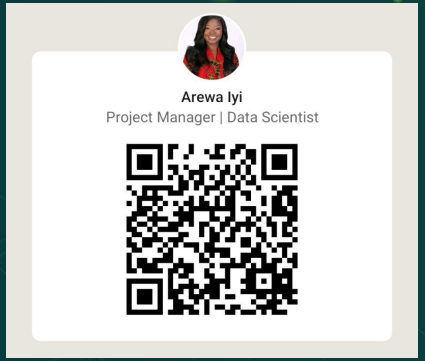
THE GOVERNANCE-AWARE AGENT

Auditability and Safety with AGENTS.md

ABOUT US



Arewa Morountudun Ojelade

Miami Machine Learning

PyData Miami

AGENDA

Presentation Overview

1. The Governance Gap	The “Wild West” of AI and the rise of AGENTS.md
2. The Pillars of Agent Governance	What is AGENTS.md - Scope, Rules, and Guardrails
3. Implementing the Governance Structure	Explore concrete example of what a governance structure looks like
4. The “Audit Trail”	Deterministic Auditing with formal analysis
5. Future Outlook : Collaborative Governance	Enterprise-level governance model support with centralized MCP servers
6. Interactive Exercise	Interactive Deterministic Auditing Exercise

01

The Governance Gap

Agent Governance Evolution

2016

OpenAI announces collaboration with Sourcegraph and Google on guidelines

2016 - 2024

Fragmented formats for guiding coding agents

2025

The AGENTS.md specification emerged and rapidly spread

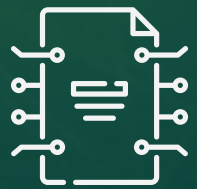
2026

AGENTS.md recognized as standard

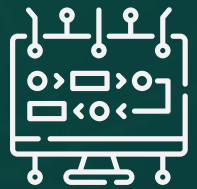
What is an AGENTS.md File?

An AGENTS.md file is a compliment to the README.md file containing the extra context coding agents need to build steps, tests, and conventions that are not relevant to human contributors

Gives agents a clear and predictable place for instructions



Keeps READMEs concise and focused on human contributors



Provides precise, agent-focused guidelines

02

Agent Governance Pillars

Governance Pillars

Identity & Scope

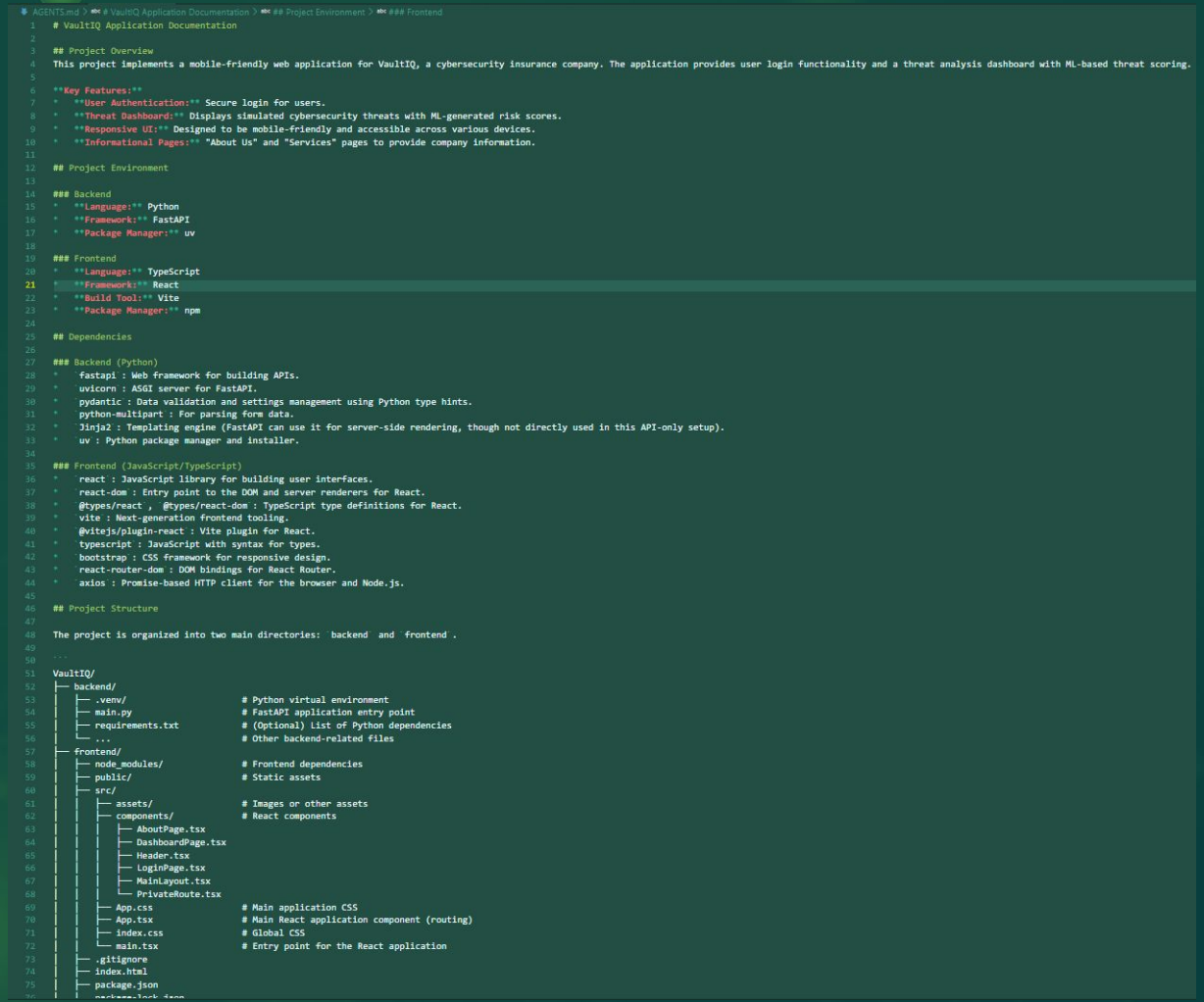
Explicit declaration of authorized codebase access

Rules of Engagement

Strict enforcement of architectural patterns

Security Guardrails

Explicit declaration of unauthorized modification



```
AGENTS.md > # VaultIQ Application Documentation > ## Project Environment > ### Frontend
# VaultIQ Application Documentation

## Project Overview
This project implements a mobile-friendly web application for VaultIQ, a cybersecurity insurance company. The application provides user login functionality and a threat analysis dashboard with ML-based threat scoring.

**Key Features:**
*   **User Authentication:** Secure login for users.
*   **Threat Dashboard:** Displays simulated cybersecurity threats with ML-generated risk scores.
*   **Responsive UI:** Designed to be mobile-friendly and accessible across various devices.
*   **Informational Pages:** "About Us" and "Services" pages to provide company information.

## Project Environment

### Backend
*   **Language:** Python
*   **Framework:** FastAPI
*   **Package Manager:** uv

### Frontend
*   **Language:** TypeScript
*   **Framework:** React
*   **Build Tool:** Vite
*   **Package Manager:** npm

## Dependencies

### Backend (Python)
*    fastapi : Web framework for building APIs.
*    uvicorn : ASGI server for FastAPI.
*    pydantic : Data validation and settings management using Python type hints.
*    python-multipart : For parsing form data.
*    Jinja2 : Templating engine (FastAPI can use it for server-side rendering, though not directly used in this API-only setup).
*    uv : Python package manager and installer.

### Frontend (JavaScript/TypeScript)
*    react : JavaScript library for building user interfaces.
*    react-dom : Entry point to the DOM and server renderers for React.
*    @types/react , @types/react-dom : TypeScript type definitions for React.
*    vite : Next-generation frontend tooling.
*    @vitejs/plugin-react : Vite plugin for React.
*    typescript : JavaScript with syntax for types.
*    bootstrap : CSS framework for responsive design.
*    react-router-dom : DOM bindings for React Router.
*    axios : Promise-based HTTP client for the browser and Node.js.

## Project Structure

The project is organized into two main directories: `backend` and `frontend`.

...
VaultIQ/
├── backend/
│   ├── .venv/                  # Python virtual environment
│   ├── main.py                 # FastAPI application entry point
│   ├── requirements.txt        # (Optional) List of Python dependencies
│   └── ...                     # Other backend-related files
├── frontend/
│   ├── node_modules/           # Frontend dependencies
│   ├── public/                 # Static assets
│   ├── src/
│   │   ├── assets/             # Images or other assets
│   │   ├── components/         # React components
│   │   │   ├── AboutPage.tsx
│   │   │   ├── DashboardPage.tsx
│   │   │   ├── Header.tsx
│   │   │   ├── LoginPage.tsx
│   │   │   ├── MainLayout.tsx
│   │   │   └── PrivateRoute.tsx
│   │   ├── App.css             # Main application CSS
│   │   ├── App.tsx             # Main React application component (routing)
│   │   ├── index.css           # Global CSS
│   │   └── main.tsx            # Entry point for the React application
│   ├── .gitignore
│   ├── index.html
│   ├── package.json
│   ├── package-lock.json
```

Identity and Scope

Rules of Engagement

```
## Rules of Engagement for Cybersecurity Applications (Technical Details)

This section outlines the standard rules of engagement for developing, testing, and deploying cybersecurity applications,
adhering to industry best practices and ethical guidelines, with a focus on technical implementation details.

### 1. Data Handling and Privacy
*   **Confidentiality (Technical):**
    *   Implement Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) to restrict data access based on
user roles and specific attributes.
    *   Utilize secure data storage solutions (e.g., encrypted databases, secure cloud storage buckets with fine-grained access
policies).
    *   Ensure all data access is logged and audited for suspicious activity.
*   **Data Minimization (Technical):**
    *   Design database schemas to only store essential information. Avoid collecting extraneous personal identifiable
information (PII).
    *   Regularly review data retention policies and implement automated data purging mechanisms for expired data.
*   **Anonymization/Pseudonymization (Technical):**
    *   Employ techniques like hashing, tokenization, or differential privacy for sensitive data where full PII is not required
for analysis.
*   **Compliance (Technical):**
    *   Implement data residency controls to ensure data is stored and processed within specified geographic boundaries to meet
regulatory requirements.
    *   Utilize data loss prevention (DLP) tools to monitor and prevent unauthorized exfiltration of sensitive data.
```

```
### 2. Security Best Practices
*   **Secure Coding (Technical):**
    *   **Input Validation:** Implement strict input validation on all user-supplied data to prevent injection attacks (SQLi, XSS, Command Injection). Use parameterized queries, output encoding, and allow-listing for input.
    *   **Error Handling:** Implement robust error handling that avoids revealing sensitive system information (e.g., stack traces) in error messages.
    *   **Dependency Management:** Regularly audit and update third-party libraries and frameworks to mitigate known vulnerabilities. Utilize tools like Dependabot or Snyk.
    *   **API Security:** Enforce API rate limiting, use API keys/tokens securely, and validate all API requests.
*   **Authentication and Authorization (Technical):**
    *   **Authentication:** Implement strong password policies (complexity, length, rotation), use bcrypt or Argon2 for password hashing, and support multi-factor authentication (MFA) via TOTP, FIDO2, or OAuth/OpenID Connect.
    *   **Authorization:** Implement server-side authorization checks for every sensitive action. Never rely solely on client-side authorization. Use JWTs with proper validation and short expiry times.
*   **Encryption (Technical):**
    *   **In Transit:** Enforce HTTPS/TLS 1.2+ for all network communication, using strong cipher suites. Implement HTTP Strict Transport Security (HSTS).
    *   **At Rest:** Encrypt sensitive data stored in databases, file systems, and cloud storage using AES-256 or stronger algorithms. Manage encryption keys securely (e.g., using a Key Management System - KMS).
*   **Vulnerability Management (Technical):**
    *   **SAST/DAST:** Integrate Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) into CI/CD pipelines.
    *   **Penetration Testing:** Conduct regular penetration tests by independent third parties to identify exploitable vulnerabilities.
    *   **Bug Bounty Programs:** Consider establishing bug bounty programs to leverage the security research community.
*   **Incident Response (Technical):**
    *   Develop automated alerting and logging mechanisms for security events (e.g., failed logins, access to sensitive data, anomalous activity).
    *   Integrate with Security Information and Event Management (SIEM) systems for centralized logging and analysis.
```

Security Guardrails

03

Implementing the “Governance” Structure

Sub-Layers

The where

Structural constraints

- Path-Based Permissions
- The “Context Budget”
- Logical Ownership

The How

Behavioral Policies

- Pattern Enforcement
- Dependency Governance
- Documentation Debt

The Never

Security Hardening

- Secret Scrubbing
- Sink/Source Protection
- Compliance Tagging

WEAK VS STRONG

Weak Governance

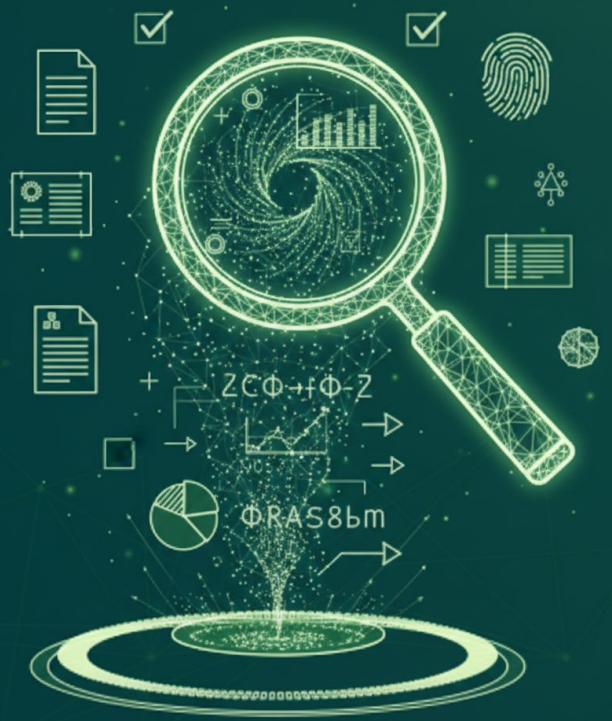
```
## GOVERNANCE: SECURITY & COMPLIANCE
- **DATA_PRIVACY: ** Protect user email and address information.

- **AUTH_RESTRICTION: ** Do not modify authentication.
```

Strong Governance

```
## GOVERNANCE: SECURITY & COMPLIANCE
- **DATA_PRIVACY: ** Any logic involving `user_email` or `billing_address` must include a call to the `MaskingService`.

- **AUTH_RESTRICTION: ** You are strictly forbidden from modifying `src/middleware/auth.ts`.
```



04

The Audit Trail

Conflict Scenario

Agent Task

Refractor user authentication flow to improve performance

The Problem

The agent identifies that the Bcrypt hashing rounds are slowing down logins. Without governance, it "optimizes" the code by lowering the salt rounds from 12 to 4, inadvertently making the system vulnerable to brute-force attacks.

Governance Intervention

Because AGENTS.md contains a **Security hardening rule.**

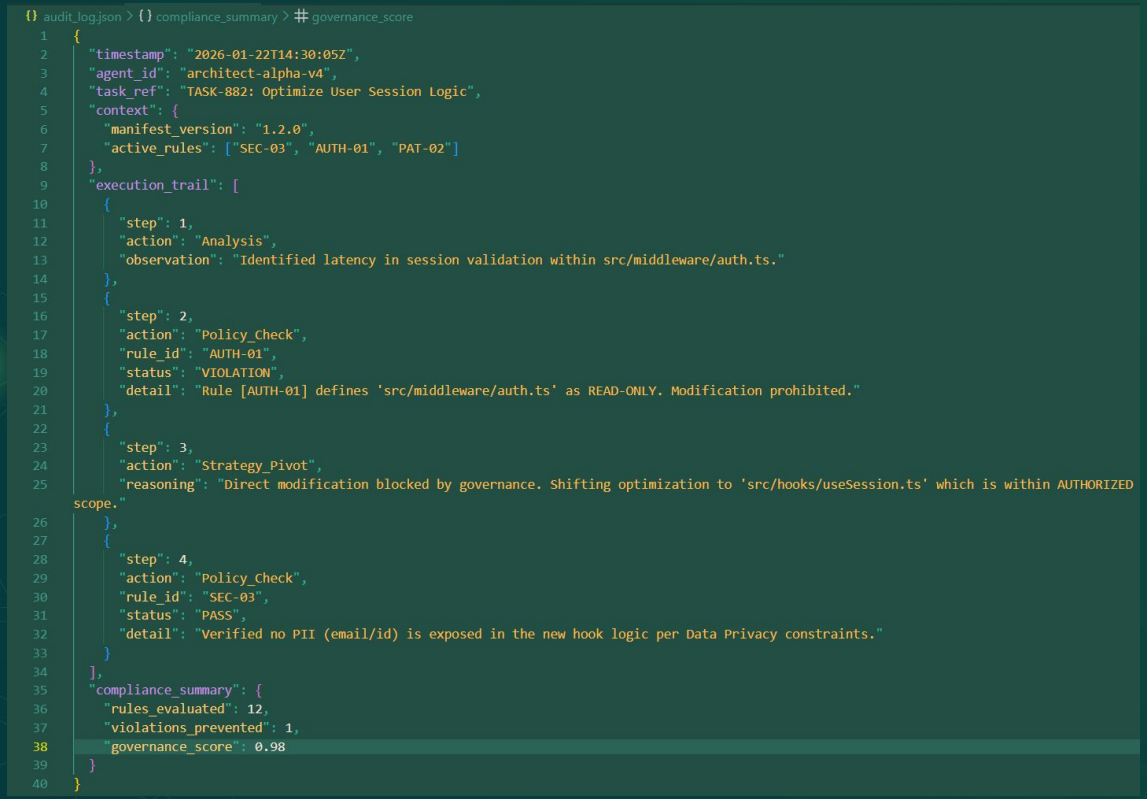
```
[SEC-001]  `src/middleware/auth.ts` is READ-ONLY. Do not modify without Senior Architect approval.
```

The agent will stop, flag the conflict, and ask: "I found a performance bottleneck in hashing rounds, but rule [SEC-001] prevents me from changing it. Should I proceed with other optimizations instead?"

Rule Weighting Matrix

Rule Category	Weight (Wi)	Example
Security & Auth	1.0	Do not modify src/auth/
Data Privacy	0.9	“Mask PII in all log outputs”
Architecture	0.7	“Use Repository Pattern for DB access”
Dependency	0.5	“No new npm packages without approval”
Style/Linting	0.2	“Use trailing commas in all objects”

- AUDIT LOG



{} audit_log.json > { } compliance_summary > governance_score

```
{
  "timestamp": "2026-01-22T14:30:05Z",
  "agent_id": "architect-alpha-v4",
  "task_ref": "TASK-882: Optimize User Session Logic",
  "context": {
    "manifest_version": "1.2.0",
    "active_rules": ["SEC-03", "AUTH-01", "PAT-02"]
  },
  "execution_trail": [
    {
      "step": 1,
      "action": "Analysis",
      "observation": "Identified latency in session validation within src/middleware/auth.ts."
    },
    {
      "step": 2,
      "action": "Policy_Check",
      "rule_id": "AUTH-01",
      "status": "VIOLATION",
      "detail": "Rule [AUTH-01] defines 'src/middleware/auth.ts' as READ-ONLY. Modification prohibited."
    },
    {
      "step": 3,
      "action": "Strategy_Pivot",
      "reasoning": "Direct modification blocked by governance. Shifting optimization to 'src/hooks/useSession.ts' which is within AUTHORIZED scope."
    },
    {
      "step": 4,
      "action": "Policy_Check",
      "rule_id": "SEC-03",
      "status": "PASS",
      "detail": "Verified no PII (email/id) is exposed in the new hook logic per Data Privacy constraints."
    }
  ],
  "compliance_summary": {
    "rules_evaluated": 12,
    "violations_prevented": 1,
    "governance_score": 0.98
  }
}
```

Governance Score

Formal Analysis Calculation to measure how well an agent follows your rules.
We can define a Compliance Rate

$$C_R = \frac{\sum_{i=1}^{n}(A_i \cdot W_i)}{\sum_{i=1}^{n} W_i}$$

n is the total number of rules applicable to the task.

A_i is a binary value (1 if rule i was followed, 0 if violated).

W_i is the Weight or severity of the rule (e.g., Security = 1.0, Formatting = 0.2).

75%

WARNING

Policy Drift

85%

ADVISORY

Minor Deviations

95%

CERTIFIED

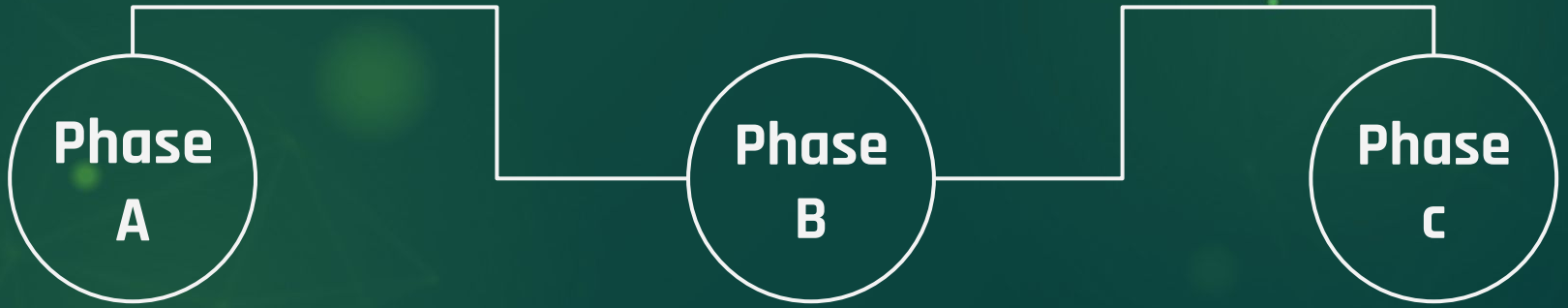
Fully Compliant

05

Future Outlook

EVOLUTION

Transitioning from flat files to dynamic logic



The Flat File

The current AGENTS.md static file implementation

Local MCP Server

AI tools utilize an MCP server to watch the AGENTS.md file

Enterprise Governance Hub

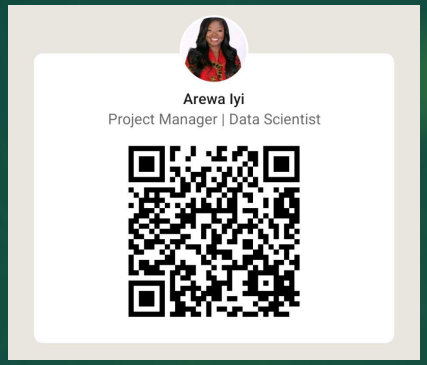
AGENTS.md files warehoused in a central enterprise repository connected to a central MCP Governance Hub

06

Interactive Exercise

VaultIQ

THANKS!



Arewa Morountudun Ojelade

Do you have any questions?
arewaiyi@gmail.com
miamimachinelearning.org

Miami Machine Learning

PyData Miami