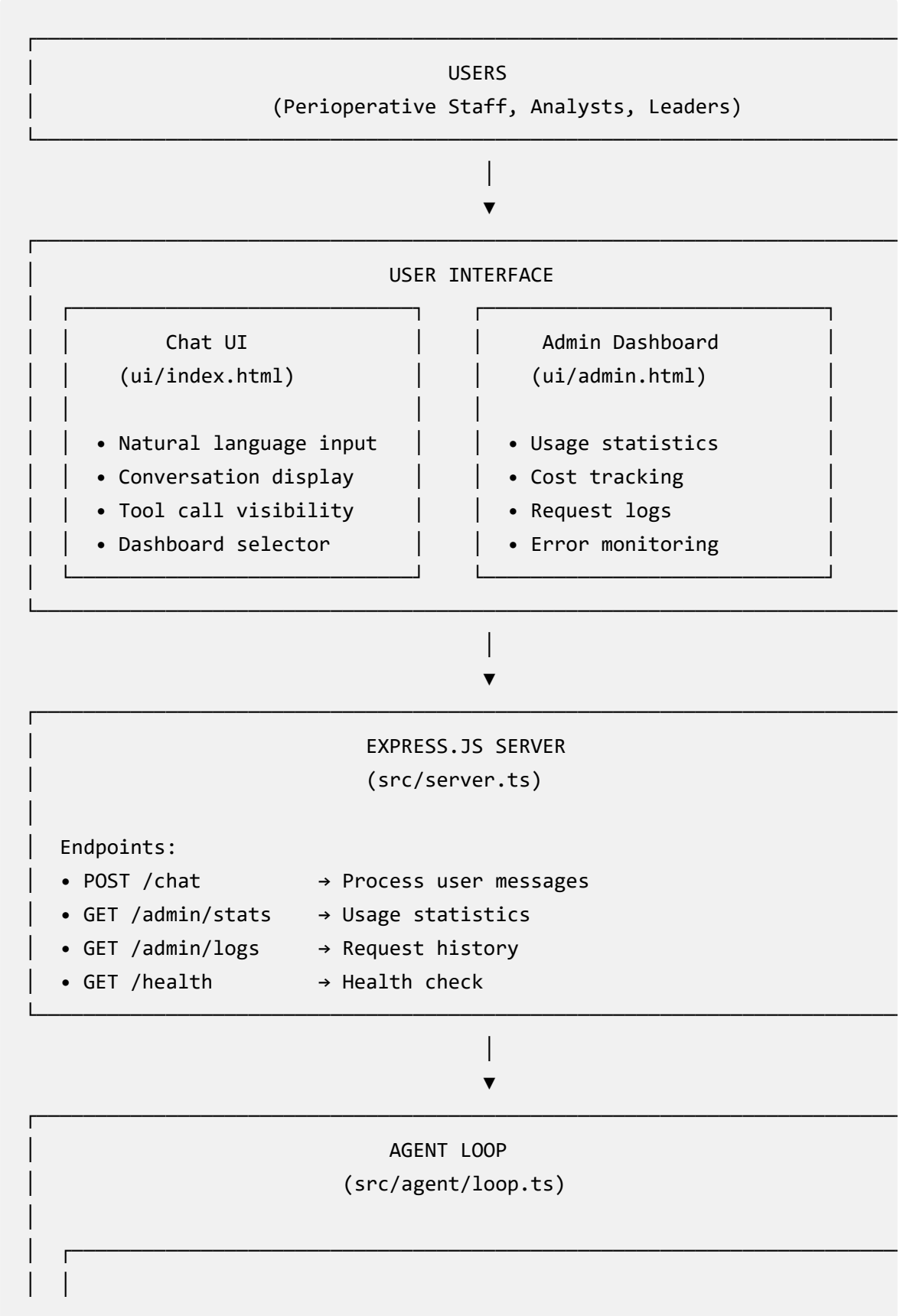
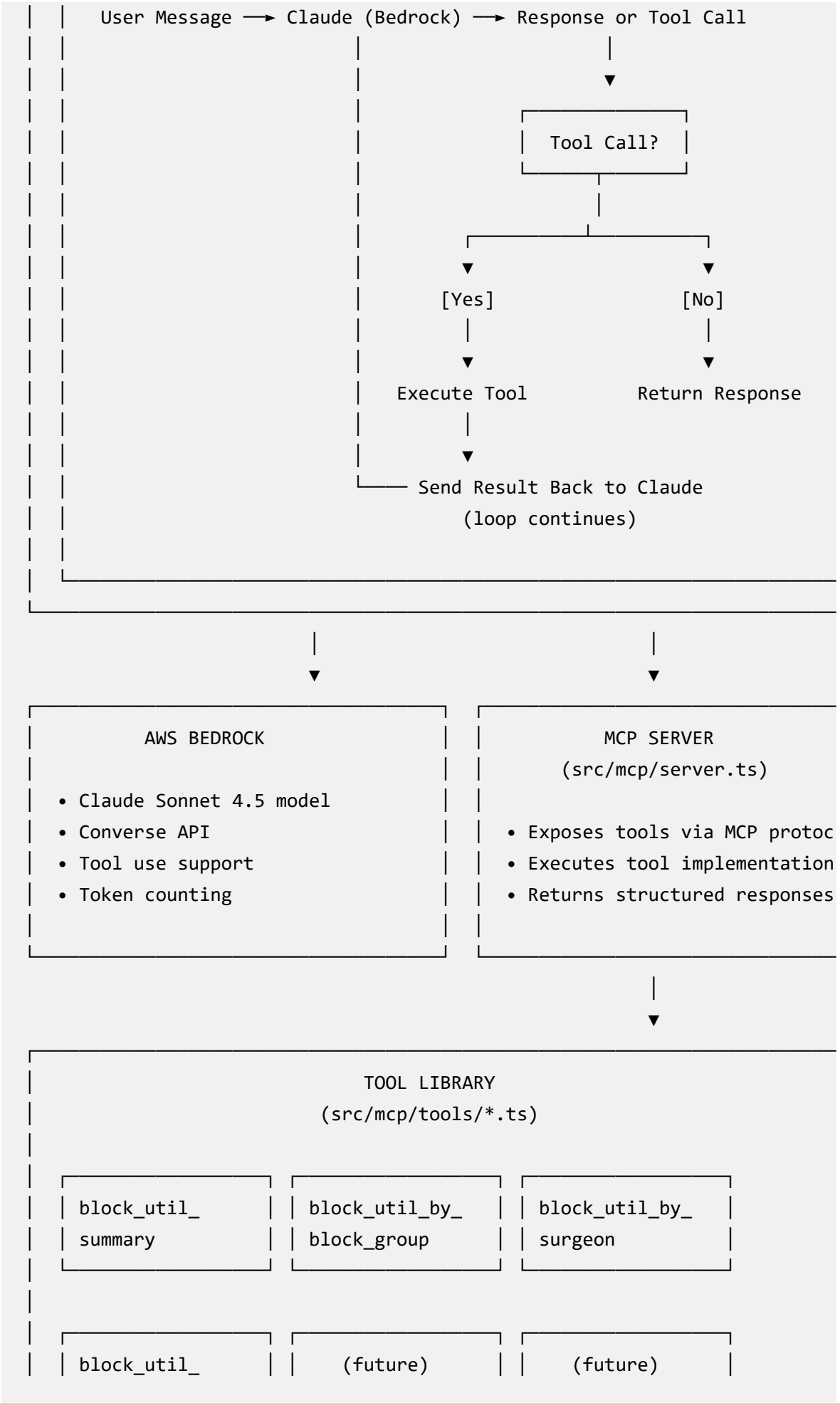
# Lumen Architecture

## High-Level System Architecture




User Message → Claude (Bedrock) → Response or Tool Call
Tool Call?
[Yes]
[No]
Execute Tool
Return Response
Send Result Back to Claude
(loop continues)
AWS BEDROCK
• Claude Sonnet 4.5 model
• Converse API
• Tool use support
• Token counting
MCP SERVER
(src/mcp/server.ts)
• Exposes tools via MCP protoc
• Executes tool implementation
• Returns structured responses
TOOL LIBRARY
(src/mcp/tools/*.ts)
block_util_ summary
block_util_by_ block_group
block_util_by_ surgeon
block_util_
(future)
(future)

```
│  │ drill_down       │ │                  │ │                  │
│  └──────────────────┘ └──────────────────┘ └──────────────────┘
│
│  Currently: Stubbed data
│  Future: Calls to LiveData Insights APIs
└──────────────────────────────────────────────────────────────────────────────
```

## Configuration Layer

```
┌──────────────────────────────────────────────────────────────────────────────
│                          CONFIGURATION LAYER
│
│  ┌──────────────────────────────────────────────────────────────────────────
│  │                          SITE CONFIG
│  │                    (src/config/site.json)
│  │
│  │  Per-client/deployment configuration:
│  │  • API endpoints & authentication
│  │  • LLM settings (model, region, tokens)
│  │  • Data sanitization rules
│  │  • Feature flags
│  │  • Data scope (facilities, service lines, date limits)
│  │  • Boundaries (what assistant can/cannot answer)
│  └──────────────────────────────────────────────────────────────────────────
│                                      │
│                                      ▼
│  ┌──────────────────────────────────────────────────────────────────────────
│  │                      DASHBOARD MANIFESTS
│  │              (src/config/dashboards/*.json)
│  │
│  │  Per-dashboard configuration:
│  │  • Which tools are available
│  │  • Out-of-scope topics
│  │  • Suggested questions
│  │
│  │  ┌──────────────────────┐  ┌──────────────────────┐
│  │  │  block-utilization   │  │     (future)         │
│  │  │       .json          │  │                      │
│  │  └──────────────────────┘  └──────────────────────┘
│  └──────────────────────────────────────────────────────────────────────────
│                                      │
│                                      ▼
│  ┌──────────────────────────────────────────────────────────────────────────
```

```
│   │                        SYSTEM PROMPT
│   │                   (src/agent/prompts.ts)
│   │
│   │  Dynamically built from:
│   │  • Base assistant instructions
│   │  • Site config boundaries
│   │  • Data scope constraints
│   │  • Available tools
│   └──────────────────────────────────────────────────────────────────────────┘
└──────────────────────────────────────────────────────────────────────────────┘
```

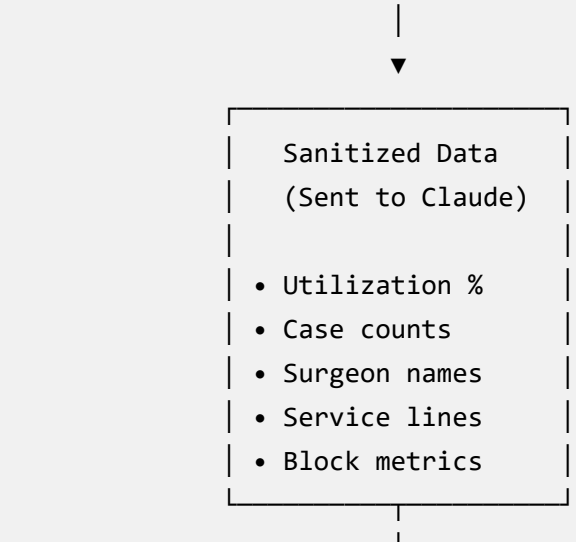
## PHI Protection & Data Sanitization

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                      PHI PROTECTION LAYER
│
│  Data flows through multiple sanitization checkpoints to ensure no PHI
│  (Protected Health Information) is exposed to the LLM or end users.
│
└──────────────────────────────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────────────────────┐
│
│                          ┌──────────────────────┐
│                          │   Insights API       │
│                          │   (Raw Data)         │
│                          │                      │
│                          │ • Case IDs           │
│                          │ • Patient MRNs       │
│                          │ • Surgeon names      │
│                          │ • Procedure details  │
│                          │ • Timestamps         │
│                          └──────────┬───────────┘
│                                     │
│                                     ▼
│   ┌──────────────────────────────────────────────────────────────────────┐
│   │                     SANITIZATION CHECKPOINT 1
│   │                        (Tool Execution)
│   │
│   │  Controlled by site.json → sanitization config:
│   │
│   │  ┌──────────────────────────────────────────────────────────────┐
│   │  │  {                                                           │
```

```
"sanitization": {
  "stripCaseIds": true,        ← Remove surgical case IDs
  "allowSurgeonNames": true,   ← Keep surgeon names (not PHI)
  "allowProcedureTypes": true, ← Keep procedure categories
  "redactPatientInfo": true    ← Remove all patient data
}
```

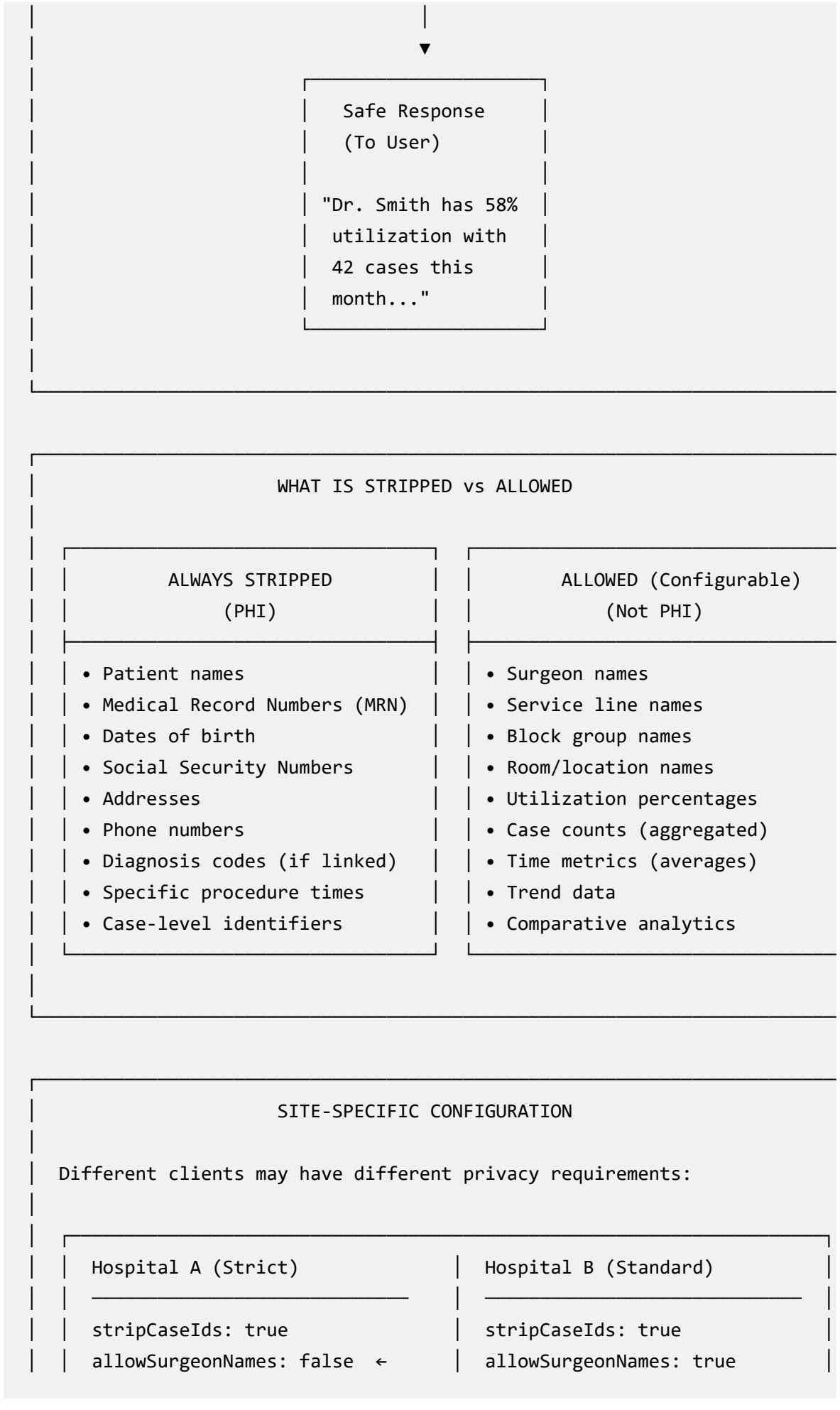
Actions:
✗ Strip case IDs (e.g., "CASE-12345" → removed)
✗ Remove patient MRNs, names, DOB
✗ Remove specific procedure times if configured
✓ Keep aggregated metrics (counts, percentages, averages)
✓ Keep surgeon names (configurable per site)
✓ Keep service line / block group names




SANITIZATION CHECKPOINT 2
(Response Filtering)

Before returning response to user:

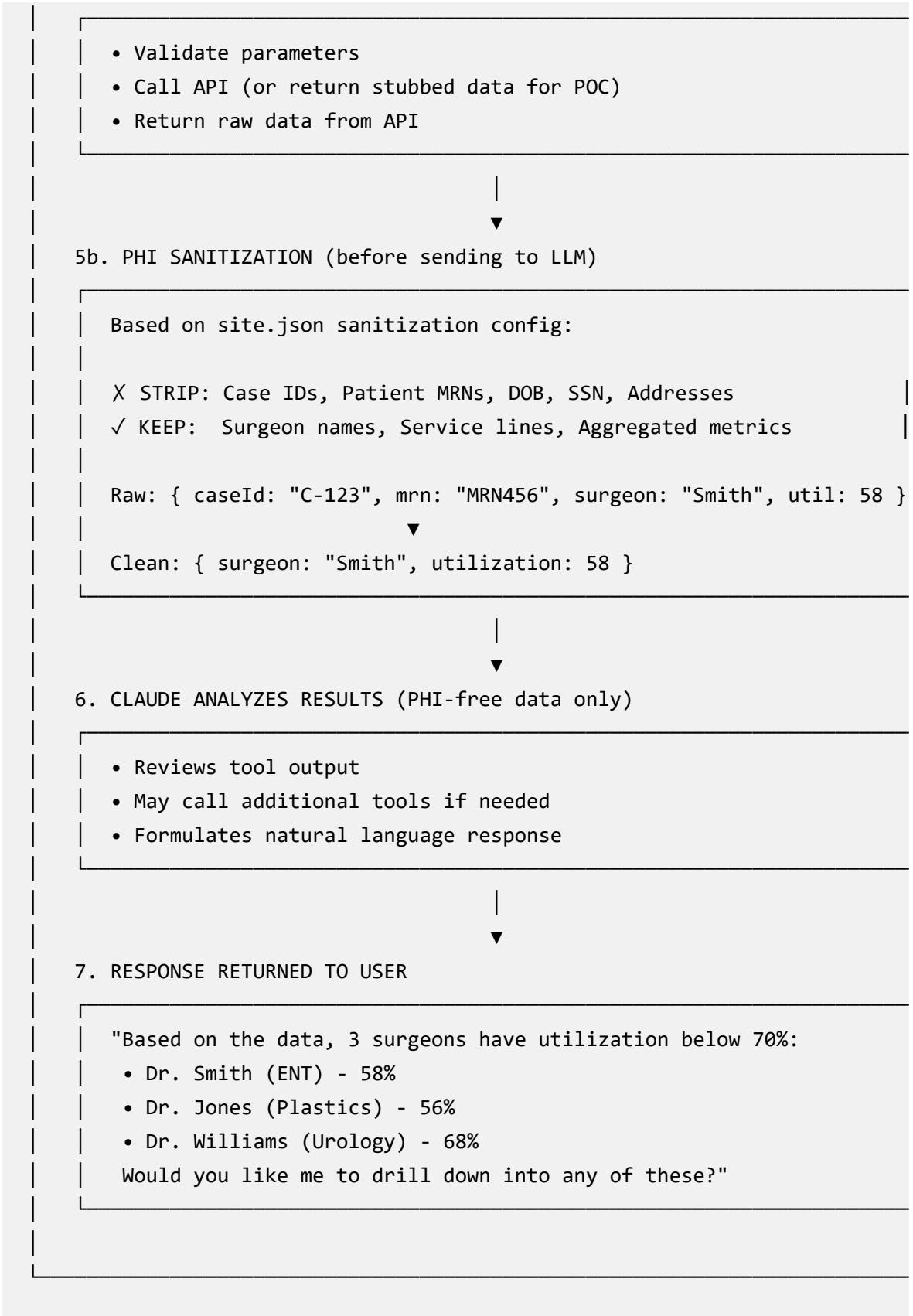
✗ Scan for accidentally leaked identifiers
✗ Remove any patterns matching MRN/SSN formats
✗ Filter stack traces or internal errors
✓ Allow aggregated analytics language


Safe Response
(To User)
"Dr. Smith has 58% utilization with 42 cases this month..."
WHAT IS STRIPPED vs ALLOWED
ALWAYS STRIPPED
(PHI)
• Patient names
• Medical Record Numbers (MRN)
• Dates of birth
• Social Security Numbers
• Addresses
• Phone numbers
• Diagnosis codes (if linked)
• Specific procedure times
• Case-level identifiers
ALLOWED (Configurable)
(Not PHI)
• Surgeon names
• Service line names
• Block group names
• Room/location names
• Utilization percentages
• Case counts (aggregated)
• Time metrics (averages)
• Trend data
• Comparative analytics
SITE-SPECIFIC CONFIGURATION
Different clients may have different privacy requirements:
Hospital A (Strict)
stripCaseIds: true
allowSurgeonNames: false ←
Hospital B (Standard)
stripCaseIds: true
allowSurgeonNames: true

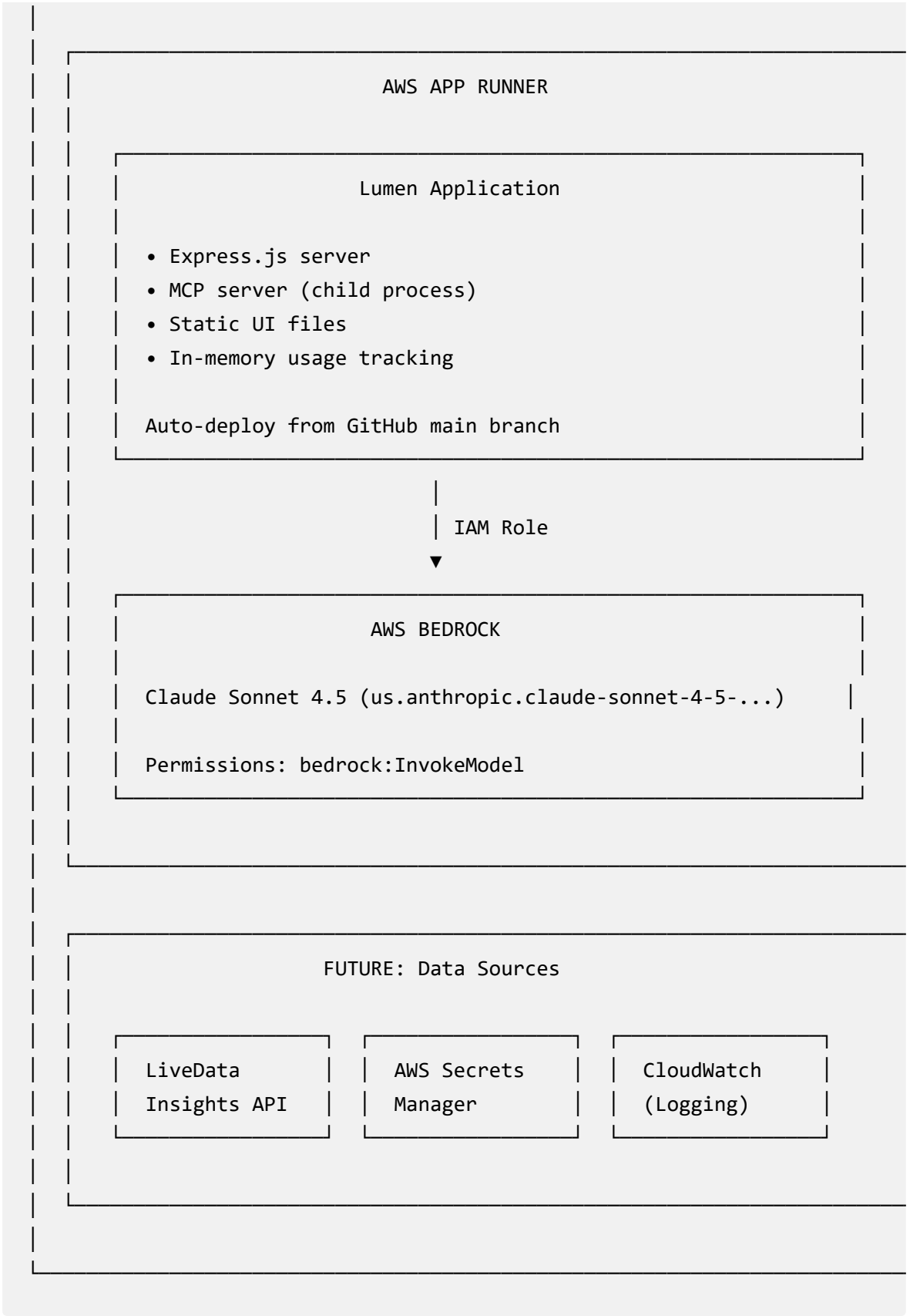
```
│  │  allowProcedureTypes: false ←     │  allowProcedureTypes: true      │
│  │  redactPatientInfo: true          │  redactPatientInfo: true        │
│  │                                   │                                 │
│  │  Response: "Service line A has    │  Response: "Dr. Smith has 58%   │
│  │  58% utilization..."              │  utilization in Orthopedics..." │
│  └──────────────────────────────────────────────────────────────────────┘
│
└──────────────────────────────────────────────────────────────────────────
```

## Data Flow

```
┌──────────────────────────────────────────────────────────────────────────
│                              DATA FLOW
│
│   1. USER ASKS QUESTION
│   ┌──────────────────────────────────────────────────────────────────────
│   │  "Which surgeons have utilization below 70% this month?"
│   └──────────────────────────────────────────────────────────────────────
│                                     │
│                                     ▼
│   2. SERVER RECEIVES REQUEST
│   ┌──────────────────────────────────────────────────────────────────────
│   │  POST /chat { message: "...", dashboard: "block-utilization" }
│   └──────────────────────────────────────────────────────────────────────
│                                     │
│                                     ▼
│   3. AGENT LOOP STARTS
│   ┌──────────────────────────────────────────────────────────────────────
│   │  • Load site config & dashboard manifest
│   │  • Build system prompt with boundaries
│   │  • Filter available tools by dashboard
│   │  • Send to Claude via Bedrock
│   └──────────────────────────────────────────────────────────────────────
│                                     │
│                                     ▼
│   4. CLAUDE DECIDES TO USE TOOL
│   ┌──────────────────────────────────────────────────────────────────────
│   │  Tool: block_util_by_surgeon
│   │  Params: { start_date: "2026-01-01", end_date: "2026-01-13" }
│   └──────────────────────────────────────────────────────────────────────
│                                     │
│                                     ▼
│   5. MCP SERVER EXECUTES TOOL
```




## Deployment Architecture




## File Structure

```
livedata_lumen/
├── src/
```

```
│   ├── server.ts                # Express server, API endpoints
│   ├── agent/
│   │   ├── loop.ts              # Agent loop, Bedrock integration
│   │   └── prompts.ts           # System prompt builder
│   ├── mcp/
│   │   ├── server.ts            # MCP server
│   │   ├── client.ts            # MCP client
│   │   ├── types.ts             # Tool type definitions
│   │   ├── tool-registry.ts     # Tool registration
│   │   └── tools/
│   │       ├── index.ts         # Tool exports
│   │       ├── block-util-summary.ts
│   │       ├── block-util-by-block-group.ts
│   │       ├── block-util-by-surgeon.ts
│   │       └── block-util-drill-down.ts
│   ├── config/
│   │   ├── site.json            # Site configuration
│   │   ├── site.ts              # Site config loader
│   │   └── dashboards/
│   │       ├── index.ts         # Dashboard manifest loader
│   │       └── block-utilization.json
│   └── services/
│       └── usage-tracker.ts     # Usage & cost tracking
├── ui/
│   ├── index.html               # Chat interface
│   └── admin.html               # Admin dashboard
├── dist/                        # Compiled output
├── PROJECT.md                   # Project overview
├── TODO.md                      # Enhancement backlog
├── ARCHITECTURE.md              # This file
├── package.json
└── tsconfig.json
```