AgentGuard Project Brief

1. Executive Summary

AgentGuard is an AgentSecOps control plane for teams deploying AI agents into real workflows. It provides a centralized system to authenticate users, manage multi-tenant workspaces, register and govern agents, enforce policy decisions in runtime, issue and rotate agent/API credentials, trigger kill-switch controls, and preserve tamper-evident audit evidence.

This product solves a high-impact enterprise gap: AI agents are being adopted faster than governance controls. Most teams can build an agent quickly, but they cannot prove who approved actions, how policy was enforced, or whether logs were tampered with. AgentGuard addresses this gap with operational controls, risk workflows, and compliance-grade observability.

AgentGuard is currently implemented as a working MVP with:

- `Next.js + Tailwind` web control plane
- `Node.js + Express + TypeScript` backend API
- `PostgreSQL + Prisma` data layer
- `JWT user auth + agent key auth`
- `Policy engine + simulation + real runtime enforcement`
- `Hash-chain audit ledger`
- `Enterprise extensions: SSO/SCIM, custom RBAC, policy sync, runtime integrations, playbooks, forensics, compliance packs, trust center`

2. Problem Statement

AI agents in production create new risk surfaces:

- Agents can execute high-risk actions (data deletes, transfers, privileged commands)
- API keys are often unmanaged or long-lived
- Teams lack runtime governance and approval workflows
- Security and compliance teams cannot confidently audit decisions
- Existing logs are often mutable and not trustworthy as legal/security evidence

Business consequences:

- Increased probability of unauthorized or dangerous actions
- Delayed enterprise sales due to failed security reviews
- Weak incident response due to low traceability
- Higher compliance and audit costs

3. Product Vision

Become the default safety and governance control plane for AI agent operations by sitting in the decision path between agent intent and agent execution.

Vision outcomes:

- Every meaningful agent action is policy-evaluated
- Every decision is explainable
- Every event is auditable and tamper-evident
- Every enterprise buyer can map controls to compliance frameworks

4. Target Users

- Security teams (SOC, AppSec, GRC)
- Platform and AI engineering teams
- Compliance and audit stakeholders
- Enterprise IT admins

5. Core Value Proposition

AgentGuard transforms AI-agent risk into controlled, auditable operations by combining:

- Preventive controls: policy engine + runtime enforcement + kill-switch
- Detective controls: anomaly signals + forensic replay
- Corrective controls: playbooks (disable/revoke/approval/webhook)
- Governance controls: signed approvals, RBAC, SSO/SCIM
- Assurance controls: compliance evidence packs + trust attestations

6. Product Scope (Implemented MVP + Advanced Layer)

6.1 Identity and Tenant Controls

- Email/password signup/login
- Multi-workspace tenancy
- Workspace membership with role model (`OWNER`, `MEMBER`)
- Custom RBAC override engine (`ALLOW`, `DENY`, default fallback)
- SSO provider registry with JIT provisioning controls
- SCIM token issuance and lifecycle provisioning endpoints

6.2 Agent Lifecycle and Access Controls

- Agent CRUD in workspace scope
- Agent status (`ACTIVE`, `DISABLED`)
- One-time raw key issuance; only key hash persisted
- API key rotation and revocation tracking
- Last-used timestamps
- Kill-switch endpoint that blocks future actions

6.3 Policy Governance

- Policy CRUD with JSON rules
- Policy modes (`STRICT`, `BALANCED`)

- Agent policy assignment
- Policy versioning
- Approval workflow (`DRAFT`, `PENDING_APPROVAL`, `APPROVED`, `REJECTED`)
- Signed policy approvals with payload hash + signature record
- Policy-as-code sync config and import runs

6.4 Runtime Enforcement and Decisioning

- Simulation endpoint for controlled testing
- Agent action endpoint using `X-Agent-Key`
- Runtime provider action endpoints:

- `OPENAI`
- `ANTHROPIC`
- `LANGCHAIN`
- `CREWAI`

- Policy evaluation with deny-overrides-allow logic
- Unknown action/tool behavior under strict mode

6.5 Risk, Approvals, and Response

- Risk scoring engine with anomaly signals
- Behavioral baseline tracking per `agent+tool+action`
- Drift, off-hours, and new-pattern signals
- Human approval queue for risky actions
- Automated response playbooks:

- Disable agent
- Revoke active keys
- Create approval request
- Notify webhook

- Playbook execution logs

6.6 Audit, Forensics, and Compliance

- Audit event storage with rich metadata
- Tamper-evident hash chain (`prev_hash`, `hash`)
- Queryable audit logs with filters and CSV export
- Forensic timeline replay and chain integrity checks
- Compliance evidence pack generation (SOC2/ISO27001/HIPAA/GDPR)
- Immutable digest (`SHA256`) per generated pack
- Public trust center endpoint + trust attestations

6.7 Deployment and Operations

- Local one-command startup (`dev:local`) with embedded PostgreSQL
- Docker local stack

- Private deployment compose profile
- Render deployment file (API + Postgres)
- Vercel notes (web)
- Deployment profile endpoint (region + private mode metadata)

7. How AgentGuard Works (End-to-End Flow)

1. User logs in and selects workspace.
2. Team registers an agent and receives a one-time API key.
3. Team creates/assigns policies and approves them.
4. Agent sends action request using `X-Agent-Key` (or runtime provider endpoint).
5. AgentGuard validates key, agent state, and active policy.
6. Policy engine returns decision (`ALLOW`/`BLOCK`) with reason/signals.
7. Risk score is computed using current signals and behavior baseline context.
8. If required, action is routed to human approval flow.
9. Event is written into hash-chain audit ledger.
10. Playbooks evaluate conditions and trigger automated responses if matched.
11. SIEM/webhook integrations can receive forwarded events.
12. Security/compliance users query logs, run forensic replay, and generate evidence packs.

8. Why This Product Is Useful

- Prevents uncontrolled agent actions before damage occurs
- Gives security teams direct control over AI runtime behavior
- Shortens enterprise procurement by providing governance artifacts
- Reduces incident response time through decision-level traceability
- Converts operational telemetry into audit-ready compliance evidence

9. Differentiation

- Runtime decision enforcement, not just static dashboards
- Signed policy approvals with version-aware evidence
- Tamper-evident ledger and replay for forensic integrity
- Unified security + governance + compliance workflow in one plane

10. Technical Architecture

10.1 Monorepo Structure

- `apps/web`: Next.js App Router UI
- `apps/api`: Express API (TypeScript)
- `packages/shared`: shared types

10.2 Backend Stack

- Node.js + Express + TypeScript
- Zod request validation
- Prisma ORM + PostgreSQL
- JWT user authentication
- Agent key hash verification

10.3 Frontend Stack

- Next.js App Router
- Tailwind CSS
- Component primitives: button, input, select, table, modal, drawer, tabs, toast, badge

10.4 Data and Storage

Primary entities include:

- users
- workspaces
- workspace_members
- workspace_role_permissions
- workspace_identity_providers
- workspace_scim_tokens
- agents
- agent_api_keys
- policies
- policy_versions
- policy_approval_signatures
- policy_git_sync_configs
- policy_sync_runs
- runtime_connections
- action_approval_requests
- workspace_playbooks
- playbook_executions
- audit_log_events
- audit_chain_state
- agent_action_baselines
- compliance_evidence_packs
- trust_attestations

11. Security Model

- User auth via bearer JWT
- Workspace-scoped authorization checks across all sensitive routes

- Agent auth via hashed key comparison (salted SHA256)
- Revoked/disabled agent enforcement
- SCIM auth via hashed bearer token lookup
- Signed policy approvals for non-repudiation signals
- Tamper-evident chain for audit integrity

12. API Surface Summary

Major route groups:

- `/auth/*` (including SSO)
- `/workspaces/*`
- `/agents/*`
- `/policies/*`
- `/policy-sync/*`
- `/runtime/*`
- `/simulate`
- `/agent/actions`
- `/audit-logs/*`
- `/forensics/*`
- `/approvals/*`
- `/playbooks/*`
- `/integrations/*`
- `/sso/*`
- `/scim/*`
- `/compliance/*`
- `/trust-attestations`
- `/public/trust-center`
- `/deployment/profile`

13. Frontend Experience

Implemented routes include:

- Marketing: `/`, `/pricing`, `/security`, `/trust-center`
- App: `/app`, `/app/login`, `/app/signup`, `/app/agents`, `/app/policies`, `/app/simulate`, `/app/approvals`, `/app/audit-logs`, `/app/settings`
- Advanced app pages:

- `/app/runtime`

- `/app/playbooks`

- `/app/forensics`

- `/app/compliance`

- `/app/identity`
- `/app/rbac`
- `/app/policy-sync`

14. Business and Commercial Angle

14.1 Buyer Pain Solved

- Security review blockers for AI deployments
- Lack of confidence in runtime controls
- Expensive, manual audit preparation

14.2 Revenue Potential

- B2B SaaS subscription with security/compliance premium tiers
- Pricing drivers:

- Number of agents
- Events per month
- Advanced controls (SSO/SCIM, private deployment, compliance packs)

14.3 Strong Enterprise Signals

- SSO/SCIM support
- Custom RBAC
- Signed approvals
- Private deployment mode
- Trust center artifacts

15. Go-To-Market Positioning

- Category: AgentSecOps / AI runtime governance
- Primary wedge: secure runtime policy control for production AI agents
- Expansion motion:

1. Start with one agent team
2. Expand to central security governance
3. Expand to compliance and audit operations

16. Risks and Mitigations

- Risk: false positives in anomaly/risk signals

- Mitigation: baseline adaptation + human approval fallback

- Risk: integration complexity

- Mitigation: modular route groups and provider-specific runtime connectors

- Risk: buyer concerns on data residency/private control

- Mitigation: deployment profile + private compose mode + region controls

17. Current Status

The codebase is runnable and tested with:

- Build passing for API and web
- API and web tests passing
- Migration and seed workflows operational
- Full demo credentials and sample enterprise controls seeded

18. Local Run and Demo

18.1 Run

1. `npm install`

2. `npm run dev:local`

18.2 Demo Credentials

- Admin: `admin@agentguard.demo`
- Member: `analyst@agentguard.demo`
- Password: `Admin123!ChangeMe`
- Demo agent key: `agk_demo_active_1234567890abcdef`
- Demo SCIM token: `scim_demo_token_1234567890abcdef`
- Demo SSO shared secret: `sso_demo_shared_secret_change_me`

19. Suggested Next Milestones

- Add enterprise SSO standards adapters (OIDC/SAML specifics)
- Add cryptographic signature algorithm upgrades (HMAC or asymmetric signatures)
- Extend runtime connectors and policy templates by vertical
- Add usage metering and billing instrumentation
- Add alerting connectors (PagerDuty/Jira/Slack incident workflows)

20. Final Positioning Statement

AgentGuard is not just an AI dashboard. It is an operational trust layer for AI agents: enforcing policy decisions in runtime, preserving forensic-quality evidence, and giving enterprises the governance controls required to ship agent automation safely at scale.