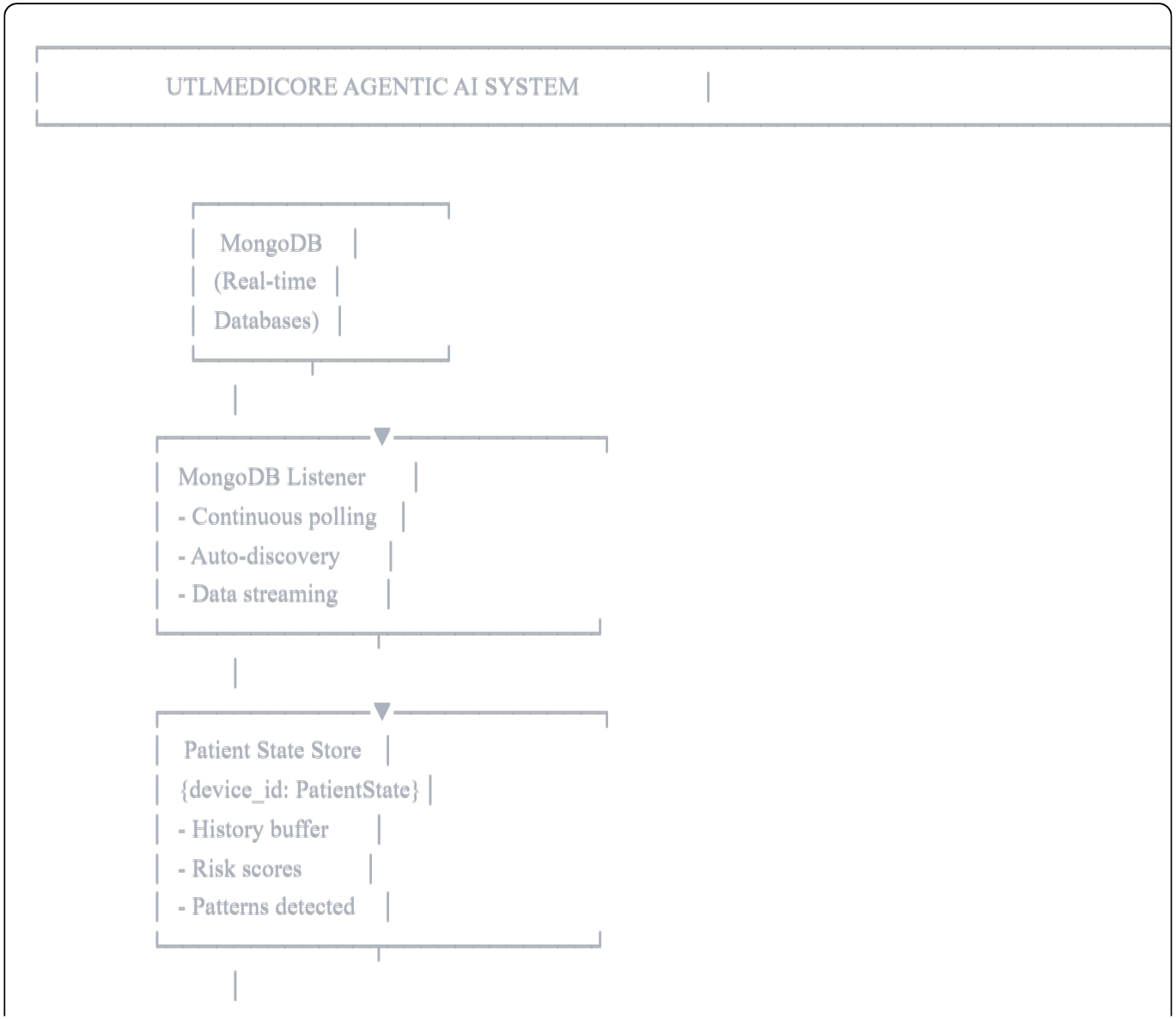
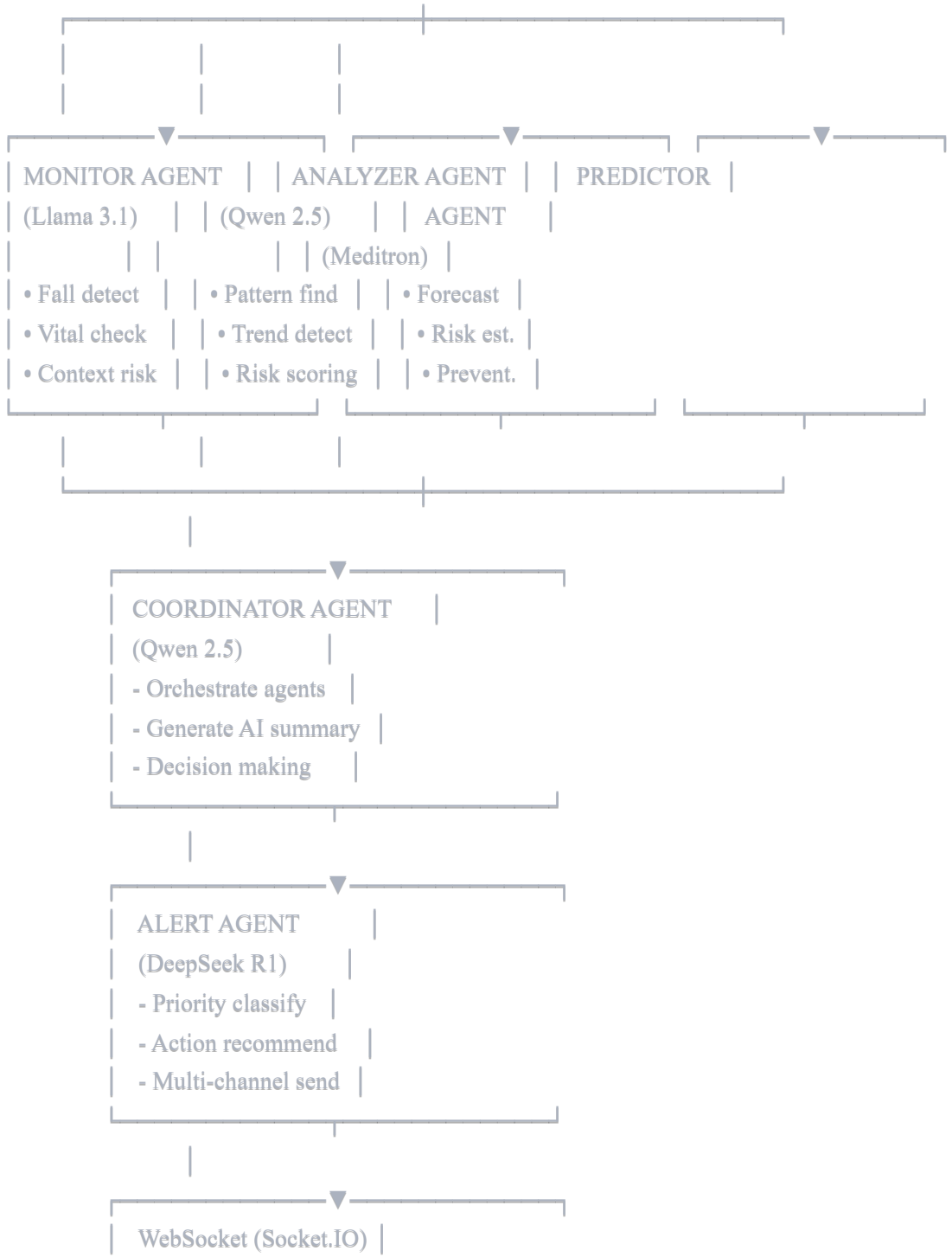
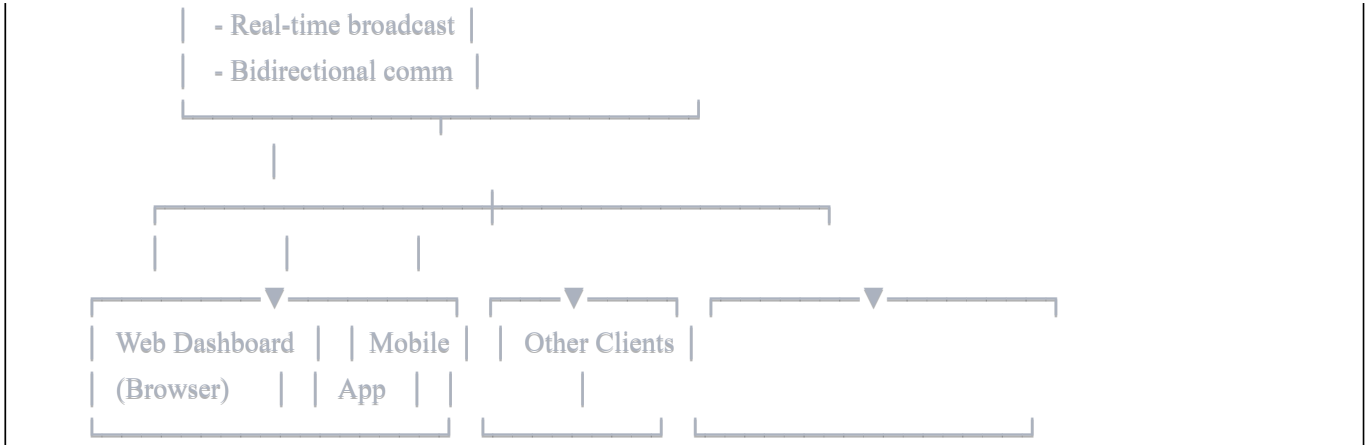
🎨 AGENTIC AI ARCHITECTURE DIAGRAMS

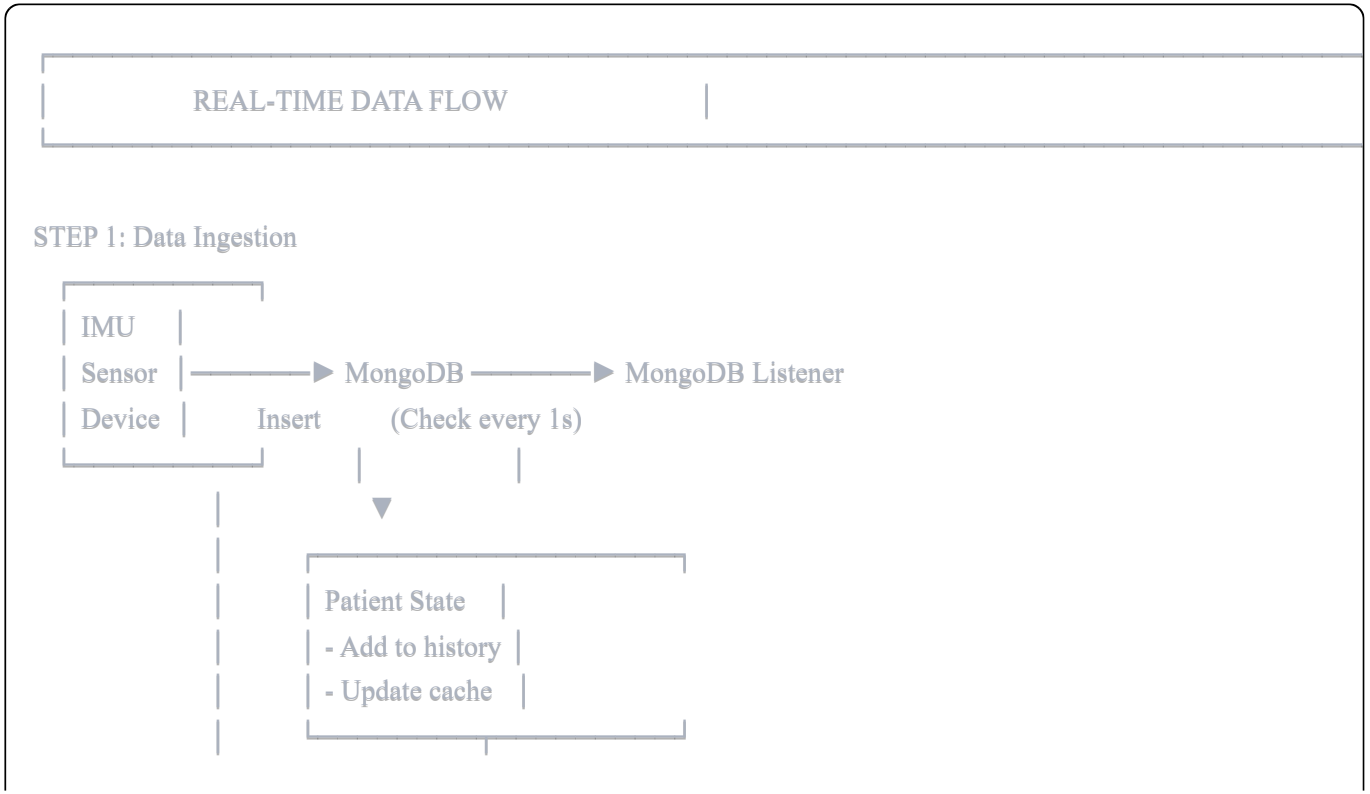
📐 SYSTEM OVERVIEW (ASCII)

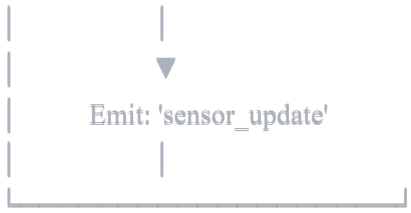



MONITOR AGENT
(Llama 3.1)
• Fall detect
• Vital check
• Context risk
ANALYZER AGENT
(Qwen 2.5)
• Pattern find
• Trend detect
• Risk scoring
PREDICTOR
AGENT
(Meditron)
• Forecast
• Risk est.
• Prevent.
COORDINATOR AGENT
(Qwen 2.5)
- Orchestrate agents
- Generate AI summary
- Decision making
ALERT AGENT
(DeepSeek R1)
- Priority classify
- Action recommend
- Multi-channel send
WebSocket (Socket.IO)

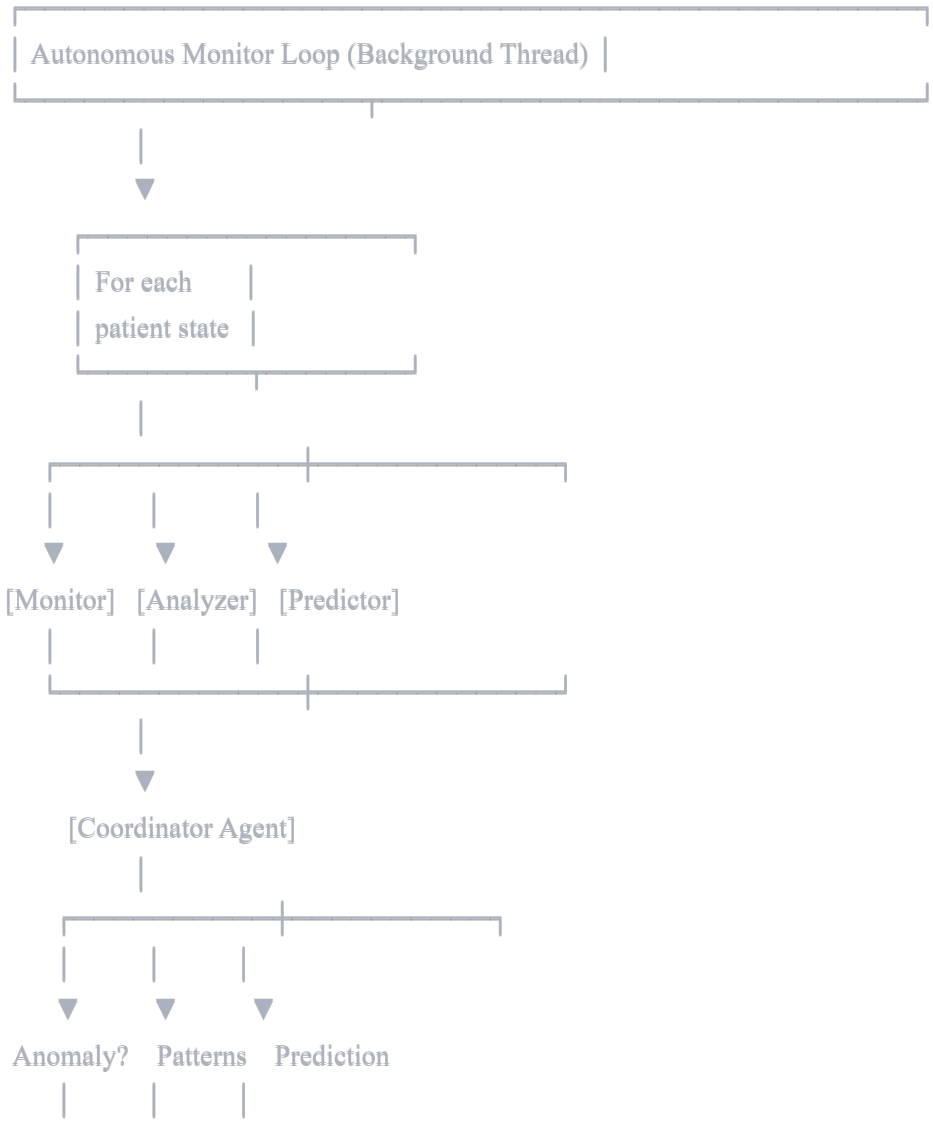


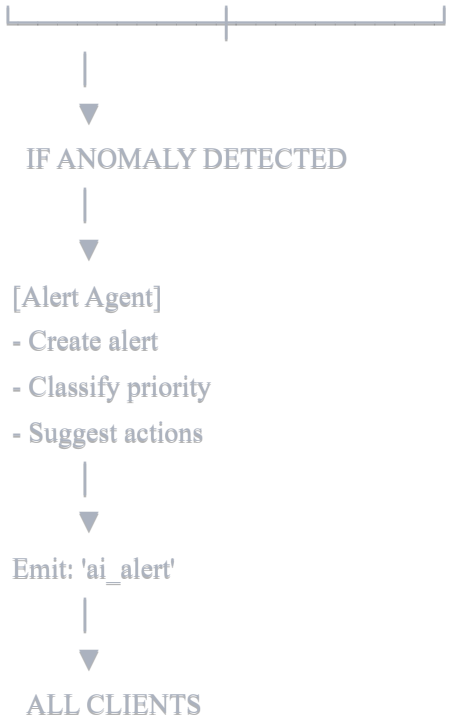
🔄 DATA FLOW DIAGRAM





STEP 2: Autonomous Analysis (Every 30s)





STEP 3: UI Updates (Real-time)

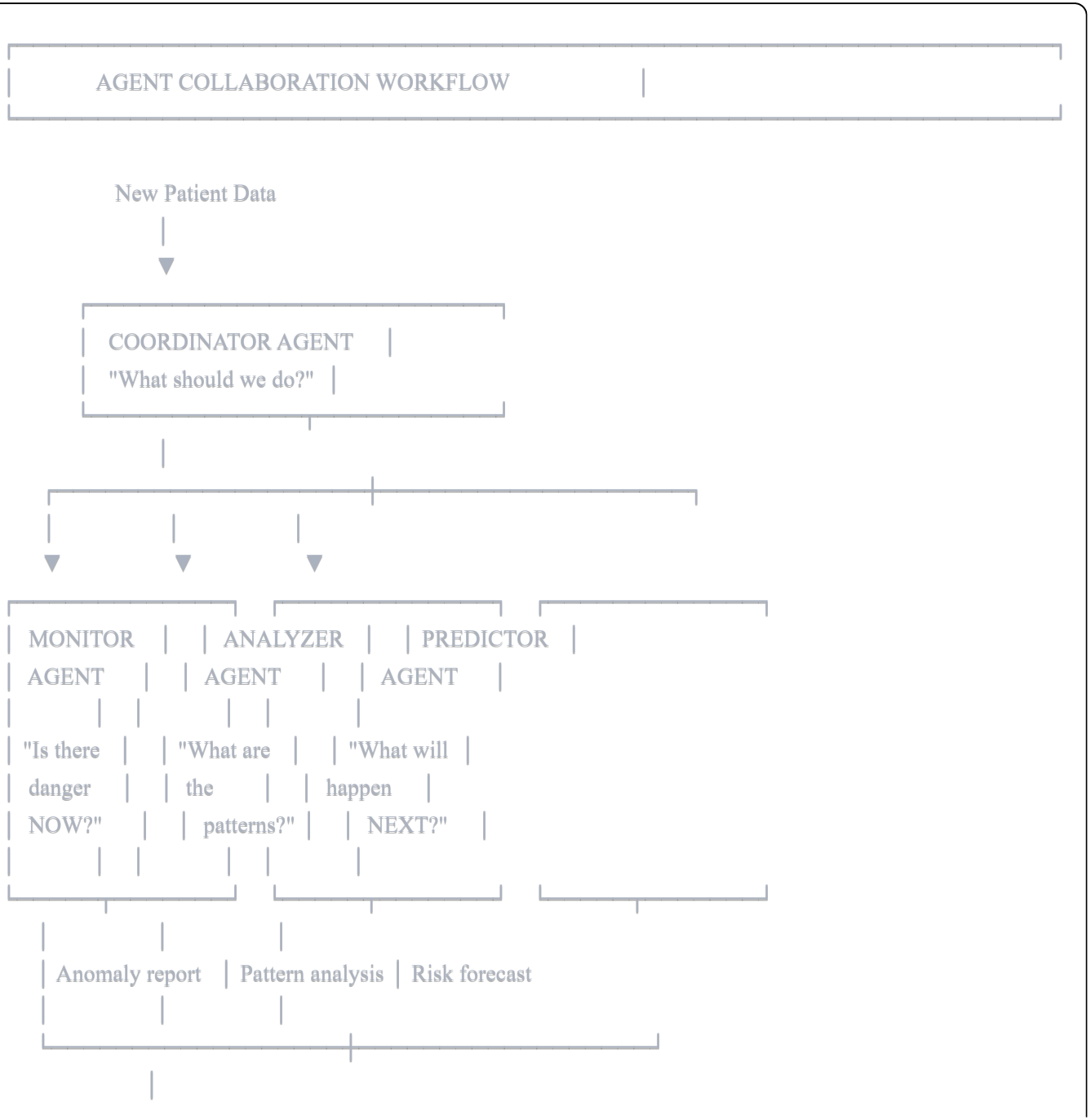
WebSocket Events:

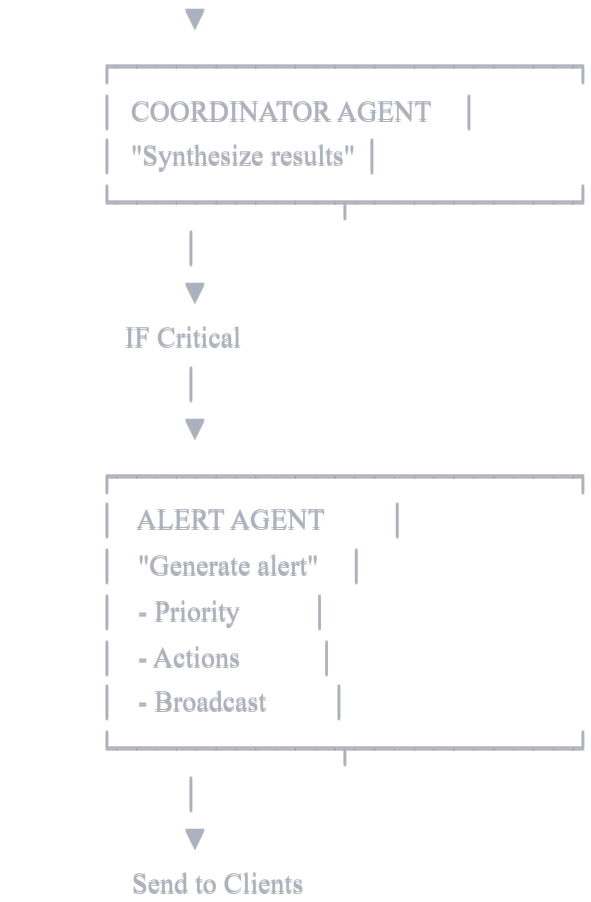
'sensor_update' ——▶ Update patient cards
Update vital stats

'ai_alert' ——▶ Show notification
Play sound (critical)
Update alerts panel

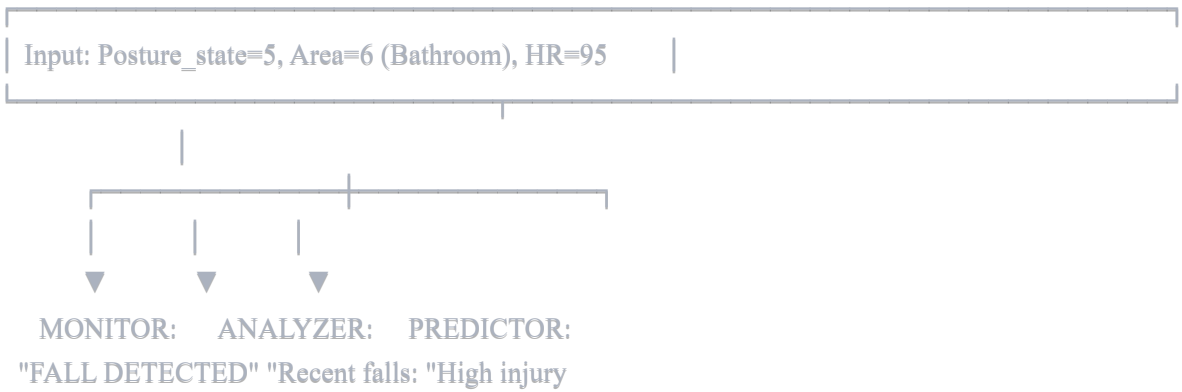
'analysis_result' ——▶ Display AI summary
Show risk scores
Update agent status

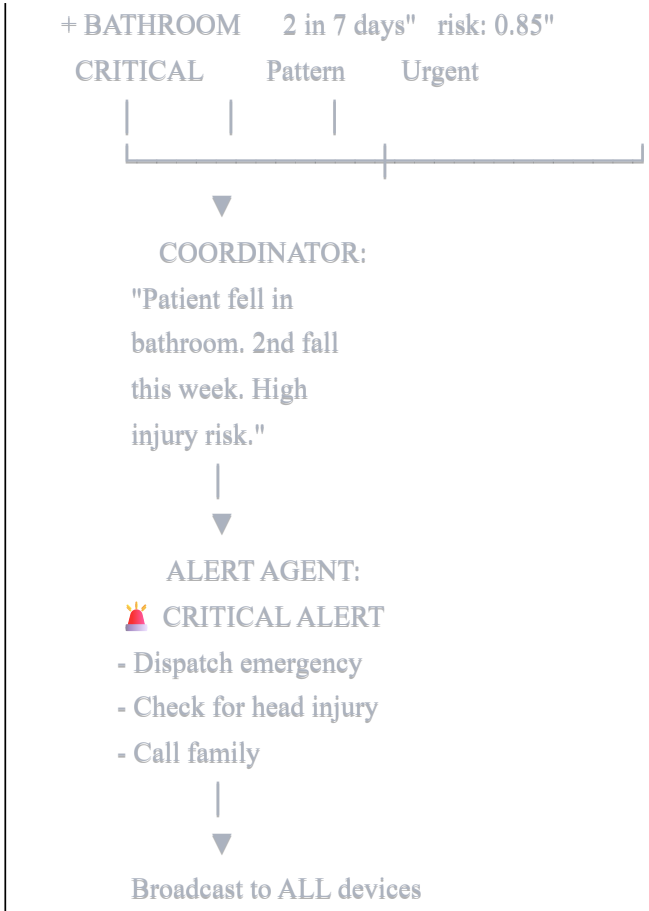
🧠 MULTI-AGENT COLLABORATION



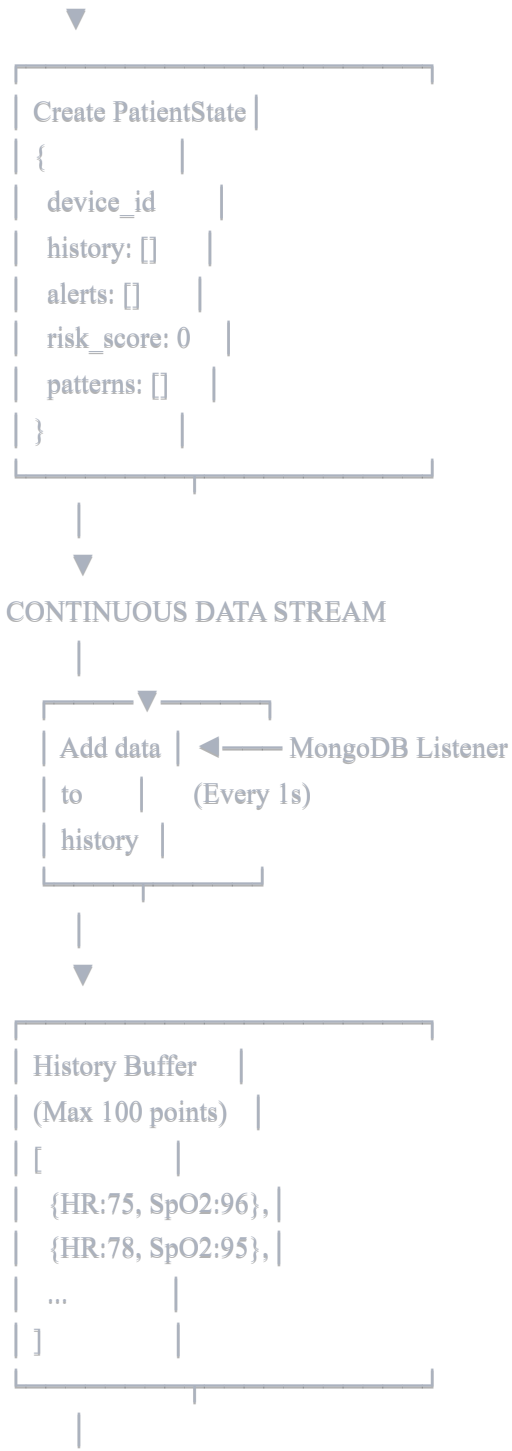


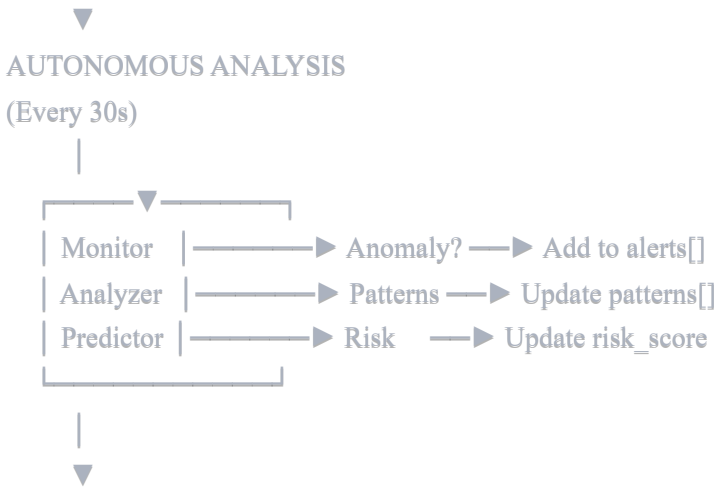
EXAMPLE SCENARIO: Fall Detection

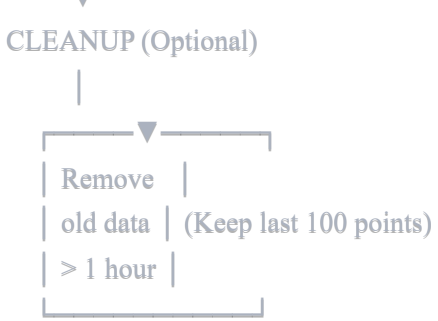


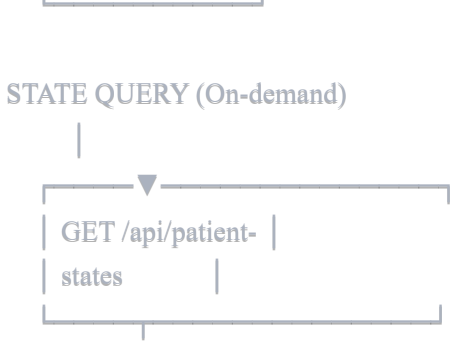


PATIENT STATE LIFECYCLE


Create PatientState
{
device_id
history: []
alerts: []
risk_score: 0
patterns: []
}
CONTINUOUS DATA STREAM
Add data to history
MongoDB Listener
(Every 1s)
History Buffer
(Max 100 points)
[
{HR:75, SpO2:96},
{HR:78, SpO2:95},
...
]

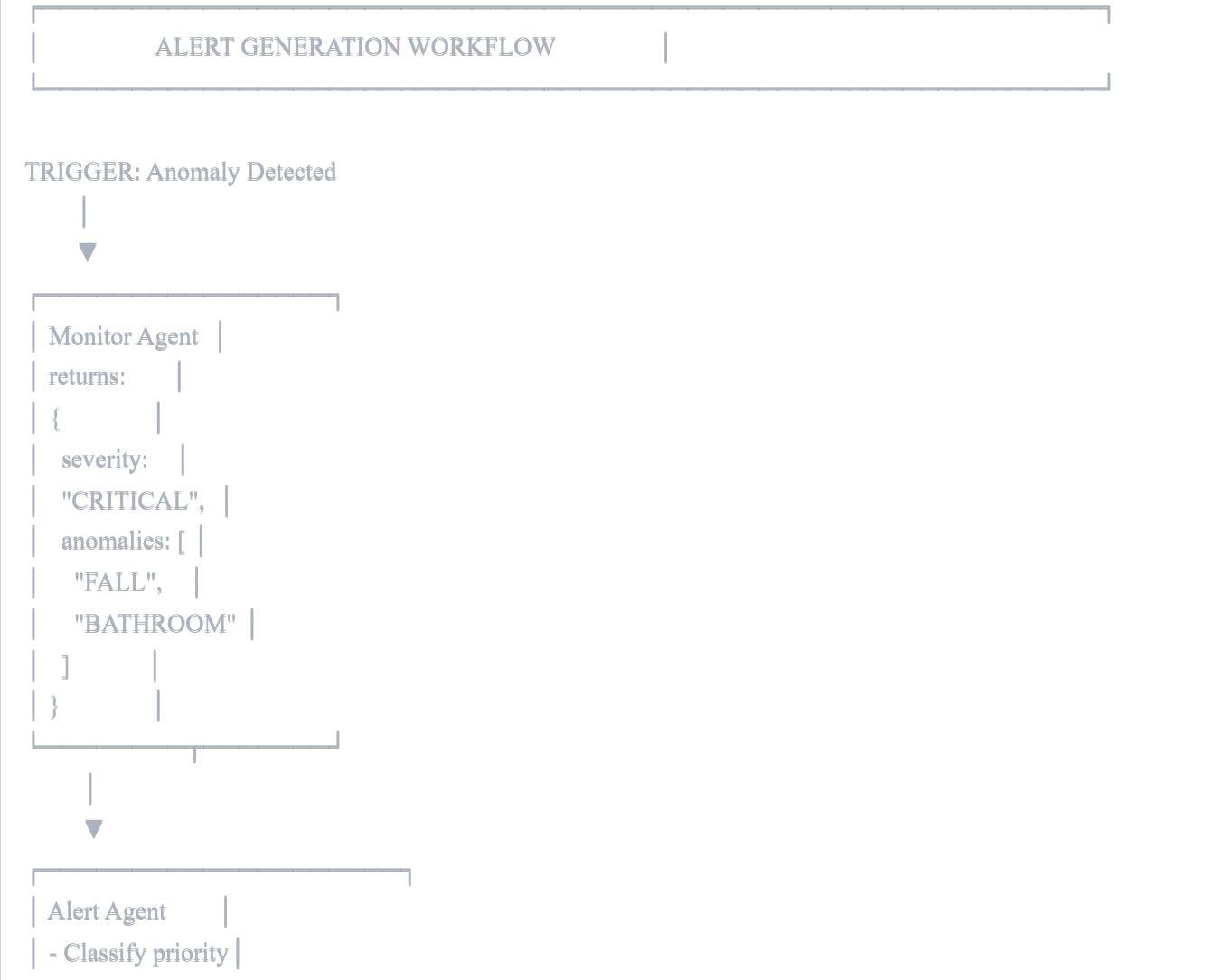

AUTONOMOUS ANALYSIS
(Every 30s)
Monitor
Analyzer
Predictor
Anomaly?
Patterns
Risk
Add to alerts[]
Update patterns[]
Update risk_score

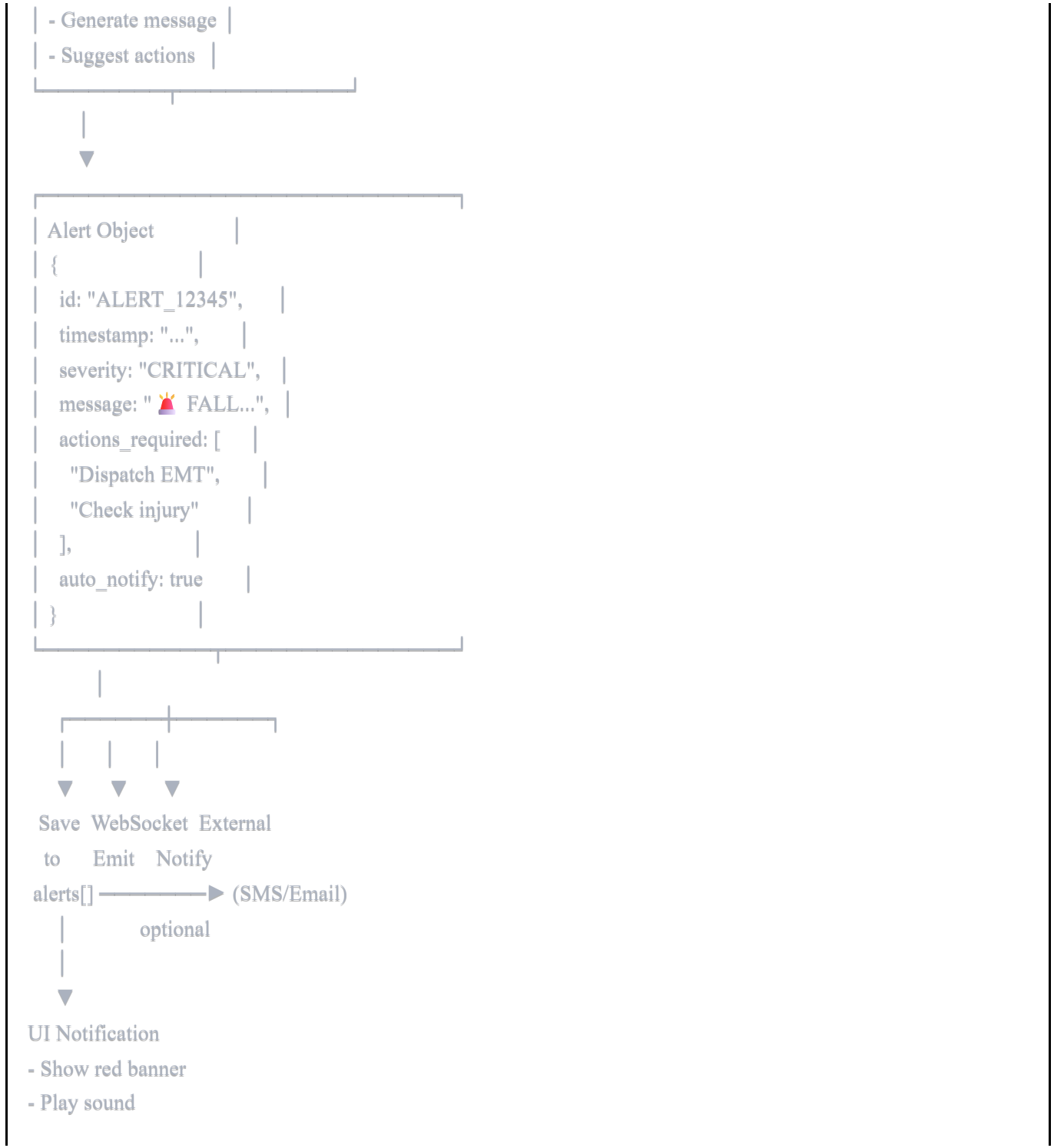

CLEANUP (Optional)
Remove
old data
> 1 hour
(Keep last 100 points)


STATE QUERY (On-demand)
GET /api/patient-
states
Return current state:
{
"device_id": "ABC123",
"risk_score": 0.42,
"data_points": 1523,

```
  "last_update": "2025-01-30T15:23:45",
  "patterns": [...]
}
```

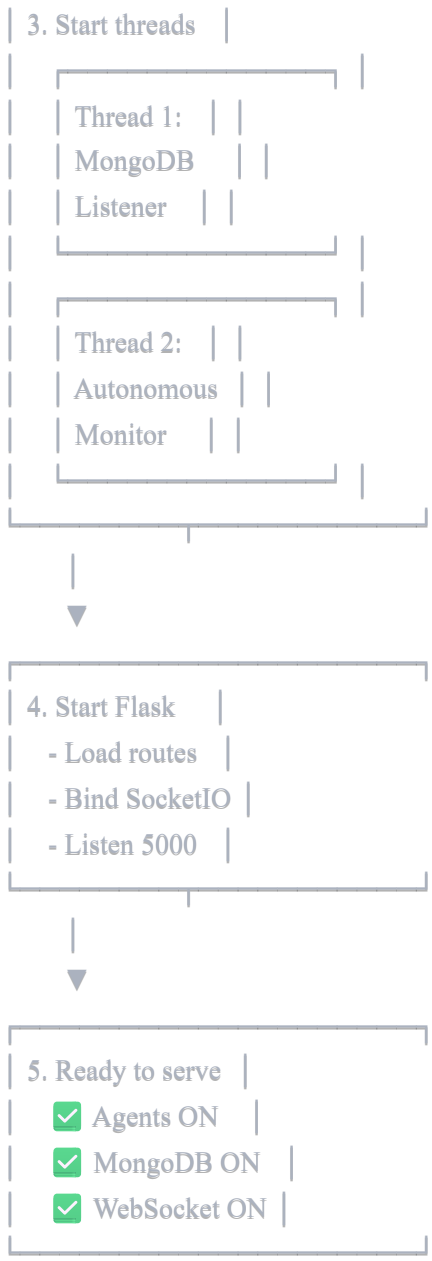
🚨 ALERT WORKFLOW




- Generate message
- Suggest actions
Alert Object
{
id: "ALERT_12345",
timestamp: "...",
severity: "CRITICAL",
message: " FALL...",
actions_required: [
"Dispatch EMT",
"Check injury"
],
auto_notify: true
}
Save to alerts[]
WebSocket Emit
External Notify
(SMS/Email)
optional
UI Notification
- Show red banner
- Play sound

- Log to alerts panel
- Update agent status

🔧 SYSTEM STARTUP SEQUENCE



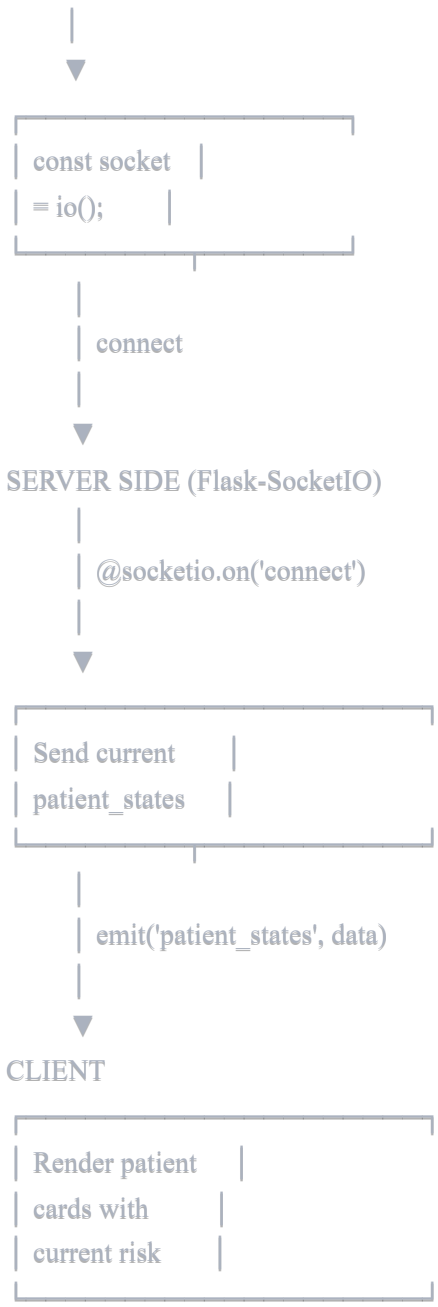
RUNTIME LOOPS (Concurrent):

Loop 1: MongoDB Listener

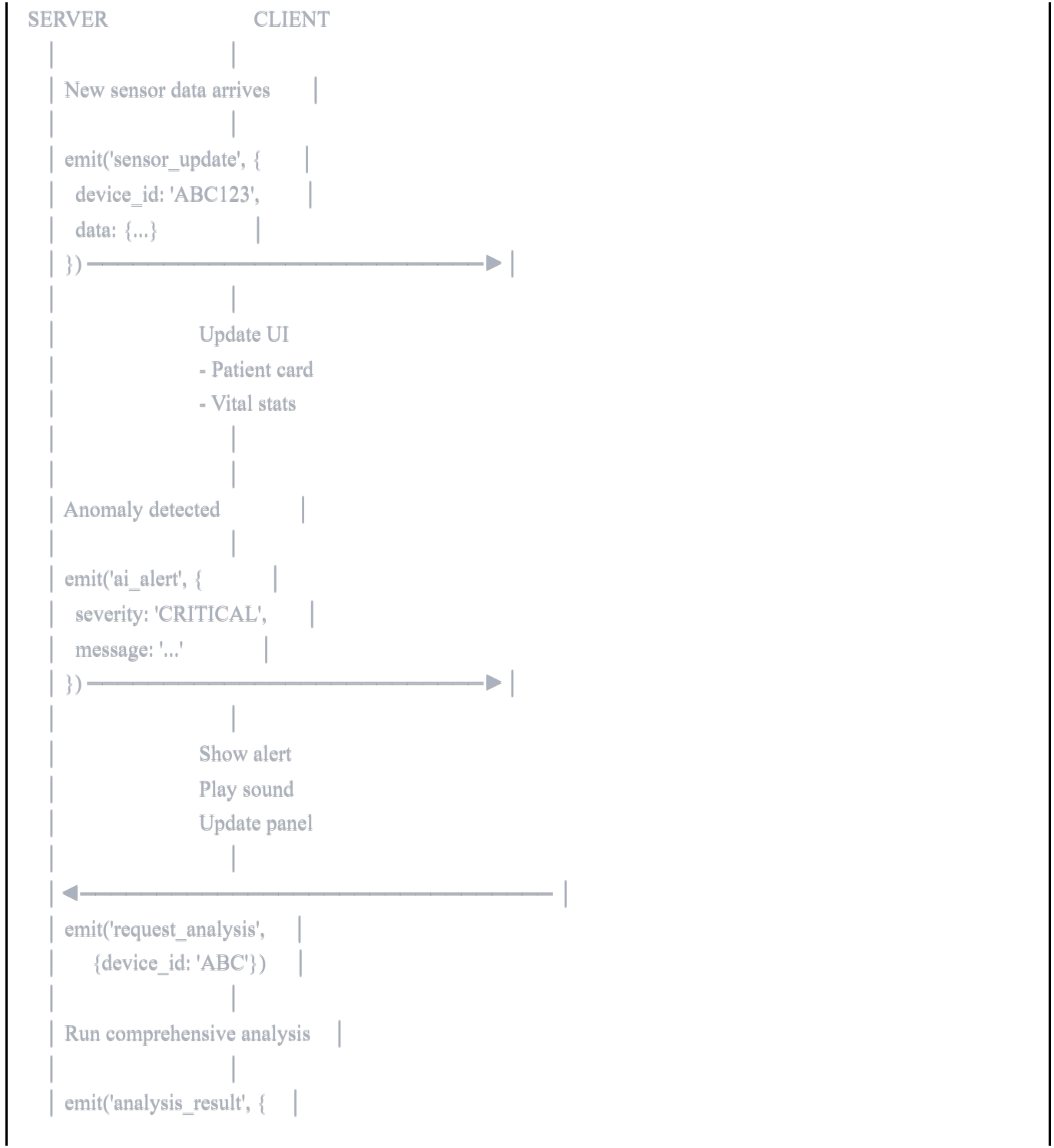
```
  while True:
    Check new documents (1s interval)
    Update patient states
    Emit sensor_update

Loop 2: Autonomous Monitor
  while True:
    For each patient:
      Run Monitor Agent
      Run Analyzer Agent
      Run Predictor Agent
      Coordinate results
      Generate alerts if needed
    Sleep 30s

Loop 3: WebSocket Server
  Event-driven:
    On connect → Send current states
    On request_analysis → Run analysis
    On disconnect → Cleanup
```

🏢 CLIENT-SERVER COMMUNICATION

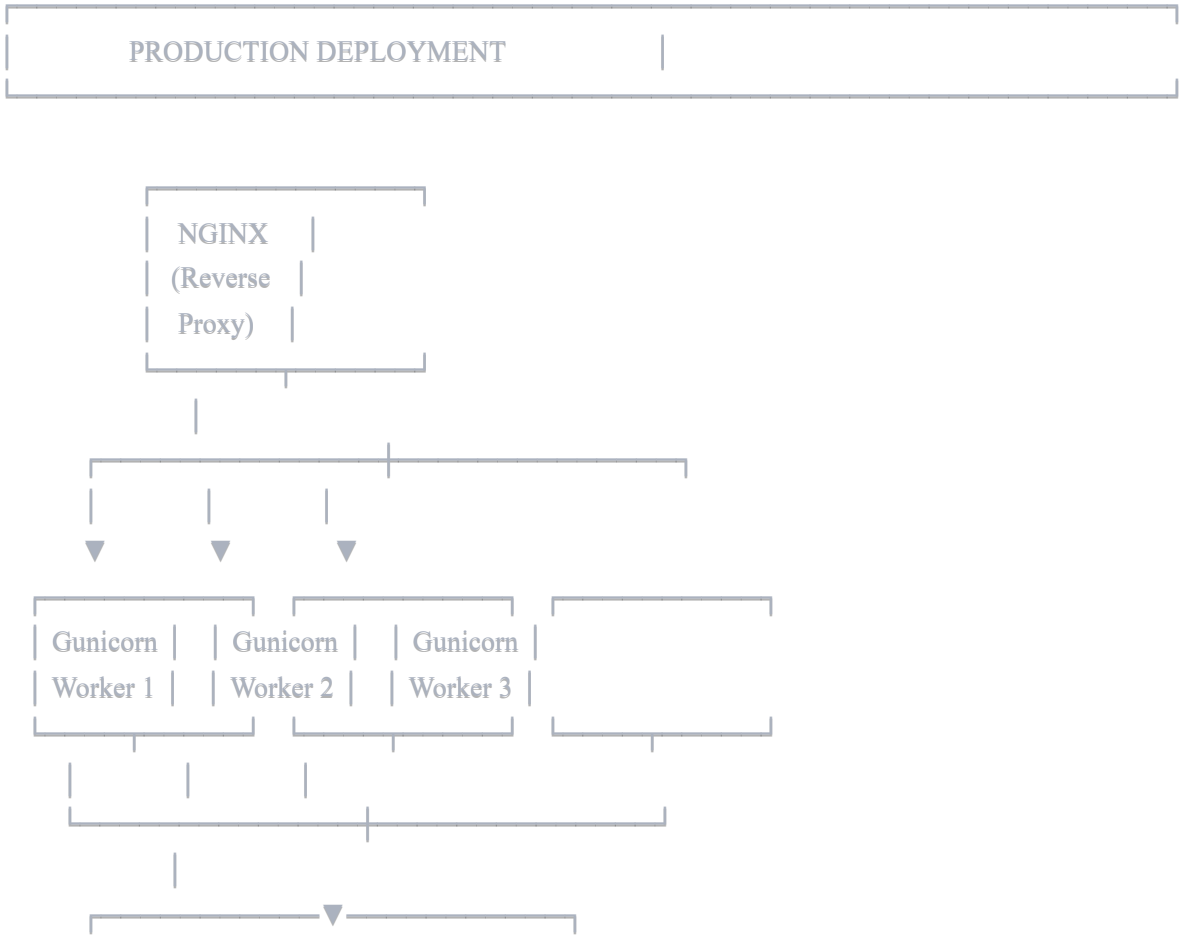
```
┌──────────────────────────────────────────────────────────────┐
│              WEBSOCKET COMMUNICATION FLOW                    │
└──────────────────────────────────────────────────────────────┘

CLIENT SIDE (Browser)
        │
        │ socket.io.js
```

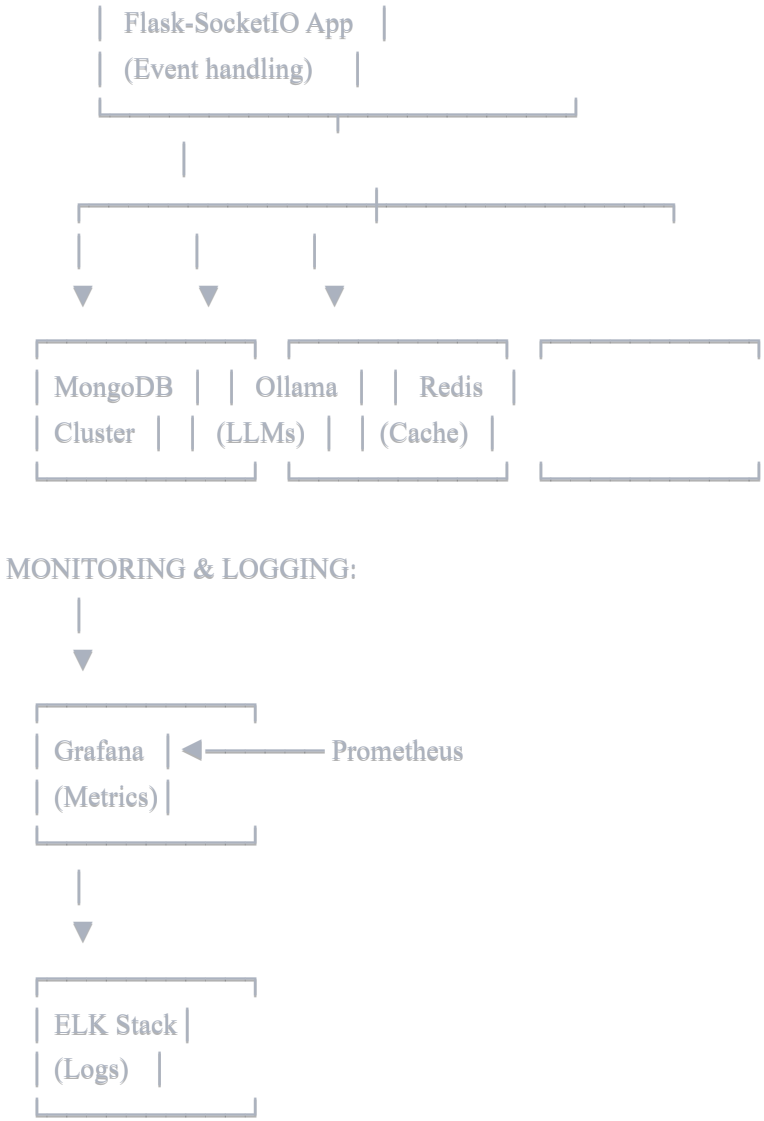


REAL-TIME UPDATES:


SERVER
CLIENT
New sensor data arrives
emit('sensor_update', {
device_id: 'ABC123',
data: {...}
})
Update UI
- Patient card
- Vital stats
Anomaly detected
emit('ai_alert', {
severity: 'CRITICAL',
message: '...'
})
Show alert
Play sound
Update panel
emit('request_analysis',
{device_id: 'ABC'})
Run comprehensive analysis
emit('analysis_result', {

🎯 DEPLOYMENT ARCHITECTURE





Gunakan diagram ini untuk memahami alur kerja system! 🚀