Systems

Corporate Strategy & Intelligence Dossier

Prepared for	Project-X Executive Leadership Team
Prepared by	Strategy & Compliance Office
Document date	October 24, 2025
Classification	Strictly Confidential – Do Not Distribute

Table of Contents

AI Governance Platform — Systems Hub

TL;DR

This hub curates subsystem runbooks that translate the platform's high-level vision into operational guidance.

Each article connects the strategic context from the **About** materials with the implementation details in the **Technical Specifications** so teams can plan, build, and run services coherently.

Why this section exists

The Systems documentation set bridges the gap between business intent and engineering execution. It:

- Explains how each platform subsystem delivers on the promises outlined in the .
- References implementation expectations from the to keep architecture, tooling, and governance aligned.
- Provides operational guidance—ownership, data flows, SLAs, and cross-service dependencies—for day-to-day platform management.

How the Systems hub relates to other documentation

- **Strategic grounding.** Use the and related market, roadmap, and risk briefs to understand *why* the subsystems exist and what outcomes they must deliver.
- **Implementation detail.** Pair each subsystem article with the matching section in the for API contracts, infrastructure choices, and coding standards.
- **Bidirectional updates.** Changes captured here should cascade back into the about and technical-specifications folders so strategy, architecture, and operations never drift apart.
- **Environment constraints.** All persistence layers run on the externally hosted PostgreSQL and MinIO services described in the . Operational steps that reference migrations, schema updates, or storage policies assume coordination through the DevOps pipeline against those managed services rather than direct instance administration.

Subsystem runbooks

Auth Service

TL;DR

- Manages identity verification, session lifecycle, and role-based authorization for every touchpoint across the AI Governance Platform.
- Built on the Express.js backend, it integrates tightly with the shared authentication libraries and enforces governance policies defined by the architecture team.

Mission-critical responsibilities

- Provide secure sign-in and token issuance aligned with the .
- Enforce RBAC policies that map to the stakeholder personas captured in the .
- Coordinate with the for encryption, secret storage, and compliance monitoring.

Operational guidance

- **Service ownership:** Platform security squad.
- **Data flows:** Issues JWTs, validates session cookies, and synchronizes user metadata with the shared PostgreSQL database.
- **SLAs:** 99.95% authentication availability, <200ms median login response.
- **Dependencies:** Relies on the user table managed by the Database Design spec and integrates with the Notification Service for multi-factor prompts.

Runbook checklist

1. Monitor authentication error rates and anomaly spikes in observability dashboards.
2. Rotate signing keys in accordance with the Security Implementation playbooks.
3. Review RBAC policy drift with compliance partners each release.

Governance Engine

TL;DR

- Automates control evaluations, risk scoring, and workflow orchestration that power the platform's compliance insights.
- Serves as the orchestration heart that connects frameworks, evidence ingestion, and task creation.

Mission-critical responsibilities

- Execute control logic defined in the and elaborated within the .
- Align evaluation cadences with regulatory expectations summarized in the .
- Produce audit-ready logs that meet the acceptance criteria.

Operational guidance

- **Service ownership:** Governance engineering squad.
- **Data flows:** Ingests framework metadata, triggers evidence probes, evaluates control logic, and publishes outcomes to the task queue.
- **SLAs:** Complete scheduled control runs within hourly windows; maintain deterministic results across identical inputs.
- **Dependencies:** Requires reliable input from the Framework Service, Evidence Repository, and Notification Service for downstream alerts.

Runbook checklist

1. Review nightly control run summaries and investigate failing evaluations.
2. Validate scoring algorithms after any framework or regulatory update.
3. Confirm audit logs are persisted and replicated per the DevOps policies.

Framework Service

TL;DR

- Curates regulatory frameworks, control catalogs, and cross-mappings that feed every governance workflow.
- Ensures authoritative content, versioning, and lifecycle management for compliance programs.

Mission-critical responsibilities

- Maintain framework metadata, control mappings, and hierarchy in line with the .

- Reflect roadmap priorities articulated in the as new frameworks roll out.
- Synchronize schema expectations with the .

Operational guidance

- **Service ownership:** Compliance content squad.
- **Data flows:** Imports regulatory datasets, version-controls updates, exposes REST endpoints for the Governance Engine, and publishes change events to the Notification Service.
- **SLAs:** Publish critical framework updates within 48 hours of policy release; guarantee backward compatibility for at least one release cycle.
- **Dependencies:** Relies on Database migrations coordinated with DevOps and ties into the Evidence Repository to ensure control evidence references remain consistent.

Runbook checklist

1. Vet new or updated framework content with legal and compliance reviewers before publishing.

2. Submit Prisma migration packages through the DevOps pipeline so they run against the provider-managed staging database before production rollout.

3. Update change logs and notify downstream teams of breaking changes.

Evidence Repository

TL;DR

- Provides secure, compliant storage for documentary and automated evidence that supports control attestations.
- Balances accessibility for auditors with strict retention, encryption, and chain-of-custody requirements.

Mission-critical responsibilities

- Implement storage practices aligned with the and standards.
- Support evidence collection journeys envisioned in the .
- Serve deterministic, versioned evidence snapshots to the Governance Engine and downstream audit exports.

Operational guidance

- **Service ownership:** Evidence management squad.
- **Data flows:** Accepts uploads and automated probe results, writes metadata to PostgreSQL, stores binary assets in MinIO, and exposes signed URLs for authorized retrieval.
- **SLAs:** Guarantee evidence availability within minutes of ingestion and sustain 7-year retention with immutable audit logs.
- **Dependencies:** Depends on encryption keys managed by the Auth Service and infrastructure policies maintained via DevOps & Infrastructure playbooks.

Runbook checklist

1. Validate retention schedules and legal hold configurations quarterly.
2. Audit MinIO bucket policies and access logs for anomalies.
3. Test retrieval workflows after each deployment to ensure signed URL issuance remains intact.

Document & Media Upload Service

TL;DR

- Issues presigned MinIO URLs and compression directives so teams can ingest evidence quickly without exposing backend credentials.
- Enforces metadata validation, image compression, and checksum verification before promoting uploads to the Evidence Repository.

Mission-critical responsibilities

- Validate upload intent, RBAC scopes, and data classification prior to generating presigned URLs.
- Orchestrate compression, malware scanning, and metadata persistence that keep evidence immutable and auditable.
- Publish lifecycle events that downstream systems (Evidence, Governance, Tasking) rely on for automation and reporting.

Operational guidance

- **Service ownership:** Evidence management squad.
- **Data flows:** Accepts metadata, issues presigned URLs, processes completion callbacks, and stores metadata alongside MinIO object keys.

- **SLAs:** Presigned URL issuance <150 ms median; compression and metadata finalization within 2 minutes of upload completion.
- **Dependencies:** Relies on RBAC enforcement, MinIO availability, ClamAV scanning, and Notification service for escalation workflows.

Runbook checklist

1. Monitor presigned URL failure rates and compression backlogs; escalate when thresholds breach runbook tolerances.

2. Review rejected uploads weekly to fine-tune MIME allowlists and file size policies.

3. Validate malware scanning signatures and quarantine automation after each infrastructure update.

Notification Service

TL;DR

- Drives multi-channel communication—email, Slack, and in-app alerts—to keep stakeholders aligned with governance activity.
- Operates on an event-driven pipeline that reacts to subsystem signals and user preferences.

Mission-critical responsibilities

- Implement message orchestration patterns described in the and .
- Reinforce change management commitments defined in the .
- Observe security and privacy controls from the when handling personally identifiable information.

Operational guidance

- **Service ownership:** Communications platform squad.
- **Data flows:** Consumes events from the Governance Engine and Task Service, applies routing rules, and dispatches through provider adapters with delivery analytics captured in PostgreSQL.
- **SLAs:** Deliver high-priority alerts within 60 seconds; ensure message deduplication and preference enforcement across channels.
- **Dependencies:** Relies on provider credentials managed via DevOps secrets and respects throttling policies defined in Integration Architecture.

Runbook checklist

1. Monitor provider health metrics and failover to backup channels when latency exceeds thresholds.

2. Reconcile delivery receipts with task and evidence states weekly to ensure notifications align with system reality.

3. Review template catalog for outdated messaging during each release planning cycle.

Admin & Configuration System

TL;DR

- Governs tenant provisioning, global settings, and integration lifecycle management through the shared admin control plane.
- Embeds workflows in the Governance Engine UI to coordinate platform-wide policy and configuration changes.

Mission-critical responsibilities

- Execute tenant onboarding and delegation journeys aligned with the and RBAC guardrails.
- Enforce security posture defined in the , including secrets management and administrative access controls.
- Coordinate integration governance patterns from the to keep probes and connectors in sync across tenants.

Operational guidance

- **Service ownership:** Platform operations squad.
- **Data flows:** Automates tenant provisioning, publishes configuration events, orchestrates integration authorization, and emits audit telemetry.
- **SLAs:** Complete tenant bootstrap within 30 minutes of approval; replicate configuration changes to dependent services within 5 minutes.
- **Dependencies:** Relies on Probe Management for scheduling automation, Governance Engine for UI embedding, and External Integrations for connector catalog updates.

Runbook checklist

1. Review pending provisioning requests and ensure verification artifacts meet compliance standards.

2. Audit configuration change approvals weekly and reconcile with audit logs for accuracy.

3. Validate integration credentials ahead of expiry and coordinate rotations through the control plane.

Task Service

TL;DR

- Manages remediation and follow-up work generated by governance activities, ensuring accountability and audit visibility.
- Provides the operational backbone for collaboration across compliance, engineering, and business stakeholders.

Mission-critical responsibilities

- Fulfil task orchestration responsibilities defined in the .
- Reflect organizational workflows highlighted in the and roadmap milestones set in the .
- Enforce coding standards and workflow automation patterns described in the .

Operational guidance

- **Service ownership:** Workflow automation squad.
- **Data flows:** Consumes events from the Governance Engine, persists task assignments and states, emits notifications, and exposes APIs for frontend task boards.
- **SLAs:** Persist new tasks within 5 seconds of triggering event; deliver state changes to dependent services in near real-time.
- **Dependencies:** Integrates with Auth for ownership permissions, Notification Service for updates, and DevOps pipelines that execute automated migrations against the externally hosted PostgreSQL service.

Runbook checklist

1. Validate that task state transitions comply with governance policies every sprint.
2. Audit automation rules and escalations for runaway loops or misrouted work.
3. Review analytics dashboards to ensure SLA adherence and backlog health.

Maintaining this hub

- Treat subsystem runbooks as living documents—update them alongside code or infrastructure changes.
- Cross-link new or changed content to both the About and Technical Specification sections when dependencies shift.
- Keep summaries short and actionable so readers can triage responsibilities quickly.

User Management System

Location: /server/src/modules/auth

TL;DR

The user management system anchors identity, authentication, and session governance for the AI Governance Platform.

It lives in server/src/modules/auth, coordinating JWT issuance, Casbin enforcement, and user lifecycle workflows.

This guide walks through the service structure, login and recovery flows, administrative tooling, and the client integrations that depend on it.

Backend Specification

Backend Location & Directory Layout

The User Management backend lives under server/src/modules/auth, following the platform's feature-module convention where HTTP entry points, business logic, and infrastructure concerns are co-located.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L52-L86■

```
server/src/modules/auth/
■■■ controllers/        # Express handlers for /api/v1/auth endpoints
■■■ services/           # Registration, login, token rotation, and email orchestration
■■■ repositories/       # Prisma data access for users, sessions, and roles
■■■ middleware/         # JWT validation, rate limiting, MFA hooks
■■■ emails/             # Nodemailer templates and delivery helpers
■■■ routes.ts           # Mounts the /api/v1/auth router into the application
```

Service Responsibilities & Collaborators

Within this module the Auth Service owns identity storage, credential verification, token lifecycle, and orchestration of RBAC, email, and audit pipelines mandated by the platform architecture.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L140■■F:docs/03-systems/02-rbac-syste

m/readme.md†L25-L115■

- **Core duties:** manage JWT access/refresh tokens, enforce Casbin policies, orchestrate password and MFA flows, and emit audit signals for every privileged action.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L86■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L144■
- **Upstream/Downstream:** collaborates with the RBAC module for authorization checks, pushes login/reset notifications through the Notification System, opens remediation tasks when suspicious access is detected, and streams events into the Audit Logging subsystem for immutable retention.■F:docs/03-systems/02-rbac-system/readme.md†L25-L115■■F:docs/03-systems/04-notification-system/readme.md†L1-L153■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L1-L120■■F:docs/03-systems/13-task-management-system/readme.md†L1-L115■
- **Integration surfaces:** exposes partner-ready endpoints for SSO callbacks, inbound webhooks, and service accounts that follow the platform's integration architecture contracts and webhook retry semantics.■F:docs/02-technical-specifications/07-integration-architecture.md†L60-L187■

Authentication & Session Flows

The Auth Service coordinates user lifecycle operations, session control, and security integrations for the AI Governance Platform. It relies on JWT-based authentication, Casbin RBAC enforcement, and Nodemailer to deliver transactional emails.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L86■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L75■

Registration (POST /api/v1/auth/register)

1. Validate input payload (email, password, organization context).

2. Hash the password with bcrypt (≥12 rounds) and persist the user record with default RBAC role assignments.

3. Trigger optional verification or welcome emails via Nodemailer integration.

4. Log the event for audit purposes and apply rate limiting to prevent abuse.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L86■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L75■

Login (POST /api/v1/auth/login)

1. Authenticate credentials against stored bcrypt hashes.

2. Issue a signed JWT containing user ID, role, and expiration, and generate a refresh token with longer validity for session continuity.

3. Store refresh token metadata (device, IP, expiry) for revocation and anomaly monitoring.

4. Enforce rate limiting and log attempts for security analytics.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L86■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L75■

Logout (POST /api/v1/auth/logout)

1. Invalidate the active refresh token by removing or flagging the record in the session store.

2. Record the logout event in audit logs, including device and timestamp.

3. Optionally revoke other active sessions through administrative controls when suspicious activity is detected.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L86■■F:docs/01-about/04-security-and-data-protection.md†L230-L249■

Refresh Tokens (POST /api/v1/auth/refresh)

1. Validate the refresh token against stored metadata and enforce rotation policies.

2. Issue a new access token and (optionally) a new refresh token with updated expiry limits.

3. Reject reused or expired tokens and log invalidation attempts to support threat detection.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L86■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L75■

Password Reset Emails (POST /api/v1/auth/forgot-password, POST /api/v1/auth/reset-password)

1. Generate a time-bound reset token stored alongside user metadata.

2. Send password reset instructions via Nodemailer using templated content and secure links.

3. Validate the token, enforce password strength policies, and update the hashed password upon completion.

4. Log successful and failed reset attempts for compliance tracking.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L80■

Enterprise Authentication Methods

- **SSO callbacks (/api/v1/auth/sso/:provider/callback):** Validate SAML or OIDC assertions, map identity claims to existing tenants, provision just-in-time accounts, and fall back to MFA enrolment when role policies demand it.■F:docs/01-about/04-security-and-data-protection.md†L206-L259■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L96■

- **MFA challenge (POST /api/v1/auth/mfa/verify):** Enforce secondary factors for privileged roles with TOTP or WebAuthn checks before final JWT issuance, persisting device metadata for governance reporting.■F:docs/01-about/04-security-and-data-protection.md†L206-L216■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L96■
- **Service and partner tokens (POST /api/v1/auth/service-tokens):** Issue scoped API credentials used by probes and integrations under the shared Partner API governance model, including rotation and revocation hooks.■F:docs/01-about/04-security-and-data-protection.md†L219-L226■■F:docs/02-technical-specifications/07-integration-architecture.md†L145-L187■

API Surface & Contracts

The module exposes REST endpoints aligned with the platform's OpenAPI standards; controllers reside in controllers/ and leverage shared error handlers to return the canonical { status, message, data, error } response shape.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L118■

Endpoint	*Purpose*	*Expected Inputs*	*Responses & Side Effects*
POST /api/v1/auth/register	Create a local account and bootstrap RBAC assignments.	Email, password, tenant metadata, optional SSO hints.	201 with user profile payload, emits welcome email, seeds auth_role_assignments, enqueues audit event.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L118■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L96■

Endpoint	Purpose	Expected Inputs	Responses & Side Effects
POST /api/v1/auth/login	Authenticate user credentials or SSO handoff.	Email + password or provider token, MFA hints.	200 with access/refresh tokens, writes session row, logs attempt, triggers risk checks and notifications on anomalies.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L86■■F:docs/01-about/04-security-and-data-protection.md†L206-L239■
POST /api/v1/auth/logout	Terminate active session.	Refresh token identifier.	204, revokes session records, cascades token invalidation, and records audit entry for traceability.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L86■■F:docs/01-about/04-security-and-data-protection.md†L243-L259■
POST /api/v1/auth/refresh	Rotate access tokens.	Refresh token, device info.	200 with new tokens, rotation metadata updated, invalid attempts flagged for monitoring.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L86■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L96■

Endpoint	Purpose	Expected Inputs	Responses & Side Effects
POST /api/v1/auth/forgot-password / POST /api/v1/auth/reset-password	Manage password recovery workflow.	Email for initiation; token + password for completion.	202 for initiation, 200 for reset, schedule email send, enforce password policies, log completion.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L118■
POST /api/v1/auth/mfa/enroll / POST /api/v1/auth/mfa/verify	Manage MFA secrets and verifications.	TOTP seed or WebAuthn challenge data.	200, persists MFA factors, flags accounts requiring step-up auth, records governance review items.■F:docs/01-about/04-security-and-data-protection.md†L206-L239■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L96■
POST /api/v1/auth/service-tokens / DELETE /api/v1/auth/service-tokens/:id	Issue and revoke service credentials.	Token label, scope, expiry.	201 with credential metadata or 204 on revoke, updates integration registry, emits webhook for dependent services.■F:docs/01-about/04-security-and-data-protection.md†L219-L226■■F:docs/02-technical-specifications/07-integration-architecture.md†L145-L187■

Background Jobs & Event Hooks

- **Session cleanup worker:** Scheduled task revokes stale refresh tokens, enforces inactivity policies, and notifies users about forced sign-outs through the Notification System.■F:docs/03-systems/04-notification-system/readme.md†L1-L153■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L96■
- **Audit dispatchers:** Every login, logout, role change, and credential update emits structured events consumed by the Audit Logging system and forwarded to SIEM sinks with immutable retention.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L1-L120■■F:docs/01-about/04-security-and-data-protection.md†L261-L313■
- **Access review triggers:** Signals to the RBAC and Task Management systems when privilege thresholds or dormant accounts require remediation, aligning with quarterly access review policies.■F:docs/03-systems/02-rbac-system/readme.md†L51-L115■■F:docs/01-about/04-security-and-data-protection.md†L230-L239■■F:docs/03-systems/13-task-management-system/readme.md†L1-L115■

Administrative Controls

Platform administrators operate dedicated tooling that interacts with the auth module to assign and delegate roles, manage authentication providers, and remediate risk.■F:docs/01-about/04-security-and-data-protection.md†L200-L259■ Key back-office automations include:

- Revoking user access, expiring active sessions, and orchestrating user lifecycle events with immutable logging for audits.■F:docs/01-about/04-security-and-data-protection.md†L230-L249■
- Configuring SSO connections, enforcing MFA policies, and managing passwordless enrollment options.■F:docs/01-about/04-security-and-data-protection.md†L206-L249■
- Executing break-glass procedures that grant temporary elevated access with real-time alerts and mandatory post-event reviews.■F:docs/01-about/04-security-and-data-protection.md†L253-L259■

Frontend Specification

Frontend Location & Directory Layout

React-based authentication experiences live within client/src/features/auth, complemented by shared state and UI primitives that surface authentication state across the app shell.■F:docs/02-technical-specifications/03-frontend-architecture.md†L40-L140■

```
client/src/features/auth/
■■■ pages/
■   ■■■ LoginPage.tsx
■   ■■■ RegisterPage.tsx
■   ■■■ ForgotPasswordPage.tsx
■   ■■■ ResetPasswordPage.tsx
■■■ components/
■   ■■■ MfaEnrollmentWizard.tsx
■   ■■■ PasswordStrengthMeter.tsx
■■■ hooks/
■   ■■■ useAuthForm.ts
■■■ index.ts

client/src/state/auth/
■■■ AuthContext.tsx
■■■ authSlice.ts
```

Reusable Components & UI Flows

- **Shared Components:** client/src/components/forms/ControlledInput.tsx, client/src/components/layout/AuthGuard.tsx, and client/src/components/feedback/SessionTimeoutModal.tsx are reused across login, registration, and MFA flows to deliver consistent experiences.
- **Authentication Screens:** Login, registration, password reset, and MFA setup pages mirror backend /api/v1/auth endpoints with form validation, CSRF headers, and contextual messaging for account states.■F:docs/02-technical-specifications/03-frontend-architecture.md†L50-L139■
- **Session Awareness:** Global AuthContext persists JWT tokens, handles refresh logic, and surfaces role metadata so protected routes can enforce access in layouts and navigation.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L139■
- **Security UX:** UI flows integrate security feedback (password strength meter, device trust prompts, session timeout modals) to align user experience with backend governance controls.■F:docs/02-technical-specifications/03-frontend-architecture.md†L143-L160■
- **Admin Adjacent Screens:** Auth screens link to RBAC and admin consoles for privilege reviews, reusing guard components defined in the RBAC feature set to keep enforcement consistent in client routing.■F:docs/03-systems/02-rbac-system/readme.md†L60-L88■■F:docs/02-technical-specifications/03-frontend-architecture.md†L52-L140■
- **Localization & Accessibility:** Forms load copy from JSON locale bundles and adhere to WCAG keyboard/focus contracts mandated by the frontend architecture, ensuring login experiences remain localized and accessible.■F:docs/02-technical-specifications/03-frontend-architecture.md†L143-L176■

Schema Specification

- **Users (auth_users):** Stores identity attributes (id, email, hashed password, MFA status, tenant associations) with auditing metadata for onboarding and lifecycle tracking.
- **Sessions (auth_sessions):** Persists refresh tokens, device fingerprints, IP addresses, and expiry timestamps for revocation workflows.
- **Password Resets (auth_password_resets):** Maintains reset token hashes, expiration, and consumption state.
- **Role Links:** Joins to RBAC tables (auth_roles, auth_role_assignments) to bootstrap privileges immediately after registration and during administrative updates.
- Relationships between these entities ensure rotation policies, password resets, and session governance are enforced consistently across tenants.■F:docs/02-technical-specifications/06-security-implementation.md†L54-L95■
- **Service Tokens (auth_service_tokens):** Captures partner credential metadata (scope, expiry, lastUsedAt) and hooks into integration registries for cross-system revocation audits.■F:docs/02-technical-specifications/07-integration-architecture.md†L145-L187■
- **Audit Links (auth_event_ledger):** Appends authentication and lifecycle events with correlations to audit archives for immutability guarantees, feeding the enterprise logging fabric.■F:docs/01-about/04-security-and-data-protection.md†L261-L313■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L1-L120■

Operational Playbooks & References

Session Handling and RBAC

- JWT validation middleware protects all authenticated routes and rehydrates user context on each request.
- Refresh tokens extend sessions without reauthentication but are constrained by timeout and rotation policies.
- Casbin-backed RBAC policies ensure users only access resources permitted to their role, with admin tooling to review and revoke sessions as needed.■F:docs/02-technical-specifications/06-security-implementation.md†L54-L95■■F:docs/01-about/04-security-and-data-protection.md†L206-L238■

Configuration, Secrets, and Environments

- Store per-environment credentials (JWT secrets, email providers, SSO certificates) in cloud-managed vaults and inject them via runtime environment variables following DevOps configuration standards.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L56-L210■

- Containerized services expose the auth module as a dedicated pod or service with liveness/readiness probes; blue-green deployments rotate secrets and migrations in lockstep across environments.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L108-L152■
- Infrastructure-as-Code definitions for auth dependencies (PostgreSQL schemas, Redis caches, secret stores) reside under infra/ and must pass policy checks before apply.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L82-L176■

Testing & Quality Gates

- Backend controllers, services, and repositories are covered by Jest unit/integration suites, while frontend flows leverage Vitest and Cypress scenarios that exercise login, MFA, and recovery journeys.■F:docs/02-technical-specifications/09-testing-and-qa.md†L40-L171■
- API contracts are validated through Newman collections derived from OpenAPI definitions, and security/performance tests (OWASP ZAP, k6) are mandatory before production releases for this module.■F:docs/02-technical-specifications/09-testing-and-qa.md†L83-L156■
- CI pipelines fail on coverage regression below platform thresholds (≥85% unit, ≥70% integration) and surface results via GitHub Actions notifications to the auth engineering team.■F:docs/02-technical-specifications/09-testing-and-qa.md†L166-L200■

Supported Authentication Methods

- **Single Sign-On (SSO):** Supports SAML 2.0 and OpenID Connect providers for federated login workflows.■F:docs/01-about/04-security-and-data-protection.md†L206-L216■
- **Multi-Factor Authentication (MFA):** Required for administrative and privileged accounts, with hooks in the login pipeline to validate second factors before issuing tokens.■F:docs/01-about/04-security-and-data-protection.md†L206-L216■
- **Passwordless Options:** Optional FIDO2 or hardware security keys can be registered to bypass passwords while maintaining strong assurance levels.■F:docs/01-about/04-security-and-data-protection.md†L206-L216■
- **Session Governance:** Automatic timeouts, refresh limits, and device tracking are enforced to manage risk across sessions and devices.■F:docs/01-about/04-security-and-data-protection.md†L206-L216■

Related Documentation

- Backend Architecture & APIs – Auth Service overview.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L74-L86■
- Security Implementation – Authentication, session handling, and RBAC policies.■F:docs/02-technical-specifications/06-security-implementation.md†L54-L145■

- Security and Data Protection – Enterprise authentication methods and administrative governance.■F:docs/01-about/04-security-and-data-protection.md†L206-L259■
- Integration Architecture – Partner API, webhook semantics, and service credential governance.■F:docs/02-technical-specifications/07-integration-architecture.md†L145-L187■
- DevOps & Infrastructure – Deployment, secrets management, and runtime configuration for auth workloads.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L56-L210■
- RBAC, Audit Logging, and Notification systems – Downstream modules consuming auth events and enforcing access governance.■F:docs/03-systems/02-rbac-system/readme.md†L25-L123■■F:docs/03-systems/04-notification-system/readme.md†L1-L153■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L1-L158■

Story: Auth Backend Layout

Summary

Capture the auth module directory structure so contributors know where controllers, services, middleware, and integrations live.

As a…

As an onboarding engineer

I want to…

I want to understand the auth module layout

So that…

So that I can navigate controllers, services, and middleware confidently

Acceptance Criteria

- [ ] Documentation lists key subdirectories under server/src/modules/auth and their responsibilities.
- [ ] Guidance references middleware, repositories, and routing files required for auth feature development.

Story: Auth Service Collaboration

Summary

Outline identity, token, RBAC, email, and audit responsibilities alongside upstream and downstream integrations.

As a…

As a platform architect

I want to…

I want clarity on auth service collaborators

So that…

So that dependent systems align with identity and audit workflows

Acceptance Criteria

- [ ] Responsibilities include JWT lifecycle, Casbin enforcement, password and MFA orchestration, and audit signaling.
- [ ] Integration notes describe coordination with RBAC, Notifications, Audit Logging, Task Management, and partner webhooks.

Story: User Registration Flow

Summary

Ensure account registration validates inputs, hashes credentials, seeds RBAC defaults, and emits onboarding signals.

As a…

As a prospective platform user

I want to…

I want to register securely with required tenant context

So that…

So that my account is provisioned with the right roles and compliance logging

Acceptance Criteria

- [ ] Registration rejects invalid payloads and persists bcrypt-hashed passwords with tenant-aware RBAC defaults.
- [ ] Successful registrations trigger welcome or verification emails and append audit trail entries.

Story: User Login Flow

Summary

Coordinate credential verification, token issuance, and anomaly logging whenever a user signs in.

As a…

As an authenticated platform member

I want to…

I want to log in with my credentials

So that…

So that I receive active access and monitoring of my session

Acceptance Criteria

- [ ] Login issues JWT access and refresh tokens with device metadata recorded for revocation.
- [ ] Failed or suspicious attempts are rate limited and logged for security analytics.

Story: User Logout Governance

Summary

Handle logout requests by invalidating refresh tokens, logging events, and enabling administrators to revoke other sessions when necessary.

As a…

As a security-conscious user

I want to…

I want to end my session cleanly

So that…

So that lingering tokens cannot be abused

Acceptance Criteria

- [ ] Logout requests invalidate the target refresh token and remove active session records.
- [ ] Audit entries capture device, timestamp, and optional admin-triggered global revocation.

Story: Token Refresh Lifecycle

Summary

Rotate access tokens through validated refresh tokens with reuse detection and logging.

As a…

As a returning authenticated user

I want to…

I want to refresh my session

So that…

So that long-lived access stays secure without re-login

Acceptance Criteria

- [ ] Refresh tokens are checked against stored metadata and rotation policies before issuing new tokens.
- [ ] Rejected or expired refresh attempts are logged and surfaced for threat detection.

Story: Password Reset Workflow

Summary

Support time-bound password resets with secure email delivery and policy enforcement.

As a…

As a user who forgot my password

I want to…

I want a reliable reset process

So that…

So that I can regain access without compromising security

Acceptance Criteria

- [ ] Reset initiation generates expiring tokens, sends templated emails, and records requests for auditing.
- [ ] Reset completion validates the token, enforces password strength, and updates hashed credentials.

Story: Enterprise Authentication Methods

Summary

Provide SSO callbacks, MFA verification, and service token issuance aligned with governance controls.

As a…

As a platform security architect

I want to…

I want federated and multi-factor authentication options

So that…

So that enterprise policies and integrations can be enforced

Acceptance Criteria

- [ ] SSO callbacks validate assertions, map tenants, and trigger MFA enrollment when required.
- [ ] Service and MFA token operations persist metadata for rotation, revocation, and reporting.

Story: Auth API Contracts

Summary

Document /api/v1/auth endpoints, payload expectations, and response semantics for core identity operations.

As a…

As an API consumer

I want to…

I want to understand auth endpoints

So that…

So that I can integrate registration, login, reset, MFA, and service token workflows

Acceptance Criteria

- [ ] Each endpoint lists purpose, inputs, and side effects following the platform's response envelope.
- [ ] Contracts call out status codes, notifications, and audit/event emissions for downstream systems.

Story: Auth Background Jobs

Summary

Operate background workers for session cleanup, audit dispatching, and access review triggers.

As a…

As a compliance operations lead

I want to…

I want automated auth maintenance

So that…

So that risk signals propagate without manual intervention

Acceptance Criteria

- [ ] Scheduled jobs revoke stale refresh tokens and notify users about forced sign-outs.
- [ ] Audit and access review events emit structured payloads consumed by downstream governance systems.

Story: Auth Administrative Controls

Summary

Equip administrators with tooling to revoke access, manage providers, and execute break-glass workflows.

As a…

As a platform administrator

I want to…

I want centralized auth controls

So that…

So that I can remediate risk and configure identity providers

Acceptance Criteria

- [ ] Admins can revoke user access, expire sessions, and log lifecycle events with immutable trails.
- [ ] Controls enforce MFA, SSO, and break-glass policies with alerts and post-event reviews.

Story: Auth Frontend Structure

Summary

Describe the React feature folders supporting authentication pages and shared state.

As a…

As a frontend contributor

I want to…

I want to know where auth UI code lives

So that…

So that I can extend login and registration experiences

Acceptance Criteria

- [ ] Documentation points to client/src/features/auth pages, components, and hooks plus shared auth state modules.
- [ ] Guidance explains how UI shells reuse guards and contexts to propagate authentication state.

Story: Auth Frontend Components

Summary

Highlight reusable auth UI flows, security UX elements, and localization requirements.

As a…

As a product designer

I want to…

I want consistent auth interactions

So that…

So that security cues and accessibility remain aligned across flows

Acceptance Criteria

- [ ] Shared components like controlled inputs, guards, and session timeout modals are enumerated for reuse.
- [ ] Flows cover login, registration, reset, and MFA pages with localization and accessibility mandates.

Story: Auth Schema Governance

Summary

Model users, sessions, resets, service tokens, and audit ledgers for lifecycle enforcement.

As a…

As a data platform engineer

I want to…

I want well-governed auth tables

So that…

So that identity events remain traceable and enforce policy

Acceptance Criteria

- [ ] Schema captures identity, session, reset, service token, and audit link entities with required metadata.
- [ ] Relationships with RBAC tables ensure privileges and rotations stay synchronized.

Story: Session Handling Operations

Summary

Document session validation, refresh limits, and Casbin RBAC enforcement for operational playbooks.

As a…

As a support engineer

I want to…

I want clear session handling guidance

So that…

So that I can troubleshoot access issues consistently

Acceptance Criteria

- [ ] Operational playbooks outline JWT validation, refresh policies, and Casbin checks on each request.
- [ ] Guidance covers administrative tooling to review and revoke sessions aligned with security mandates.

Story: Auth Configuration & Secrets

Summary

Explain environment variables and secret management needed for auth services across environments.

As a…

As a DevOps engineer

I want to…

I want to manage auth configuration

So that…

So that deployments carry correct secret values and policies

Acceptance Criteria

- [ ] Configuration guidance covers JWT keys, email providers, MFA, and RBAC policy sources managed via secure vaults.
- [ ] Docs describe environment-specific secrets handling and rotation expectations for auth infrastructure.

Story: Auth Quality Gates

Summary

Capture automated testing expectations for auth services covering unit, integration, and end-to-end flows.

As a…

As a QA lead

I want to…

I want quality criteria for auth

So that…

So that CI pipelines enforce coverage and regression safeguards

Acceptance Criteria

- [ ] Testing guidance outlines suites for middleware, endpoints, and UI flows with required coverage thresholds.
- [ ] CI expectations include security checks, rate-limit regression coverage, and failure notifications.

Story: Supported Authentication Methods

Summary

Summarize supported options like SSO, MFA, passwordless, and session governance requirements.

As a…

As a security program owner

I want to…

I want to catalogue allowed auth methods

So that…

So that policy documents map to implemented controls

Acceptance Criteria

- [ ] Supported methods list SSO, MFA, passwordless, and session governance features available to tenants.
- [ ] Documentation ties each method to enforcement and reporting expectations for compliance.

RBAC System

Location: /server/src/modules/auth

TL;DR

The AI Governance Platform enforces role-based access control (RBAC) using **Casbin** across the Node.js backend. Policies are persisted in **PostgreSQL**, cached for high throughput, and evaluated in middleware before any module logic runs. This document explains the role hierarchy, permission granularity, enforcement layers, and operational workflows that keep access secure and auditable.

Casbin is a mandated dependency for RBAC. Replacing it with any other authorization library or custom implementation requires a formal architecture review and sign-off from the security governance committee.

Backend Specification

Backend Location & Directory Layout

Casbin-backed RBAC logic ships with the Auth module in server/src/modules/auth, with cross-cutting middleware under server/src/middleware to enforce permissions before feature handlers execute.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L52-L86■

```
server/src/modules/auth/
■■■ casbin/
■   ■■■ rbac_with_domains_model.conf
■   ■■■ policy.seed.json
■■■ controllers/
■   ■■■ rbac.controller.ts
■■■ services/
■   ■■■ role.service.ts
■   ■■■ policy.service.ts
■■■ repositories/
■   ■■■ role.repository.ts
■   ■■■ policy.repository.ts
■■■ casbin-adapter.ts
■■■ routes/rbac.routes.ts      # Mounts /api/v1/auth RBAC endpoints

server/src/middleware/
■■■ authorization.ts
```

Role Model & Enforcement

The RBAC system resides within the Auth Service, where controllers orchestrate login, token issuance, and authorization decisions. A dedicated Casbin adapter loads policies from PostgreSQL into an in-memory enforcer shared across Express middleware. Prisma migrations run through the release pipeline to keep the managed PostgreSQL schema aligned with Casbin's tables.

Role Hierarchy

RBAC follows hierarchical inheritance that mirrors the platform-wide security model. Canonical roles are **Admin**, **Compliance Officer**, **Engineer**, **Auditor**, and **System Service**, with an operational **Super Admin** overlay reserved for the CISO break-glass workflow.■F:docs/02-technical-specifications/06-security-implementation.md†L84-L108■■F:docs/01-about/04-security-and-data-protection.md†L238-L282■ Each role inherits capabilities from the role beneath it to uphold least-privilege principles. Custom roles derive from these definitions or from other custom roles, storing metadata in auth_roles (owner tenant, description, review cadence) and mapping to granular policies for scoped privileges. Separation-of-duty rules enforce that approval actions require distinct reviewers, impersonation is audited, and conflicting admin/auditor assignments trigger automated reviews with escalations to the Security Council.■F:docs/01-about/04-security-and-data-protection.md†L238-L299■■F:docs/01-about/08-operations-and-teams.md†L79-L128■

Permission Granularity

Policies combine resource type, identifier, and action verb (e.g., framework:123:update, control:*:approve). Casbin domains provide tenant scoping, wildcards enable read vs. write bundles, and middleware-level attribute checks add ABAC-style conditions for contextual enforcement.

Enforcement Surface

Middleware in server/src/middleware/authorization.ts validates JWTs, builds Casbin subjects/domains/objects/actions, and denies or allows requests with audit logging. Module-specific helpers extend enforcement into Governance Engine, Evidence, Notifications, and Tasks modules to guarantee consistent authorization across feature services.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L56-L115■ Observability hooks log decisions to audit_logs for compliance and forensics, forwarding structured JSON to the centralized monitoring stack mandated by DevOps and security governance.■F:docs/02-technical-specifications/06-security-implementation.md†L120-L156■■F:docs/02-technical-specifications/05-devops-infrastructure.md†L64-L141■

API Contracts & Module Interfaces

RBAC functionality is exposed through dedicated REST endpoints served by the Auth Service and versioned under /api/v1 in line with the platform's API standards.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L90-L151■■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L200-L251■

Endpoint	*Method*	*Purpose*
/api/v1/auth/roles	GET/POST	List existing roles, create custom roles, or clone baseline templates.
/api/v1/auth/roles/:id	GET/PATCH/DELETE	Inspect role metadata, adjust inheritance, or retire roles while preserving revision history.
/api/v1/auth/policies	GET/POST	Manage granular Casbin p rules for resources and actions with dry-run support.
/api/v1/auth/policies/:id	PATCH/DELETE	Update or revoke specific policies; mutation events invalidate caches.
/api/v1/auth/access-reviews	POST	Launch scheduled or ad-hoc access recertification workflows that integrate with the Task Service.
/api/v1/auth/service-accounts	POST/PATCH	Issue scoped credentials for automation and partner integrations with domain-bound permissions.

All endpoints require JWT authentication and are guarded by Casbin middleware before reaching controller logic. Responses follow the standardized { status, message, data, error } schema and publish OpenAPI documentation through /api/v1/docs to keep implementation and documentation in sync.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L208-L251■

Domain Events & Integrations

RBAC emits and consumes domain events to keep downstream systems synchronized:

- **policy.updated** – Published whenever roles or policies change. Subscribers include cache workers, the Notification Service, and Governance Engine recalculations. Retries follow the exponential backoff standards in the integration architecture.■F:docs/02-technical-specifications/07-integration-architecture.md†L122-L176■
- **access-review.opened** / **access-review.completed** – Coordinate remediation tasks and compliance dashboards, feeding evidence exports and KPI reporting.
- **Partner API hooks** – RBAC scopes partner tokens and enforces API access restrictions, ensuring external integrations remain within approved privileges.■F:docs/02-technical-specifications/07-integration-architecture.md†L176-L212■

Inbound events—such as automated evidence probes requesting temporary elevation—are validated with signed payloads, evaluated against Casbin domains, and either accepted or rejected with audit trails for regulatory review.■F:docs/02-technical-specifications/07-integration-architecture.md†L212-L236■

Frontend Specification

Frontend Location & Directory Layout

Role management experiences live inside the admin area of the React client under client/src/features/admin/rbac, alongside shared guard components that protect routes and dashboards based on evaluated permissions.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L139■ The RBAC pages are exposed through the same admin console shell described in the Admin and Configuration system, giving platform and tenant administrators a dedicated "Access Control" navigation cluster for listing roles, editing inheritance, and reviewing policy matrices without leaving the governance control plane.■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L71-L121■

```
client/src/features/admin/rbac/
■■■ pages/
■   ■■■ RoleListPage.tsx
■   ■■■ RoleDetailPage.tsx
■   ■■■ PolicyEditorPage.tsx
■■■ components/
■   ■■■ PermissionMatrix.tsx
■   ■■■ RoleInheritanceGraph.tsx
■   ■■■ AccessReviewSummary.tsx
■■■ hooks/
■   ■■■ useRoleAssignments.ts
■■■ api/
    ■■■ rbacClient.ts

client/src/components/guards/
■■■ RequirePermission.tsx
```

Reusable Components & UI Flows

- **Shared Guards:** RequirePermission and RequireRole components gate admin navigation, dashboards, and inline actions with Casbin-backed checks exposed via the Auth Context.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L139■
- **Policy Management Screens:** Admin pages provide CRUD experiences for roles, inheritance, and policies, invoking /api/v1/auth/roles, /api/v1/auth/policies, and /api/v1/auth/access-reviews endpoints with optimistic updates and conflict resolution modals. The list → detail → policy editor flow surfaces contextual breadcrumbs (Access Control / Roles) so admins always understand which tenant domain and Casbin scope they are adjusting.
- **Access Review Workflows:** AccessReviewSummary surfaces outstanding reviews, integrates with task notifications, and links into the Governance Engine for remediation triggers.
- **Visualizations:** Components such as RoleInheritanceGraph and PermissionMatrix reuse charting primitives from client/src/components/charts to display effective permissions per tenant.

Schema Specification

- **auth_roles:** Stores role metadata (id, tenant domain, name, inheritance parent, review cadence) and drives UI listings. Indexed on tenant and name for fast lookups under the externally hosted PostgreSQL model.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L180-L209■
- **auth_role_assignments:** Links users or service accounts to roles with effective and expiry timestamps for delegation tracking and supports soft deletes for historical review.
- **auth_policies:** Canonical Casbin policy rules (p, g, g2) with resource/action descriptors and domain segmentation. Policies persist as immutable rows with revision pointers.

- **auth_policy_revisions:** Append-only audit ledger capturing change metadata, justification, and reviewer approvals consistent with immutable logging expectations.■F:docs/02-technical-specifications/06-security-implementation.md†L138-L156■
- **Redis Cache (casbin:tenant:<id>):** Maintains short-lived policy snapshots to reduce database load, invalidated via policy.updated events and redeployed alongside Kubernetes rollouts.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L88-L141■

Schema migrations are executed through Prisma migration bundles submitted via the CI/CD change queue so RBAC tables remain governed even on managed database hosts.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L172-L190■ Security controls enforce RBAC-aligned database roles with TLS and AES-256 encryption per the database design standards.■F:docs/02-technical-specifications/04-database-design.md†L169-L196■

Operational Playbooks & References

Access Reviews and Governance Workflows

- Quarterly access reviews snapshot current assignments, send review tasks to Compliance Officers, and require sign-off before completion in alignment with enterprise governance processes.■F:docs/01-about/04-security-and-data-protection.md†L282-L314■■F:docs/01-about/08-operations-and-teams.md†L129-L181■
- Event-driven reviews detect dormant or conflicting privileges and open remediation tasks in the Task Service, surfacing alerts via Notification Service SLAs.
- Certification exports write reports to the Evidence Repository, while delegations create time-bound policies enforced by scheduled jobs and audited through immutable log retention.

Updating Casbin Policies

1. **Plan:** Document desired permission updates and map the affected endpoints using the . Capture rationale and risk assessment for Security Council review.

2. **Edit:** Use the Admin UI or /api/v1/auth/policies API to adjust p (permission) or g (inheritance) rules; bulk updates leverage the RBAC sync CLI to push YAML manifests through the adapter.

3. **Validate:** Run /api/v1/auth/policies/dry-run to preview effects, execute automated regression suites, and confirm separation-of-duty constraints enforced by Casbin and ABAC checks.■F:docs/02-technical-specifications/06-security-implementation.md†L96-L116■

4. **Deploy:** Ship accompanying migrations when introducing new roles/resources, coordinate cache invalidation across regions, and allow CI/CD promotion gates to verify policy load success before production rollout.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L108-L167■

5. **Audit:** Verify auth_policy_revisions captures the change with justification and reviewer sign-off, then export summaries for immutable evidence storage.

Testing RBAC Scenarios

- **Unit Tests:** server/src/modules/auth/__tests__/authorization.spec.ts mocks the Casbin enforcer to cover allow/deny paths and inheritance edge cases.
- **Integration Tests:** API tests under server/src/routes/__tests__ seed PostgreSQL with fixtures to verify middleware and module-specific enforcement, exercising the standardized error schema.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L200-L228■
- **Manual Verification:** Leverage the shared Postman collection, run npm run test:rbac, and execute the security hardening checklist before releases, including cache flushes and policy reload verificatio n.■F:docs/02-technical-specifications/06-security-implementation.md†L96-L116■■F:docs/02-techni cal-specifications/05-devops-infrastructure.md†L128-L180■

Configuration, Deployment & Monitoring

- **Environment Configuration:** RBAC services read Casbin model paths, adapter DSNs, and cache TTLs from environment variables managed via cloud vaults and Terraform in the infra/ directory.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L48-L117■
- **Deployment Workflow:** CI pipelines lint, test, and build Docker images, then stage rollouts with blue-green or rolling strategies. Production promotion requires manual approval from security leadership when RBAC policies change.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L117-L195■
- **Observability:** Metrics for authorization latency, policy cache hit rate, and deny counts stream to Grafana dashboards; alerts trigger PagerDuty when failure rates exceed thresholds or policy sync jobs lag behind schedule.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L195-L238■

Related Documentation

Story: RBAC Backend Layout

Summary

Describe where Casbin models, adapters, controllers, and middleware live within the auth module.

As a…

As an authorization developer

I want to…

I want to locate RBAC code

So that…

So that I can extend policies without breaking conventions

Acceptance Criteria

- [ ] Directory guidance covers casbin configs, services, repositories, and middleware entry points.
- [ ] Routes and adapters show how RBAC endpoints mount under /api/v1/auth.

Story: RBAC Role Model

Summary

Explain the hierarchical role structure, permission granularity, and enforcement surface used to evaluate access decisions.

As a…

As a security designer

I want to…

I want to understand RBAC inheritance

So that…

So that least privilege and separation of duties remain intact

Acceptance Criteria

- [ ] Roles such as Admin, Compliance Officer, Engineer, Auditor, and System Service inherit capabilities according to policy.
- [ ] Enforcement combines Casbin domains, middleware checks, and audit logging for deny/allow decisions.

Story: RBAC API Contracts

Summary

Detail role, policy, access review, and service account endpoints exposed under /api/v1/auth.

As a…

As an API integrator

I want to…

I want RBAC endpoint clarity

So that…

So that I can manage roles and policies programmatically

Acceptance Criteria

- [ ] Endpoints list CRUD operations for roles, policies, access reviews, and service accounts with expected verbs.
- [ ] Contracts highlight authentication requirements and cache invalidation behavior.

Story: RBAC Domain Events

Summary

Summarize outbound events such as policy.updated and access review signals plus inbound validation flows.

As a…

As a platform integrator

I want to…

I want RBAC event visibility

So that…

So that dependent systems stay synchronized with authorization changes

Acceptance Criteria

- [ ] Stories note which events publish to caches, notifications, governance, and partner APIs.
- [ ] Inbound event validation ensures signed payloads respect tenant domains before applying privileges.

Story: RBAC Frontend Structure

Summary

Map the admin console directories powering RBAC management pages and guard components.

As a…

As a frontend maintainer

I want to…

I want to navigate RBAC UI code

So that…

So that I can adjust role management experiences

Acceptance Criteria

- [ ] Documentation references client/src/features/admin/rbac pages, components, hooks, and API clients.
- [ ] Shared guard components like RequirePermission and RequireRole are highlighted for reuse across admin surfaces.

Story: RBAC Frontend Flows

Summary

Describe UI patterns for policy management, access reviews, and visualization widgets.

As a…

As a product manager

I want to…

I want insight into RBAC UI capabilities

So that…

So that admins can manage permissions intuitively

Acceptance Criteria

- [ ] Flows cover list/detail/editor experiences with optimistic updates and breadcrumbs.
- [ ] Visualization components show inheritance graphs and permission matrices integrated with metrics hooks.

Story: RBAC Access Reviews

Summary

Capture quarterly and event-driven review workflows, evidence exports, and remediation integrations.

As a…

As a compliance lead

I want to…

I want a clear access review process

So that…

So that certification cycles meet governance requirements

Acceptance Criteria

- [ ] Stories describe scheduled and event-triggered reviews that coordinate with tasks and notifications.
- [ ] Evidence exports and delegation rules ensure reviewers document approvals with immutable history.

Story: RBAC Policy Updates

Summary

Outline the plan, edit, validate, deploy, and audit steps for modifying Casbin policies.

As a…

As a release manager

I want to…

I want a controlled RBAC update workflow

So that…

So that policy changes are reviewed and traceable

Acceptance Criteria

- [ ] Process includes documenting changes, using APIs or CLI tooling, and running dry-run validations.
- [ ] Deployment steps mention migrations, cache invalidation, CI gates, and audit ledger verification.

Story: RBAC Testing Expectations

Summary

Define unit, integration, and manual verification practices for RBAC middleware and endpoints.

As a…

As a QA engineer

I want to…

I want RBAC testing guidance

So that…

So that coverage exists for allow/deny scenarios

Acceptance Criteria

- [ ] Tests mock the Casbin enforcer, seed Postgres fixtures, and validate standardized error responses.
- [ ] Manual verification leverages Postman collections, npm scripts, and security hardening checklists.

Story: RBAC Configuration & Monitoring

Summary

Summarize environment variables, deployment workflow, and observability metrics for RBAC services.

As a…

As a DevOps engineer

I want to…

I want to operate RBAC infrastructure

So that…

So that rollouts and incidents are handled consistently

Acceptance Criteria

- [ ] Environment configuration covers Casbin model paths, adapter DSNs, cache TTLs, and vault-managed secrets.
- [ ] Deployment guidance outlines CI pipelines, promotion approvals, and metrics/alerts for authorization health.

Document and Media Upload System

Location: /server/src/modules/uploads

TL;DR

The upload service brokers secure, auditable ingestion of documents and media into MinIO using presigned URLs and metadata contracts.

It validates payload intent, orchestrates client-side uploads, compresses imagery, and persists immutable evidence records for downstream workflows.

All image assets must be compressed before finalizing an upload. Compression is enforced at the worker tier and any bypass is treated as a policy violation requiring a release rollback.

Backend Specification

Backend Location & Directory Layout

The upload service runs under server/src/modules/uploads, coordinating Express controllers, MinIO integrations, and BullMQ workers for presigned URL issuance and completion processing.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L44-L171■

```
server/src/modules/uploads/
■■■ controller.ts
■■■ routes.ts
■■■ services/
■   ■■■ metadata.service.ts
■   ■■■ presign.service.ts
■■■ worker/
■   ■■■ index.ts
■   ■■■ compression.processor.ts
■■■ integrations/
■   ■■■ minio.client.ts
■■■ events/
    ■■■ evidence.publisher.ts
```

Storage Architecture

The Document and Media Upload system provides a single entry point for governance evidence and shared artifacts. It issues short-lived MinIO presigned URLs, enforces content validation, compresses images, and stores immutable metadata in PostgreSQL so downstream services (Evidence Repository, Governance Engine, Reporting) can reference a consistent record.■F:docs/02-technical-specifications/01-system-architecture.md†L173-L202■■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L115■ Core components include:

- **Upload Controller:** Validates RBAC scopes, orchestrates presigned URL issuance, and tracks upload state transitions.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L44-L135■
- **MinIO Integration:** Wraps the SDK for presigned PUT URLs, lifecycle inspection, and bucket health checks.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L57-L135■
- **BullMQ Worker:** Processes completion callbacks, runs compression, calculates fingerprints, and publishes status events to dependent services.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L124-L171■■F:docs/03-systems/12-governance-engine/readme.md†L7-L104■
- **Bucket Strategy:** Buckets follow {tenant}/{classification}/{yyyy}/{mm} naming, with classifications (evidence, policy, report, media) aligning lifecycle policies. Provisioning occurs during tenant onboarding alongside Admin & Configuration access policies; versioning stays enabled for immutable audit history.■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L65-L154■■F:docs/02-technical-specifications/05-devops-infrastructure.md†L60-L119■

Upload Lifecycle

1. **Intake & Validation:** Clients submit metadata (filename, mimeType, checksum, classification, sizeBytes) to /api/v1/uploads/request. The service validates MIME types, size limits (512 MB documents, 50 MB images), and RBAC scopes (evidence:write, media:write). Replacement uploads verify that the prior record is editable.■F:docs/03-systems/02-rbac-system/readme.md†L7-L192■

2. **Presigned URL Issuance:** Controllers request MinIO presigned PUT URLs with 10-minute expirations and enforce Content-MD5 headers to detect tampering. Response payloads include upload IDs, headers, and compression flags so clients can prepare assets.■F:docs/02-technical-specifications/01-system-architecture.md†L173-L202■

3. **Completion & Versioning:** Clients confirm via /api/v1/uploads/complete after MinIO acknowledges the PUT. Workers queue compression jobs, compute SHA-256 fingerprints, and persist metadata (size, checksum, path, uploader). If versions exist, version increments while prior paths remain for auditors; completion events (evidence.uploaded) notify the Governance Engine and Task Service.■F:docs/03-systems/11-evidence-management-system/readme.md†L47-L111■■F:docs/03-systems/12-governance-engine/readme.md†L52-L113■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■

Media Compression & Transformations

- **Compression Pipeline:** BullMQ workers route images through sharp with tenant presets (max width 2560 px, JPEG/WEBP 85 % quality, PNG quantization). Status updates track pending, compressed, or failed; uploads remain hidden until compression succeeds.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L124-L171■
- **Supported Formats:** Documents (.pdf, .docx, .rtf, .txt, .md) and images (.png, .jpg, .jpeg, .webp) pass validation; oversize files, checksum mismatches, or blocked formats (e.g., .heic, .nef) return HTTP 422. Malware scanning integrates with asynchronous ClamAV, quarantining suspicious objects in a hold prefix.■F:docs/02-technical-specifications/06-security-implementation.md†L108-L196■
- **Derivatives & Accessibility:** Thumbnails (320 px) support UI previews, extracted text feeds search indexing, and alt-text metadata is mandatory for imagery displayed in dashboards.■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■■F:docs/01-about/04-security-and-data-protection.md†L206-L259■ Notifications fire when compression repeatedly fails for manual remediation.■F:docs/03-systems/04-notification-system/readme.md†L7-L222■

API Endpoints & Contracts

The service exposes REST endpoints under /api/v1/uploads, conforming to the platform's OpenAPI conventions and Express middleware stack (JWT auth + Casbin RBAC + Joi validation).■F:docs/02-technical-specifications/01-system-architecture.md†L160-L198■■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L1-L77■ Every route returns JSON envelopes with { status, data, error } and consistent error codes (400 validation, 401/403 auth, 409 version conflicts, 422 payload issues, 500 internal).

Endpoint	Method	Description	Request Payload	Success Response
/api/v1/uploads/request	POST	Validate metadata, create upload_requests row, and issue presigned PUT contract.	{ filename, mimeType, checksum, classification, sizeBytes, altText?, tags?, parentEvidenceId? }	{ uploadId, presignedUrl, headers, expiresAt, requiresCompression } with 409 if the referenced evidence cannot be replaced.■F:docs/03-systems/02-rbac-system/readme.md†L7-L192■■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L44-L135■
/api/v1/uploads/complete	POST	Confirm object persistence, enqueue compression, and persist evidence entry.	{ uploadId, contentMd5, storageKey, sizeBytes }	{ evidenceId, version, compressionStatus }; background worker emits evidence.uploaded event on success.■F:docs/03-systems/12-governance-engine/readme.md†L52-L113■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■
/api/v1/uploads/:id	GET	Fetch metadata and status for UI history / retries.	id path param, optional includeEvents query flag.	{ evidenceId, compressionStatus, events[] }; respects RBAC scopes and redacts quarantined objects.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L200■

Webhook-style callbacks (/api/v1/uploads/callbacks/minio) can be enabled for MinIO event listeners; they validate HMAC signatures from MinIO, translate events into worker jobs, and respond within 3 s to stay under MinIO retry thresholds.■F:docs/02-technical-specifications/07-integration-architecture.md†L88-L176■

Frontend Specification

Frontend Location & Directory Layout

Client upload experiences live in client/src/features/uploads, integrating with shared file input primitives and evidence viewers to deliver presigned flows and progress updates.■F:docs/02-technical-specifications/03-frontend-architecture.md†L50-L139■

```
client/src/features/uploads/
■■■ pages/
■   ■■■ UploadRequestPage.tsx
■   ■■■ UploadReviewPage.tsx
■   ■■■ UploadHistoryPage.tsx
■■■ components/
■   ■■■ FileDropzone.tsx
■   ■■■ CompressionStatusBadge.tsx
■   ■■■ PresignedUrlInstructions.tsx
■■■ hooks/
■   ■■■ useUploadRequest.ts
■   ■■■ useUploadProgress.ts
■■■ api/
    ■■■ uploadsClient.ts

client/src/components/evidence/
■■■ EvidencePreviewCard.tsx
■■■ EvidenceTimeline.tsx
```

Reusable Components & UI Flows

- **Shared Components:** FileDropzone and CompressionStatusBadge leverage design-system inputs and toasts to standardize drag-and-drop, checksum validation, and compression visibility across modules.
- **Presigned Flow:** Upload request pages call /api/v1/uploads/request, present required headers, and hand off to the browser or native client to PUT objects directly to MinIO. Progress hooks update status banners until /api/v1/uploads/complete finalizes.
- **Evidence Surfacing:** EvidencePreviewCard and EvidenceTimeline components show thumbnails, extracted text, and version history across the Evidence Repository and Reporting features.
- **Accessibility & Guidance:** PresignedUrlInstructions educates users on compression requirements, alt-text capture, and failure recovery, reusing messaging from the Notification system for consistency.■F:docs/03-systems/04-notification-system/readme.md†L7-L222■

Configuration & Dependencies

- **Environment Variables:** MINIO_ENDPOINT, MINIO_ACCESS_KEY, MINIO_SECRET_KEY, MINIO_BUCKET_UPLOADS, UPLOADS_MAX_DOCUMENT_BYTES, UPLOADS_MAX_IMAGE_BYTES, and CLAMAV_HOST are injected via .env per environment and sourced from the platform's vault-backed configuration workflow.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L60-L103■■F:docs/02-technical-specifications/01-system-architecture.md†L204-L223■ Workers additionally require BULLMQ_REDIS_URL and COMPRESSION_QUEUE_CONCURRENCY to throttle jobs.
- **Service Integrations:** MinIO clients reuse shared connection pools under server/src/integrations/minio.client.ts; ClamAV scanning runs through the security integration bus defined for the broader platform to guarantee policy parity.■F:docs/02-technical-specifications/06-security-implementation.md†L108-L196■■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L115■
- **Deployment Footprint:** Upload controllers live in the API container, while compression workers run as dedicated queue consumers with autoscaling rules tuned to keep backlog under 1 000 jobs and latency below 800 ms, matching observability thresholds used across systems.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L108-L140■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L200■
- **Language Standardization:** Backend and worker code remain JavaScript-only with Jest-based unit tests, aligning with the platform-wide ban on TypeScript to preserve tooling consistency.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L1-L32■■F:docs/02-technical-specifications/09-testing-and-qa.md†L40-L76■

Schema Specification

- **evidence & evidence_snapshots:** Persist file fingerprints, compression status, classifications, uploader identity, and version lineage for immutable history.■F:docs/02-technical-specifications/04-database-design.md†L88-L146■
- **upload_requests:** Tracks presigned URL issuance, expiration, checksum expectations, and tenant scope to reconcile orphaned uploads.
- **upload_events:** Append-only log capturing state transitions (requested, completed, compressed, failed) and correlation IDs for observability.
- Relationships integrate with RBAC (auth_roles/auth_role_assignments) for scope validation and with Governance Engine entities to trigger evidence-driven workflows.■F:docs/03-systems/12-governance-engine/readme.md†L52-L113■

Testing & QA Expectations

- **Unit & Integration Tests:** Controllers, services, and workers use Jest suites stored under server/tests/uploads to cover validation, presign issuance, and compression orchestration with ≥85 % coverage, mirroring platform targets.■F:docs/02-technical-specifications/09-testing-and-qa.md†L55-L176■
- **API & Contract Tests:** Postman/Newman collections assert OpenAPI parity for /api/v1/uploads/* and replay failure scenarios (checksum mismatch, oversized file) in CI to enforce shift-left validation.■F:docs/02-technical-specifications/09-testing-and-qa.md†L83-L158■
- **E2E Flows:** Cypress tests under client/tests/uploads execute drag-and-drop, presign PUT, and completion confirmation, verifying accessibility cues (alt-text prompts, compression warnings) and integration with Evidence timelines.■F:docs/02-technical-specifications/09-testing-and-qa.md†L88-L154■■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L104■
- **Security & Performance:** OWASP ZAP scans ensure upload routes resist injection and auth bypass, while k6 load tests run against presign issuance to validate SLA commitments (≤800 ms P95 latency, ≤2 % failure rate).■F:docs/02-technical-specifications/09-testing-and-qa.md†L93-L154■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L154-L200■

Operational Playbooks & References

Security, Compliance, and Governance

- RBAC middleware enforces scope checks; service-to-service tokens handle machine uploads, and classification tags propagate into the Governance Engine for control mappings.■F:docs/03-systems/02-rbac-system/readme.md†L7-L192■■F:docs/03-systems/09-control-management-system/readme.md†L7-L133■
- MinIO applies SSE-S3 encryption with quarterly key rotation, retention defaults to seven years, and legal holds surface in the Evidence UI; cross-region replication satisfies durability targets.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L60-L226■■F:docs/03-systems/11-evidence-management-system/readme.md†L92-L115■
- Audit trails capture uploader metadata, IP, object key, checksum, and compression outcome; download presigns are watermarked, rate limited, and logged for governance review.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L200■■F:docs/03-systems/10-framework-mapping-system/readme.md†L55-L217■

Monitoring and Incident Response

- Dashboards watch throughput, compression latency, presign errors, and MinIO 5xx rates; thresholds (issuance failures >2 %, compression backlog >1000, latency >800 ms) trigger Evidence squad PagerDuty alerts.■F:docs/03-systems/04-notification-system/readme.md†L7-L222■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L200■

- Synthetic probes validate flows per classification, while incident response runbooks cover manual compression retries, malware quarantine handling, and MinIO failover with checksum verification.■F:docs/03-systems/07-probe-management-system/readme.md†L7-L226■■F:docs/02-technical-specifications/06-security-implementation.md†L108-L196■■F:docs/02-technical-specifications/05-devops-infrastructure.md†L168-L226■

Capacity & Cost Management

- Monthly reviews assess bucket growth, compression savings, and lifecycle tiering; cost allocation tags feed Reporting dashboards for budgeting transparency.■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■
- API gateway caching and deployment sizing follow the Deployment & Environment guide to scale presigned traffic efficiently.■F:docs/02-technical-specifications/08-deployment-and-environment-guide.md†L53-L186■

Related Documentation

Story: Uploads Backend Layout

Summary

Document the upload module's controllers, services, workers, and integrations directory structure.

As a…

As a backend developer

I want to…

I want to find upload service files

So that…

So that I can extend presign or worker logic

Acceptance Criteria

- [ ] Directory listing includes controllers, services, worker processors, integrations, and event publishers.
- [ ] Notes explain how BullMQ workers and MinIO clients are organized within the module.

Story: Uploads Storage Architecture

Summary

Explain how presigned URLs, metadata validation, and MinIO buckets manage secure evidence ingestion.

As a…

As a storage architect

I want to…

I want to understand upload storage design

So that…

So that evidence stays immutable and traceable

Acceptance Criteria

- [ ] Architecture covers controller validation, MinIO integration, worker orchestration, and bucket naming conventions.
- [ ] Guidance links uploads to downstream evidence, governance, and reporting services via metadata.

Story: Upload Lifecycle

Summary

Describe the request, presign, completion, and versioning steps for ingesting documents and media.

As a…

As an integration engineer

I want to…

I want to follow the upload lifecycle

So that…

So that clients can complete uploads reliably

Acceptance Criteria

- [ ] Flow covers metadata validation, presigned URL issuance with required headers, and completion callbacks.
- [ ] Lifecycle records versioning, fingerprinting, and event publication to dependent services.

Story: Compression & Transformations

Summary

Detail the compression pipeline, supported formats, and derivative generation for media uploads.

As a…

As a media specialist

I want to…

I want automated compression controls

So that…

So that imagery meets policy before being shared

Acceptance Criteria

- [ ] Workers enforce tenant-specific compression presets, track statuses, and block release until success.
- [ ] Supported format validation, malware quarantine, and derivative/alt-text requirements are documented.

Story: Uploads API Contracts

Summary

Capture /api/v1/uploads endpoints, payloads, and webhook callbacks for presign and completion flows.

As a…

As an API consumer

I want to…

I want to integrate upload endpoints

So that…

So that I can request presigned URLs and confirm completions

Acceptance Criteria

- [ ] Endpoints describe request bodies, responses, and error codes for request, complete, and metadata retrieval routes.
- [ ] Webhook behavior includes HMAC validation, job enqueueing, and retry expectations.

Story: Uploads Frontend Structure

Summary

Identify the React folders, pages, components, and hooks supporting upload experiences.

As a…

As a frontend engineer

I want to…

I want to locate upload UI code

So that…

So that I can enhance progress tracking and guidance

Acceptance Criteria

- [ ] Structure lists client/src/features/uploads pages, components, hooks, and shared evidence UI modules.
- [ ] Guidance explains how presigned flows and progress hooks update clients during uploads.

Story: Uploads UI Flows

Summary

Summarize reusable upload components, evidence previews, and accessibility messaging.

As a…

As a UX lead

I want to…

I want consistent upload interactions

So that…

So that users understand compression requirements and statuses

Acceptance Criteria

- [ ] Components like FileDropzone and CompressionStatusBadge standardize drag-and-drop and status indicators.
- [ ] Instructions cover alt-text capture, localization, and integration with notification messaging for failures.

Story: Uploads Configuration & Dependencies

Summary

List environment variables, integrations, and deployment topology for controllers and workers.

As a…

As an SRE

I want to…

I want upload configuration guidance

So that…

So that services connect to MinIO, ClamAV, and queues securely

Acceptance Criteria

- [ ] Configuration enumerates MinIO credentials, size limits, and queue settings managed via environment variables.
- [ ] Deployment notes distinguish API containers from worker consumers and describe autoscaling expectations.

Story: Uploads Schema

Summary

Describe database tables for evidence, upload requests, and events plus relationships to RBAC and governance.

As a…

As a data steward

I want to…

I want upload metadata models

So that…

So that lineage and access controls remain enforceable

Acceptance Criteria

- [ ] Schema entries include evidence, snapshots, upload*requests, and upload*events with key attributes.
- [ ] Relationships map to RBAC scopes and governance entities for consistent enforcement.

Story: Uploads Testing & QA

Summary

Outline unit, integration, E2E, security, and performance testing expectations for upload services.

As a…

As a QA manager

I want to…

I want upload quality standards

So that…

So that regressions or vulnerabilities are caught early

Acceptance Criteria

- [ ] Testing coverage spans Jest suites, Postman/Newman contract checks, and Cypress E2E workflows.
- [ ] Security and load testing requirements include OWASP scans and k6 targets for latency and failure rates.

Story: Uploads Security & Governance

Summary

Capture RBAC enforcement, encryption, legal holds, and audit logging applied to uploaded evidence.

As a…

As a compliance officer

I want to…

I want to ensure uploaded evidence is protected

So that…

So that regulatory obligations are met

Acceptance Criteria

- [ ] Policies describe scope checks, service tokens, encryption, retention, and legal hold expectations.
- [ ] Audit trails record uploader metadata, checksums, and download watermarking for governance review.

Story: Uploads Monitoring & Incident Response

Summary

Summarize dashboards, alert thresholds, synthetic probes, and runbooks for handling failures.

As a…

As an on-call engineer

I want to…

I want monitoring guidance

So that…

So that I can triage upload issues quickly

Acceptance Criteria

- [ ] Metrics track throughput, compression latency, presign errors, and MinIO health with defined alert thresholds.
- [ ] Runbooks cover manual retries, quarantine handling, and failover verification steps.

Story: Uploads Capacity & Cost Management

Summary

Describe reviews of bucket growth, compression savings, and deployment sizing for presign traffic.

As a…

As a finance partner

I want to…

I want cost oversight for uploads

So that…

So that storage and compute remain efficient

Acceptance Criteria

- [ ] Monthly reviews assess bucket utilization, lifecycle tiering, and compression effectiveness.
- [ ] Scaling strategies detail API gateway caching and autoscaling policies matching deployment guides.

Notification System

Location: /server/src/modules/notifications

TL;DR

The notification system orchestrates outbound communication across email, collaboration tools, and external ticketing channels while preserving regulatory tone and consent requirements.

It is centered around the server/src/modules/notifications package, which exposes shared workflow utilities, template management, channel adapters, and delivery policies aligned with the platform's marketing and risk commitments.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L137-L148■■F:docs/01-about/07-marketing-strategy.md†L70-L154■■F:docs/01-about/10-risk-management-and-Mitigation.md†L80-L153■

This guide details event contracts, API surfaces, integrations with ServiceNow, Jira, Slack, and in-app messaging, plus the observability and testing patterns required to extend notification coverage confidently.■F:docs/02-technical-specifications/07-integration-architecture.md†L1-L177■■F:docs/02-technical-specifications/09-testing-and-qa.md†L141-L203■

Backend Specification

Backend Location & Directory Layout

Notification orchestration lives under server/src/modules/notifications, bundling event processing, template rendering, delivery policies, and channel dispatchers behind a consistent API surface.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L137-L149■

```
server/src/modules/notifications/
■■■ events/
■   ■■■ index.ts
■   ■■■ schemas/
■■■ channels/
■   ■■■ email/
■   ■■■ servicenow/
■   ■■■ jira/
■   ■■■ slack/
■■■ templates/
■   ■■■ registry.json
■   ■■■ email/
■■■ schedulers/
■   ■■■ digest.worker.ts
■■■ services/
■   ■■■ dispatcher.service.ts
■   ■■■ renderer.service.ts
■■■ config/
■   ■■■ delivery-policies.yml
■■■ logging/
    ■■■ notifications.logger.ts
```

Workflow Orchestration

The notification system ensures domain events reach internal and external stakeholders using the correct channel, format, audience segmentation, and timing mandated by marketing and governance policies.■F:docs/01-about/07-marketing-strategy.md†L70-L178■■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L137-L149■ Core packages include events, templates, channels, logging, config, and testing, each handling ingestion, rendering, dispatch, and observability duties. Typical workflow phases progress from event ingestion, eligibility checks, template resolution, payload rendering, channel dispatch, to post-processing with metrics, consent tracking, and retries.

Event Sources & Contracts

- **Governance Engine:** Emits remediation, scoring, and control status events that trigger regulatory and operational alerts (governance.*).■F:docs/03-systems/12-governance-engine/readme.md†L49-L60■
- **Task Management:** Publishes task lifecycle events (created, escalated, resolved) requiring owner notifications and escalations across channels (tasks.*).■F:docs/03-systems/13-task-management-system/readme.md†L51-L115■
- **Framework & Probe Integrations:** Version updates and probe failures are surfaced to stakeholders to keep compliance programs aligned with roadmap expectations (frameworks.*, probes.*).■F:docs/02-technical-specifications/07-integration-architecture.md†L1-L177■

- **Manual Triggers & Preferences:** Operators can invoke /api/v1/notifications/send or /api/v1/notifications/trigger for ad-hoc outreach while respecting opt-in/opt-out states stored in preference records.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L145-L148■

API Surface & Message Pipeline

- **REST Endpoints:** /api/v1/notifications/send, /api/v1/notifications/templates, and /api/v1/notifications/logs expose synchronous operations for sending, template management, and audit queries, following the platform's OpenAPI/REST conventions and Casbin enforcement.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L145-L210■
- **Queue Processing:** Event payloads are normalised into internal schemas (events/schemas) and queued for workers that apply eligibility rules, deduplicate, and route to channel strategies before persistence in notifications_deliveries and alerts tables.■F:docs/02-technical-specifications/04-database-design.md†L97-L133■
- **Template Resolution:** Registry metadata determines locale, branding, consent gates, and marketing persona alignment so copy stays consistent with the go-to-market playbook and targeted segments.■F:docs/01-about/07-marketing-strategy.md†L70-L210■
- **Delivery Policies:** config/delivery-policies.yml enforces throttling, blackout windows, and fallback sequencing aligned with crisis-management SLAs from the business continuity plan.■F:docs/01-about/10-risk-management-and-Mitigation.md†L123-L159■

Email Workflows

- **Triggering Events:** Producers publish messages to the queue through the events package, scheduled jobs in schedulers launch digests, and operators can force sends via POST /api/v1/notifications/trigger.
- **Template Management:** Email templates (templates/email/<locale>/<notification-type>.hbs) register metadata in templates/registry.json, rely on Handlebars helpers, and expose preview endpoints for operations teams.
- **Personalisation & Rendering:** Context builders merge profile service data with event payloads, redacting sensitive fields via maskSensitive. Localised subjects draw from i18n/messages.json, and attachments land in object storage with signed links.
- **Dispatch & Delivery:** Email delivery defaults to AWS SES (override via NOTIFICATIONS_EMAIL_PROVIDER), reuses shared retry policies (3 attempts, exponential backoff), and normalises responses to DeliveryResult. Bulk sends flow through background consumers to avoid throttling.
- **Logging & Audit Trails:** Dispatches log correlation IDs, emit structured events via notifications.logger, and persist receipts in notifications_deliveries, archiving logs older than 180 days per governance rules.

Channel Integrations

Adapters under channels/ extend coverage beyond email while inheriting the integration security controls for OAuth, API keys, and audit logging outlined in the integration architecture.■F:docs/02-technical-specifications/07-integration-architecture.md†L1-L177■

- **ServiceNow:** OAuth 2.0 client credentials (servicenow/notifications) authorise incident/task APIs. High-priority incidents (≥ P2) and change approvals trigger notifications, with delivery results updating incidents via /api/now/table/incident. Nightly reconciliation verifies outstanding tasks.
- **Jira:** Personal access tokens or OAuth 1.0a credentials (jira/notifications) power project automation APIs. Issue transitions and sprint events broadcast updates, while failed deliveries tag issues with notification_failed and acknowledgements push state via webhooks.
- **Slack:** Bot tokens (slack/bot-token) with chat:write, im:write, users:read, and channels:read scopes post real-time alerts and daily digests. Message timestamps enable follow-up edits, and slash commands (/notify-ack, /notify-snooze) synchronise acknowledgement state.
- **In-App & Partner Channels:** Notification contexts in the web client listen for delivery receipts and update UI badges, while outbound webhooks inform partner ecosystems that subscribe to governance events.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L140■■F:docs/02-technical-specifications/07-integration-architecture.md†L74-L149■

Extending Notification Types

- **Define Schema:** Add payload definitions in events/schemas and register them in events/index.ts alongside channel mappings.
- **Implement Strategies:** Extend channels/<channel>/strategies.ts to handle the new type, wiring configuration through config/channels.yml.
- **Register Templates & Localisations:** Create email/Slack/Jira templates, update templates/registry.json, add translation strings to i18n/messages.json, and run yarn notifications:validate for integrity checks.

Frontend Specification

Frontend Location & Directory Layout

Administrative tooling for notifications resides in client/src/features/notifications, giving operators dashboards to preview templates, inspect deliveries, and adjust preferences.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L160■

```
client/src/features/notifications/
■■■ pages/
■   ■■■ NotificationConsolePage.tsx
■   ■■■ TemplatePreviewPage.tsx
■   ■■■ DeliveryLogPage.tsx
■■■ components/
■   ■■■ ChannelToggle.tsx
■   ■■■ DeliveryStatusTable.tsx
■   ■■■ TemplateMetadataForm.tsx
■■■ hooks/
■   ■■■ useNotificationMetrics.ts
■   ■■■ useTemplateRegistry.ts
■■■ api/
    ■■■ notificationsClient.ts

client/src/components/preferences/
■■■ NotificationPreferencesForm.tsx
```

Reusable Components & UI Flows

- **Template Previews:** TemplatePreviewPage reuses TemplateMetadataForm and notificationsClient to render test payloads, applying localisation and persona tagging so preview copy reflects GTM messaging guidelines.■F:docs/01-about/07-marketing-strategy.md†L104-L210■■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L177■
- **Delivery Observability:** DeliveryStatusTable integrates with metrics hooks to display queue depth, failure rates, provider statuses, and webhook health, embedding drill-down links to Grafana dashboards consumed by risk and operations teams.■F:docs/01-about/10-risk-management-and-Mitigation.md†L100-L153■
- **Preference Management:** NotificationPreferencesForm allows users to opt in/out of channels, mirroring profile service records and validating compliance requirements around consent timestamps and GDPR alignment.■F:docs/02-technical-specifications/03-frontend-architecture.md†L123-L177■■F:docs/01-about/04-security-and-data-protection.md†L219-L259■
- **Channel Toggles & Overrides:** ChannelToggle components manage throttling overrides, quiet hours, and blackout windows, tying into the backend DeliveryPolicyOverride API and business continuity escalation tiers.■F:docs/01-about/10-risk-management-and-Mitigation.md†L123-L159■

Schema Specification

- **notifications_events:** Captures inbound domain events, eligibility outcomes, scheduling metadata, and the originating subsystem reference for traceability back to governance or task workflows.■F:docs/03-systems/12-governance-engine/readme.md†L49-L60■■F:docs/03-systems/13-task-management-system/readme.md†L51-L115■

- **notifications_deliveries:** Stores provider message IDs, timestamps, status codes, correlation IDs, and raw payloads for audit purposes in line with immutable logging expectations.■F:docs/01-about/04-security-and-data-protection.md†L261-L312■
- **notifications_templates:** Tracks template versions, locales, persona tags, required context keys, and checksum validation results so marketing copy stays versioned and reviewable.■F:docs/01-about/07-marketing-strategy.md†L70-L210■
- **notification_preferences:** Persists user-level opt-in/out flags, preferred channels, consent timestamps, and legal basis for contact, enabling GDPR-grade auditing of communication choices.■F:docs/01-about/04-security-and-data-protection.md†L219-L259■
- **alerts:** Represents risk or incident alerts emitted to the notification system, linking to tasks for follow-up and aligning with the shared alerts schema described in the database design.■F:docs/02-technical-specifications/04-database-design.md†L97-L133■
- Relationships link to ServiceNow/Jira sync tables and RBAC entities to enforce channel eligibility, audit access, and SLA escalations.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L137-L210■

Operational Playbooks & References

Delivery Policies & Configuration

- Scheduling, throttling, and blackout windows live in config/delivery-policies.yml; per-user limits default to five critical notifications/hour and twenty informational notifications/day so outreach respects customer trust targets and regulatory fatigue thresholds.■F:docs/01-about/07-marketing-strategy.md†L70-L154■
- Retry policies configure attempt counts, delay strategies, and jitter per notification type. Exhausted retries escalate incidents via ServiceNow with the notification-delivery-failure tag, satisfying crisis-response SLAs for critical communications.■F:docs/02-technical-specifications/07-integration-architecture.md†L74-L149■■F:docs/01-about/10-risk-management-and-Mitigation.md†L123-L153■
- User preferences come from the profile service; GDPR-compliant double opt-in and consent records are mandatory for marketing-style messages, with audit logs exposed in the governance portal per security and privacy mandates.■F:docs/01-about/04-security-and-data-protection.md†L219-L312■

Monitoring & Failure Handling

- Prometheus exports notifications_* metrics (queued, sent, delivered, failed, latency) powering Grafana dashboards that highlight failing templates, backlog depth, and webhook retries for the risk committee's reporting cadence.■F:docs/01-about/10-risk-management-and-Mitigation.md†L100-L153■
- Alerts fire when failure rates spike or queue depth exceeds 5,000 for 15+ minutes, paging on-call responders with links to dashboards and runbooks outlining triage steps (provider status,

deployment review, replay, RCA) that align with the incident escalation windows in the continuity plan.■F:docs/01-about/10-risk-management-and-Mitigation.md†L123-L153■

- Log retention spans 30-day hot, 180-day warm, 365-day archive storage with PII redaction; break-glass access is SIEM-monitored so that auditability remains intact during emergency access events.■F:docs/01-about/04-security-and-data-protection.md†L253-L312■

Quality Gates & Testing

- Notification modules include Jest/Vitest suites for template rendering, channel adapters, and event schemas, executed automatically in CI alongside integration and E2E tests covering end-to-end alert delivery.■F:docs/02-technical-specifications/09-testing-and-qa.md†L141-L203■
- CI pipelines notify developers via Slack/email on failure, exercising the service itself and preventing regressions from silently shipping to production.■F:docs/02-technical-specifications/09-testing-and-qa.md†L155-L159■

Related Documentation

- — Primary producer of compliance events and remediation triggers.
- — Consumes notifications for assignments, escalations, and completion workflows.
- — Adapter standards, OAuth scopes, and webhook policies for external channels.
- — Consent, logging, and emergency access rules governing outreach.

Story: Notifications Backend Layout

Summary

Explain the notification module's events, channels, templates, schedulers, and services directories.

As a…

As a messaging engineer

I want to…

I want to locate notification code

So that…

So that I can extend delivery workflows

Acceptance Criteria

- [ ] Directory overview lists events, channel adapters, templates, schedulers, services, and logging modules.
- [ ] Structure clarifies how dispatchers and configuration files power orchestrated deliveries.

Story: Notification Workflow Orchestration

Summary

Describe ingestion, eligibility, template resolution, dispatch, and post-processing phases.

As a…

As a workflow designer

I want to…

I want to map notification steps

So that…

So that events reliably reach the right audiences

Acceptance Criteria

- [ ] Event sources include governance, tasks, frameworks, probes, and manual triggers with schema validation.
- [ ] Pipeline details cover REST endpoints, queue processing, template registry, and delivery policies.

Story: Email Notification Workflows

Summary

Summarize how events become localized emails, from template management to SES delivery and logging.

As a…

As a communications manager

I want to…

I want consistent email operations

So that…

So that regulatory tone and retry policies are honored

Acceptance Criteria

- [ ] Process outlines triggers, template registration, personalization, and dispatch with retry controls.
- [ ] Logging expectations capture correlation IDs, provider responses, and archival retention requirements.

Story: Notification Channel Integrations

Summary

Detail ServiceNow, Jira, Slack, in-app, and webhook adapters with required scopes and behaviors.

As a…

As an integrations engineer

I want to…

I want multi-channel coverage

So that…

So that notifications appear in the systems stakeholders use

Acceptance Criteria

- [ ] Stories describe authentication methods, triggered events, and update callbacks for each external channel.
- [ ] Adapters inherit security policies for OAuth, API keys, and audit logging per integration standards.

Story: Extending Notification Types

Summary

Provide steps for adding new schemas, strategies, templates, and localization assets.

As a…

As a feature developer

I want to…

I want to introduce new notification types

So that…

So that emerging events can be communicated

Acceptance Criteria

- [ ] Workflows cover defining payload schemas, implementing channel strategies, and wiring configuration.
- [ ] Template registration, translation updates, and validation tooling ensure additions pass integrity checks.

Story: Notifications Frontend Structure

Summary

Map the React admin console folders supporting template previews, delivery logs, and preference management.

As a…

As a frontend engineer

I want to…

I want to navigate notification UI code

So that…

So that I can evolve console features

Acceptance Criteria

- [ ] Structure highlights client/src/features/notifications pages, components, hooks, and API clients.
- [ ] Shared preference forms integrate with profile services to respect consent controls.

Story: Notification UI Flows

Summary

Describe template previews, delivery observability dashboards, and channel toggles in the admin experience.

As a…

As an operations lead

I want to…

I want actionable notification dashboards

So that…

So that I can monitor delivery health and adjust policies

Acceptance Criteria

- [ ] UI flows provide localized previews, metrics integrations, and Grafana drill-down links.
- [ ] Preference management enforces consent timestamps, channel throttles, and blackout window overrides.

Story: Notification Delivery Policies

Summary

Summarize throttling, blackout windows, retry behavior, and consent tracking configurations.

As a…

As a risk manager

I want to…

I want notification policy clarity

So that…

So that outreach respects customer trust and compliance

Acceptance Criteria

- [ ] Delivery policies enforce per-user limits, crisis escalation tags, and ServiceNow escalation paths.
- [ ] User preferences source consent metadata with GDPR-compliant double opt-in and audit exposure.

Story: Notification Monitoring & Incident Response

Summary

Outline metrics, alerts, escalation paths, and log retention for notification reliability.

As a…

As an on-call responder

I want to…

I want notification observability

So that…

So that I can resolve delivery failures quickly

Acceptance Criteria

- [ ] Prometheus metrics track queue depth, failure rates, latency, and webhook retries with alert thresholds.
- [ ] Incident response playbooks cover provider status checks, replay strategies, and SIEM-monitored break-glass access.

Story: Notification Quality & Testing

Summary

Detail automated testing, CI alerts, and validation tooling for templates and adapters.

As a…

As a QA specialist

I want to…

I want notification test expectations

So that…

So that rendering and delivery remain trustworthy

Acceptance Criteria

- [ ] Suites include Jest/Vitest tests, integration checks, and E2E flows across channels.
- [ ] CI notifications inform developers of failures and ensure regressions do not ship unnoticed.

Admin and Configuration System

Location: /server/src/modules/admin

TL;DR

The admin and configuration system orchestrates multi-tenant onboarding, policy controls, and integration governance so the rest of the platform can operate with auditable guardrails.

It exposes Express.js REST APIs and a React console to manage tenants, enforce security baselines, and govern external connectors in alignment with platform-wide workflows.

This guide details the JavaScript-only implementation approach, module layout, API contracts, UI flows, data models, and operational dependencies required to ship production-ready features.

Implementation Overview

The admin system anchors the control plane for the continuous governance cycle by enabling teams to provision tenants, configure policies, and route integrations without breaking the evidence-to-remediation loop described in the concept summary.■F:docs/01-about/03-concept-summary.md†L203-L358■

It runs as part of the Node.js + Express backend using JavaScript-only modules, interacting with PostgreSQL via Prisma and adhering to shared API, security, and deployment conventions documented for the platform.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L55-L171■■F:docs/02-technical-specifications/08-deployment-and-environment-guide.md†L138-L142■

Administrative decisions feed downstream services such as Governance Engine, Notifications, Task Management, and Evidence Repository so every configuration change remains traceable and actionable. ■F:docs/03-systems/12-governance-engine/readme.md†L33-L126■■F:docs/03-systems/04-notification-system/readme.md†L56-L126■■F:docs/03-systems/13-task-management-system/readme.md†L1-L226■■F:docs/03-systems/11-evidence-management-system/readme.md†L47-L115■

Backend Specification

Backend Location & Directory Layout

Administrative capabilities live under server/src/modules/admin, following the platform’s feature-based layout, JavaScript naming conventions, and Express routing patterns.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L55-L68■

```
server/src/modules/admin/
■■■ api/
■   ■■■ tenants.controller.js
■   ■■■ settings.controller.js
■   ■■■ integrations.controller.js
■   ■■■ admin.routes.js
■■■ workflows/
■   ■■■ verify-tenant.workflow.js
■   ■■■ bootstrap-environment.workflow.js
■   ■■■ rotate-credential.workflow.js
■■■ services/
■   ■■■ tenant.service.js
■   ■■■ configuration.service.js
■   ■■■ feature-flags.service.js
■   ■■■ integration.service.js
■■■ repositories/
■   ■■■ tenant.repository.js
■   ■■■ settings.repository.js
■   ■■■ integration.repository.js
■■■ events/
■   ■■■ admin.event-publisher.js
■   ■■■ schemas/
■■■ telemetry/
■   ■■■ admin.audit.logger.js
■   ■■■ metrics.js
■■■ scripts/
    ■■■ seed-admin-data.js
```

Routes register in server/src/routes/admin.routes.js, mounting the /api/v1/admin router and reusing shared middleware for authentication, validation, rate limiting, and Casbin enforcement consistent with other mod ules.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L56-L104■■F:docs/02-te chnical-specifications/06-security-implementation.md†L60-L99■

Core Responsibilities & Workflows

- **Tenant Provisioning Lifecycle:** POST /api/v1/admin/tenants invokes verify-tenant.workflow.js, validating organization metadata, RBAC seeds, and billing context before queuing bootstrap-environment.workflow.js to create evidence namespaces, default notification templates, dashboards, and probe stubs required by onboarding standards.■F:docs/03-systems/07-probe-man agement-system/readme.md†L27-L142■■F:docs/02-technical-specifications/08-deployment-and-en vironment-guide.md†L138-L142■ Failures emit admin.tenant.failed events consumed by Notifications for follow-up and Task Management for remediation tickets.■F:docs/03-systems/04-not ification-system/readme.md†L56-L126■■F:docs/03-systems/13-task-management-system/readme. md†L1-L226■

- **Delegated Administration:** Tenant owners manage user invites, role assignments, and approval workflows, relying on Casbin-backed middleware and RBAC runbooks to enforce separation of duties and tenant scoping.■F:docs/03-systems/02-rbac-system/readme.md†L23-L116■■F:docs/02-technical-specifications/06-security-implementation.md†L65-L99■ Governance Engine surfaces configuration diffs and pending approvals, ensuring administrators can reconcile changes alongside control scoring.■F:docs/03-systems/12-governance-engine/readme.md†L33-L132■
- **Global Policy & Secrets Governance:** configuration.service.js centralizes platform policy toggles (auth requirements, session rules, evidence retention) with dual-approval workflows, while rotate-credential.workflow.js coordinates Vault/Key Vault updates, audit logging, and notification hooks to meet security standards.■F:docs/02-technical-specifications/06-security-implementation.md†L100-L166■■F:docs/03-systems/15-external-integrations-system/readme.md†L96-L211■
- **Integration Control Plane:** Admin operators register third-party connectors, map scopes, and schedule health checks that orchestrate Probe Management, External Integrations, and Governance Engine subscribers for continuous evidence intake.■F:docs/03-systems/07-probe-management-system/readme.md†L139-L210■■F:docs/02-technical-specifications/07-integration-architecture.md†L69-L172■■F:docs/03-systems/15-external-integrations-system/readme.md†L52-L211■

Administrative API Surface

APIs follow REST and OpenAPI standards shared across the backend, including pagination, optimistic concurrency via If-Match headers, and correlation IDs for auditability.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L194-L200■ Key endpoints:

Method	*Path*	*Description*
POST	/api/v1/admin/tenants	Create tenant, run verification workflow, seed RBAC defaults, and emit provisioning events.
GET	/api/v1/admin/tenants/:tenantId	Retrieve tenant metadata, lifecycle status, and compliance readiness indicators for governance dashboards.
PATCH	/api/v1/admin/tenants/:tenantId/status	Transition tenant lifecycle states (pending, active, suspended, deprovisioning) with audit comment requirements.
GET/PUT	/api/v1/admin/settings	Read or update global policies (password rotation, MFA enforcement, evidence retention) guarded by dual approvals and change tickets.
GET/PUT	/api/v1/admin/settings/:tenantId	Override policies per tenant with automatic inheritance checks and rollback support.

Method	Path	Description
POST	/api/v1/admin/feature-flags	Define feature toggles, rollout cohorts, and kill switches referencing LaunchDarkly identifiers with staged release metadata.■F:docs/02-technical-specifications/07-integration-architecture.md†L69-L145■
POST	/api/v1/admin/integrations	Register external connectors, capture OAuth/API credentials, configure scopes, and schedule health checks shared with Probe Management.■F:docs/03-systems/15-external-integrations-system/readme.md†L52-L172■
POST	/api/v1/admin/integrations/:id/rotate	Trigger credential rotation workflow, update secrets manager, notify stakeholders, and document evidence artifacts.■F:docs/03-systems/15-external-integrations-system/readme.md†L173-L211■
GET	/api/v1/admin/audit-events	Stream configuration change history, dual approvals, and workflow results linked to audit logging and evidence systems.■F:docs/02-technical-specifications/06-security-implementation.md†L124-L144■■F:docs/03-systems/11-evidence-management-system/readme.md†L47-L115■

Configuration, Security & Observability

- **Authentication & Authorization:** All routes require JWT authentication, Casbin checks, and tenant-domain scoping enforced by shared middleware, aligning with security implementation and RBAC mandates.■F:docs/02-technical-specifications/06-security-implementation.md†L60-L115■■F:docs/03-systems/02-rbac-system/readme.md†L23-L88■
- **Secrets & Environment Variables:** Environment config uses uppercase snake-case variables (e.g., ADMIN_LAUNCHDARKLY_SDK_KEY, ADMIN_SECRETS_PROVIDER) managed through deployment pipelines and secret vault integrations outlined in security guidance.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L66-L68■■F:docs/02-technical-specifications/06-security-implementation.md†L100-L121■
- **Eventing & Audit:** admin.event-publisher.js emits structured events (admin.tenant.created, admin.setting.updated, admin.integration.health) to the shared message bus so Notifications, Governance, and Audit Logging systems maintain state parity and compliance evidence.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L1-L200■■F:docs/03-systems/04-notification-system/readme.md†L56-L126■■F:docs/03-systems/12-governance-engine/readme.md†L90-L156■
- **Telemetry & Metrics:** Metrics exported via metrics.js include onboarding duration, approval latency, integration heartbeat failures, and rotation throughput; these feed Grafana dashboards referenced in operations guides for notifications and audit logging.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L120-L200■■F:docs/03-systems/04-notification-system/readme.md†L103-L126■
- **Error Handling & Rate Limits:** Shared middleware enforces validation, rate limiting on sensitive endpoints, and structured error responses consistent with platform standards to mitigate abuse and maintain SLA commitments.■F:docs/02-technical-specifications/06-security-implementation.md†L147-L182■

Frontend Specification

Frontend Location & Directory Layout

The admin console is embedded within the Governance Engine UI under client/src/features/admin, implemented in JavaScript (no TypeScript) and organized per the feature-based React architecture guidelines.■F:docs/02-technical-specifications/03-frontend-architecture.md†L17-L78■

```
client/src/features/admin/
■■■ pages/
■   ■■■ tenant-onboarding-wizard.jsx
■   ■■■ settings-overview-page.jsx
■   ■■■ integration-catalog-page.jsx
■   ■■■ audit-trail-page.jsx
■■■ components/
■   ■■■ delegated-admin-table.jsx
■   ■■■ feature-flag-toggle.jsx
■   ■■■ secret-rotation-modal.jsx
■   ■■■ integration-health-card.jsx
■■■ hooks/
■   ■■■ use-tenant-provisioning.js
■   ■■■ use-admin-settings.js
■   ■■■ use-integration-health.js
■■■ api/
■   ■■■ admin-client.js
■■■ state/
    ■■■ admin-store.js

client/src/components/governance/
■■■ config-diff-viewer.jsx
```

Shared route guards (RequirePermission) from the RBAC system wrap admin routes to ensure UI state matches backend authorization checks.■F:docs/03-systems/02-rbac-system/readme.md†L60-L88■ All files adhere to lowercase hyphen naming conventions with PascalCase component exports, aligning with frontend standards.■F:docs/02-technical-specifications/03-frontend-architecture.md†L29-L72■

UI Workflows & State Management

- **Onboarding Wizard:** Multi-step wizard collects tenant metadata, verifies prerequisites, and polls provisioning status using use-tenant-provisioning.js, surfacing failure reasons tied to Notification events and remediation tasks.■F:docs/03-systems/07-probe-management-system/readme.md†L27-L142■■F:docs/03-systems/04-notification-system/readme.md†L56-L103■
- **Policy & Feature Management:** settings-overview-page.jsx and feature-flag-toggle.jsx enforce dual approvals by requiring two distinct session confirmations before enabling admin-client.js mutations, with UI copy referencing security posture expectations such as MFA and encryption policies.■F:docs/02-technical-specifications/06-security-implementation.md†L71-L166■

- **Integration Catalog & Health:** integration-catalog-page.jsx visualizes connector scopes, heartbeat metrics, and incident summaries, linking to Probe Management and External Integration runbooks for deeper triage guidance.■F:docs/02-technical-specifications/07-integration-architecture.md†L69-L198■■F:docs/03-systems/07-probe-management-system/readme.md†L139-L210■
- **Audit Visibility & Diffing:** audit-trail-page.jsx pairs with config-diff-viewer.jsx to present immutable change history with filters for actor, tenant, resource, and timeframe, ensuring parity with audit logging retention policies and evidence linking requirements.■F:docs/02-technical-specifications/06-security-implementation.md†L124-L144■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L1-L200■
- **State Management:** Local state leverages React hooks and Context (auth, notifications) as described in frontend architecture, while feature-specific caches (e.g., onboarding status) use SWR-style polling via useEffect loops inside hooks for predictable updates.■F:docs/02-technical-specifications/03-frontend-architecture.md†L84-L132■

Schema Specification

Admin persistence relies on Prisma models living in the shared PostgreSQL schema, applying the database design principles around normalization, UUID keys, and soft deletes.■F:docs/02-technical-specifications/04-database-design.md†L38-L133■ Core tables:

- **admin_tenants** – Stores tenant identifiers, lifecycle state, billing tier, data residency, approval metadata, and timestamps, linking to RBAC assignments and governance contexts for onboarding workflows.■F:docs/02-technical-specifications/04-database-design.md†L69-L139■■F:docs/03-systems/02-rbac-system/readme.md†L91-L96■
- **admin_settings** – Captures global configuration values, effective/expiry windows, dual-approval records, and references to evidence artifacts documenting policy changes.■F:docs/02-technical-specifications/04-database-design.md†L38-L133■■F:docs/03-systems/11-evidence-management-system/readme.md†L47-L115■
- **admin_flags** – Manages feature toggle definitions, rollout cohorts, audit trails, and launch metadata for experimentation aligned with integration architecture guidance.■F:docs/02-technical-specifications/07-integration-architecture.md†L69-L145■
- **admin_integrations** – Persists connector configuration (type, scopes, credential references, health status) and scheduling metadata tied to Probe Management and External Integrations systems.■F:docs/03-systems/07-probe-management-system/readme.md†L139-L210■■F:docs/03-systems/15-external-integrations-system/readme.md†L52-L211■
- **admin_audit_events** – Append-only ledger recording actor, action, payload diff, correlation IDs, and evidence references, enforcing retention and immutability expectations set by security and audit logging documentation.■F:docs/02-technical-specifications/06-security-implementation.md†L124-L144■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L1-L200■

Foreign keys connect admin tables to users, roles, notifications, tasks, and evidence tables following the relationship model described in the database design specification so downstream services can join data efficiently.■F:docs/02-technical-specifications/04-database-design.md†L125-L141■

Operational Playbooks & References

Audit & Compliance Expectations

- **Logging & Retention:** admin.audit.logger.js writes structured events to the centralized audit pipeline with the 36-month retention baseline, exposing dashboards for auditors through Governance Engine views.■F:docs/02-technical-specifications/06-security-implementation.md†L124-L144■■F:docs/03-systems/12-governance-engine/readme.md†L110-L132■
- **Evidence Capture:** Provisioning, policy changes, and rotations automatically store supporting records (artifacts, approvals, tickets) in the Evidence Repository so auditors can trace configuration history.■F:docs/03-systems/11-evidence-management-system/readme.md†L47-L115■
- **Periodic Reviews:** Quarterly/annual reviews reconcile RBAC assignments, overrides, and integration scopes, triggering automated tasks and notifications for remediation when discrepancies appear, keeping governance cycles unbroken.■F:docs/03-systems/02-rbac-system/readme.md†L97-L115■■F:docs/03-systems/13-task-management-system/readme.md†L1-L226■

Runbooks & Inter-Service References

- — scheduling and health orchestration for connectors and evidence collection.
- — alerting paths for onboarding failures, policy approvals, and credential rotations.
- — embedded admin experience, configuration diffing, and risk scoring integration.
- — connector catalog, authorization flows, and incident handling.
- — encryption, MFA, incident response, and audit controls.
- — external API standards, webhook governance, and feature flag rollout.

Story: Admin Backend Layout

Summary

Describe where administrative controllers, services, jobs, and configuration stores live.

As a…

As a platform engineer

I want to…

I want to navigate admin backend code

So that…

So that I can extend tenant provisioning tools

Acceptance Criteria

- [ ] Directory overview covers controllers, services, schedulers, policy stores, and integration adapters.
- [ ] Routing and job orchestration notes explain how admin APIs interact with governance modules.

Story: Admin Core Workflows

Summary

Summarize tenant onboarding, policy propagation, secrets rotation, and environment configuration responsibilities.

As a…

As a program manager

I want to…

I want to understand admin workflows

So that…

So that tenants launch with compliant defaults

Acceptance Criteria

- [ ] Workflows include provisioning tenants, seeding RBAC roles, syncing configs, and rotating secrets.
- [ ] Automation hooks trigger notifications, audit logs, and cross-system updates during lifecycle changes.

Story: Admin API Surface

Summary

Detail REST endpoints that manage tenants, configuration bundles, and delegated administration.

As a…

As an API integrator

I want to…

I want admin endpoint clarity

So that…

So that external automation can manage org settings

Acceptance Criteria

- [ ] Endpoints describe CRUD operations for tenants, settings, secrets, and delegation relationships.
- [ ] Contracts highlight authentication, audit logging, and rollback hooks for misconfigurations.

Story: Admin Configuration & Security

Summary

Explain infrastructure, secrets management, and observability controls enforced by the admin service.

As a…

As a DevOps owner

I want to…

I want secure admin configuration

So that…

So that sensitive settings stay protected

Acceptance Criteria

- [ ] Guidance covers vault integration, encryption, Terraform workflows, and config drift detection.
- [ ] Observability notes include audit logging, metrics, and alerting for administrative changes.

Story: Admin Frontend Structure

Summary

Map the React admin console directories supporting configuration pages and shared state.

As a…

As a frontend contributor

I want to…

I want to find admin UI code

So that…

So that I can maintain configuration experiences

Acceptance Criteria

- [ ] Structure highlights navigation shells, feature bundles, forms, and state managers dedicated to admin tasks.
- [ ] Links to RBAC, notification, and tenant management modules show how the console stays cohesive.

Story: Admin UI Workflows

Summary

Describe dashboard navigation, inline editing, review flows, and optimistic updates in the admin experience.

As a…

As a product owner

I want to…

I want consistent admin UI patterns

So that…

So that operators can configure tenants without confusion

Acceptance Criteria

- [ ] UI flows cover onboarding wizards, configuration diff viewers, and approval modals for risky changes.
- [ ] State management explains React Query caches, feature flags, and cross-module breadcrumbs.

Story: Admin Audit & Compliance

Summary

Summarize audit logging, approval checkpoints, and evidence capture for administrative actions.

As a…

As a compliance officer

I want to…

I want admin control traceability

So that…

So that auditors can verify changes

Acceptance Criteria

- [ ] Every admin action records immutable audit entries with justifications and reviewer metadata.
- [ ] Controls enforce dual-approval, time-bound delegations, and evidence exports for compliance reviews.

Story: Admin Runbooks & References

Summary

Document incident response, cross-service dependencies, and linked documentation for admin operations.

As a…

As an incident commander

I want to…

I want admin runbooks

So that…

So that I can coordinate responses across systems

Acceptance Criteria

- [ ] Runbooks map dependencies to RBAC, Governance, and Infrastructure modules with escalation paths.
- [ ] References include health checks, rollback steps, and checklists for tenant lifecycle events.

Audit Logging and Monitoring

Location: /server/src/lib/logging

TL;DR

The audit logging and monitoring system guarantees immutable, end-to-end visibility into security-critical activity.

Built around server/src/lib/logging/audit.js, it captures structured events, enforces retention and tamper controls, and feeds observability pipelines for governance oversight.

Use this guide to understand capture scope, storage contracts, operational safeguards, and developer instrumentation practices.

Backend Specification

Backend Location & Directory Layout

Audit instrumentation is implemented under server/src/lib/logging, exposing shared helpers and transports that every service consumes for structured event capture.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L124-L171■

```
server/src/lib/logging/
■■■ audit.ts
■■■ middleware/
■   ■■■ attach-audit-context.ts
■   ■■■ request-metadata.ts
■■■ transports/
■   ■■■ winston-console.ts
■   ■■■ winston-file.ts
■   ■■■ winston-https.ts
■■■ formatters/
■   ■■■ redact-fields.ts
■   ■■■ correlation.ts
■■■ retention/
■   ■■■ glacier-archiver.ts
■■■ tests/
    ■■■ audit.logger.spec.ts
```

Core Modules & Responsibilities

- **audit.ts:** Factory that returns a Winston logger scoped to the calling module, wraps JSON serialization, and enforces shared default fields (service, tenant, correlationId).
- **middleware/attach-audit-context.ts:** Express middleware that injects tenant, actor, and request metadata, aligning with Casbin-based RBAC enforcement before handlers run.■F:docs/01-about/04-security-and-data-protection.md†L182-L259■■F:docs/02-technical-specifications/06-security-implementation.md†L58-L109■
- **formatters/redact-fields.ts:** Utility that masks sensitive attributes (PII, credentials) according to the data classification policy and privacy constraints.■F:docs/01-about/04-security-and-data-protection.md†L120-L169■
- **retention/glacier-archiver.ts:** Cron-safe job that packages batches for cold storage, publishes integrity hashes, and updates the ledger tables.
- **transports/winston-https.ts:** Streams structured events into the centralized log collector (OpenSearch, Loki, Splunk) with mTLS enforced per environment.■F:docs/02-technical-specifications/01-system-architecture.md†L204-L224■

API Endpoints & Event Contracts

- **REST hooks:** /api/v1/audit/events (ingest events from trusted services), /api/v1/audit/exports (generate signed archive manifests), /api/v1/audit/ledger/:batchId (validate integrity) – secured via service tokens with Casbin scopes audit:write, audit:read, and audit:verify respectively.■F:docs/02-technical-specifications/06-security-implementation.md†L58-L144■
- **Event bus topics:**
- audit.event.appended.v1 — emitted for every persisted event and consumed by Notification and Task services to trigger escalations.■F:docs/03-systems/04-notification-system/readme.md†L7-L184■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■

- audit.archive.created.v1 — signals completion of immutable snapshots so Evidence Management can reconcile retention schedules.■F:docs/02-technical-specifications/04-database-design.md†L148-L162■
- audit.breakglass.access.v1 — broadcast when privileged access is granted, aligning with emergency access procedures.■F:docs/01-about/04-security-and-data-protection.md†L253-L260■

Configuration & Environment Controls

- **Environment variables:** AUDIT_TRANSPORTS, AUDIT_REMOTE_ENDPOINT, AUDIT_RETENTION_DAYS, AUDIT_GLACIER_VAULT, AUDIT_LEDGER_SALT. Secrets are delivered through the platform's secret management process with dual control and rotation windows aligned to security council directives.■F:docs/01-about/08-operations-and-teams.md†L124-L170■■ F:docs/02-technical-specifications/01-system-architecture.md†L204-L235■
- **Initialization:** server/src/index.js loads attach-audit-context ahead of route registration to guarantee traceability for every request path, including health checks and integration callbacks.
- **Deployment:** Container builds must include the winston-* transports and @opentelemetry/api peer dependency, matching the logging guidance in the system architecture overview.■F:docs/02-technical-specifications/01-system-architecture.md†L227-L235■

Observability & Metrics

- Metrics exported via OpenTelemetry counters (audit_events_total, audit_failures_total, audit_latency_seconds) feed Datadog and Grafana dashboards managed by the Security & Compliance squad to satisfy 24/7 monitoring commitments.■F:docs/01-about/08-operations-and-teams.md†L29-L170■
- Log ingestion lag, archive duration, and ledger verification failures emit to Prometheus alert rules with routing through Notification's high-priority channel set.■F:docs/03-systems/04-notification-system/readme.md†L7-L184■
- Weekly automated conformance reports reconcile SIEM counts with PostgreSQL partitions to prove end-to-end completeness.

Capture Scope & Storage Pipelines

- **Capture Scope:** Log authentication events, privileged actions, configuration changes, data exports, workflow transitions, critical business operations, and system lifecycle events. Each entry includes timestamp (UTC), actor identity, action, target resource, request origin, and outcome.
- **Application Layer:** Winston transports emit structured JSON tagged with category: "audit" and correlation IDs. Logs write to rotating files (≤24 hours) for local debugging and to stdout for container capture.
- **Centralized Logging:** HTTPS or sidecar agents stream audit events into dedicated indices/streams (OpenSearch, Loki, Splunk) with RBAC-separated access from general application logs.

- **Developer Instrumentation:** Teams use auditLogger helpers, mask sensitive fields via redactFields, attach request/session IDs, and validate with npm run lint:audit and npm run test:audit before merging.
- **Event shape:**

```
{
  "timestamp": "2026-06-04T12:34:56.123Z",
  "service": "governance-engine",
  "actor": { "id": "user_123", "role": "compliance_officer" },
  "action": "CONTROL_OVERRIDE_APPROVED",
  "target": { "type": "control", "id": "ctl-42001-17" },
  "origin": { "ip": "203.0.113.10", "userAgent": "Mozilla/5.0" },
  "correlationId": "req-abc-123",
  "outcome": "success",
  "hash": "sha256:e3f6…",
  "redactions": ["justification"]
}
```

- **Integrity pipeline:** After local persistence, events are batched every 5 minutes, hashed with the ledger salt, written to audit_integrity_ledger, and forwarded to cold storage when the archive job runs. Failed batches raise audit.event.retry metrics and open remediation tasks for the operations sq uad.■F:docs/02-technical-specifications/04-database-design.md†L97-L183■■F:docs/03-systems/1 3-task-management-system/readme.md†L7-L226■

Retention & Immutability Controls

- **Retention Policies:** Maintain centralized audit data for ≥36 months (minimum regulatory floor 400 days) and extend per jurisdiction. Archive immutable snapshots to object storage (WORM) every 30 days for at least seven years; purge local file transports after 24 hours once uploads succeed.
- **Tamper Protection:** Enforce append-only permissions on audit indices; restrict delete rights to security/compliance with change-control approval. WORM storage (S3 Glacier Vault Lock equivalents) protects archives, while batch hashes stored in an integrity ledger detect tampering.
- **Monitoring & Alerting:** Feed audit streams into the SIEM for correlation. Alert on privilege escalations, repeated denials, or failed admin logins, with dashboards tracking event volume and failure rates linked to triage runbooks.
- **Partitioning:** PostgreSQL tables follow monthly partitions with automatic retention policies and ledger checkpoints to align with the database design's partitioning guidance.■F:docs/02-technical-specifications/04-database-design.md†L55-L162■
- **Regulatory alignment:** Retention schedules mirror the privacy and accountability commitments defined in the security and data protection overview, ensuring transparency, lawful access, and audit-ready reporting.■F:docs/01-about/04-security-and-data-protection.md†L120-L180■

Frontend Specification

Frontend Location & Directory Layout

Audit visibility surfaces in the governance console under client/src/features/observability/audit, providing dashboards and review tooling for security teams.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L160■

```
client/src/features/observability/audit/
■■■ pages/
■   ■■■ AuditTimelinePage.tsx
■   ■■■ AuditReviewQueuePage.tsx
■   ■■■ RetentionPolicyPage.tsx
■■■ components/
■   ■■■ AuditEventTable.tsx
■   ■■■ IntegrityHashViewer.tsx
■   ■■■ BreakGlassRequestModal.tsx
■■■ hooks/
■   ■■■ useAuditStream.ts
■   ■■■ useRetentionPolicies.ts
■■■ api/
    ■■■ auditClient.ts

client/src/components/security/
■■■ CriticalAlertBanner.tsx
```

Reusable Components & UI Flows

- **Audit Timeline:** AuditEventTable streams events with filters for actor, resource, and outcome, embedding links to SIEM dashboards for deeper analysis.
- **Review Queue:** AuditReviewQueuePage manages quarterly integrity checks, break-glass approvals, and remediation tracking, using BreakGlassRequestModal to capture justification and expiry.
- **Retention Administration:** RetentionPolicyPage visualizes lifecycle status (hot, warm, archive) and integrates with useRetentionPolicies to modify schedules under dual-control workflows.
- **Critical Alerts:** CriticalAlertBanner surfaces SIEM-driven incidents, linking responders to runbooks and providing context from correlated audit entries.
- **State management:** Hooks use React Query caching keyed by tenantId and timeRange, ensuring least-privilege data access while honoring operations governance for secure, agile delivery.■F:docs/01-about/08-operations-and-teams.md†L29-L139■■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L160■
- **Accessibility:** All tables render with keyboard navigation, aria-live regions for streaming updates, and redaction badges that mirror privacy-by-design commitments.■F:docs/01-about/04-security-and-data-protection.md†L165-L180■

API & Data Contracts

- auditClient.ts exposes listEvents, getLedgerBatch, requestArchiveExport, and acknowledgeAlert methods built on the shared HTTP client, applying JWT tokens and Casbin scopes from the Auth service.■F:docs/02-technical-specifications/06-security-implementation.md†L58-L109■
- WebSocket subscriptions use /ws/v1/audit/stream with token re-auth every 30 minutes; reconnect logic abides by exponential backoff policies shared across observability features.
- Filters persist via URL search params and Redux-backed feature flags so admins can share deep links during compliance reviews.

Schema Specification

- **audit_events (central index/table):** Timestamp, actor, action, target, request origin, outcome, correlation ID, hash, and redaction metadata.
- **audit_archives:** Records monthly snapshot manifests, storage location, hash digests, and WORM lock identifiers.
- **audit_access_requests:** Tracks break-glass approvals, requester, approver, reason, scope, and expiration.
- **audit_integrity_ledger:** Stores cryptographic signatures for batches, enabling tamper verification across archives.
- Relationships integrate with RBAC subjects, Notification escalations, and Task Service remediation tickets to maintain traceability.■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■
- Schema definitions follow Prisma models managed within the externally hosted PostgreSQL cluster, inheriting partitioning, retention, and encryption strategies from the database design specification.■F:docs/02-technical-specifications/04-database-design.md†L41-L183■
- Access controls map to Casbin policies, ensuring roles (Admin, Auditor, Compliance Officer) retrieve only scoped records in alignment with RBAC commitments.■F:docs/02-technical-specifications/06-security-implementation.md†L80-L109■

Operational Playbooks & References

Access Restrictions & Review Processes

- Grant read-only access to security, compliance, and incident response roles; default-deny engineering access. Temporary investigative access follows break-glass procedures with approval logging and expiry enforcement.

- Schedule weekly automated reports summarizing critical events for security leadership sign-off, plus quarterly formal reviews documenting retention adherence, integrity checks, and remediation status in the compliance tracker.
- Post-review action items convert into Task Service tickets, and outcomes feed Operations governance dashboards to uphold KPI visibility for leadership.■F:docs/01-about/08-operations-and-teams.md†L142-L170■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■

Related Documentation

- — change governance and audit event producers.
- — escalation channels for high-risk events.
- — SIEM integrations and tamper controls.
- — logging sidecar and retention configuration per environment.

Implementation Checklist

- Confirm middleware registration for attach-audit-context in every Express app before route definitions, ensuring consistent trace metadata.■F:docs/02-technical-specifications/01-system-architecture.md†L204-L235■
- Write unit tests for new audit events (audit.logger.spec.ts) plus integration tests exercising ledger updates and archive exports; include them in CI by invoking npm run test:audit and retaining ≥90% coverage to satisfy QA policy.■F:docs/02-technical-specifications/09-testing-and-qa.md†L1-L120■
- Update Prisma schema migrations when introducing new audit entities, run npm run migrate:audit, and coordinate with DevOps for external Postgres promotion as described in the database design guide.■F:docs/02-technical-specifications/04-database-design.md†L41-L162■
- Document operational impacts (new alerts, retention changes) in the weekly security council briefing so governance, compliance, and operations squads maintain shared situational awareness.■F:docs/01-about/08-operations-and-teams.md†L124-L170■

Story: Audit Backend Layout

Summary

Describe module directories for collectors, pipelines, and retention services handling audit logs.

As a…

As a logging engineer

I want to…

I want to locate audit code

So that…

So that I can extend ingestion or export flows

Acceptance Criteria

- [ ] Layout covers collectors, pipelines, API handlers, storage drivers, and configuration modules.
- [ ] Notes explain how middleware feeds audit controllers and how logs move to immutable storage.

Story: Audit Core Responsibilities

Summary

Summarize modules capturing events, exposing APIs, configuring environments, and surfacing observability metrics.

As a…

As a platform architect

I want to…

I want to understand audit duties

So that…

So that every action is recorded with context

Acceptance Criteria

- [ ] Responsibilities include event ingestion, query APIs, retention enforcement, and monitoring instrumentation.
- [ ] Configuration controls govern environments, rate limits, and metric exports for audit pipelines.

Story: Audit Capture Pipeline

Summary

Explain end-to-end capture scope, storage tiers, and data flows for audit events.

As a…

As a compliance engineer

I want to…

I want to trace audit data movement

So that…

So that evidence stays immutable

Acceptance Criteria

- [ ] Pipelines describe ingestion from services, normalization, enrichment, and streaming to data lakes.
- [ ] Storage tiers outline hot, warm, and archive buckets with encryption and access controls.

Story: Audit Retention Controls

Summary

Detail retention policies, immutability guarantees, and export tooling for audits.

As a…

As a regulator liaison

I want to…

I want retention assurance

So that…

So that audits meet regulatory timelines

Acceptance Criteria

- [ ] Retention policies define durations, legal hold handling, and write-once storage characteristics.
- [ ] Export tooling supports compliance requests with tamper-evident packaging and chain-of-custody logs.

Story: Audit Frontend Structure

Summary

Identify console directories for audit viewers, search interfaces, and evidence exports.

As a…

As a frontend engineer

I want to…

I want to navigate audit UI code

So that…

So that I can maintain log browsing tools

Acceptance Criteria

- [ ] Structure references client/src/features/audit pages, components, and API clients used to display logs.
- [ ] UI hooks coordinate filters, pagination, and export requests tied to authorization checks.

Story: Audit UI Flows

Summary

Describe dashboards, detail views, and export workflows available to authorized reviewers.

As a…

As an auditor

I want to…

I want intuitive log review experiences

So that…

So that I can investigate events efficiently

Acceptance Criteria

- [ ] UI flows surface timeline views, correlation explorers, and immutable export actions.
- [ ] Access controls restrict visibility based on RBAC scopes and track review acknowledgements.

Story: Audit Access & Review Processes

Summary

Summarize access restrictions, approval workflows, and monitoring for audit data access.

As a…

As a security officer

I want to…

I want controlled audit access

So that…

So that sensitive logs remain protected

Acceptance Criteria

- [ ] Policies enforce least privilege, break-glass tracking, and periodic entitlement reviews.
- [ ] Monitoring alerts on anomalous queries, mass exports, or privilege escalations.

Story: Audit Implementation Checklist

Summary

Provide step-by-step rollout guidance and validation tasks for deploying audit logging components.

As a…

As a delivery lead

I want to…

I want an audit deployment checklist

So that…

So that new environments are configured correctly

Acceptance Criteria

- [ ] Checklist covers provisioning storage, configuring pipelines, setting environment variables, and enabling metrics.
- [ ] Validation tasks include running smoke tests, verifying retention policies, and confirming alert wiring.

Probe Management System

Location: /server/src/modules/probes

TL;DR

The Probe Management System orchestrates how evidence-collection probes are registered, deployed, scheduled, and monitored across environments.

It exposes a Probe SDK (implemented in server/src/modules/probes) that standardizes authentication, retries, payload schemas, and version negotiation.

This guide explains lifecycle workflows, API contracts, configuration patterns, and operational practices for integrating probes with enterprise systems.

Backend Specification

Backend Location & Directory Layout

Probe orchestration and SDK utilities live in server/src/modules/probes, coordinating registry APIs, deployment workflows, scheduling, and health monitoring.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L124-L171■

```
server/src/modules/probes/
■■■ api/
■   ■■■ probes.controller.ts
■   ■■■ deployments.controller.ts
■   ■■■ schedules.controller.ts
■■■ services/
■   ■■■ registry.service.ts
■   ■■■ deployment.service.ts
■   ■■■ scheduler.service.ts
■   ■■■ health.service.ts
■■■ sdk/
■   ■■■ ProbeClient.ts
■   ■■■ ProbeScheduler.ts
■   ■■■ ProbeHealthClient.ts
■   ■■■ ProbeConfigLoader.ts
■■■ events/
■   ■■■ probe.heartbeat.ts
■   ■■■ probe.evidence.ts
■   ■■■ probe.failure.ts
■■■ workflows/
■   ■■■ registerProbe.workflow.ts
■   ■■■ rolloutProbe.workflow.ts
■■■ tests/
    ■■■ probe.management.spec.ts
```

Configuration & Environment

- Modules run on the platform's JavaScript-only stack and draw configuration from the shared .env pattern (no TypeScript or bespoke config loaders).■F:docs/02-technical-specifications/01-system-architecture.md†L41-L145■
- Extend the global template with probe-specific keys such as PROBE_REGISTRY_WEBHOOK_SECRET, PROBE_HEARTBEAT_INTERVAL, PROBE_DEPLOYMENT_TOPIC, and PROBE_SDK_VERSION_MIN, loaded through dotenv and secret vaults as prescribed in the deployment guide.■F:docs/02-technical-specifications/08-deployment-and-environment-guide.md†L100-L142■
- Scheduler workers consume the platform-wide external PostgreSQL and MinIO endpoints (DATABASE_URL, MINIO_ENDPOINT, etc.), ensuring probes never embed infrastructure coordinates directly in code.■F:docs/02-technical-specifications/01-system-architecture.md†L50-L180■■F:docs/02-technical-specifications/08-deployment-and-environment-guide.md†L51-L188■

System Components

- **Registry Service:** Persists probe metadata (ID, owner, framework bindings, evidence types, version) and exposes /api/v1/probes CRUD operations, lifecycle transitions, credential issuance, and environment overlays.

- **Deployment Coordinator:** Generates manifests from registry templates and environment overlays, integrates with CI/CD to build/tag artifacts, publishes deployment intents, and tracks rollout states for auditability.
- **Scheduler & Execution Plane:** Maintains cron, event-driven, and ad-hoc schedules through ProbeScheduler, translating definitions into orchestration primitives (Kubernetes CronJobs, Airflow DAGs, serverless invocations) with tenant-isolated sandboxes.
- **Observability & Alerting:** Streams heartbeat, status, and metrics into OpenTelemetry collectors (Prometheus + Loki), normalizing logs with probe identifiers and control mappings for correlation.

API Endpoints & Contracts

The REST layer follows the platform's resource-first patterns, JSON payload envelopes, and Swagger documentation, with JWT + Casbin guardrails on every route.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L194-L216■

Method	*Path*	*Purpose*	*Required RBAC Role*	*Sample Payload*
GET	/api/v1/probes	Paginated list with filters (status, frameworkId, owner).	Compliance Officer / Engineer	GET /api/v1/probes?status=active&frameworkId=nist-800-53
POST	/api/v1/probes	Register probe drafts, attach frameworks, set evidence schema references.	Engineer	{ "name": "probe-snowflake", "frameworkBindings": ["nist-800-53"], "evidenceSchema": { ... } }
POST	/api/v1/probes/:id/deployments	Kick off rollout with artifact version, environment overlay, and canary ratio.	Engineer	{ "version": "2.1.0", "environment": "prod", "overlayId": "overlay-prod", "canaryPercent": 10 }
POST	/api/v1/probes/:id/schedules	Manage cron/event/ad-hoc schedules, priorities, and control bindings.	Compliance Officer	{ "type": "cron", "expression": "0 */6 * * *", "priority": "high", "controls": ["CTRL-1001"] }

Method	Path	Purpose	Required RBAC Role	Sample Payload
POST	/api/v1/probes/:id/run	Execute on-demand runs with contextual metadata (e.g., remediation ticket).	Compliance Officer / Admin	{ "trigger": "manual", "context": { "taskId": "TASK-123" } }
GET	/api/v1/probes/:id/metrics	Surface heartbeat, failure, latency aggregates for dashboards.	Auditor/ Compliance Officer	—

Errors return the shared { status, message, data, error } envelope, and mutation endpoints publish audit logs for immutability and forensic review.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L200-L216■■F:docs/02-technical-specifications/06-security-implementation.md†L124-L144■

Probe Lifecycle

1. **Registration:** Engineers submit metadata, evidence schemas, and supported frameworks via /api/v1/probes or the admin UI. registerProbe.workflow.ts validates schemas, capabilities, and authentication modes before Governance admins approve environments.

2. **Credential Issuance:** Upon approval, the registry issues scoped API keys or mTLS certificates per environment and transitions the probe to active.

3. **Deployment:** CI builds artifacts (e.g., probe-snowflake@2.1.0), the coordinator merges defaults with environment overrides, performs selfTest() preflight checks, and applies manifests. Health checks confirm heartbeats and ingestion, recording audit events.

4. **Scheduling:** Schedules include cron expressions, webhook-driven triggers, and ad-hoc executions (/api/v1/probes/:id/run). Priority queues consider risk tiers and regulatory deadlines.

5. **Deprecation & Rollback:** Deprecated probes retain historical evidence but cannot run new jobs. Failed rollouts revert to prior versions and alert owners.

Workflow Coordination & Integrations

- **Control Coverage:** Registry templates enforce mappings between probes, checks, and governance controls so posture analytics remain accurate in the Governance Engine and Control Management modules.■F:docs/03-systems/09-control-management-system/readme.md†L52-L66■

- **Task Automation:** Repeated failures publish probe.failure.v1 events that the Task Management system converts into remediation tasks and synchronizes with Jira/ServiceNow adapters when required.■F:docs/03-systems/13-task-management-system/readme.md†L51-L108■
- **Notification Hooks:** Deployment approvals, schedule changes, and heartbeat degradations route through Notification System channels (email, Slack) leveraging webhook retry semantics defined by the integration architecture.■F:docs/03-systems/04-notification-system/readme.md†L7-L200■■F:docs/02-technical-specifications/07-integration-architecture.md†L95-L143■
- **Evidence Pipeline:** Successful runs stream evidence payloads into the Evidence Management system, which persists metadata in PostgreSQL and objects in external MinIO for auditor consumption.■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L170■■F:docs/02-technical-specifications/07-integration-architecture.md†L68-L90■

Probe SDK & Platform Behaviors

- **Core Classes:** ProbeClient handles authentication and payload signing (submitEvidence, submitHeartbeat); ProbeScheduler registers schedules; ProbeHealthClient runs selfTest and reports diagnostics; ProbeConfigLoader merges environment overlays; ProbeVersionManager negotiates compatibility.
- **REST Endpoints:** /api/v1/probes (register/update), /api/v1/probes/:id/deployments (trigger rollout), /api/v1/probes/:id/schedules (manage schedules), /api/v1/probes/:id/run (ad-hoc), /api/v1/probes/:id/metrics (heartbeat/failure telemetry).
- **Event Contracts:** probe.heartbeat.v1, probe.evidence.v1, probe.failure.v1, and probe.deployment.v1 propagate health, payload, failure, and rollout events to downstream systems.
- **Authentication & Secrets:** Probes authenticate via signed JWTs or mTLS; credentials rotate every 30 days and are revocable. Secrets load from vault providers; payloads include SHA-256 HMAC headers for verification.
- **Retry Semantics:** SDK applies exponential backoff with jitter (2 s start, max 2 min, five attempts), idempotency keys, and circuit breakers. Scheduler retries transient failures up to three times; systemic failures escalate to alerting.
- **Versioning:** Semantic versioning governs probes and SDK runtime. ProbeVersionManager enforces minimum versions, canary rollouts shift incremental traffic, and sunset dates block submissions after deprecation.
- **Usage Pattern:** Probes import the SDK, instantiate ProbeClient with issued credentials, execute domain collectors, and submit evidence as { payload, controlMappings, checksum }, ensuring schemas align with registry definitions and Casbin policies govern data access.■F:docs/02-technical-specifications/07-integration-architecture.md†L68-L90■■F:docs/02-technical-specifications/06-security-implementation.md†L54-L95■

- **Configuration Loading:** ProbeConfigLoader reads .env values, merges environment overlays, and validates required keys using shared configuration utilities so runtime instances honor the repository-wide configuration discipline.■F:docs/02-technical-specifications/01-system-architecture.md†L72-L132■

Frontend Specification

Frontend Location & Directory Layout

Probe administration lives in client/src/features/probes, giving operators visibility into registry data, deployments, schedules, and health metrics.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L160■

```
client/src/features/probes/
■■■ pages/
■   ■■■ ProbeRegistryPage.tsx
■   ■■■ ProbeDeploymentPage.tsx
■   ■■■ ProbeSchedulePage.tsx
■   ■■■ ProbeHealthDashboard.tsx
■■■ components/
■   ■■■ ProbeDetailsPanel.tsx
■   ■■■ DeploymentTimeline.tsx
■   ■■■ ScheduleEditor.tsx
■   ■■■ HealthStatusCard.tsx
■■■ hooks/
■   ■■■ useProbeRegistry.ts
■   ■■■ useDeploymentStatus.ts
■   ■■■ useProbeMetrics.ts
■■■ api/
    ■■■ probesClient.ts

client/src/components/evidence/
■■■ ControlMappingMatrix.tsx
```

Reusable Components & UI Flows

- **Registry Management:** ProbeRegistryPage lists probes, lifecycle states, and framework bindings, linking to ProbeDetailsPanel for metadata edits and credential rotation.
- **Deployment Tracking:** DeploymentTimeline visualizes rollout phases with audit breadcrumbs and rollback controls. useDeploymentStatus polls events for live updates.
- **Scheduling UX:** ScheduleEditor supports cron, webhook, and ad-hoc triggers, validating control mappings via ControlMappingMatrix before persisting.
- **Health Monitoring:** ProbeHealthDashboard combines HealthStatusCard widgets (heartbeats, failure counts, latency) and surfaces alert acknowledgements with direct links to incident runbooks.

- **State & Data Access:** Hooks rely on Axios clients that attach JWT headers, map responses into shared stores (Redux/Zustand), and memoize selectors per the frontend architecture guidance for feature slices.■F:docs/02-technical-specifications/03-frontend-architecture.md†L50-L160■
- **Cross-System Surfacing:** UI embeds related control status, evidence previews, and remediation tasks so operators triage without context switching across modules.■F:docs/03-systems/09-control-management-system/readme.md†L52-L107■■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L170■■F:docs/03-systems/13-task-management-system/readme.md†L51-L102■

Schema Specification

- **probes:** Registry metadata (id, owner, framework bindings, evidence schema, version, lifecycle state, environment overlays).
- **probe_deployments:** Deployment records with environment, artifact version, rollout status, timestamps, and audit references.
- **probe_schedules:** Cron/event definitions, priority, control mappings, and last execution metadata.
- **probe_metrics:** Aggregated heartbeat intervals, failure counts, latency stats, and error codes.
- **probe_credentials:** Issued secrets (JWT, mTLS) with rotation history and revocation status.
- **probe_events:** Append-only stream capturing heartbeat, evidence, deployment, and failure events for replay/testing.
- Field definitions align with the broader Probes & Checks schema, enforcing referential integrity with checks, controls, and evidence tables managed via Prisma on externally hosted PostgreSQL.■F:docs/02-technical-specifications/04-database-design.md†L82-L115■■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L169-L190■
- Relationships connect to Governance controls, Evidence ingestion pipelines, Notification alerts, and Task Service remediation tickets.■F:docs/03-systems/09-control-management-system/readme.md†L52-L66■■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L170■■F:docs/03-systems/04-notification-system/readme.md†L7-L200■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■

Operational Playbooks & References

Monitoring & Alerting

- Heartbeat SLA: default five-minute interval; missing two beats escalates to warning, three triggers incident. Evidence ingestion errors emit structured codes consumed by SIEM dashboards.

- Alert tiers: Probe owners (Slack/Email), Platform SRE (PagerDuty escalation after 30 minutes), Governance Leads (regulatory-impact outages >4 hours). Alerts integrate with ServiceNow and Task Service for follow-up.
- Continuous improvement: Post-incident reviews capture root cause, configuration drift, and mitigation tasks; registry templates update with new validation rules.

Deployment & Runbook Checklist

- **Pre-Deployment:** Confirm probe-specific environment variables and secrets through CI pipelines, run Prisma migrations, and validate container images in staging per the platform deployment checkli st.■F:docs/02-technical-specifications/08-deployment-and-environment-guide.md†L151-L188■
- **Rollout Verification:** Execute SDK selfTest() routines, monitor /api/v1/probes/:id/metrics, and verify webhook/event delivery with retry telemetry enabled in observability tooling.■F:docs/02-technical-specifications/07-integration-architecture.md†L123-L143■
- **Failure Recovery:** Publish probe.deployment.v1 with revert status, disable schedules, notify stakeholders through Notification adapters, and open remediation tasks when thresholds are exceed ed.■F:docs/03-systems/04-notification-system/readme.md†L7-L200■■F:docs/03-systems/13-task-management-system/readme.md†L51-L109■

Related Documentation

- — tenant onboarding hooks and delegated administration.
- — evidence ingestion mapping and storage contracts.
- — framework/control associations.
- — alert delivery for probe failures.

Story: Probe Backend Layout

Summary

Detail the modules coordinating probe definitions, scheduling, execution, and reporting.

As a…

As a backend developer

I want to…

I want to locate probe code

So that…

So that I can enhance probe orchestration

Acceptance Criteria

- [ ] Layout includes controllers, services, schedulers, SDK integrations, and results repositories.
- [ ] Configuration notes explain environment setup and dependency management for probe runners.

Story: Probe Configuration

Summary

Summarize environment variables, secrets, and infrastructure requirements for probe services.

As a…

As a DevOps engineer

I want to…

I want probe configuration clarity

So that…

So that probes run reliably across environments

Acceptance Criteria

- [ ] Configuration lists credentials, timeouts, concurrency, and queue settings managed via secure storage.
- [ ] Guidance references infrastructure automation assets provisioning supporting services.

Story: Probe System Components

Summary

Explain collectors, schedulers, result processors, and notification hooks that make up the probe system.

As a…

As a systems architect

I want to…

I want a component overview

So that…

So that I can understand interactions between probe modules

Acceptance Criteria

- [ ] Components include registry services, execution workers, result aggregators, and alert dispatchers.
- [ ] Integration points tie into notification, governance, and evidence systems for downstream actions.

Story: Probe API Contracts

Summary

Document REST endpoints for registering probes, managing runs, and retrieving telemetry.

As a…

As an integration engineer

I want to…

I want to call probe APIs

So that…

So that I can onboard new probes and view results

Acceptance Criteria

- [ ] Endpoints cover CRUD for probes, scheduling requests, execution status, and result retrieval.
- [ ] Contracts reference authentication, RBAC scopes, and pagination or filtering semantics.

Story: Probe Lifecycle

Summary

Describe the stages from registration, scheduling, execution, verification, to archival.

As a…

As a product manager

I want to…

I want probe lifecycle visibility

So that…

So that probes deliver reliable monitoring

Acceptance Criteria

- [ ] Lifecycle steps track approval gates, credential injection, execution, and remediation task creation.
- [ ] Archival processes retain historical runs, attach evidence, and surface anomalies to governance workflows.

Story: Probe Workflow Integrations

Summary

Summarize coordination with notifications, governance engine, tasks, and evidence systems.

As a…

As an integrations lead

I want to…

I want to know probe dependencies

So that…

So that cross-system workflows stay aligned

Acceptance Criteria

- [ ] Integrations publish events for failures, escalate tasks, and attach evidence snapshots to governance scoring.
- [ ] Inbound hooks validate partner probes, enforce authentication, and reconcile status updates.

Story: Probe SDK Behaviors

Summary

Explain SDK expectations, runtime contracts, and security posture for probe authors.

As a…

As a probe developer

I want to…

I want to understand SDK capabilities

So that…

So that I can build compliant probes

Acceptance Criteria

- [ ] SDK guidance covers configuration loading, telemetry submission, authentication, and error handling.
- [ ] Security notes enforce sandboxing, rate limits, and required metadata for governance analytics.

Story: Probe Frontend Structure

Summary

Identify UI directories for probe catalogues, scheduling consoles, and run history views.

As a…

As a frontend engineer

I want to…

I want to navigate probe UI code

So that…

So that I can improve operator tooling

Acceptance Criteria

- [ ] Structure references client/src/features/probes pages, components, and state management hooks.
- [ ] Shared design system elements handle probe status indicators, run detail panels, and filters.

Story: Probe UI Flows

Summary

Describe list, detail, scheduling, and troubleshooting experiences within the console.

As a…

As an operations analyst

I want to…

I want intuitive probe dashboards

So that…

So that I can monitor coverage and triage issues

Acceptance Criteria

- [ ] UI flows include catalogue filters, run detail timelines, and bulk scheduling tools.
- [ ] State synchronization ensures status badges, notification acknowledgements, and evidence links stay updated.

Story: Probe Monitoring & Alerting

Summary

Summarize metrics, alerts, dashboards, and incident response playbooks for probe execution.

As a…

As an on-call engineer

I want to…

I want probe observability

So that…

So that I can detect and resolve probe failures

Acceptance Criteria

- [ ] Monitoring tracks execution success rates, queue depth, runtime latency, and SDK version drift.
- [ ] Runbooks cover retry strategies, credential rotation, and communication with partner teams.

Story: Probe Deployment Checklist

Summary

Provide deployment, rollout, and verification steps for releasing probe management updates.

As a…

As a release manager

I want to…

I want a probe deployment plan

So that…

So that new features roll out safely

Acceptance Criteria

- [ ] Checklist includes environment promotion, worker scaling, feature flag toggles, and SDK compatibility checks.
- [ ] Post-deploy validation ensures telemetry ingestion, notification triggers, and dashboards operate as expected.

Check Management System

Location: /server/src/modules/governance

TL;DR

The check management system operationalizes governance requirements inside the Governance Engine.

It coordinates check definitions, execution workflows, evidence capture, and publication workflows across automated and manual paths.

Use this runbook to understand lifecycle states, execution patterns, and operational playbooks for managing compliance checks.

Backend Specification

Backend Location & Directory Layout

Runtime services that orchestrate checks live alongside the Governance Engine in server/src/modules/governance; the canonical catalogue of check definitions persists in PostgreSQL (checks, results). There is no standalone checks subdirectory—the Governance Engine modules coordinate registrations, scheduling, evidence links, and publishing. All runtime code is authored in JavaScript, matching the Express.js conventions defined in the technical architecture overview.■F:docs/02-technical-specifications/01-system-architecture.md†L18-L118■■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L124-L171■

```
server/src/modules/governance/
■■■ checks/
■   ■■■ checks.service.js
■   ■■■ execution.service.js
■   ■■■ lifecycle.service.js
■■■ controllers/
■   ■■■ checks.controller.js
■   ■■■ results.controller.js
■   ■■■ review-queue.controller.js
■■■ schedulers/
■   ■■■ governance.scheduler.js
■■■ mappers/
■   ■■■ control-mapping.service.js
■■■ events/
■   ■■■ check.failed.js
■   ■■■ check.published.js
■■■ ui/
    ■■■ admin-console/
```

Core Modules & Responsibilities

- **checks.service.js** – CRUD endpoints for definitions, validation of control/probe bindings, serialization of metadata, and enforcement of lifecycle guards (draft → ready → active → retired).■F:docs/01-about/03-concept-summary.md†L214-L318■
- **execution.service.js** – Orchestrates automated, manual, and hybrid executions, invokes probe adapters, hydrates evidence references, and persists raw outputs for downstream analytics.■F:docs/03-systems/07-probe-management-system/readme.md†L7-L172■
- **lifecycle.service.js** – Applies governance approvals, manages version bumps, archives retired checks, and emits audit events consumed by the Governance Engine pipeline.■F:docs/03-systems/12-governance-engine/readme.md†L7-L120■
- **Controllers** – Express routers mounted under /api/v1/governance/checks that expose definition, result, and review-queue APIs with shared middleware for JWT authentication, Casbin authorization, and request validation.■F:docs/02-technical-specifications/06-security-implementation.md†L35-L146■
- **governance.scheduler.js** – Configures cron/event triggers, seeds BullMQ queues, and coordinates retries and backoff policies aligned with Probe Management cadence definitions.■F:docs/03-systems/07-probe-management-system/readme.md†L32-L166■
- **Event Emitters** – check.failed.js and check.published.js push structured payloads onto the platform event bus for Notification, Task, and Dashboard systems to consume.■F:docs/03-systems/04-notification-system/readme.md†L7-L192■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L21-L108■

System Components & Check Types

- **Conceptual Overview:** Checks convert governance requirements into machine- and human-executable validations, coordinate probe integrations, persist outcomes, and preserve audit history tied to controls and frameworks. Every check maps back to the continuous governance flow described in the concept summary and produces data used by controls, frameworks, and reporting layers.■F:docs/01-about/03-concept-summary.md†L214-L359■
- **Core Services:**
- **Definition Service:** Normalizes metadata, enforces taxonomy alignment, binds controls/probes, and generates OpenAPI documentation for the exposed endpoints.
- **Execution Engine:** Resolves run context (trigger, schedule, manual), composes probe payloads, evaluates results against thresholds or rule scripts, and records diagnostics for observability.
- **Lifecycle Manager:** Applies approval workflows, manages version transitions, and coordinates publishing with the Governance Engine to maintain auditability across iterations.
- **Check Types:**
- **Automated:** Programmatic validations executed by probes or integrations against APIs, logs, configuration stores, or model metadata. Automated checks must declare probe_contract schemas to ensure payload compatibility.
- **Manual:** Human attestation tasks completed by compliance officers when automation is unavailable or requires interpretation. Manual runs collect evidence uploads via the Evidence Management system and enforce reviewer assignments.
- **Hybrid:** Automated collection paired with reviewer sign-off for nuanced findings (e.g., bias tests needing human confirmation). Hybrid flows attach automated payloads and require dual-approval for publication in high-risk domains.
- **Distinctions:** Automated checks trigger via schedules or events; manual/hybrid checks enter review queues. Automated outputs are structured JSON; manual/hybrid store attachments and free-form observations. Manual/hybrid require reviewer approval before publication, and all variants emit audit logs for downstream monitoring.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■

API Surface & Integration Contracts

- **Definitions:**
- GET /api/v1/governance/checks – Paginated catalogue with filtering by lifecycle state, control, probe, framework, and risk tier.
- POST /api/v1/governance/checks – Create draft definitions with schema validation, Casbin policy checks, and automatic audit capture.
- PUT /api/v1/governance/checks/:id – Update metadata, bind probes, and enqueue governance approvals when lifecycle transitions occur.

- POST /api/v1/governance/checks/:id/activate – Transition ready_for_validation definitions to active once approvals complete.
- **Execution & Results:**
- POST /api/v1/governance/checks/:id/run – Trigger ad-hoc executions, immediately enqueueing BullMQ jobs with tenant-specific priorities.
- GET /api/v1/governance/checks/:id/results – Fetch historical outcomes with filtering by severity, publication state, or time window.
- POST /api/v1/governance/results/:resultId/publish – Validate reviewer sign-off and promote pending results into published dashboards.
- **Review Queue:**
- GET /api/v1/governance/review-queue – List manual/hybrid tasks with SLA metadata, role-gated by Casbin policies.
- POST /api/v1/governance/review-queue/:itemId/complete – Capture reviewer decision, attach evidence references, and route to publishing workflow.
- **Cross-System Hooks:** REST responses include hypermedia links to Control, Evidence, Notification, and Task endpoints so that client applications can jump between modules without additional lookups .■F:docs/03-systems/09-control-management-system/readme.md†L7-L159■■F:docs/03-systems/11-evidence-management-system/readme.md†L47-L115■■F:docs/03-systems/04-notification-system/readme.md†L7-L192■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■

Execution Workflows

- **Automated Checks:** Scheduler selects eligible checks (frequency, next_run_at), triggers probes with context, evaluates responses against thresholds/patterns, records outcomes in results, and dispatches remediation tasks/alerts for failures or warnings. Execution jobs run inside BullMQ workers to provide retry semantics, concurrency control, and telemetry hooks published to the monitoring stack.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L48-L155■
- **Manual Checks:** Controls flagged for manual validation generate queue items. Reviewers upload evidence (stored in the Evidence Repository), assess status/severity, submit results into pending_validation, then publish once validated. Manual flows require reviewer acknowledgements captured in audit trails and escalate overdue items through Task Management.
- **Hybrid Checks:** Automated pass produces requires_review outcomes; reviewers receive probe output with guidance, adjust severity/status, and resubmit for publication. Hybrid state transitions preserve automated payloads in MinIO via evidence links so human reviewers can inspect raw telemetry.■F:docs/03-systems/11-evidence-management-system/readme.md†L47-L115■
- **Mapping to Probes & Controls:** Each check binds to a single control and optionally a probe. Control coverage metrics aggregate active checks; evidence links tie to the Evidence Repository, and status/severity propagate to control-level risk scores. Control scoring changes trigger downstream framework updates and dashboard refreshes for real-time posture visibility.■F:docs/03-

systems/09-control-management-system/readme.md†L7-L159■■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L21-L108■

- **Notification & Remediation:** Failed or warning results emit events that trigger Notification templates (email, Slack) and create remediation tasks with due dates derived from control enforcement levels. Published results feed Governance Engine aggregation pipelines for continuous monitoring.■F:docs/03-systems/04-notification-system/readme.md†L7-L192■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■■F:docs/03-systems/12-governance-engine/readme.md†L25-L120■

Lifecycle Governance

- **Definition Stages:** draft → ready_for_validation → active → retired, ensuring review before scheduling. Governance reviewers validate metadata, probe contracts, severity rationale, and evidence retention.
- **Result States:** pending_validation, requires_review, pass, fail, warning, culminating in published once approved. Only published results feed dashboards and reports.
- **Versioning:** Increment checks.version for logic/severity changes. Migration scripts seed new versions while historical results remain readable. Deprecation occurs after replacement checks publish successful results, maintaining coverage overlap.
- **Access Control:** JWT-authenticated requests pass through Casbin policy checks ensuring only authorized roles can mutate definitions or approve results. Sensitive evidence links inherit signed URL expirations aligned with security implementation standards.■F:docs/02-technical-specifications/06-security-implementation.md†L35-L146■
- **Auditability:** Every lifecycle transition writes to the Audit Logging system, including reviewer identity, timestamps, and payload hashes for integrity verification.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■

Frontend Specification

Frontend Location & Directory Layout

Governance UI experiences surface under client/src/features/governance/checks, enabling check authors, reviewers, and auditors to manage definitions, reviews, and outcomes. Components follow the JavaScript-only, feature-first conventions defined in the frontend architecture guide and integrate directly with Axios API clients and shared contexts for authentication and theming.■F:docs/02-technical-specifications/03-frontend-architecture.md†L1-L160■

```
client/src/features/governance/checks/
■■■ pages/
■   ■■■ CheckCatalogPage.tsx
■   ■■■ CheckDesignerPage.tsx
■   ■■■ ReviewQueuePage.tsx
■   ■■■ ResultExplorerPage.tsx
■■■ components/
■   ■■■ CheckDefinitionForm.tsx
■   ■■■ CheckLifecycleTimeline.tsx
■   ■■■ ReviewTaskDrawer.tsx
■   ■■■ EvidenceLinkList.tsx
■■■ hooks/
■   ■■■ useCheckDefinitions.ts
■   ■■■ useReviewQueue.ts
■   ■■■ useCheckResults.ts
■■■ api/
    ■■■ checksClient.ts

client/src/components/governance/
■■■ ControlCoverageChart.tsx
```

Reusable Components & UI Flows

- **Catalog & Designer:** CheckCatalogPage lists definitions with lifecycle/status filters; CheckDefinitionForm manages creation/edits, guiding authors through control mapping, probe selection, severity defaults, and frequency settings. Form state uses React Hook Form-style custom hooks with schema validation matching backend expectations, while contextual help surfaces governance guidance sourced from shared content files.■F:docs/02-technical-specifications/03-frontend-architecture.md†L62-L139■
- **Review Queue:** ReviewQueuePage surfaces manual/hybrid work items with SLA indicators; ReviewTaskDrawer presents probe output, evidence links, and approval workflow (two-person rule when required). Components leverage Notification context to show escalations and Task Service integrations for remediation shortcuts.■F:docs/03-systems/13-task-management-system/readme.md†L159-L226■
- **Result Exploration:** ResultExplorerPage visualizes published outcomes, severity trends, and drill-down to evidence via EvidenceLinkList. ControlCoverageChart highlights framework coverage gaps and links to Control Management for comparative scoring analysis.■F:docs/03-systems/09-control-management-system/readme.md†L69-L159■
- **Shared Hooks:** useCheckDefinitions, useReviewQueue, and useCheckResults wrap Axios clients with caching, optimistic updates for lifecycle transitions, and toast notifications tied to Notification templates, following the API integration patterns defined in the frontend architecture spec.■F:docs/02-technical-specifications/03-frontend-architecture.md†L140-L191■
- **Accessibility & Localization:** Components honor the platform’s accessibility and localization guidelines, including keyboard navigation, ARIA labelling, and locale-aware formatting, ensuring compliance with corporate governance requirements.■F:docs/02-technical-specifications/03-frontend-architecture.md†L166-L231■

Schema Specification

- **checks:** Definition metadata (id, control*id, probe*id, type, name, description, severity*default, status, version, frequency, created*by/updated_by, metadata JSON). Prisma enforces enums for type, status, and severity_default, while partial indexes accelerate queries on status = 'active' and control_id for catalogue filtering.■F:docs/02-technical-specifications/04-database-design.md†L33-L128■
- **check_versions:** Historical table capturing logic snapshots, migration notes, rollback references, and diff metadata. Version rows include JSON patches to support audit reconstruction and regression testing of deprecated logic.
- **results:** Execution records (id, check*id, control*id, probe*run*id, status, severity, evidence*link*id, notes, executed*at, validated*at, published*at, created*by, raw_output) with indexes for remediation and reporting. Raw payloads persist as JSONB to support search across warning/failure diagnostics.
- **review_queue_items:** Manual/hybrid tasks (check*id, assigned*to, due_at, priority, state, SLA metadata) syncing with Task Management for accountability. Queue records embed escalation_level, evidence_bundle_id, and acknowledged_at timestamps so escalations align with remediation policies.■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■
- **check_control_links:** Join table aligning check coverage to controls with weight, enforcement_level, and evidence_requirements fields powering scoring and remediation logic.■F:docs/03-systems/09-control-management-system/readme.md†L69-L159■
- **check_notifications:** Tracks outbound alert templates, recipients, delivery status, and correlation IDs for audit reconciliation with the Notification system.■F:docs/03-systems/04-notification-system/readme.md†L7-L192■
- Relationships connect to probes, controls, evidence links, notifications, dashboards, and audit logs to ensure consistent governance data. Prisma migrations manage schema evolution, and retention policies align with the database design guidelines for backups, archiving, and encryption.■F:docs/02-technical-specifications/04-database-design.md†L1-L154■

Operational Playbooks & References

Core Playbooks

- **Adding a Check:** Author definition (UI or YAML), attach probe if applicable, submit for validation with evidence samples, Governance review in staging, activate on approval. Deployment checklist includes updating OpenAPI docs, running Prisma migrations, and verifying scheduler entries in non-production before promoting to production.■F:docs/02-technical-specifications/08-deployment-and-environment-guide.md†L42-L168■
- **Manual Review Handling:** Queue intake assigns reviewers, structured forms capture attestations and evidence uploads, senior reviewers validate and publish results, with escalation for overdue

items. SLA breaches auto-create remediation tasks and dispatch notifications following incident management procedures.■F:docs/01-about/08-operations-and-teams.md†L112-L198■■F:docs/03-systems/13-task-management-system/readme.md†L159-L226■

- **Monitoring:** Dashboards track failure/warning volumes, SLA breaches, and coverage; repeated false positives trigger logic review and severity tuning. SRE teams monitor BullMQ queue depth, worker health, and scheduler drift through the DevOps observability stack.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L48-L155■
- **Change Management:** Lifecycle changes (activate, retire, revise) require change tickets, impact analysis on control coverage, and notification to stakeholders. All changes recorded in the Audit Logging system and distributed via governance communications to maintain transparency.■F:docs/01-about/04-security-and-data-protection.md†L140-L214■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■

Related Documentation

- — automated evidence integrations and scheduling.
- — framework/catalog governance.
- — evidence storage and linkage.
- — alerting for failed checks.

Story: Check Backend Layout

Summary

Document module directories for check definitions, execution engines, and result storage.

As a…

As a backend engineer

I want to…

I want to locate check management code

So that…

So that I can enhance check orchestration

Acceptance Criteria

- [ ] Layout covers controllers, services, repositories, and scheduler integrations for checks.
- [ ] Notes explain how check modules integrate with governance, tasks, and notifications.

Story: Check Core Responsibilities

Summary

Summarize module roles for cataloging checks, templating, execution planning, and compliance scoring.

As a…

As a systems architect

I want to…

I want insight into check responsibilities

So that…

So that governance requirements translate into automated checks

Acceptance Criteria

- [ ] Responsibilities include building catalogs, templates, and pipelines for automated and manual checks.
- [ ] Execution planning ties into evidence ingestion, risk scoring, and escalation workflows.

Story: Check Types & Components

Summary

Detail check categories, engines, and supporting components used to evaluate controls.

As a…

As a product strategist

I want to…

I want clarity on check types

So that…

So that offerings cover procedural, technical, and questionnaire assessments

Acceptance Criteria

- [ ] Components describe automated scans, attestations, questionnaires, and custom plug-ins with RBAC enforcement.
- [ ] Each type links to storage, evaluation metrics, and downstream reporting requirements.

Story: Check API Contracts

Summary

Capture endpoints for managing checks, templates, runs, and evidence attachments.

As a…

As an API consumer

I want to…

I want to automate check workflows

So that…

So that I can manage compliance programs programmatically

Acceptance Criteria

- [ ] Endpoints cover CRUD operations, scheduling, run status, and evidence linking with error semantics.
- [ ] Contracts reference RBAC scopes, pagination, and webhook callbacks for status updates.

Story: Check Execution Workflows

Summary

Describe orchestration steps from scheduling, execution, approval, to remediation handoff.

As a…

As an operations lead

I want to…

I want end-to-end check visibility

So that…

So that findings trigger timely remediation

Acceptance Criteria

- [ ] Workflows include task creation, notification triggers, and governance engine scoring updates.
- [ ] Retry and escalation policies handle failed checks, overdue reviews, and exception approvals.

Story: Check Lifecycle Governance

Summary

Explain how checks are versioned, reviewed, retired, and audited.

As a…

As a compliance manager

I want to…

I want to govern check lifecycles

So that…

So that evidence remains trustworthy

Acceptance Criteria

- [ ] Policies require version reviews, approval workflows, and immutable logging of changes.
- [ ] Retirement processes archive history, notify stakeholders, and link to replacement controls.

Story: Check Frontend Structure

Summary

Map UI directories for check catalogs, run dashboards, and review panels.

As a…

As a frontend developer

I want to…

I want to find check UI code

So that…

So that I can evolve review experiences

Acceptance Criteria

- [ ] Structure references client/src/features/checks pages, components, and hooks.
- [ ] Shared state patterns explain how filters, timelines, and evidence panels stay synchronized.

Story: Check UI Flows

Summary

Summarize operator workflows for configuring checks, reviewing findings, and approving remediations.

As a…

As a UX designer

I want to…

I want consistent check experiences

So that…

So that teams act on findings efficiently

Acceptance Criteria

- [ ] UI flows cover creation wizards, finding detail modals, and bulk approval tools.
- [ ] Integrations connect to task assignments, notifications, and evidence previews within the console.

Story: Check Management Playbooks

Summary

Provide operational runbooks for alerting, exception handling, and integrations with other governance systems.

As a…

As a support lead

I want to…

I want check management playbooks

So that…

So that I can respond to issues systematically

Acceptance Criteria

- [ ] Playbooks describe alert thresholds, SLA escalations, and communication channels for incidents.
- [ ] Guidance maps dependencies to probes, governance engine, and reporting outputs.

Control Management System

Location: /server/src/modules/governance/controls

TL;DR

The control management system anchors governance, risk, and compliance posture across the platform.

It maintains a canonical control catalog, maps controls to frameworks and checks, and orchestrates scoring, remediation, and reporting flows.

Use this reference to understand data models, lifecycle procedures, and integrations that keep control health accurate and auditable.

Domain Context & Alignment

- **Governance Loop Positioning:** Control management sits between check execution and framework reporting in the closed-loop governance cycle, turning validated check outputs into authoritative control scores and narratives for auditors.■F:docs/01-about/03-concept-summary.md†L203-L301■
- **Cross-Functional Ownership:** Compliance analysts curate taxonomy, engineering maintains APIs and integrations, and product/operations ensure roadmap delivery; the squad structure mirrors the broader organizational pillars (Engineering, Product & Design, Security & Compliance, Customer Success).■F:docs/01-about/08-operations-and-teams.md†L24-L84■
- **Regulatory Coverage:** Controls must map to EU AI Act, ISO/IEC 42001, NIST AI RMF, and sector frameworks surfaced elsewhere in governance modules, ensuring posture roll-ups remain audit-ready and multi-jurisdictional.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L110-L118■■F:docs/03-systems/10-framework-mapping-system/readme.md†L7-L107■
- **Product Outcomes:** Deliver reliable control scorecards, remediation triggers, and framework exports that feed reporting dashboards and customer attestations — key outputs promised in platform narratives and go-to-market collateral.■F:docs/01-about/03-concept-summary.md†L269-L320■■F:docs/01-about/08-operations-and-teams.md†L108-L121■

Non-Functional & Compliance Requirements

- **Security Posture:** Enforce security-by-design, RBAC, and encryption expectations at every layer. Controllers must validate JWTs/Casbin policies; repositories protect tenant boundaries and leverage immutable audit logging per platform-wide security doctrine.■F:docs/01-about/04-security-and-data-protection.md†L3-L107■■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L194-L217■
- **Data Protection:** All control metadata, evidence pointers, and score histories carry classification, retention, and residency constraints; deletion workflows must honor configurable retention windows and region-specific storage policies.■F:docs/01-about/04-security-and-data-protection.md†L118-L179■
- **Operational Governance:** Release cadence follows agile delivery with sign-offs from Product, QA, and Security; production deployments require CTO/CISO approval and integrate into enterprise OKR tracking for compliance outcomes.■F:docs/01-about/08-operations-and-teams.md†L29-L144■
- **Performance & Reliability:** APIs must sustain multi-tenant loads, leverage caching on read-heavy scorecards, and expose health metrics to centralized monitoring, aligning with infrastructure SLAs (>90% uptime) and DevOps orchestration baselines.■F:docs/01-about/08-operations-and-teams.md†L34-L39■■F:docs/02-technical-specifications/05-devops-infrastructure.md†L52-L119■

Backend Specification

Backend Location & Directory Layout

Control governance logic resides in server/src/modules/governance/controls, which exposes REST APIs, scoring engines, and lifecycle workflows for the Governance Engine.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L124-L171■

```
server/src/modules/governance/controls/
■■■ controllers/
■   ■■■ controls.controller.ts
■   ■■■ scoring.controller.ts
■   ■■■ mappings.controller.ts
■■■ services/
■   ■■■ control.service.ts
■   ■■■ scoring.service.ts
■   ■■■ mapping.service.ts
■   ■■■ lifecycle.service.ts
■■■ repositories/
■   ■■■ control.repository.ts
■   ■■■ control-framework.repository.ts
■   ■■■ control-check.repository.ts
■■■ workflows/
■   ■■■ createControl.workflow.ts
■   ■■■ updateControl.workflow.ts
■■■ events/
    ■■■ control.failed.ts
    ■■■ control.updated.ts
```


Control Taxonomy & Relationships

- **Taxonomy Structure:** Domains → Categories → Controls (with optional sub-controls). Every control maps to at least one framework requirement and one verification check, ensuring traceability from enterprise objectives to validations.
- **Control Metadata:** controls table stores identifiers, taxonomy keys, title/description, rationale, implementation guidance, owner team, status (draft, active, deprecated), risk tier, version, audit timestamps, and authorship.
- **Framework Links:** control_framework_links join table associates controls with frameworks/requirements, including coverage level, evidence references, and validity windows.
- **Check Links:** control_check_links tracks which automated/manual checks substantiate a control, including assertion type, frequency, weight/enforcement level, and cadence expectations. Relationships power coverage metrics and remediation triggers.■F:docs/03-systems/08-check-management-system/readme.md†L7-L167■

Service Responsibilities & Integration Points

- **Controllers:** controls.controller.ts, scoring.controller.ts, and mappings.controller.ts expose REST endpoints that honour the shared response envelope (status, message, data, error) and enforce JWT + Casbin middleware prior to invoking services.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L194-L217■
- **Services:** control.service.ts orchestrates CRUD, deduplication, and taxonomy invariants; mapping.service.ts manages framework/check associations and emits control.mapping.updated; scoring.service.ts recalculates aggregates when check results land; lifecycle.service.ts coordinates approvals with Governance Engine schedulers and framework versioning workflows.■F:docs/03-sys

tems/12-governance-engine/readme.md†L15-L101■■F:docs/03-systems/10-framework-mapping-system/readme.md†L40-L110■

- **Repositories & Policies:** Repository classes encapsulate Prisma queries with tenant scoping, eager loading for mappings, and optimistic locking for concurrent edits. Casbin policies align with RBAC duties for compliance analysts, auditors, and administrators.■F:docs/01-about/04-security-and-data-protection.md†L192-L200■
- **Workflow & Event Handlers:** createControl.workflow.ts and updateControl.workflow.ts coordinate approvals, queue scoring recalculations, and notify downstream systems (Tasks, Notifications, Dashboards) via domain events (control.created, control.updated, control.failed).■F:docs/03-systems/04-notification-system/readme.md†L1-L126■■F:docs/03-systems/13-task-management-system/readme.md†L1-L115■■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■
- **External Systems:** Evidence repository lookups validate attached artefacts, while Framework Mapping ensures cross-standard parity before publish; every mutation records audit events for transparency and compliance reporting.■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L170■■F:docs/03-systems/10-framework-mapping-system/readme.md†L40-L170■

API Surface & DTO Contracts

Endpoint	*Method*	*Purpose*	*Notes*
/api/v1/controls	GET	List controls with taxonomy, framework, and scoring filters.	Supports pagination, risk tier filters, ownership scoping, and ETag caching for UI grids.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L194-L207■
/api/v1/controls	POST	Create draft or active control records.	Payload validated via Zod DTOs requiring taxonomy path, rationale, enforcement weight, and at least one mapping stub; emits control.created for governance subscribers.■F:docs/03-systems/12-governance-engine/readme.md†L33-L88■

Endpoint	Method	Purpose	Notes
/api/v1/controls/:id	GET	Fetch control details, mappings, scores, evidence references, and audit cursors.	Accepts includeDraftMappings for reviewers and returns aggregated metrics.
/api/v1/controls/:id	PATCH	Update metadata, weights, lifecycle status, or mappings.	Version bump triggers when risk tier or enforcement changes; approvals enforced before publish.
/api/v1/controls/:id/archive	POST	Soft-archive controls with deprecation rationale.	Schedules remap jobs and notifies check owners to reassess coverage.
/api/v1/controls/:id/mappings	PUT	Replace framework/check mapping matrix.	Validates referential integrity via Framework Mapping service before committing transaction.■F:docs/03-systems/10-framework-mapping-system/readme.md†L57-L170■

Endpoint	Method	Purpose	Notes
/api/v1/controls/:id/scores	GET	Retrieve historical score trends and contributing checks.	Offers granularity (daily/weekly/monthly) and cursor pagination for dashboards.
/api/v1/controls/:id/remediation	POST	Force remediation cycle (open task, notify owners).	Bridges to Task & Notification services; returns task IDs and notification receipts for traceability.■F:docs/03-systems/13-task-management-system/readme.md†L51-L115■■F:docs/03-systems/04-notification-system/readme.md†L56-L126■

DTO Conventions

- Request DTOs use camelCase keys, enumerations for status (draft, active, deprecated), enforcement (advisory, mandatory), and risk (low, medium, high). Response objects wrap payloads inside { status, message, data } with consistent error contracts (code, details).■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L194-L207■
- Mapping payloads include nested arrays for frameworkMappings (frameworkId, requirementId, coverageLevel, effectiveFrom/To) and checkLinks (checkId, weight, enforcementLevel, frequencyCadence). Validation ensures at least one active mapping before activation.
- Audit trails expose changeId, actor, before, after, and comment fields in line with immutable logging and transparency requirements.■F:docs/01-about/04-security-and-data-protection.md†L93-L147■

Scoring & Lifecycle Management

- **Scoring Model:** Check results produce normalized scores (pass=1, warning=0.5, fail=0). Weighted averages (per link weight) roll up to control scores, scaled by risk tier multipliers (High=1.5, Medium=1.0, Low=0.75). Thresholds: ≥0.85 Passing, 0.60–0.84 Needs Attention, <0.60 Failing.
- **Failure Handling:** Failing controls trigger incidents/remediation tasks when enforcement levels demand it, persist evidence, and notify owner teams plus subscribed framework stakeholders.
- **Lifecycle Procedures:**

- **Create Control:** Draft change request, deduplicate against catalog, governance board approval, insert into controls with active status, establish required framework/check mappings, and schedule initial checks.
- **Update Control:** Versioned change request, increment version, adjust mappings and weights, notify dependent teams, and log updates via audit events.
- **Cross-Framework Mapping:** Maintain coverage matrix, document compensating controls, update control_framework_links, and refresh posture dashboards.
- **Audit Trail:** All mutations emit audit events, quarterly audits sample controls for accuracy/evidence freshness, discrepancies create remediation tasks tracked in Task Service.■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■
- **Score Publication:** Aggregated results publish to Governance Engine subscribers, update framework scorecards, and feed dashboard materialization jobs; failures raise alerts through Notification and Task systems to maintain continuous compliance posture.■F:docs/03-systems/12-governance-engine/readme.md†L15-L160■■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L27-L118■■F:docs/03-systems/04-notification-system/readme.md†L56-L126■

Frontend Specification

Frontend Location & Directory Layout

Control management UI lives at client/src/features/governance/controls, enabling catalog maintenance, mappings, and posture analytics.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L160■

```
client/src/features/governance/controls/
■■■ pages/
■   ■■■ ControlCatalogPage.tsx
■   ■■■ ControlDetailPage.tsx
■   ■■■ FrameworkMappingPage.tsx
■   ■■■ ControlScoreboardPage.tsx
■■■ components/
■   ■■■ ControlForm.tsx
■   ■■■ MappingMatrix.tsx
■   ■■■ ScoreTrendChart.tsx
■   ■■■ RemediationTaskList.tsx
■■■ hooks/
■   ■■■ useControls.ts
■   ■■■ useControlMappings.ts
■   ■■■ useControlScores.ts
■■■ api/
    ■■■ controlsClient.ts

client/src/components/governance/
■■■ FrameworkCoverageHeatmap.tsx
```

Reusable Components & UI Flows

- **Catalog Maintenance:** ControlCatalogPage lists controls with taxonomy filters; ControlForm supports creation/version updates with change-ticket capture and governance approval routing.
- **Mapping Management:** FrameworkMappingPage leverages MappingMatrix to edit framework links, set coverage levels, attach evidence references, and configure validity dates.
- **Scoring Analytics:** ControlScoreboardPage and ScoreTrendChart visualize posture trends, risk tiers, and historical performance. RemediationTaskList integrates Task Service data to track open issues.
- **Dashboard Insights:** FrameworkCoverageHeatmap highlights gaps across frameworks, linking to failing controls and associated checks.

State, Security & Localization Patterns

- **State Management:** Feature hooks (useControls, useControlMappings, useControlScores) wrap Axios clients with React Context-aware caching, mirroring the platform pattern of Context + custom hooks for feature-level state.■F:docs/02-technical-specifications/03-frontend-architecture.md†L91-L140■
- **Design System:** Components compose TailwindCSS and shadcn/ui atoms/molecules with lucide icons and semantic compliance colors to meet accessibility targets and maintain brand cohesion.■F:docs/02-technical-specifications/03-frontend-architecture.md†L72-L159■
- **Security Guards:** API clients attach JWT headers, enforce CSRF tokens, and propagate authorization failures through Notification Context for consistent messaging; protected routes leverage RBAC-aware wrappers to hide privileged actions.■F:docs/02-technical-specifications/03-frontend-architecture.md†L125-L159■
- **Localization:** Locale bundles support multilingual deployments; formatting utilities draw on Admin & Configuration tenant settings to honour regional policies and data residency commitments.■F:docs/02-technical-specifications/03-frontend-architecture.md†L163-L176■■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L1-L180■

Schema Specification

- **controls:** Canonical control definitions (id, domain, category, title, description, rationale, guidance, owner*team, status, risk*tier, version, created/updated metadata).
- **control_framework_links:** Framework mappings with requirement IDs, coverage level, evidence references, effective/expiry dates.
- **control_check_links:** Associations between controls and checks, storing assertion type, frequency, enforcement level, and weighting.
- **control_scores:** Materialized scores with timestamps, aggregated weight sums, normalized values, and status classification.

- **control_audit_events:** Append-only log recording before/after snapshots, approvers, and change-ticket references.
- Relationships join to Task Service remediation tasks, Notification alerts, dashboards, and Evidence Repository assets for end-to-end traceability.■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■

Testing, Deployment & Operational Readiness

- **Testing Strategy:** Unit tests cover services and repositories using Prisma test harnesses; integration suites validate lifecycle workflows, mapping transactions, and scoring recalculations against seeded data before merges. CI pipelines enforce linting, unit/integration coverage, and contract tests for REST endpoints across environments.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L52-L166■
- **Staging & Release Gates:** Feature flags allow dark launches of new scoring formulas. Staging deployments run smoke tests with Governance Engine interactions and require manual QA plus security approval per operational governance process before promotion.■F:docs/01-about/08-operations-and-teams.md†L129-L144■■F:docs/02-technical-specifications/05-devops-infrastructure.md†L72-L166■
- **Observability:** Services emit Prometheus metrics (score latency, workflow duration), structured logs with correlation IDs, and distributed traces for API calls; alerts track scoring lag, workflow failures, and queue depth within centralized monitoring stacks.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L96-L140■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■
- **Runbooks & On-Call:** On-call engineers follow remediation playbooks for scoring anomalies, mapping conflicts, or evidence validation failures, coordinating with Task and Notification systems to inform stakeholders and document outcomes for audits.■F:docs/03-systems/13-task-management-system/readme.md†L51-L115■■F:docs/03-systems/04-notification-system/readme.md†L56-L126■

Operational Playbooks & References

Playbooks

- **Handling Failures:** When a control drops below threshold, create incidents/remediation tasks, notify owner team, and ensure evidence captured with timestamps for audit.
- **Quarterly Reviews:** Governance team reviews taxonomy accuracy, framework coverage, risk tiers, and evidence freshness; outcomes recorded in compliance tracker and dashboards.
- **Reporting:** Export control posture to BI tooling and regulatory reports, ensuring mappings and scores align with framework obligations.

Related Documentation

- — verification workflows feeding control scores.
- — cross-framework alignment logic.
- — visualization of control health.
- — remediation tracking.

Story: Control Backend Layout

Summary

Document where control models, services, controllers, and integrations live within the codebase.

As a…

As a backend developer

I want to…

I want to find control management files

So that…

So that I can manage control lifecycle logic

Acceptance Criteria

- [ ] Layout covers controllers, services, repositories, and integration clients powering control management.
- [ ] Notes explain how control APIs connect to governance, evidence, and framework mapping services.

Story: Control Taxonomy & Relationships

Summary

Explain how controls map to frameworks, control families, and related evidence requirements.

As a…

As a compliance architect

I want to…

I want a control taxonomy

So that…

So that I can model relationships across regulations

Acceptance Criteria

- [ ] Taxonomy defines core entities, parent-child relationships, and mapping metadata.
- [ ] Relationships connect controls to frameworks, checks, and evidence artifacts for traceability.

Story: Control Service Responsibilities

Summary

Summarize core duties, integration points, and event flows for the control management service.

As a…

As a platform architect

I want to…

I want control service clarity

So that…

So that governance workflows orchestrate correctly

Acceptance Criteria

- [ ] Responsibilities include cataloging controls, syncing statuses, handling attestations, and broadcasting updates.
- [ ] Integration points reach governance engine, evidence, notifications, and reporting systems.

Story: Control API Contracts

Summary

Detail endpoints for control CRUD, relationships, attestations, and evidence linkage.

As a…

As an API consumer

I want to…

I want control endpoint specs

So that…

So that I can automate control lifecycle tasks

Acceptance Criteria

- [ ] Endpoints describe payloads for creating, updating, scoring, and linking controls with metadata.
- [ ] Contracts cover RBAC enforcement, pagination, and webhook notifications for control events.

Story: Control Scoring & Lifecycle

Summary

Describe scoring models, status transitions, reviews, and exception handling for controls.

As a…

As a compliance analyst

I want to…

I want to manage control lifecycle

So that…

So that control posture reflects real-world evidence

Acceptance Criteria

- [ ] Scoring logic evaluates effectiveness, risk, and review cadences tied to governance policies.
- [ ] Lifecycle stages cover drafting, review, approval, monitoring, and archival with escalation rules.

Story: Control Frontend Structure

Summary

Identify UI directories for control catalogs, detail panes, and mapping explorers.

As a…

As a frontend engineer

I want to…

I want to find control UI code

So that…

So that I can enhance operator tooling

Acceptance Criteria

- [ ] Structure references client/src/features/controls pages, components, and state management.
- [ ] Shared UI modules integrate with framework mapping and evidence viewers for context.

Story: Control UI Flows

Summary

Summarize list, detail, scoring, and approval experiences across the console.

As a…

As a UX lead

I want to…

I want cohesive control workflows

So that…

So that teams can review and update controls efficiently

Acceptance Criteria

- [ ] UI flows include status dashboards, edit forms, attestation timelines, and evidence linking interactions.
- [ ] Components enforce RBAC, localization, and audit trails for inline changes.

Story: Control State & Security Patterns

Summary

Explain state management, caching, localization, and security practices applied to control UI and APIs.

As a…

As a security engineer

I want to…

I want to ensure control data stays protected

So that…

So that sensitive mappings do not leak

Acceptance Criteria

- [ ] State patterns describe normalized caches, optimistic updates, and translation loading.
- [ ] Security practices include permission guards, sanitized queries, and tenant scoping across APIs.

Story: Control Management Playbooks

Summary

Provide operational procedures for publishing updates, handling incidents, and coordinating with adjacent systems.

As a…

As an operations manager

I want to…

I want control playbooks

So that…

So that changes roll out with clear communication

Acceptance Criteria

- [ ] Playbooks cover release cadences, audit preparation, and rollback strategies tied to governance engine dependencies.
- [ ] Incident guidance explains alert handling, stakeholder communication, and evidence reconciliation.

Framework Mapping System

Location: /server/src/modules/frameworks

TL;DR

The framework mapping system governs lifecycle management for regulatory frameworks and their control mappings.

It centralizes metadata, aligns controls across standards, and exposes APIs for onboarding, versioning, and exporting frameworks.

Use this guide to understand data contracts, CRUD flows, and operational guardrails that keep mappings authoritative.

Backend Specification

Backend Location & Directory Layout

Framework governance lives under server/src/modules/frameworks, following the layered JavaScript architecture defined for the backend service.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L52-L135■■F:docs/02-technical-specifications/10-coding-standards-and-governance.md†L69-L118■

```
server/src/modules/frameworks/
■■■ controllers/          # Express route handlers (e.g., framework-controller.js)
■■■ routers/              # Route registration in kebab-case files (framework-routes.js)
■■■ services/             # Business orchestration for lifecycle, mapping, and versioning
■■■ repositories/         # Prisma data access for frameworks, controls, mappings, versions
■■■ validators/           # Joi/celebrate schemas reused across controllers and tasks
■■■ policies/             # Casbin helper utilities enforcing role- and tenant-scoped rules
■■■ jobs/                 # BullMQ processors for imports, exports, and scheduled syncs
■■■ subscribers/          # Event listeners (governance engine, reporting, notifications)
■■■ index.js              # Module bootstrap that mounts routers into `server/src/routes`
```

Controllers expose REST endpoints, validators guarantee payload integrity, services orchestrate repositories and integrations, policies enforce Casbin authorization, BullMQ jobs handle asynchronous ingestion/export, and subscribers broadcast mapping changes to the governance engine, evidence management, notification, and reporting modules.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L99-L171■■F:docs/01-about/04-security-and-data-protection.md†L206-L259■■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■

Data Model & Metadata Contracts

- **Frameworks (frameworks):** id, slug, title, version, domain, jurisdiction, publisher, valid_from/to, status, metadata. Metadata captures regulatory IDs, localization, weighting hints, lifecycle flags, and source hashes to satisfy auditability and provenance requirements.■F:docs/02-technical-specifications/04-database-design.md†L41-L114■■F:docs/01-about/04-security-and-data-protection.md†L206-L259■
- **Controls (controls):** framework_id, code, title, description, category, risk_level, evidence_requirements, metadata; controls belong to a single framework version and link directly to governance checks for scoring.■F:docs/02-technical-specifications/04-database-design.md†L76-L99■■F:docs/03-systems/12-governance-engine/readme.md†L58-L153■
- **Mappings (framework_mappings):** source_framework_id, source_control_id, target_framework_id, target_control_id, mapping_strength (exact, partial, informative), justification, tags, status, metadata. Historical rows in framework_mapping_history retain previous relationships for compliance traceability.■F:docs/03-systems/09-control-management-system/readme.md†L49-L168■■F:docs/02-technical-specifications/06-security-implementation.md†L85-L141■
- **Versions (framework_versions):** Semantic versions capturing diff hashes, changelog narratives, approvals, rollout metadata, and compatibility flags so governance engine scoring can reference the correct snapshot.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L103-L171■■F:docs/01-about/10-risk-management-and-Mitigation.md†L146-L214■
- **Imports/Exports (framework_imports, framework_exports):** Track asynchronous ingestion, normalization, and artifact generation (CSV/JSON/XLSX) with payload URIs, statuses, ownership, and detailed error reports for audit and operational review.■F:docs/02-technical-specifications/07-integration-architecture.md†L73-L180■■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L41-L179■

Framework & Mapping Workflows

- **Framework Lifecycle:**

1. POST /api/v1/frameworks validates payloads against celebrate/Joi schemas, generates a kebab-case slug, persists an initial version (v1.0.0), seeds default controls when provided, and emits framework.created with tenant context.

2. GET /api/v1/frameworks supports pagination, jurisdiction, lifecycle, and publisher filters; GET /api/v1/frameworks/:frameworkId returns metadata, active version, control counts, coverage, and connected tasks so risk officers can track obligations.■F:docs/02-technical-specifications/02-backend-arc

hitecture-and-apis.md†L103-L171■■F:docs/01-about/06-mvp-and-roadmap.md†L224-L295■

3. PATCH /api/v1/frameworks/:frameworkId records field-level diffs and routes breaking changes into draft versions for approval, logging every mutation into the audit log module.

4. DELETE /api/v1/frameworks/:frameworkId performs soft archive with effective date windows, while POST /api/v1/frameworks/:frameworkId/restore revalidates references, reinstates mappings, and regenerates downstream caches.

- **Control Registration:** REST endpoints and /api/v1/frameworks/:frameworkId/controls/import bulk jobs create or update controls. Draft updates attach to unpublished versions; breaking changes flag new revisions. BullMQ jobs normalize CSV/JSON uploads, enforce referential integrity, and produce downloadable validation reports.
- **Mapping Management:** CRUD operations (/api/v1/frameworks/:frameworkId/mappings, /api/v1/mappings/import) enforce referential integrity, support reciprocal edges, maintain framework_mapping_history, and emit mapping.* events consumed by governance, reporting, and external integration services.
- **Version Governance:** Draft versions clone current state, aggregate diffs for review, and publish via Admin & Configuration approval workflows that increment semantic versions. Rollbacks reactivate prior versions, trigger recalculation jobs in the governance engine, and notify owners about potential score shifts.■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L41-L179■■F:docs/03-systems/12-governance-engine/readme.md†L150-L205■
- **Multi-Framework Support:** Metadata tags align controls across standards, coverage analytics compute mapping percentages per framework pair, and import/export pipelines ensure schema validation, referential integrity, and policy enforcement. Cached aggregates feed dashboards and external partner exports.■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■■F:docs/03-systems/15-external-integrations-system/readme.md†L41-L167■

Frontend Specification

Frontend Location & Directory Layout

Framework administration UIs live under client/src/features/frameworks, implemented in React (JavaScript) with shared shadcn/Tailwind primitives for consistent accessibility and compliance cues.■F:docs/02-technical-specifications/03-frontend-architecture.md†L50-L160■■F:docs/01-about/04-security-and-data-protection.md†L200-L259■

```
client/src/features/frameworks/
■■■ pages/
■   ■■■ framework-catalog-page.jsx
■   ■■■ framework-detail-page.jsx
■   ■■■ mapping-matrix-page.jsx
■   ■■■ version-history-page.jsx
■■■ components/
■   ■■■ framework-form.jsx
■   ■■■ control-list.jsx
■   ■■■ mapping-editor.jsx
■   ■■■ version-diff-viewer.jsx
■■■ hooks/
■   ■■■ use-frameworks.js
■   ■■■ use-framework-mappings.js
■   ■■■ use-framework-versions.js
■■■ api/
    ■■■ frameworks-client.js

client/src/components/governance/
■■■ coverage-matrix-chart.jsx
```

Reusable Components & UI Flows

- **Catalog & Detail:** framework-catalog-page.jsx surfaces jurisdiction, lifecycle, and publisher filters with saved views; framework-form.jsx handles create/update flows with inline validation and localization toggles; control-list.jsx manages framework-specific controls with inline edits, segmentation by categories, and bulk import triggers.
- **Mapping Operations:** mapping-matrix-page.jsx and mapping-editor.jsx enable control alignment across frameworks, set mapping strength, capture justifications with markdown support, and preview coverage metrics sourced from governance engine scores.
- **Version Governance:** version-history-page.jsx lists drafts and published versions while version-diff-viewer.jsx visualizes added/removed controls and mapping changes. Hooks surface task progress for imports/exports, stream BullMQ job status, and expose rollback actions routed through the Admin & Configuration approvals.

Schema Specification

- **frameworks / framework_versions:** Canonical metadata, semantic versions, diff hashes, approvals, lifecycle flags, and jurisdictional scope; retention policies align with platform-wide data protection mandates.■F:docs/02-technical-specifications/04-database-design.md†L76-L136■■F:docs/01-about/04-security-and-data-protection.md†L206-L259■
- **framework_controls:** Framework-specific controls with risk metadata, evidence requirements, localization fields, and governance engine references for scoring.

- **framework_mappings / framework_mapping_history:** Cross-framework equivalence matrix with mapping strength, tags, justifications, status, and revision history supporting audit readiness and rollback analysis.
- **framework_imports / framework_exports:** Async job tracking tables (source, format, status, URIs, timestamps, error bundles) underpinning operational transparency and SLA reporting.
- **framework_audit_log:** Append-only ledger capturing actor, payload diffs, version transitions, and approval references; feeds the Audit Logging & Monitoring system for holistic oversight.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L190■
- Relationships integrate with Control Management, Governance Engine, Check Management, Reporting, Task Management, and External Integrations to maintain compliance traceability across the platform.■F:docs/03-systems/09-control-management-system/readme.md†L7-L168■■F:docs/03-systems/12-governance-engine/readme.md†L58-L205■■F:docs/03-systems/13-task-management-system/readme.md†L7-L171■■F:docs/03-systems/15-external-integrations-system/readme.md†L41-L167■

Operational Playbooks & References

Playbooks

- **Onboard Framework:** Create framework, import controls, align mappings via preview/import flows, review draft diff, secure dual-approval through Admin & Configuration, publish the initial version, trigger exports, and notify downstream services and stakeholders.
- **Update Mappings After Regulatory Change:** Monitor integrations for updates, submit bulk mapping edits, run dry-run previews, route for governance approval with risk classifications, publish minor versions, and allow governance engine recalculations and reporting refreshes to propagate.■F:docs/01-about/10-risk-management-and-Mitigation.md†L146-L214■
- **Preserve Backward Compatibility:** Maintain retired mappings with status flags, leverage versioned exports for historical reports, enforce retention windows, and ensure reporting services respect effective_from/to intervals for financial and regulatory evidence packages.■F:docs/01-about/09-financial-plans-and-projections.md†L217-L264■

Related Documentation

- — authoritative control catalog and taxonomy alignment.
- — visualization of framework coverage and trend analysis.
- — regulatory feed connectors and export distribution.
- — governance approvals, lifecycle controls, and policy enforcement.
- — scoring, risk computation, and compliance automation triggered by mapping updates.

Story: Framework Mapping Backend Layout

Summary

Document where mapping services, repositories, and synchronization jobs live.

As a…

As a backend engineer

I want to…

I want to locate framework mapping code

So that…

So that I can manage mapping logic

Acceptance Criteria

- [ ] Layout identifies services handling frameworks, mappings, sync workers, and API routes.
- [ ] Notes explain integration with control management, evidence, and reporting modules.

Story: Framework Data Model

Summary

Explain metadata structures capturing frameworks, controls, mappings, and versioning.

As a…

As a data steward

I want to…

I want to understand framework schemas

So that…

So that mappings remain accurate and auditable

Acceptance Criteria

- [ ] Data model includes framework metadata, mapping relationships, and version fields for change tracking.
- [ ] Contracts describe how mappings align with control records and evidence references.

Story: Framework Mapping Workflows

Summary

Describe ingestion, normalization, mapping, and synchronization processes for frameworks.

As a…

As an operations lead

I want to…

I want mapping workflow clarity

So that…

So that new or updated frameworks roll out smoothly

Acceptance Criteria

- [ ] Workflows cover importing frameworks, deduplicating controls, and aligning mappings to control catalogs.
- [ ] Processes trigger governance recalculations, task creation, and notifications when mappings change.

Story: Framework Frontend Structure

Summary

Map UI directories for mapping explorers, comparison tools, and mapping editors.

As a…

As a frontend developer

I want to…

I want to find framework UI code

So that…

So that I can improve mapping visualization

Acceptance Criteria

- [ ] Structure references client/src/features/frameworks pages, components, and hooks.
- [ ] Shared modules integrate with controls UI for cross-linking and filtering.

Story: Framework UI Flows

Summary

Summarize comparison, mapping, and review experiences available in the client.

As a…

As a compliance analyst

I want to…

I want intuitive mapping tools

So that…

So that I can reconcile frameworks and controls

Acceptance Criteria

- [ ] Flows include side-by-side comparisons, mapping editors, and review queues with status indicators.
- [ ] State management ensures filters, search, and selection stay synchronized across modules.

Story: Framework Mapping Playbooks

Summary

Provide operational guidance for version updates, stakeholder communication, and dependency coordination.

As a…

As a release coordinator

I want to…

I want mapping runbooks

So that…

So that framework updates roll out predictably

Acceptance Criteria

- [ ] Playbooks cover monitoring vendor feeds, staging updates, and notifying control owners of changes.
- [ ] Runbooks include rollback strategies, audit evidence capture, and reporting alignment.

Evidence Management System

Location: /server/src/modules/evidence

TL;DR

The evidence management system governs how compliance artifacts are collected, secured, and distributed.

Implemented in server/src/modules/evidence, it orchestrates presigned upload/download flows, metadata persistence, and immutable audit trails.

This reference documents ingestion patterns, schema design, operational safeguards, and integrations with controls, checks, and tasks.

Backend Specification

Backend Location & Directory Layout

Evidence orchestration lives in server/src/modules/evidence, integrating with shared MinIO clients and exposing upload/download APIs that enforce encryption, presigned URL lifecycles, and linkage to governance entities.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L44-L171■

```
server/src/modules/evidence/
■■■ controllers/
■   ■■■ upload.controller.ts
■   ■■■ download.controller.ts
■   ■■■ metadata.controller.ts
■■■ services/
■   ■■■ upload.service.ts
■   ■■■ download.service.ts
■   ■■■ metadata.service.ts
■■■ repositories/
■   ■■■ evidence.repository.ts
■   ■■■ evidence-links.repository.ts
■■■ integrations/
■   ■■■ minio.client.ts
■■■ events/
■   ■■■ evidence.created.ts
■   ■■■ evidence.accessed.ts
■■■ tasks/
    ■■■ retention.scheduler.ts
```

API Surface & Controllers

Endpoint	Method	Controller	Purpose
/api/v1/evidence/upload	POST	upload.controller.ts	Issues presigned upload sessions, validates RBAC scopes, and coordinates checksum-validated completion callbacks.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L122-L171■
/api/v1/evidence/:id/download	GET	download.controller.ts	Authorizes access, minting short-lived presigned GET URLs with binding to stored metadata and audit context.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L122-L171■■F:docs/01-about/04-security-and-data-protection.md†L248-L338■
/api/v1/evidence/:id/metadata	GET/PUT	metadata.controller.ts	Retrieves or amends descriptive metadata, version lineage, retention flags, and governance linkages under audit review policies.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L122-L171■■F:docs/02-technical-specifications/04-database-design.md†L82-L139■
/api/v1/evidence/:id/links	POST/DELETE	metadata.controller.ts	Adds or removes associations to controls, checks, and tasks while enforcing compensating control documentation.■F:docs/02-technical-specifications/04-database-design.md†L82-L146■■F:docs/03-systems/09-control-management-system/readme.md†L28-L103■

Controllers register with Express routers inside server/src/modules/evidence/routes.ts and rely on shared middleware for authentication (JwtGuard), Casbin scope checks, and structured error handling to maintain consistent API posture across the platform.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L44-L171■■F:docs/02-technical-specifications/06-security-implementation.md†L6-L148■

Service Responsibilities & Cross-Cutting Concerns

- **upload.service.ts:** Generates presigned PUT URLs, records expected checksums, and queues completion tasks for immutability verification. Handles both human and automated probe submissions so ingestion parity remains consistent.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L44-L171■■F:docs/02-technical-specifications/07-integration-architecture.md†L70-L117■
- **download.service.ts:** Resolves storage keys, enforces download throttling, and assembles audit payloads prior to generating signed GET URLs. Integrates with Notification and Audit Logging modules for anomaly detection hooks.■F:docs/01-about/04-security-and-data-protection.md†L248-L338■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L200■
- **metadata.service.ts:** Applies validation schemas, persists retention and version markers, and orchestrates link mutations to governance entities. Soft-delete semantics preserve historical context while preventing orphaned records.■F:docs/02-technical-specifications/04-database-design.md†L82-L177■
- **Repositories:** Prisma-backed repositories centralize query patterns, enforce referential integrity, and expose transactional helpers for multi-table updates (evidence + links + events). Indices on tags, control IDs, and retention states keep retrieval performant for dashboards.■F:docs/02-technical-specifications/04-database-design.md†L82-L150■
- **Integrations & Events:** Shared MinIO client wraps credential injection and bucket routing; event publishers emit evidence.created and evidence.accessed messages into the governance event bus consumed by Control, Task, and Notification systems.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L44-L171■■F:docs/03-systems/04-notification-system/readme.md†L7-L222■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■

Workflow, Events & Integrations

1. **Session Initiation:** Clients (user or probe) request uploads; RBAC middleware evaluates scopes (evidence:write, probe:evidence) and enforces tenant isolation. Casbin policies align with RBAC system definitions to prevent privilege drift.■F:docs/03-systems/02-rbac-system/readme.md†L7-L192■■F:docs/02-technical-specifications/06-security-implementation.md†L85-L148■

2. **Presign Orchestration:** Upload service calculates object keys using {tenant}/{classification}/{yyyy}/{mm} conventions inherited from the Upload System to ensure consistent retention policies and compression behavior.■F:docs/03-systems/03-document-and-media-upload/readme.md†L33-L131■

3. **Completion & Validation:** Workers verify size, checksum, and malware scan status before promoting evidence to active state, persisting metadata, and publishing evidence.created events that the Governance Engine consumes to refresh coverage matrices.■F:docs/03-systems/03-document-and-media-upload/readme.md†L107-L177■■F:docs/03-systems/12-governance-engine/readme.md†L52-L113■

4. **Access & Audit:** Download requests gather metadata (version, retention, legal holds), log immutable audit entries, and stream via expiring URLs. Audit logs feed dashboards and compliance reports for regulators.■F:docs/01-about/04-security-and-data-protection.md†L248-L338■■F:docs/03-systems/14-dashbo

ard-and-reporting-system/readme.md†L7-L118■

5. **Retention Automation:** retention.scheduler.ts evaluates retention policies nightly, transitioning artifacts to archival storage, pausing deletions under legal hold, and opening remediation tasks when manual approval is required.■F:docs/02-technical-specifications/04-database-design.md†L162-L190■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■

Ingestion & Distribution Flows

- **Manual & Assisted Uploads:** Privileged users request upload sessions (/evidence/upload), receive short-lived presigned URLs to stream files into MinIO, and on completion metadata (size, checksum, MIME type, actor, control/check/task references) persists to evidence and evidence_links.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L122-L133■■F:docs/02-technical-specifications/01-system-architecture.md†L166-L193■
- **Automated Probes:** System credentials issued via Casbin policies allow probes to submit artifacts with identical metadata schema and version tracking, preserving parity between automated and manual submissions.■F:docs/02-technical-specifications/04-database-design.md†L80-L99■■F:docs/01-about/04-security-and-data-protection.md†L263-L287■
- **Secure Downloads:** RBAC-validated requests (/evidence/:id/download) generate presigned URLs bound to object keys and expirations. Downloads reference stored metadata for checksum validation and append immutable audit entries.■F:docs/01-about/04-security-and-data-protection.md†L263-L311■

Metadata & Chain of Custody

- Evidence metadata in PostgreSQL captures identifiers, storage references, versions, lifecycle timestamps, and relationships to controls, checks, and tasks. Integrity constraints prevent orphaning while soft deletes preserve history.■F:docs/02-technical-specifications/04-database-design.md†L86-L130■
- Audit trails log uploads, approvals, downloads, and edits with timestamps, actor identities, network origin, and cryptographic hashes stored in append-only ledgers for tamper detection.■F:docs/01-about/04-security-and-data-protection.md†L87-L311■
- Versioning, tagging, and retention policies retain complete histories (default 36 months) with automated archival, GDPR-compliant deletion, and encrypted backups (nightly full, hourly differential).■F:docs/02-technical-specifications/04-database-design.md†L147-L171■
- **Indexing & Searchability:** Text search indexes on evidence names, tags, and linked controls power global search, while composite indexes on (control_id, retention_state) enable fast filtering in dashboards without degrading transactional writes.■F:docs/02-technical-specifications/04-database-design.md†L112-L150■
- **Data Integrity Safeguards:** Soft deletes, referential constraints, and versioned updates guarantee traceable history while preventing cascading data loss in downstream governance modules.■F:docs/02-technical-specifications/04-database-design.md†L120-L177■

Frontend Specification

Frontend Location & Directory Layout

Evidence workflows surface in client/src/features/evidence, giving compliance users upload interfaces, review dashboards, and linkage to governance objects.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L160■

```
client/src/features/evidence/
■■■ pages/
■   ■■■ EvidenceLibraryPage.tsx
■   ■■■ EvidenceUploadPage.tsx
■   ■■■ EvidenceDetailPage.tsx
■   ■■■ EvidenceRetentionPage.tsx
■■■ components/
■   ■■■ EvidenceUploadWizard.tsx
■   ■■■ EvidenceMetadataPanel.tsx
■   ■■■ EvidenceDownloadButton.tsx
■   ■■■ EvidenceLinkingForm.tsx
■■■ hooks/
■   ■■■ useEvidenceLibrary.ts
■   ■■■ useEvidenceUpload.ts
■   ■■■ useEvidenceRetention.ts
■■■ api/
    ■■■ evidenceClient.ts

client/src/components/governance/
■■■ EvidenceTimeline.tsx
```

Reusable Components & UI Flows

- **Upload Wizard:** Guides users through metadata capture (control, check, task references), calculates checksums client-side, and displays compression requirements before requesting presigned URLs.
- **Library & Detail Views:** EvidenceLibraryPage provides search/filtering by tags, framework, control, and retention status; EvidenceMetadataPanel reveals version history, audit logs, and linked remediation tasks.
- **Download & Linking:** EvidenceDownloadButton enforces revalidation before generating presigned links, while EvidenceLinkingForm allows reassociation of artifacts with additional controls or tasks under audit supervision.
- **Retention Management:** EvidenceRetentionPage and useEvidenceRetention display lifecycle stages (active, archived, purge scheduled) with approval workflows for legal holds and deletions.

State Management & API Contracts

- **API Client (api/evidenceClient.ts):** Wraps Axios with interceptors for JWT handling, error normalization, and retry logic aligned with frontend architecture guidelines.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L168■
- **Hooks:** useEvidenceLibrary, useEvidenceUpload, and useEvidenceRetention encapsulate query caching, optimistic updates for metadata edits, and websocket-driven status updates broadcast from Notification service topics.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L168■■F:docs/03-systems/04-notification-system/readme.md†L7-L222■
- **State Containers:** Feature hooks integrate with Auth and Notification contexts to respect RBAC-scoped UI and toasts, falling back to local reducers for wizard progress per the design philosophy of localized state.■F:docs/02-technical-specifications/03-frontend-architecture.md†L88-L150■
- **Validation & Accessibility:** Forms leverage shared validation schemas and shadcn/ui components to remain WCAG AA compliant, with keyboard navigable evidence tables and focus-managed modals for download confirmation.■F:docs/02-technical-specifications/03-frontend-architecture.md†L88-L146■

Localization & Reporting Hooks

- Text content draws from shared localization files, allowing evidence labels, retention states, and audit statuses to render in the user's preferred locale. Currency/date formatting matches compliance dashboards for consistent reporting narratives.■F:docs/02-technical-specifications/03-frontend-architecture.md†L168-L210■■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■
- Evidence pages emit analytics events consumed by Dashboard & Reporting to surface artifact freshness metrics and control coverage insights alongside other governance KPIs.■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■

Schema Specification

- **evidence:** Stores artifact metadata (id, storage*key, version, size, checksum, mime*type, uploader*id, source, created*at, archived*at, tags, retention*policy_id).
- **evidence_links:** Joins evidence to controls, checks, and tasks with context (role, justification, linked*by, linked*at).
- **evidence_events:** Audit ledger capturing action type (upload, download, approval), actor, origin IP, hash, and integrity verification status.
- **evidence_retention_policies:** Defines retention duration, archival storage location, legal hold flags, and purge schedules.

- **evidence_versions:** Optional table tracking previous file hashes, storage keys, and validation outcomes for immutable history.
- Relationships integrate with Control Management, Check Management, Task Management, and Notification systems for traceable governance actions.■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■■F:docs/03-systems/04-notification-system/readme.md†L7-L170■
- **Indices & Partitioning:** High-volume tables (evidence, evidence_events) adopt time-based partitioning and text search indexes to satisfy search requirements without sacrificing ingestion throughput.■F:docs/02-technical-specifications/04-database-design.md†L61-L150■
- **Retention Policies:** Default 36-month retention with configurable extensions ensures legal compliance; archival routines move aged evidence to cold storage while preserving immutable references for auditors.■F:docs/02-technical-specifications/04-database-design.md†L162-L190■

Operational Playbooks & References

Storage & Security Operations

- Configure MinIO endpoints and credentials per environment via secure vaults; confirm upload smoke tests post-deploy. Infrastructure-as-code manages bucket lifecycle, versioning, and capacity planning.■F:docs/02-technical-specifications/08-deployment-and-environment-guide.md†L100-L205■■F:docs/02-technical-specifications/05-devops-infrastructure.md†L155-L226■
- Enforce AES-256 encryption at rest (MinIO + PostgreSQL) and TLS in transit. Presigned URLs must target HTTPS endpoints; key rotation follows managed KMS policies.■F:docs/01-about/04-security-and-data-protection.md†L87-L150■
- Integrity verification combines MinIO checksums, stored metadata hashes, and immutable log signatures. Deployment rituals include validating upload/download flows and monitoring alerts for anomaly detection.■F:docs/02-technical-specifications/08-deployment-and-environment-guide.md†L182-L205■

Monitoring, Alerting & Incident Response

- Observability dashboards track API latency, presign success rate, retention job duration, and MinIO health, with PagerDuty alerts when thresholds breach SLOs (99.9 % uptime for production evidence services).■F:docs/02-technical-specifications/05-devops-infrastructure.md†L179-L206■
- Audit and notification pipelines emit anomalies (e.g., repeated download failures, unexpected bulk exports) to SIEM tooling for investigation, aligning with the security-by-design mandate.■F:docs/01-about/04-security-and-data-protection.md†L248-L338■■F:docs/02-technical-specifications/06-security-implementation.md†L6-L148■

- Incident runbooks cover presign queue backlogs, retention job stalls, and cross-region failover. Disaster recovery exercises validate multi-region MinIO replicas and Postgres snapshots before reactivating evidence access.■F:docs/02-technical-specifications/05-devops-infrastructure.md†L213-L226■■F:docs/02-technical-specifications/04-database-design.md†L162-L190■

Testing & Quality Assurance

- Automated integration suites create temporary uploads, verify metadata persistence, and validate download authorizations across role permutations as part of CI.■F:docs/02-technical-specifications/09-testing-and-qa.md†L91-L156■
- QA scripts exercise retention scheduler paths (archive, purge, legal hold) in staging environments with synthetic data to ensure audit trails remain intact before production deployment.■F:docs/02-technical-specifications/09-testing-and-qa.md†L118-L156■

Related Documentation

- — presigned URL orchestration and compression policies.
- — evidence linkage to check outcomes.
- — control catalog relationships.
- — remediation workflows referencing evidence.

Story: Evidence Backend Layout

Summary

Document controllers, services, events, and storage adapters that compose the evidence service.

As a…

As a backend developer

I want to…

I want to locate evidence code

So that…

So that I can extend ingestion and retrieval logic

Acceptance Criteria

- [ ] Layout includes controllers, services, integrations, and event handlers for evidence flows.
- [ ] Notes explain how uploads, governance engine, and reporting modules interface with evidence services.

Story: Evidence API Surface

Summary

Detail endpoints exposed for managing evidence records, attachments, and queries.

As a…

As an API consumer

I want to…

I want evidence endpoint specs

So that…

So that I can integrate ingestion and retrieval

Acceptance Criteria

- [ ] Endpoints describe CRUD, search, attachment linking, and export operations with payload requirements.
- [ ] Contracts highlight RBAC enforcement, pagination, and event emission for downstream consumers.

Story: Evidence Service Responsibilities

Summary

Summarize cross-cutting duties like chain of custody, RBAC enforcement, and retention policies.

As a…

As a compliance architect

I want to…

I want evidence service clarity

So that…

So that governance and audits rely on trustworthy artifacts

Acceptance Criteria

- [ ] Responsibilities include verifying uploads, linking to controls, maintaining provenance, and enforcing retention.
- [ ] Service coordinates with notifications, tasks, reporting, and governance for evidence-driven workflows.

Story: Evidence Workflow & Integrations

Summary

Describe ingestion, publishing, subscription, and event routing across dependent systems.

As a…

As an integrations engineer

I want to…

I want evidence workflow visibility

So that…

So that events propagate to tasks and dashboards

Acceptance Criteria

- [ ] Workflows detail event schemas, streaming, and consumer responsibilities across governance and reporting.
- [ ] Integrations manage subscription filters, retries, and reconciliation of evidence state.

Story: Evidence Ingestion & Distribution

Summary

Explain flows from upload completion through enrichment, tagging, and distribution to consumers.

As a…

As a product manager

I want to…

I want ingestion clarity

So that…

So that evidence becomes usable quickly

Acceptance Criteria

- [ ] Processes cover metadata enrichment, tagging, control associations, and distribution to dashboards.
- [ ] Distribution ensures RBAC filtering, version tracking, and subscription notifications.

Story: Evidence Metadata & Chain of Custody

Summary

Summarize metadata captured, audit trails, and verification steps safeguarding evidence integrity.

As a…

As an auditor

I want to…

I want chain of custody assurance

So that…

So that evidence can withstand regulatory scrutiny

Acceptance Criteria

- [ ] Metadata includes uploader identity, timestamps, hashes, tags, and change history for each artifact.
- [ ] Chain-of-custody records approvals, transfers, and access events with immutable logs.

Story: Evidence Frontend Structure

Summary

Map UI directories for evidence timelines, previews, and review consoles.

As a…

As a frontend engineer

I want to…

I want to locate evidence UI code

So that…

So that I can enhance review experiences

Acceptance Criteria

- [ ] Structure references client/src/features/evidence pages, components, and state management hooks.
- [ ] Shared modules integrate with uploads, controls, and reporting features for context.

Story: Evidence UI Flows

Summary

Describe viewer, tagging, approval, and export experiences in the client.

As a…

As a compliance analyst

I want to…

I want intuitive evidence tools

So that…

So that I can review and attach artifacts

Acceptance Criteria

- [ ] UI flows include preview cards, timelines, approval modals, and evidence linking to tasks.
- [ ] State contracts ensure filters, localization, and API requests stay aligned with backend expectations.

Story: Evidence State & API Contracts

Summary

Explain state management patterns, caching, and API client usage across evidence features.

As a…

As a frontend architect

I want to…

I want consistent evidence data handling

So that…

So that UI remains performant and accurate

Acceptance Criteria

- [ ] State patterns use normalized caches, pagination, and optimistic updates tied to API contracts.
- [ ] Contracts cover GraphQL or REST usage, error handling, and subscription updates.

Story: Evidence Localization & Reporting Hooks

Summary

Summarize localization support, reporting exports, and integration hooks.

As a…

As a localization lead

I want to…

I want evidence internationalization guidance

So that…

So that global teams access artifacts

Acceptance Criteria

- [ ] Localization ensures copy from locale bundles, timezone-aware timestamps, and accessible descriptions.
- [ ] Reporting hooks expose exports, data feeds, and KPI widgets connected to dashboards.

Story: Evidence Storage & Security Operations

Summary

Describe storage encryption, access controls, and operational safeguards for evidence data.

As a…

As a security engineer

I want to…

I want evidence protection details

So that…

So that data remains secure at rest and in transit

Acceptance Criteria

- [ ] Storage operations cover encryption, retention, replication, and key rotation policies.
- [ ] Security procedures manage access reviews, anomaly detection, and incident escalation.

Story: Evidence Monitoring & Incident Response

Summary

Outline observability metrics, alerts, and runbooks for evidence services.

As a…

As an on-call responder

I want to…

I want evidence monitoring guidance

So that…

So that I can handle outages or data issues

Acceptance Criteria

- [ ] Metrics monitor ingestion latency, processing backlogs, and export throughput with alert thresholds.
- [ ] Runbooks include investigation steps, communication protocols, and recovery tasks.

Story: Evidence Testing & QA

Summary

Define testing suites ensuring evidence workflows remain reliable and compliant.

As a…

As a QA lead

I want to…

I want evidence testing standards

So that…

So that automated coverage protects the service

Acceptance Criteria

- [ ] Testing covers unit, integration, E2E, and contract suites for ingestion, retrieval, and UI flows.
- [ ] Quality gates enforce performance checks, security scans, and compliance validations.

Governance Engine

Location: /server/src/modules/governance

TL;DR

The governance engine is the platform core that turns evidence into actionable compliance intelligence.

Located at server/src/modules/governance, it executes checks, aggregates control and framework scores, and drives remediation workflows.

This guide explains lifecycle orchestration, execution internals, extensibility hooks, and system dependencies.

Backend Specification

Backend Location & Directory Layout

server/src/modules/governance follows the feature-oriented layout, coordinating check execution, control aggregation, framework scoring, and remediation orchestration. All backend code is authored in JavaScript (Node.js + Express) with feature folders keeping controllers, services, jobs, and subscribers co-located for discoverability.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L12-L103■

```
server/src/modules/governance/
■■■ checks/
■   ■■■ checks.service.js
■   ■■■ execution.pipeline.js
■   ■■■ lifecycle.service.js
■■■ controls/
■   ■■■ scoring.service.js
■   ■■■ posture.service.js
■■■ frameworks/
■   ■■■ mapping.service.js
■■■ orchestration/
■   ■■■ scheduler.service.js
■   ■■■ remediation.service.js
■   ■■■ workflow.builder.js
■■■ subscribers/
■   ■■■ evidence.subscriber.js
■   ■■■ mapping.subscriber.js
■■■ controllers/
■   ■■■ governance.controller.js
■   ■■■ review-queue.controller.js
■   ■■■ scoring.controller.js
■■■ routes/
■   ■■■ governance.routes.js
■■■ ui/
    ■■■ admin-console/
```

Supporting modules live under server/src/routes for route registration, server/src/integrations for shared providers (MinIO, Nodemailer, Casbin), and server/src/config for environment-driven configuration consumed by the governance services.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L63-L103■

Lifecycle Orchestration

- **Evidence Intake:** Subscribers react to probe uploads and manual submissions, enqueuing evaluations for relevant checks.■F:docs/01-about/03-concept-summary.md†L214-L359■
- **Scheduling:** Cron/event schedules maintained by the Probe Management system feed into scheduler.service.js, which prioritizes check runs based on frequency, risk tier, and outstanding remediation items.■F:docs/03-systems/07-probe-management-system/readme.md†L7-L226■
- **Execution Pipeline:** execution.pipeline.js fetches definitions, resolves probe outputs, applies validation logic, and writes results into the Check Management tables.■F:docs/03-systems/08-check-management-system/readme.md†L7-L165■
- **Feedback Loop:** Remediation service opens tasks, dispatches notifications, and updates dashboards. Results flow into control aggregation, framework scoring, and evidence linkage for continuous monitoring.■F:docs/01-about/03-concept-summary.md†L282-L359■

Execution & Aggregation Model

- **Check Runners:** Pluggable strategy objects evaluate automated, manual, or hybrid checks; manual submissions enter review queues before publication. Strategy instances use shared validators in server/src/modules/governance/checks/validators and persist normalized payloads via Prisma repositories to satisfy reporting and remediation requirements.■F:docs/01-about/03-concept-summary.md†L224-L297■■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L32-L89■
- **Control Aggregation:** Aggregates check outcomes using configured weights, risk tiers, and thresholds to produce control posture scores. Scores propagate to frameworks for compliance reporting and support downstream dashboard visualizations.■F:docs/03-systems/09-control-management-system/readme.md†L7-L159■■F:docs/01-about/03-concept-summary.md†L298-L343■
- **Remediation Orchestration:** Failing controls trigger incident tickets, remediation tasks, and escalation notifications while preserving immutable audit records.■F:docs/03-systems/04-notification-system/readme.md†L7-L200■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■
- **Logging & Audit:** Every execution emits structured events captured by the Audit Logging system for traceability and tamper-evident compliance history.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■

Service Responsibilities & API Surface

- **Controllers & Routes:** governance.controller.js exposes REST endpoints (GET /governance/overview, GET /governance/controls/:id, POST /governance/runs, POST /governance/recalculate) mounted through governance.routes.js. Endpoints obey the shared REST, validation, and error-handling standards defined for all services.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L118-L198■
- **Schedulers:** scheduler.service.js registers cron jobs with the platform scheduler, reading cadence and prioritization metadata from the Check Management tables. Jobs enqueue work onto the Governance Engine execution pipeline and fall back to retry queues when dependent systems are degraded.■F:docs/03-systems/07-probe-management-system/readme.md†L7-L226■■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L24-L89■
- **Execution Pipeline:** execution.pipeline.js orchestrates fetch→validate→persist steps, calling Probe Management adapters, Check Management repositories, and Notification/Task services to ensure cohesive remediation flows.■F:docs/03-systems/07-probe-management-system/readme.md†L7-L226■■F:docs/03-systems/08-check-management-system/readme.md†L34-L141■■F:docs/03-systems/13-task-management-system/readme.md†L7-L173■
- **Scoring Services:** scoring.controller.js and scoring.service.js expose recalculation triggers and cached score retrieval, publishing framework score updates to Dashboard & Reporting subscribers via the message bus.■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■■F:docs/03-systems/10-framework-mapping-system/readme.md†L7-L210■
- **Security & Access:** Each endpoint enforces JWT authentication, Casbin policy checks, and request validation middleware defined in the shared backend specification. Governance-specific scopes gate

mutation routes to compliance roles only.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L118-L198■

Configuration, Dependencies, and Cross-System Contracts

- **Environment Variables:** GOVERNANCE_RUNNER_BATCH_SIZE, GOVERNANCE_MAX_CONCURRENCY, GOVERNANCE_NOTIFICATION_TEMPLATE, and storage credentials are injected via server/src/config. Feature toggles (e.g., hybrid-review enforcement) rely on the shared config loader described in the backend architecture chapter.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L63-L103■
- **Integrations:** MinIO clients handle evidence payload retrieval, Nodemailer templates deliver remediation notices, and Casbin ensures role-based access alignment across governance routes.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L24-L120■■F:docs/03-systems/04-notification-system/readme.md†L7-L200■
- **Messaging Contracts:** Evidence and mapping subscribers consume domain events from the Probe Management and Framework Mapping systems, updating governance caches and run queues accordingly. Events adhere to the naming and payload conventions documented in the respective system guides.■F:docs/03-systems/07-probe-management-system/readme.md†L57-L202■■F:docs/03-systems/10-framework-mapping-system/readme.md†L7-L210■
- **Resilience:** Circuit breakers protect downstream dependencies, and failed executions emit structured audit events plus retry jobs. All recoverable errors bubble through the shared Express error handler to maintain observability and consistent response codes.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L170-L198■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■

Extensibility Hooks

- **New Checks/Probes:** Register definitions via Check Management, add probe integrations through Probe Management, and subscribe to governance events to extend workflows.■F:docs/03-systems/07-probe-management-system/readme.md†L7-L226■■F:docs/03-systems/08-check-management-system/readme.md†L7-L165■
- **Scoring Adjustments:** Control weights and thresholds configurable through Control Management; governance services recalculate scores automatically when mappings change.■F:docs/03-systems/09-control-management-system/readme.md†L7-L159■
- **Reporting Hooks:** Subscribers broadcast changes to dashboards, reporting exports, and external integrations when frameworks or scores update.■F:docs/03-systems/10-framework-mapping-system/readme.md†L7-L210■■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■

Frontend Specification

Frontend Location & Directory Layout

Governance operations UI resides in client/src/features/governance, providing dashboards, review queues, and administrative tooling. Feature folders contain React components, hooks, localized copy, and API clients implemented in JavaScript, aligned with the shared frontend architecture guidance.■F:docs/02-technical-specifications/03-frontend-architecture.md†L80-L138■

```
client/src/features/governance/
■■■ pages/
■   ■■■ GovernanceOverviewPage.jsx
■   ■■■ ControlHealthPage.jsx
■   ■■■ FrameworkScorecardPage.jsx
■   ■■■ ReviewQueuePage.jsx
■■■ components/
■   ■■■ GovernanceScorecard.jsx
■   ■■■ ControlDrilldownPanel.jsx
■   ■■■ FrameworkTrendChart.jsx
■   ■■■ RemediationWorkflowPanel.jsx
■■■ hooks/
■   ■■■ useGovernanceOverview.js
■   ■■■ useReviewQueue.js
■   ■■■ useRemediationActions.js
■■■ api/
■   ■■■ governanceClient.js
■■■ locales/
■   ■■■ en.json
■■■ styles/
    ■■■ governance.css

client/src/components/governance/
■■■ EvidenceControlMatrix.jsx
```

Reusable Components & UI Flows

- **Overview Dashboards:** GovernanceScorecard and FrameworkTrendChart visualize aggregated posture, risk tiers, and historical performance using shadcn/ui primitives and Tailwind tokens for consistent look-and-feel.■F:docs/02-technical-specifications/03-frontend-architecture.md†L80-L132■
- **Control Drilldowns:** ControlDrilldownPanel highlights failing checks, linked evidence, and remediation progress with quick links to tasks; contextual tooltips pull localized copy from locales/en.json to support future translations.■F:docs/02-technical-specifications/03-frontend-architecture.md†L138-L174■
- **Review Queue:** ReviewQueuePage manages manual/hybrid check approvals, surfacing SLA breaches and requiring dual approvals where configured. Hooks encapsulate Axios calls with interceptors for consistent JWT handling.■F:docs/02-technical-specifications/03-frontend-architecture.md†L146-L193■
- **Remediation Workflow:** RemediationWorkflowPanel integrates with Task Management to track status, due dates, and evidence closure while rendering accessible form controls that comply with

WCAG AA guidelines.■F:docs/02-technical-specifications/03-frontend-architecture.md†L138-L193■ ■F:docs/01-about/03-concept-summary.md†L332-L359■

- **Evidence-Control Matrix:** Shared component EvidenceControlMatrix enables cross-system evidence navigation and is reused by Evidence Management, ensuring UI parity across governance modules.■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L115■

State, Data Fetching, and Security Considerations

- Feature hooks manage local state via useState/useReducer, while global posture values subscribe to the Governance Overview context so dashboards update after each run.■F:docs/02-technical-specifications/03-frontend-architecture.md†L92-L152■
- Axios clients defined in governanceClient.js attach JWT tokens, handle refresh flows, and surface errors through the shared notification context to maintain consistent UX messaging.■F:docs/02-technical-specifications/03-frontend-architecture.md†L146-L193■
- Sensitive identifiers (control IDs, framework IDs) are masked in the UI for read-only roles, aligning with the cross-role collaboration rules described in the concept summary.■F:docs/01-about/03-concept-summary.md†L206-L247■
- Components must meet accessibility and localization expectations out-of-the-box; copy, date formatting, and severity color ramps are provided via the shared design system tokens.■F:docs/02-technical-specifications/03-frontend-architecture.md†L118-L174■

Schema Specification

- **governance_runs:** Execution metadata (run*id, check*id, control*id, schedule*id, trigger*type, started*at, completed_at, status, diagnostics) indexed on status, control_id, and started_at to power dashboards and SLA reporting.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L140-L168■
- **control_scores / framework_scores:** Aggregated values with weighting context, thresholds, and timestamps for historical comparison. Changes trigger scoring webhooks and publish events consumed by the Dashboard system.■F:docs/03-systems/09-control-management-system/readme.md†L7-L159■■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■
- **governance_notifications:** Records of remediation alerts, recipients, escalation tier, and acknowledgement status. Relies on Notification templates and retains immutable audit metadata.■F:docs/03-systems/04-notification-system/readme.md†L7-L200■
- **review_queue:** Manual/hybrid approvals with assigned reviewer, due dates, SLA state, and validation timestamps.■F:docs/03-systems/08-check-management-system/readme.md†L7-L165■
- **governance_audit_events:** Append-only log capturing execution and scoring changes for traceability, consumed by the centralized audit and monitoring stack.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■

- **evidence_links:** Join table that binds governance runs to stored artifacts in the Evidence Management system, ensuring cross-module traceability.■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L115■
- **governance_settings:** Configuration overrides (e.g., scoring weights, SLA thresholds, reviewer requirements) surfaced through the admin console and enforced by orchestration services.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L63-L168■

Operational Playbooks & References

Operations

- **Monitor Runtime Health:** Track scheduler health, run queue depth, execution latency, and retry counts. Use Audit Logging dashboards to inspect failure spikes or repeated probe timeouts.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■
- **Validate Scoring Integrity:** Recalculate governance pipelines after framework updates, compare historical vs. recalculated control scores, and confirm framework rollups remain within expected tolerance bands.■F:docs/03-systems/10-framework-mapping-system/readme.md†L7-L210■■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■
- **Coordinate Remediation:** During incidents coordinate with Notification and Task systems to confirm remediation pathways are functioning, verifying escalations and acknowledgements are recorded in governance_notifications.■F:docs/03-systems/04-notification-system/readme.md†L7-L200■■F:docs/03-systems/13-task-management-system/readme.md†L7-L226■
- **Evidence Assurance:** Cross-check evidence links for failed controls to ensure every run retains supporting artifacts, aligning with the continuous monitoring expectations set in the platform concept summary.■F:docs/01-about/03-concept-summary.md†L314-L359■■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L115■

Implementation Checklist

- Adhere to the JavaScript-only backend mandate (no TypeScript) and follow shared naming conventions for controllers, services, and routes.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L12-L107■
- Document new or updated endpoints in the OpenAPI specification and announce breaking scoring changes through change management logs, keeping API consumers synchronized.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L118-L198■
- Extend governance UI components using the feature-folder conventions, ensure accessibility checks pass, and surface localization strings through the shared JSON catalogs.■F:docs/02-technical-specifications/03-frontend-architecture.md†L80-L193■

- Coordinate schema migrations with the platform-wide database governance process to maintain referential integrity and version history.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L138-L168■

Related Documentation

- — definitions and execution workflows.
- — scoring model and lifecycle governance.
- — cross-framework propagation of control results.
- — evidence storage and linkage.
- — visualization of governance outputs.
- — remediation follow-up.

Story: Governance Backend Layout

Summary

Document controllers, services, schedulers, and worker modules powering the governance engine.

As a…

As a backend engineer

I want to…

I want to locate governance code

So that…

So that I can extend orchestration logic

Acceptance Criteria

- [ ] Layout lists orchestration services, scoring engines, event processors, and configuration modules.
- [ ] Notes explain boundaries with tasks, notifications, evidence, and controls.

Story: Governance Lifecycle Orchestration

Summary

Describe intake, evaluation, scoring, and escalation cycles handled by the engine.

As a…

As a compliance strategist

I want to…

I want lifecycle clarity

So that…

So that governance actions align with policy cadences

Acceptance Criteria

- [ ] Lifecycle covers collecting signals, normalizing data, evaluating rules, and triggering remediation tasks.
- [ ] Escalation steps feed notifications, dashboards, and evidence capture for ongoing monitoring.

Story: Governance Execution & Aggregation

Summary

Explain execution pipelines, aggregation logic, and data rollups driving risk scores.

As a…

As a data analyst

I want to…

I want to understand aggregation

So that…

So that dashboards reflect accurate governance metrics

Acceptance Criteria

- [ ] Execution model runs rule evaluations, merges probe/check results, and calculates composite scores.
- [ ] Aggregation outputs feed reporting, thresholds, and trend analysis for stakeholders.

Story: Governance API Surface

Summary

Detail endpoints, events, and integration contracts exposed by the engine.

As a…

As an integrator

I want to…

I want governance interface specs

So that…

So that other services can submit signals or consume outcomes

Acceptance Criteria

- [ ] API surface includes REST endpoints, event topics, and subscription expectations for consumers.
- [ ] Contracts define payload schemas, authorization requirements, and retry semantics.

Story: Governance Configuration & Dependencies

Summary

Summarize configuration files, external dependencies, and cross-system contracts.

As a…

As a platform engineer

I want to…

I want configuration clarity

So that…

So that deployments align with shared contracts

Acceptance Criteria

- [ ] Configuration covers rule registries, weighting presets, queue connections, and dependency versions.
- [ ] Cross-system contracts explain expectations with probes, checks, notifications, and reporting.

Story: Governance Extensibility Hooks

Summary

Explain how to add new rules, signals, or scoring plugins safely.

As a…

As an extender

I want to…

I want governance hook guidance

So that…

So that I can innovate without breaking core flows

Acceptance Criteria

- [ ] Hooks describe plugin interfaces, validation steps, and rollout procedures for new signal types.
- [ ] Safeguards include feature flags, canary evaluations, and documentation updates.

Story: Governance Frontend Structure

Summary

Identify UI directories for governance dashboards, rule editors, and escalation consoles.

As a…

As a frontend developer

I want to…

I want to locate governance UI code

So that…

So that I can evolve dashboards

Acceptance Criteria

- [ ] Structure references client/src/features/governance pages, components, and hooks.
- [ ] Shared state integrates with tasks, controls, and notifications for unified operator experiences.

Story: Governance UI Flows

Summary

Describe dashboards, rule editors, investigation panels, and reporting interactions.

As a…

As a product owner

I want to…

I want comprehensive governance UX

So that…

So that teams can understand risk posture

Acceptance Criteria

- [ ] UI flows include scorecards, drill-downs, rule builders, and escalation actions with RBAC enforcement.
- [ ] Components share filters, context panels, and evidence links for cross-system context.

Story: Governance State & Security

Summary

Summarize state management, data fetching, and security considerations for governance views.

As a…

As a security engineer

I want to…

I want governance data protections

So that…

So that sensitive signals remain scoped

Acceptance Criteria

- [ ] State patterns leverage normalized caches, streaming updates, and fallback polling with tenant scoping.
- [ ] Security covers permission checks, masking sensitive fields, and securing WebSocket subscriptions.

Story: Governance Operations

Summary

Provide monitoring, alerting, scaling, and maintenance guidance for the engine.

As a…

As an SRE

I want to…

I want governance operational runbooks

So that…

So that the engine remains healthy

Acceptance Criteria

- [ ] Operations detail metrics for rule execution, queue depth, and backlog with escalation paths.
- [ ] Maintenance tasks include dependency patching, configuration reviews, and incident response steps.

Story: Governance Implementation Checklist

Summary

List steps to deploy, validate, and document governance engine updates.

As a…

As a release manager

I want to…

I want a governance rollout checklist

So that…

So that new capabilities ship safely

Acceptance Criteria

- [ ] Checklist covers environment promotion, feature flag gating, smoke tests, and documentation updates.
- [ ] Verification includes monitoring dashboards, alert tests, and stakeholder communication.

Task Management System

Location: /server/src/modules/tasks

TL;DR

The task management system operationalizes remediation by translating failed governance checks into assignable, auditable work.

Implemented in server/src/modules/tasks, it exposes REST APIs, lifecycle engines, SLA automation, and integrations that keep remediation accountable across the platform.

This guide captures every engineering detail needed to implement and operate the feature in alignment with the broader governance architecture.

System Overview & Alignment

The task management system sits in step 7 of the governance lifecycle: failed checks and controls automatically create remediation tasks with owners, deadlines, escalation paths, and evidence requirements, and successful completion retriggers control validation to confirm resolution.■F:docs/01-about/03-concept-summary.md†L282-L299■ It must uphold platform objectives around continuous compliance, measurable outcomes, and cross-functional collaboration while scaling with enterprise governance operations.■F:docs/01-about/03-concept-summary.md†L301-L335■■F:docs/01-about/08-operations-and-teams.md†L9-L101■

Key responsibilities:

- Provide a system of record for remediation work across internal and external teams, ensuring accountability to SLA targets outlined in operational objectives.■F:docs/01-about/08-operations-and-teams.md†L9-L169■
- Enforce automation-first workflows that integrate with governance, evidence, notification, audit, and dashboard systems to maintain traceability and audit readiness.■F:docs/02-technical-specifications/01-system-architecture.md†L9-L133■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■

- Support JavaScript-only implementation across backend (Express.js) and frontend (React + Vite) layers, aligning with platform technology standards.■F:docs/02-technical-specifications/01-system-architecture.md†L1-L83■

Backend Specification

Backend Location & Directory Layout

Task orchestration lives in server/src/modules/tasks, following the feature-first Express.js structure used throughout the backend.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L55-L182■

```
server/src/modules/tasks/
■■■ controllers/
■   ■■■ tasks.controller.js
■   ■■■ assignments.controller.js
■   ■■■ integrations.controller.js
■■■ services/
■   ■■■ task.service.js
■   ■■■ lifecycle.service.js
■   ■■■ escalation.service.js
■   ■■■ evidence-sync.service.js
■■■ repositories/
■   ■■■ task.repository.js
■   ■■■ task-assignment.repository.js
■   ■■■ task-metric.repository.js
■■■ policies/
■   ■■■ tasks.policy.js
■■■ workflows/
■   ■■■ sla.scheduler.js
■   ■■■ verification.queue.js
■   ■■■ sync.processor.js
■■■ integrations/
■   ■■■ jira.adapter.js
■   ■■■ servicenow.adapter.js
■■■ events/
    ■■■ task.created.js
    ■■■ task.updated.js
    ■■■ task.closed.js
```

Service Responsibilities & API Contracts

Services enforce clear separation of concerns:

- task.service.js handles creation, updates, status transitions, and read models.

- lifecycle.service.js encapsulates state machine rules (draft → open → in-progress → awaiting evidence → pending verification → resolved → closed) and validation when reopening or rolling back states.
- escalation.service.js calculates SLA breaches, updates escalation tiers, and dispatches notifications.
- evidence-sync.service.js validates linked artifacts against the Evidence Management system before verification can succeed.■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L118■

Controller endpoints conform to REST contracts documented in the backend specification and surfaced via OpenAPI. All controllers are implemented in JavaScript (no TypeScript) and reuse shared middleware for authentication, authorization, validation, and logging.■F:docs/02-technical-specifications/01-system-architecture.md†L9-L83■■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L136-L205■

Method	*Endpoint*	*Description*	*Request Shape (JSON)*	*Response Shape*	
POST	/api/v1/tasks	Create remediation task from governance trigger, mapping control/check/context metadata.	{ title, description, priority, source, controlId, checkId, frameworkId, slaDueAt, assigneeId/teamId, tags, evidenceHints[] }	{ status, message, data: { task } }	
GET	/api/v1/tasks	List/filter tasks by status, severity, framework, owner, SLA state, escalation tier.	Query params status, priority, frameworkId, assigneeId, escalationLevel, search	{ status, message, data: { tasks, pagination, aggregates } }	
GET	/api/v1/tasks/:id	Retrieve task detail, lifecycle events, evidence links, external sync state.	N/A	{ status, message, data: { task, timeline, metrics } }	
PATCH	/api/v1/tasks/:id/status	Transition task state; enforces lifecycle rules and evidence prerequisites.	{ status, comment, evidenceIds[], reopenReason }	{ status, message, data: { task } }	

Method	Endpoint	Description	Request Shape (JSON)	Response Shape	
POST	/api/v1/tasks/:id/reassign	Reassign or delegate to user/team, capturing expiry and audit context.	{ assigneeId, teamId, delegationExpiresAt, justification }	{ status, message, data: { task, assignments } }	
POST	/api/v1/tasks/:id/escalate	Manually escalate with reason (auto-escalations handled by scheduler).	{ escalationLevel, reason }	{ status, message, data: { task } }	
POST	/api/v1/tasks/:id/evidence	Attach or detach evidence references after validation.	`{ action: "add"	"remove", evidenceIds[] }`	{ status, message, data: { task, evidenceLinks } }	
POST	/api/v1/tasks/:id/sync	Force external issue sync (Jira/ServiceNow) and return status.	{ provider, forceRefresh }	{ status, message, data: { syncState } }	

All responses follow the unified JSON envelope (status, message, data, error) defined for the API server.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L182-L205■

Lifecycle Orchestration & Automation

Lifecycle automation ties remediation to platform events and SLAs:

- **Creation Triggers:** Governance engine emits task.created events when checks fail or manual remediation is requested. Manual creation remains available to privileged users for ad-hoc remediation.■F:docs/03-systems/12-governance-engine/readme.md†L29-L94■
- **State Machine Enforcement:** lifecycle.service.js ensures transitions respect business rules (e.g., cannot close without verified evidence, reopen requires justification and resets verification tasks). Status transitions emit domain events and append audit log entries captured by the monitoring system.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■
- **SLA Automation:** sla.scheduler.js runs on a configurable interval (default hourly) via the platform job runner, evaluates slaDueAt against current timestamps, and triggers escalations (level 1: notify owner, level 2: notify team leads, level 3: leadership and compliance) aligned with enterprise SLA policies.■F:docs/01-about/08-operations-and-teams.md†L162-L194■■F:docs/01-about/10-risk-management-and-Mitigation.md†L116-L138■
- **Verification Queue:** Tasks entering pending verification enqueue a job in verification.queue.js to request approvals from control owners or compliance reviewers. Completion automatically requests

governance re-validation to maintain continuous compliance.■F:docs/01-about/03-concept-summary.md†L282-L299■■F:docs/03-systems/12-governance-engine/readme.md†L33-L115■

Integrations & Messaging

Task workflows rely on internal and external integrations:

- **Notification System:** Escalation, assignment, and status change events post structured payloads to the notification service for email, Slack, or PagerDuty distribution.■F:docs/03-systems/04-notification-system/readme.md†L7-L200■
- **Evidence Management:** Evidence attachments and verification require validation against the Evidence Management API before the lifecycle can progress, ensuring artifacts remain trustworthy.■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L159■
- **Governance Engine:** Completed tasks trigger governance re-runs to update control and framework scores, closing the remediation loop.■F:docs/03-systems/12-governance-engine/readme.md†L31-L118■
- **Dashboard & Reporting:** SLA and throughput metrics feed the dashboard system to surface remediation posture in executive views.■F:docs/03-systems/14-dashboard-and-reporting-system/readme.md†L7-L118■
- **External Trackers:** Jira and ServiceNow adapters synchronize titles, statuses, comments, and assignees with third-party systems, using webhooks or scheduled syncs to avoid drift.■F:docs/02-technical-specifications/07-integration-architecture.md†L98-L134■ Field mappings and conflict resolution logic (platform source-of-truth unless overridden by admin) are configurable per environment.

Message flow is implemented via lightweight event emitters within the Express app; events are persisted to task_events and forwarded to integrations through background workers to avoid blocking API latency.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L153-L205■

Security, Compliance & Observability

Security controls follow platform-wide policies:

- **Authentication & Authorization:** All routes are protected by JWT middleware and Casbin policies defined in tasks.policy.js. Delegation workflows respect break-glass requirements and enforce time-bound access, emitting alerts for emergency overrides.■F:docs/01-about/04-security-and-data-protection.md†L243-L337■
- **Audit Logging:** Every change (status updates, reassignments, evidence edits) writes immutable entries to the audit logging system with actor, timestamp, payload diff, and origin metadata.■F:docs/01-about/04-security-and-data-protection.md†L261-L337■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■

- **Data Protection:** Task descriptions and attachments must remain free of sensitive model artifacts unless explicitly tagged; retention follows governance policies (default 36 months) with archival handled via scheduled jobs akin to evidence retention.■F:docs/01-about/04-security-and-data-protection.md†L303-L337■■F:docs/02-technical-specifications/04-database-design.md†L80-L147■
- **Observability:** Structured logs (Winston + Morgan) emit lifecycle events, and metrics (SLA breaches, reopen rate, verification latency) publish to monitoring dashboards to satisfy operational KPIs.■F:docs/02-technical-specifications/01-system-architecture.md†L34-L132■■F:docs/01-about/08-operations-and-teams.md†L118-L194■ Error budgets align with risk management thresholds (e.g., <2 SLA breaches per quarter).■F:docs/01-about/10-risk-management-and-Mitigation.md†L116-L138■

Frontend Specification

Frontend Location & Directory Layout

Remediation UI lives under client/src/features/tasks, mirroring backend boundaries and enabling feature-focused development in JavaScript (no TypeScript).■F:docs/02-technical-specifications/03-frontend-architecture.md†L9-L82■

```
client/src/features/tasks/
■■■ pages/
■   ■■■ task-inbox-page.jsx
■   ■■■ task-detail-page.jsx
■   ■■■ task-board-page.jsx
■   ■■■ sla-dashboard-page.jsx
■■■ components/
■   ■■■ task-form.jsx
■   ■■■ task-timeline.jsx
■   ■■■ evidence-attachment-list.jsx
■   ■■■ escalation-banner.jsx
■   ■■■ external-sync-status.jsx
■■■ hooks/
■   ■■■ use-task-inbox.js
■   ■■■ use-task-detail.js
■   ■■■ use-sla-metrics.js
■   ■■■ use-task-mutations.js
■■■ api/
    ■■■ tasks-client.js
```

Shared governance components such as TaskControlPanel.jsx reside in client/src/components/governance for reuse across dashboards.■F:docs/02-technical-specifications/03-frontend-architecture.md†L32-L118■

State Management, Components & UI Flows

The frontend follows the platform's React architecture (Vite, React Router, Tailwind, shadcn). Data fetching uses Axios clients wrapped in feature-specific hooks; React Context supplies auth tokens, theming, and notification preferences.■F:docs/02-technical-specifications/03-frontend-architecture.md†L9-L140■

Key flows:

- **Task Intake:** task-form.jsx renders dynamic fields (control/check lookups, SLA pickers) and leverages validation helpers shared via shared/validation. Automation auto-populates context when invoked from governance alerts; manual users can attach initial evidence placeholders. Submission posts to /api/v1/tasks and on success routes to task-detail-page.jsx.
- **Inbox & Board Views:** task-inbox-page.jsx supports filters (status, framework, owner, SLA state). task-board-page.jsx renders a Kanban board aligned with lifecycle statuses using drag-and-drop interactions; dropping triggers status mutations that respect lifecycle rules (confirmation modals for cross-tier moves). Aggregates (counts, breach totals) update via use-sla-metrics.js.
- **Detail & Timeline:** task-detail-page.jsx composes task-timeline.jsx, evidence-attachment-list.jsx, and external-sync-status.jsx to show history, linked evidence, and external ticket state. Verification actions and delegation modals check Casbin permissions returned by the backend before enabling UI controls.
- **Escalation Awareness:** escalation-banner.jsx surfaces SLA urgency (time remaining, current level) with quick actions to reassign, escalate, or add evidence. The component subscribes to notification context to display realtime alerts from WebSocket or polling channels configured globally.■F:docs/03-systems/04-notification-system/readme.md†L7-L170■
- **Accessibility & Security:** Components follow WCAG AA theming, support keyboard interactions, and ensure sensitive data is masked according to security guidance (e.g., redacting personal data in comments). All API mutations include JWT headers from auth context and refresh tokens when 401 responses occur.■F:docs/02-technical-specifications/03-frontend-architecture.md†L108-L160■■F:docs/01-about/04-security-and-data-protection.md†L243-L337■

UI tests should cover lifecycle transitions, evidence attachment flows, and escalation banners to protect core remediation paths aligned with MVP roadmap milestones for automated remediation and SLA tracking.■F:docs/01-about/06-mvp-and-roadmap.md†L180-L210■

Schema Specification

Primary tables (PostgreSQL via Prisma) capture lifecycle state, metrics, and integrations. Column names use snake_case per database conventions; Prisma schema definitions live in

server/prisma/schema.prisma.

- **tasks** – id (uuid PK), title, description, priority, status, source, control_id, check_id, framework_id, created_by, assignee_id, team_id, delegation_expires_at, sla_due_at, escalation_level, external_issue_key, external_provider, verification_required, verification_completed_at, created_at, updated_at, resolved_at, closed_at.
- **task_events** – Append-only ledger with id, task_id, event_type, payload, actor_id, actor_type, created_at, origin, providing integration with audit logging and dashboards.■F:docs/02-technical-specifications/04-database-design.md†L80-L147■
- **task_assignments** – Ownership history tracking task_id, assignee_id, team_id, delegated_by, delegation_expires_at, justification, created_at, revoked_at.
- **task_evidence_links** – Bridge table linking evidence with verification context task_id, evidence_id, link_type, reviewer_id, verification_status, verified_at.
- **task_sla_metrics** – Materialized metrics for dashboards: task_id, time_to_acknowledge, time_in_status, time_to_close, breach_count, last_breach_at.

Foreign keys enforce integrity with governance entities (controls, checks, frameworks, evidence), and indexes on status, assignee_id, sla_due_at, and escalation_level support high-volume queries. Soft deletes are avoided; instead, closure is recorded via timestamps to maintain audit history.■F:docs/02-technical-specifications/04-database-design.md†L80-L147■

Operational Playbooks & Runbooks

Operations teams rely on structured runbooks to maintain reliability and compliance:

- **Daily Monitoring:** Review SLA dashboards for breaches, verify escalation notifications reached recipients, and inspect synchronization logs for external trackers. Any repeated failure triggers incident management per risk governance policies.■F:docs/01-about/10-risk-management-and-Mitigation.md†L70-L138■■F:docs/03-systems/04-notification-system/readme.md†L7-L200■
- **Weekly Audits:** Sample closed tasks to confirm evidence sufficiency and governance re-validation outcomes; ensure audit logs contain complete histories and no unauthorized delegation occurred.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L139■
- **Release Checklist:** Before deploying changes, run integration tests for lifecycle transitions, verify Prisma migrations for schema changes, and execute end-to-end smoke tests connecting governance engine triggers, task creation, and dashboard updates.■F:docs/02-technical-specifications/08-deployment-and-environment-guide.md†L100-L205■■F:docs/01-about/08-operations-and-teams.md†L118-L194■

- **Disaster Recovery:** Leverage platform-wide DR plans (RPO < 4 hours, RTO < 24 hours) to restore task data and external sync states. Post-recovery, reconcile with external trackers to prevent duplicate or missed remediation records.■F:docs/01-about/08-operations-and-teams.md†L166-L194■■F:docs/02-technical-specifications/05-devops-infrastructure.md†L200-L214■

Related Documentation

- — Triggers remediation workflows and re-validates controls.
- — Validates supporting artifacts before closure.
- — Manages alerts for creation, escalation, and verification.
- — Captures immutable task histories and operational metrics.
- — Surfaces SLA metrics, backlog, and remediation trends.

Story: Task Backend Layout

Summary

Document controllers, services, queues, and workers powering task management.

As a…

As a backend developer

I want to…

I want to find task service code

So that…

So that I can enhance assignment workflows

Acceptance Criteria

- [ ] Layout covers task controllers, services, repositories, queue processors, and integration modules.
- [ ] Notes highlight dependencies on governance, notifications, probes, and evidence events.

Story: Task Service Responsibilities

Summary

Summarize endpoints, SLAs, templates, and automation handled by the task service.

As a…

As a product manager

I want to…

I want clarity on task responsibilities

So that…

So that teams rely on timely remediation

Acceptance Criteria

- [ ] Responsibilities include assignment creation, escalations, approvals, and SLA tracking with notifications.
- [ ] API contracts cover creation, reassignment, commenting, and completion events.

Story: Task Lifecycle Orchestration

Summary

Describe creation, assignment, escalation, resolution, and archival workflows.

As a…

As an operations lead

I want to…

I want end-to-end task lifecycle insight

So that…

So that nothing slips through

Acceptance Criteria

- [ ] Lifecycle stages cover triage, ownership, SLA monitoring, escalations, and closure reviews.
- [ ] Automation hooks sync with notifications, RBAC, and analytics for accountability.

Story: Task Integrations & Messaging

Summary

Explain messaging patterns with governance engine, notifications, RBAC, and external systems.

As a…

As an integration architect

I want to…

I want task messaging clarity

So that…

So that tasks remain synchronized

Acceptance Criteria

- [ ] Integrations publish events for creation, updates, and SLA breaches while consuming governance triggers.
- [ ] External connectors map to ServiceNow, Jira, and partner APIs with auditing.

Story: Task Security & Observability

Summary

Summarize access controls, audit logging, metrics, and alerts for the task platform.

As a…

As a security engineer

I want to…

I want task safeguards

So that…

So that remediation data stays compliant

Acceptance Criteria

- [ ] Security ensures RBAC enforcement, encryption, and audit trails for every update.
- [ ] Observability includes metrics for backlog, SLA breaches, and worker health with alert thresholds.

Story: Task Frontend Structure

Summary

Map UI directories for inboxes, detail views, and automation settings.

As a…

As a frontend engineer

I want to…

I want to navigate task UI code

So that…

So that I can extend assignment experiences

Acceptance Criteria

- [ ] Structure references client/src/features/tasks pages, components, and state management.
- [ ] Shared components include assignment lists, SLA badges, and comment threads.

Story: Task UI Flows

Summary

Describe inbox, Kanban, detail, and automation configuration flows.

As a…

As a UX lead

I want to…

I want cohesive task interactions

So that…

So that responders manage workloads effectively

Acceptance Criteria

- [ ] UI flows enable filtering, drag-and-drop prioritization, inline updates, and escalation controls.
- [ ] State management aligns with notifications, governance signals, and evidence attachments.

Dashboard and Reporting System

Location: /client/src/features/dashboards, /server/src/modules/reports

TL;DR

The dashboard and reporting system transforms compliance telemetry—framework scores, control status, remediation progress, and evidence freshness—into actionable insights and exports.

React dashboards in client/src/features/dashboards consume APIs backed by server/src/modules/reports to visualize control health, remediation execution, and evidence coverage.

This runbook describes the data pipelines, UI components, report generators, and extensibility patterns that keep analytics current and audit-ready.

Objectives & Alignment

The dashboards and exports operationalize the platform vision of making AI governance measurable, continuous, and trusted by translating check and control outcomes into decision support for compliance, engineering, and audit stakeholders.■F:docs/01-about/01-project-overview.md†L29-L133■ Dashboards must surface framework-aligned scores, remediation actions, and evidence status so organizations can prove adherence to global standards and act on emerging risks in near real time.■F:docs/01-about/01-project-overview.md†L63-L83■

Technology Stack & Dependencies

The reporting module is implemented entirely in JavaScript, sharing the platform's Express.js backend, React.js frontend, Prisma ORM, and Axios HTTP integrations while persisting to externally hosted PostgreSQL and MinIO services.■F:docs/02-technical-specifications/01-system-architecture.md†L41-L198■ These components are mandated by the system architecture to ensure consistency, security, and scalability across modules. Reporting controllers must follow the backend architectural conventions (module-based Express structure, Swagger-documented REST APIs), and UI work must comply with the Vite-powered feature folder

organization, Tailwind/shadcn design system, and Atomic Design principles defined for the client codebase.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L43-L217■■F:docs/02-technical-specifications/03-frontend-architecture.md†L39-L175■

Backend Specification

Backend Location & Directory Layout

Reporting services live in server/src/modules/reports, exposing REST endpoints and BullMQ workers that aggregate governance data and produce exports.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L103-L135■

```
server/src/modules/reports/
■■■ controllers/
■   ■■■ dashboards.controller.ts
■   ■■■ exports.controller.ts
■■■ services/
■   ■■■ score-aggregator.service.ts
■   ■■■ remediation-metrics.service.ts
■   ■■■ evidence-metrics.service.ts
■■■ workers/
■   ■■■ score-aggregator.worker.ts
■   ■■■ remediation.worker.ts
■   ■■■ export.worker.ts
■■■ repositories/
■   ■■■ scores.repository.ts
■   ■■■ metrics.repository.ts
■   ■■■ exports.repository.ts
■■■ templates/
    ■■■ export-templates/
```

Services & Responsibilities

- **dashboards.controller.ts:** Serves read-only endpoints that join governance scores, remediation metrics, and evidence data through dedicated services. Responses must follow the platform's standardized JSON envelope and expose pagination/filtering consistent with other modules.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L196-L217■
- **exports.controller.ts:** Handles export job CRUD, scheduling, and artifact retrieval while validating RBAC scopes before enqueuing background work.■F:docs/03-systems/02-rbac-system/readme.md†L83-L166■
- **Aggregators (score-aggregator.service.ts, remediation-metrics.service.ts, evidence-metrics.service.ts):** Query governance, task, and evidence repositories using Prisma to assemble dimensioned datasets for dashboards while caching hot reads per framework/control to honor database performance guidance.■F:docs/02-technical-specifications/04-database-design.md†L40-L140■

- **Repositories:** Encapsulate SQL-friendly projections for reports, ensure index usage aligns with database optimization practices, and hide raw Prisma queries behind reusable functions.■F:docs/02-technical-specifications/04-database-design.md†L108-L140■
- **Templates & Serializers:** Keep export templates versioned, localized, and extensible; serializers must normalize units, currency, and timestamps so downstream consumers receive compliant artefacts.■F:docs/01-about/01-project-overview.md†L63-L83■

API Surface

- GET /api/v1/reports/dashboards/framework-scores — Returns framework posture with historical trend slices and optional filtering by framework, domain, or time window.
- GET /api/v1/reports/dashboards/control-health — Aggregates failing controls, SLA state, ownership, and risk tier for heatmaps.
- GET /api/v1/reports/dashboards/remediation — Provides task throughput, escalation counts, and SLA adherence metrics sourced from the Task Management module.■F:docs/03-systems/13-task-management-system/readme.md†L41-L120■
- GET /api/v1/reports/dashboards/evidence — Summarizes freshness, retention windows, and coverage gaps using Evidence Management metadata.■F:docs/03-systems/11-evidence-management-system/readme.md†L49-L152■
- POST /api/v1/reports/exports — Creates export jobs with filters, format, and schedule details, validating RBAC policies and Casbin domains before enqueueing.
- GET /api/v1/reports/exports/:id — Retrieves job status and signed artifact URLs from MinIO once processing completes.■F:docs/02-technical-specifications/01-system-architecture.md†L168-L181■
- POST /api/v1/reports/exports/:id/retry — Re-queues failed jobs with idempotency safeguards and audit logging hooks.■F:docs/02-technical-specifications/06-security-implementation.md†L124-L144■

Workers & Scheduling

- BullMQ queues (score-aggregator.worker.ts, remediation.worker.ts, export.worker.ts) subscribe to governance engine broadcasts, task updates, and evidence changes to keep analytics fresh. Jobs must be idempotent, respect tenant isolation, and emit audit events for traceability.■F:docs/03-systems/12-governance-engine/readme.md†L49-L105■■F:docs/03-systems/13-task-management-system/readme.md†L41-L168■■F:docs/03-systems/11-evidence-management-system/readme.md†L41-L158■
- Schedulers align with framework cadence rules and SLA thresholds defined by Governance and Task systems; configure repeatable jobs for daily score recalculation, weekly remediation digests, and ad hoc export windows while enforcing concurrency limits to protect shared infrastructure.■F:docs/03-systems/10-framework-mapping-system/readme.md†L49-L160■■F:docs/02-technical-specifications/04-database-design.md†L54-L119■

- Queue health metrics feed observability dashboards; dead-letter queues trigger alerting playbooks coordinated with the Notification system for timely remediation.■F:docs/03-systems/04-notification-system/readme.md†L39-L170■

Data Pipelines

- **Framework Scores:** Governance engine emits check/control updates; BullMQ workers normalize payloads using Framework mappings and persist scores for dashboard consumption.■F:docs/03-systems/12-governance-engine/readme.md†L7-L205■■F:docs/03-systems/10-framework-mapping-system/readme.md†L7-L210■
- **Control Status:** Aggregates check results, remediation state, and SLA data from Task Management to render risk indicators and heatmaps.■F:docs/03-systems/13-task-management-system/readme.md†L7-L214■
- **Remediation Metrics:** Tracks task lifecycle, escalation counts, and resolution timing to feed SLA dashboards and compliance reports.
- **Evidence Snapshot:** Pulls metadata from Evidence Management for freshness, retention, and coverage analytics across controls/frameworks.■F:docs/03-systems/11-evidence-management-system/readme.md†L7-L170■

Report Generation

- **Framework Attestation Packs:** Export controller assembles framework summaries, control coverage, and evidence pointers for auditors.
- **Control Breakdown Reports:** Provide per-control status, failing checks, remediation assignments, and evidence links.
- **Remediation & Evidence Digest:** Weekly digest summarizing open tasks, SLA breaches, and evidence gaps for stakeholders.
- Exports support CSV/JSON/XLSX formats, versioned artifacts, localization, and API scheduling via export.worker.ts.

Security & Access Control

- Apply JWT authentication and Casbin RBAC guards to every endpoint, ensuring dashboards respect tenant domains and role-based scopes (Compliance Officer, Auditor, Executive).■F:docs/02-technical-specifications/06-security-implementation.md†L56-L97■■F:docs/03-systems/02-rbac-system/readme.md†L83-L166■
- Enforce least-privilege access: only Admins and Compliance Officers can configure exports; Auditors receive read-only dashboard endpoints; system service accounts manage scheduled jobs. Authorization decisions must be logged to report_audit_log for forensic review.■F:docs/02-technical-specifications/06-security-implementation.md†L124-L144■■F:docs/03-systems/02-rbac-system/readme.md†L107-L166■

- All exports produced in MinIO require presigned URLs with short-lived expiry and AES-256 encryption; checksum validation and immutable audit entries satisfy regulatory requirements.■F:docs/02-technical-specifications/01-system-architecture.md†L168-L181■■F:docs/02-technical-specifications/06-security-implementation.md†L100-L144■

Testing & Observability

- Unit-test controllers, services, and repositories using Jest with fixtures mirroring framework/task/evidence scenarios; target ≥85% coverage as mandated by QA standards.■F:docs/02-technical-specifications/09-testing-and-qa.md†L31-L120■
- Integration tests should exercise REST endpoints via Postman/Newman collections and Vitest-driven contract checks to ensure frontend hooks align with API responses.■F:docs/02-technical-specifications/09-testing-and-qa.md†L94-L154■
- Instrument workers and controllers with structured Winston logs, performance metrics, and alert hooks feeding the centralized monitoring stack to catch latency regressions or queue backlogs promptly.■F:docs/02-technical-specifications/06-security-implementation.md†L134-L181■■F:docs/02-technical-specifications/01-system-architecture.md†L183-L199■

Frontend Specification

Frontend Location & Directory Layout

Dashboards live in client/src/features/dashboards, rendering compliance telemetry with charting and table components.■F:docs/02-technical-specifications/03-frontend-architecture.md†L50-L160■

```
client/src/features/dashboards/
■■■ pages/
■   ■■■ FrameworkDashboardPage.tsx
■   ■■■ ControlHealthPage.tsx
■   ■■■ RemediationDashboardPage.tsx
■   ■■■ EvidenceCoveragePage.tsx
■■■ components/
■   ■■■ ScoreGauge.tsx
■   ■■■ ControlHeatmap.tsx
■   ■■■ RemediationTrendChart.tsx
■   ■■■ EvidenceFreshnessTable.tsx
■■■ hooks/
■   ■■■ useFrameworkScores.ts
■   ■■■ useControlMetrics.ts
■   ■■■ useRemediationMetrics.ts
■   ■■■ useEvidenceMetrics.ts
■■■ api/
    ■■■ reportsClient.ts

client/src/components/reports/
■■■ ExportSchedulerModal.tsx
```

Data Fetching & State Management

- Hooks (useFrameworkScores, useControlMetrics, useRemediationMetrics, useEvidenceMetrics) must encapsulate Axios data access, leverage the shared API client interceptors, and integrate with Auth Context for JWT propagation as documented in the frontend architecture.■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L140■
- Minimize redundant requests by memoizing responses within feature hooks and reusing the documented API interceptors/polling cadence, normalizing outputs to shared chart/table props so UI composition stays declarative.■F:docs/02-technical-specifications/03-frontend-architecture.md†L39-L140■
- Persist user-selected filters (framework, domain, severity, time range) via URL search params or local storage consistent with feature-folder conventions so dashboards remain shareable and resumable across sessions.■F:docs/02-technical-specifications/03-frontend-architecture.md†L50-L103■

Reusable Components & UI Flows

- **Framework Dashboard:** ScoreGauge and useFrameworkScores display overall posture and trends, enabling drill-down into domains and frameworks.
- **Control Health:** ControlHeatmap surfaces failing controls by domain/owner with filters for severity and risk tier.
- **Remediation Monitoring:** RemediationTrendChart and SLA tables visualize task throughput, escalations, and overdue work.
- **Evidence Coverage:** EvidenceFreshnessTable highlights artifacts nearing expiration, missing evidence, and retention status.
- **Export Scheduling:** ExportSchedulerModal allows users to configure recurring exports, formats, recipients, and localization options.

Access Control & Routing

- Wrap dashboard routes with RequirePermission guards so only authorized roles can view sensitive metrics, aligning with RBAC policies and frontend guard patterns established in the admin module.■F:docs/03-systems/02-rbac-system/readme.md†L122-L170■■F:docs/02-technical-specifications/03-frontend-architecture.md†L96-L175■
- Ensure navigation integrates with global layouts and breadcrumb patterns, exposing contextual links back to Governance, Tasks, and Evidence features to maintain consistent user journeys defined across systems.■F:docs/03-systems/12-governance-engine/readme.md†L94-L118■■F:docs/03-systems/13-task-management-system/readme.md†L93-L152■

- Respect security measures (CSP, CSRF mitigation, session timeout) and accessibility requirements (WCAG AA) when composing charts, tables, and export dialogs.■F:docs/02-technical-specifications/03-frontend-architecture.md†L145-L175■

Schema Specification

- **report_scores:** Aggregated framework/control scores with timestamps, dimensions (framework, domain, control), and metadata for charting.
- **report_metrics:** Remediation and evidence KPIs (open tasks, SLA breaches, evidence freshness, retention stats).
- **report_exports:** Export job definitions, formats, filters, scheduling metadata, and artifact URIs.
- **report_audit_log:** Records generation events, consumers, and checksum validation for audit readiness.
- **report_widgets:** Optional configuration table for enabling/disabling dashboard widgets per tenant.
- Tables inherit standard PostgreSQL practices: UUID primary keys, foreign keys to governance/task/evidence tables, time-based partitioning for audit trails, and indexes optimized for high-read dashboard queries.■F:docs/02-technical-specifications/04-database-design.md†L54-L140■
- Prisma schema definitions must align with shared naming conventions, enforce soft deletes where appropriate, and register migrations through the managed change queue controlling the externally hosted database.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L169-L217■
- Exports reference MinIO object metadata and retention policies to honor backup/archival requirements and compliance retention windows documented for evidence and audit data.■F:docs/02-technical-specifications/04-database-design.md†L148-L184■

Operational Playbooks & References

Playbooks

- **Pipeline Health:** Monitor worker queues, ingestion latency, and dashboard API performance; replay backlog on failure.
- **Export Validation:** Verify generated artifacts after schema changes; ensure localization and retention metadata remain accurate.
- **Accessibility & Performance:** Adhere to WCAG guidelines, optimize chart rendering, and enforce caching strategies for large datasets.

- **Security Reviews:** Validate Casbin policy coverage, JWT expiry handling, and encryption settings after changes; re-run vulnerability scans if dependencies or templates shift.■F:docs/02-technical-specifications/06-security-implementation.md†L56-L198■
- **Testing Cadence:** Enforce automated Jest/Vitest suites and Cypress dashboard flows in CI, reviewing coverage and regression reports before release as defined by QA governance.■F:docs/02-technical-specifications/09-testing-and-qa.md†L31-L170■

Related Documentation

- — source of scoring events.
- — remediation metrics.
- — evidence coverage inputs.
- — mapping metadata for reporting.
- — permission enforcement for dashboards and exports.
- — alerting pathways for export and queue incidents.

Story: Reporting Backend Layout

Summary

Document services, workers, and pipelines responsible for dashboards and scheduled reports.

As a…

As a backend engineer

I want to…

I want to locate reporting code

So that…

So that I can enhance analytics delivery

Acceptance Criteria

- [ ] Layout covers services handling dashboards, API endpoints, workers, and data pipeline integrations.
- [ ] Notes explain how reporting interfaces with governance, evidence, and control systems.

Story: Reporting Services & Responsibilities

Summary

Summarize analytics generation, metric aggregation, and visualization responsibilities.

As a…

As a product strategist

I want to…

I want reporting service clarity

So that…

So that stakeholders trust platform insights

Acceptance Criteria

- [ ] Services compute KPIs, compile dashboards, and coordinate with data warehouses for analytics.
- [ ] Responsibilities include managing caching, role-based visibility, and drill-down capabilities.

Story: Reporting API Surface

Summary

Detail endpoints delivering dashboard data, exports, and embedded widgets.

As a…

As an API consumer

I want to…

I want reporting endpoint specs

So that…

So that I can embed metrics in external tools

Acceptance Criteria

- [ ] API surface describes REST routes, filters, and export options with SLA expectations.
- [ ] Contracts enforce RBAC scopes, caching headers, and pagination for large datasets.

Story: Reporting Workers & Scheduling

Summary

Explain scheduled jobs, background workers, and cron strategies generating reports.

As a…

As an operations engineer

I want to…

I want scheduling clarity

So that…

So that reports run on time

Acceptance Criteria

- [ ] Workers schedule recurring reports, data refreshes, and anomaly detection tasks.
- [ ] Scheduling strategies handle retries, prioritization, and multi-region coordination.

Story: Reporting Data Pipelines

Summary

Describe data ingestion, transformation, and storage powering dashboards.

As a…

As a data engineer

I want to…

I want pipeline visibility

So that…

So that analytics remain accurate

Acceptance Criteria

- [ ] Pipelines ingest governance, task, evidence, and integration data into curated marts.
- [ ] Processes include ETL steps, schema validations, and data quality monitoring.

Story: Report Generation

Summary

Summarize how reports are templated, localized, and delivered to stakeholders.

As a…

As a communications lead

I want to…

I want report generation clarity

So that…

So that scheduled reports meet branding and compliance

Acceptance Criteria

- [ ] Generation process renders templates, applies localization, and packages outputs with visual assets.
- [ ] Delivery includes email, portal downloads, and partner webhooks with audit logging.

Story: Reporting Security & Access Control

Summary

Explain role-based access, data redaction, and secure delivery of analytics content.

As a…

As a security officer

I want to…

I want reporting safeguards

So that…

So that sensitive metrics remain scoped

Acceptance Criteria

- [ ] Security enforces RBAC, row-level filters, and watermarking for exported content.
- [ ] Access policies integrate with session management, audit logging, and anomaly detection.

Story: Reporting Testing & Observability

Summary

Detail testing suites, performance checks, and monitoring for reporting workloads.

As a…

As a QA manager

I want to…

I want reporting quality gates

So that…

So that dashboards and exports remain reliable

Acceptance Criteria

- [ ] Testing includes unit, integration, visual regression, and load tests for dashboard APIs.
- [ ] Observability tracks latency, job success rates, and data freshness with alerts.

Story: Reporting Frontend Structure

Summary

Map UI directories for dashboards, widgets, and report builders.

As a…

As a frontend developer

I want to…

I want to locate reporting UI code

So that…

So that I can expand analytics experiences

Acceptance Criteria

- [ ] Structure references client/src/features/reporting pages, components, and state managers.
- [ ] Shared visualization components integrate with design system charts and layout primitives.

Story: Reporting Data Fetching & State

Summary

Summarize data fetching strategies, caching, and state management for dashboards.

As a…

As a frontend architect

I want to…

I want consistent reporting data patterns

So that…

So that dashboards stay performant

Acceptance Criteria

- [ ] Strategies include query caching, subscription updates, and drill-down synchronization across widgets.
- [ ] State management enforces pagination, filtering, and error handling for large datasets.

Story: Reporting UI Flows

Summary

Describe dashboard exploration, customization, and export experiences.

As a…

As a UX lead

I want to…

I want intuitive reporting flows

So that…

So that users can derive insights quickly

Acceptance Criteria

- [ ] Flows include configurable dashboards, widget editors, and export/download actions with RBAC checks.
- [ ] Interactions support localization, saved views, and sharing with governance stakeholders.

Story: Reporting Access & Routing

Summary

Explain routing guards, navigation clusters, and permission enforcement in the reporting UI.

As a…

As a platform admin

I want to…

I want to control reporting access

So that…

So that only authorized roles view analytics

Acceptance Criteria

- [ ] Routing integrates with auth guards, tenant scoping, and navigation breadcrumbs for dashboards.
- [ ] Permissions align with RBAC scopes and audit logging for dashboard access events.

Story: Reporting Playbooks

Summary

Provide operational runbooks for data refresh failures, incident response, and stakeholder communication.

As a…

As an on-call engineer

I want to…

I want reporting runbooks

So that…

So that I can resolve analytics incidents

Acceptance Criteria

- [ ] Playbooks document alert thresholds, investigation steps, and rollback strategies.
- [ ] Communication plans outline stakeholder updates, status dashboards, and post-incident reviews.

External Integrations System

Location: /server/src/integrations

TL;DR

This document catalogs Project X's third-party integrations, covering supported connectors, authentication models, data synchronization patterns, and operational runbooks.

For every integration we highlight configuration steps, environment variables, failure-handling playbooks, and how the connector ties into notifications, tasks, and evidence lifecycle.

Use this reference when onboarding new environments, planning releases, or troubleshooting production incidents.

Backend Specification

Backend Location & Directory Layout

Integration connectors live under server/src/integrations, each exposing configuration, authentication, synchronization, and failure-handling logic reused by governance modules.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L53-L68■

```
server/src/integrations/
■■■ servicenow/
■   ■■■ client.js
■   ■■■ mapper.js
■   ■■■ scheduler.js
■■■ jira/
■■■ onetrust/
■■■ slack/
■■■ email-bridge/
■■■ evidence-providers/
■■■ shared/
    ■■■ auth.js
    ■■■ scheduler.js
    ■■■ telemetry.js
```

Core Modules & Responsibilities

- **Integration Registry (registry.js):** Declares connector metadata (capabilities, scopes, feature flags) and enforces naming conventions aligned with the platform-wide JavaScript implementation standar d.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L41-L68■■F:docs/02-t echnical-specifications/07-integration-architecture.md†L49-L66■
- **Base Connector (shared/connector-base.js):** Provides lifecycle hooks (configure, authenticate, sync, handleWebhook, teardown) so all connectors share retries, error normalization, and audit logging.
- **Service Clients (<connector>/client.js):** Wrap third-party SDKs/REST APIs with opinionated helpers for pagination, rate limiting, and concurrency budgets.
- **Mappers (<connector>/mapper.js):** Convert external payloads into governance domain entities (notifications, tasks, evidence) while enforcing schema validation before data leaves the integration boundary.■F:docs/03-systems/13-task-management-system/readme.md†L52-L109■■F:docs/03-sy stems/11-evidence-management-system/readme.md†L51-L101■
- **Schedulers & Workers (<connector>/scheduler.js):** Register BullMQ queues and background jobs responsible for polling and reconciliation tasks.
- **Shared Utilities (shared/):** Centralize OAuth flows, credential caching, telemetry emitters, and secret-resolution logic so connectors inherit proven patterns.■F:docs/02-technical-specifications/07-integration-architecture.md†L53-L142■

Configuration & Secrets Governance

- Connector configuration is persisted via the Admin & Configuration system; secrets (client IDs, client secrets, signing keys) are injected at runtime from the platform vault and rotated through dual-control workflows.■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L25-L118■
- All credentials are encrypted at rest, transmitted via TLS, and scoped to least-privilege permissions consistent with security-by-design expectations (AES-256 at rest, TLS 1.2+ in transit, tenant isolation).■F:docs/01-about/04-security-and-data-protection.md†L74-L177■
- Emergency rotations trigger admin workflows, Notification alerts, and follow-up Task assignments so downstream systems acknowledge the change window before resuming syncs.■F:docs/03-systems/ 05-admin-and-configuration-system/readme.md†L59-L118■■F:docs/03-systems/04-notification-syst em/readme.md†L47-L120■■F:docs/03-systems/13-task-management-system/readme.md†L52-L10 9■

REST & Webhook Endpoints

- **Admin APIs (/api/v1/integrations):** CRUD endpoints expose catalog metadata, configuration schemas, and installation state. Requests/response bodies follow the OpenAPI-driven REST conventions (resource-centric routes, JSON envelopes, JWT authorization).■F:docs/02-technical-sp ecifications/02-backend-architecture-and-apis.md†L196-L248■

- **Credential APIs (/api/v1/integrations/:id/credentials):** Issue credential setup links, validate scoped tokens, and kick off rotation workflows managed by Admin services.■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L25-L118■
- **Health APIs (/api/v1/integrations/:id/health & /api/v1/integrations/:id/metrics):** Surface heartbeat timestamps, recent job outcomes, queue depth, and SLA compliance for dashboards and SRE monitors.■F:docs/03-systems/07-probe-management-system/readme.md†L120-L133■
- **Inbound Webhooks (/api/v1/integrations/:type/webhook):** Validate HMAC signatures or JWT assertions, enqueue payloads for async processing, and emit structured audit entries before invoking mappers.■F:docs/02-technical-specifications/07-integration-architecture.md†L123-L186■
- **Outbound Webhooks/Partner APIs:** When connectors publish events to customer systems, payloads inherit governance metadata (control IDs, evidence URIs, correlation IDs) and respect retry/backoff strategies defined in the integration framework.■F:docs/02-technical-specifications/07-integration-architecture.md†L123-L186■

Schedulers & Data Flow

- Polling jobs run on named BullMQ queues (e.g., integrations:servicenow:sync) with idempotency keys, concurrency caps, and exponential backoff policies. Each job logs start/end markers, error stacks, and correlation IDs consumed by audit pipelines.■F:docs/02-technical-specifications/07-integration-architecture.md†L53-L142■
- Webhook handlers push raw payloads to durable queues before enrichment to protect against downtime and rate limits. Replay logic uses deduplication hashes stored alongside integration_events.
- Task, Notification, and Evidence modules subscribe to integration events to create remediation items, dispatch alerts, or attach imported evidence, maintaining cross-system traceability.■F:docs/03-systems/04-notification-system/readme.md†L47-L170■■F:docs/03-systems/13-task-management-system/readme.md†L52-L109■■F:docs/03-systems/11-evidence-management-system/readme.md†L51-L101■
- Probe-managed schedules and blackout windows coordinate with integration workers so maintenance events or regulatory quiet periods suspend outbound communication gracefully.■F:docs/03-systems/07-probe-management-system/readme.md†L120-L133■

Data Contracts & Mapping

- **ServiceNow:** OAuth 2.0 client credentials; syncs incident, change_request, and custom tables. Mappers enforce field parity for priority, state, assignment group, evidence links, and ServiceNow correlation IDs.■F:docs/02-technical-specifications/07-integration-architecture.md†L97-L122■

- **Jira:** Supports PAT (server) and OAuth 1.0a (cloud). Synchronizes issue status, comments, assignees, and custom fields referencing control/task identifiers. Bidirectional updates respect SLA transitions defined by Task Management policies.■F:docs/02-technical-specifications/07-integration-architecture.md†L103-L119■■F:docs/03-systems/13-task-management-system/readme.md†L52-L109■
- **OneTrust:** API-key based ingestion of policy inventory, risk registers, and assessment outcomes to enrich control metadata and evidence references.■F:docs/02-technical-specifications/07-integration-architecture.md†L109-L119■■F:docs/03-systems/11-evidence-management-system/readme.md†L51-L101■
- **Slack:** Bot tokens with scoped permissions deliver alert threads, interactive approvals, and slash-command callbacks. Connectors correlate Slack message IDs with Notification delivery entries for end-to-end acknowledgment tracking.■F:docs/02-technical-specifications/07-integration-architecture.md†L114-L119■■F:docs/03-systems/04-notification-system/readme.md†L47-L132■
- **Email/Webhook Bridge:** Normalizes inbound email/webhook payloads into governance events, verifying DKIM/signatures, parsing attachments, and queuing evidence ingestion jobs with retention policies derived from security requirements.■F:docs/02-technical-specifications/07-integration-architecture.md†L123-L186■■F:docs/01-about/04-security-and-data-protection.md†L118-L169■
- **Evidence Providers (S3, SharePoint, etc.):** Honor tenant retention, encryption, and residency policies before transferring artifacts into the Evidence Management domain.■F:docs/03-systems/11-evidence-management-system/readme.md†L51-L116■■F:docs/01-about/04-security-and-data-protection.md†L126-L169■

Observability, Testing & Quality Gates

- Structured logs emit integration.* events with correlation IDs and sanitized payload snapshots routed into the audit logging pipeline.■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L160■
- Metrics exported (success rate, latency, queue depth, credential expiry counters) feed Grafana dashboards and alert thresholds aligned with probe heartbeat SLAs.■F:docs/03-systems/07-probe-management-system/readme.md†L120-L133■
- Automated tests cover contract validation (JSON schema fixtures), sandbox API simulations, and regression suites executed during CI/CD with OpenAPI contract checks and mocked third-party responses.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L221-L248■
- Canary deployments run in staging tenants using feature flags before promoting connectors to production, ensuring safe rollout per integration governance policies.■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L59-L118■

Frontend Specification

Frontend Location & Directory Layout

Integration administration UI lives in client/src/features/integrations, exposing configuration consoles, health dashboards, and credential management flows.■F:docs/02-technical-specifications/03-frontend-architecture.md†L83-L140■

```
client/src/features/integrations/
■■■ pages/
■   ■■■ IntegrationCatalogPage.tsx
■   ■■■ IntegrationDetailPage.tsx
■   ■■■ CredentialManagementPage.tsx
■   ■■■ IntegrationHealthPage.tsx
■■■ components/
■   ■■■ IntegrationConfigForm.tsx
■   ■■■ CredentialRotationWizard.tsx
■   ■■■ SyncStatusTable.tsx
■   ■■■ FailureRunbookPanel.tsx
■■■ hooks/
■   ■■■ useIntegrations.ts
■   ■■■ useIntegrationHealth.ts
■   ■■■ useCredentialRotations.ts
■■■ api/
    ■■■ integrationsClient.ts
```

UI State & Data Access Patterns

- Feature-level hooks wrap Axios clients with cached SWR-style fetching, optimistic updates for configuration edits, and error channels aligned with global notification context.■F:docs/02-technical-specifications/03-frontend-architecture.md†L83-L140■
- Local state stores pending credential secrets in memory only; sensitive values never persist to local storage to honor frontend security posture (JWT headers, CSRF tokens, CSP).■F:docs/02-technical-specifications/03-frontend-architecture.md†L123-L160■
- Access control uses RBAC-aware route guards so only authorized roles see credential forms or rotation actions, matching platform-wide least privilege policies.■F:docs/01-about/04-security-and-data-protection.md†L81-L199■

Reusable Components & UI Flows

- **Catalog & Detail:** Lists available connectors, status, environment bindings, and quick actions. Detail pages show configuration, credentials, sync schedules, recent runs, and linked tasks or notifications for context.
- **Credential Management:** CredentialRotationWizard orchestrates dual-approval flows, captures change-ticket IDs, and calls rotation APIs; success transitions emit toast + inline alerts referencing Notification runbooks.■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L59-L118■■F:docs/03-systems/04-notification-system/readme.md†L47-L132■

- **Health Monitoring:** SyncStatusTable and IntegrationHealthPage display queue depth, failure counts, SLA timers, and heartbeat markers with drill-down links into Task remediation views.■F:docs/03-systems/07-probe-management-system/readme.md†L120-L133■■F:docs/03-systems/13-task-management-system/readme.md†L52-L109■
- **Runbooks:** FailureRunbookPanel surfaces connector-specific troubleshooting steps, environment matrices, and escalation targets, ensuring operators follow documented pathways before triggering incident management.

Schema Specification

- **integration_configs:** Stores connector type, environment, credentials (encrypted), scopes, schedules, webhook secrets, approval metadata, and feature-flag bindings.
- **integration_runs:** Logs synchronization executions with timestamps, status, error codes, latency, payload hashes, and downstream entity references (task IDs, notification IDs, evidence IDs).
- **integration_events:** Records inbound/outbound webhook deliveries, retries, deduplication hashes, and delivery statuses for replay and audit.
- **integration_credentials_audit:** Tracks rotation history, approvers, expiry dates, dual-control attestations, and incident links.
- **integration_health_metrics:** Aggregated KPIs for dashboards (success rate, latency, backlog depth, heartbeat lag) with rollups by tenant and connector.
- Relationships tie into Notification, Task, Evidence, and Governance modules to maintain end-to-end traceability for third-party interactions.■F:docs/03-systems/04-notification-system/readme.md†L47-L170■■F:docs/03-systems/13-task-management-system/readme.md†L52-L109■■F:docs/03-systems/11-evidence-management-system/readme.md†L51-L101■

Operational Playbooks & References

Environment Configuration

- Maintain environment-specific matrices for endpoints, credentials, scopes, rate limits, and feature flags. Use the Admin & Configuration system to manage secrets, tenant overrides, and staged rollout toggles.■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L25-L118■
- Validate integrations in staging with sandbox credentials before production rollout; execute change-management checklists covering smoke tests, audit logging verification, and rollback plans.■F:docs/02-technical-specifications/07-integration-architecture.md†L49-L186■
- Enforce encryption, segregation, and residency requirements when storing mirrored datasets or evidence artifacts to satisfy enterprise security principles.■F:docs/01-about/04-security-and-data-protection.md†L118-L169■

Daily Checks & Incident Response

- Monitor scheduled jobs, webhook failures, credential expirations, and heartbeat gaps via shared dashboards. Alerts route through Notification channels and escalate using Task workflows when SLAs breach.■F:docs/03-systems/04-notification-system/readme.md†L47-L170■■F:docs/03-systems/13-task-management-system/readme.md†L52-L109■■F:docs/03-systems/07-probe-management-system/readme.md†L120-L133■
- Incident response playbooks detail escalation paths, rollback procedures, communication channels, and evidence capture requirements. Post-incident reviews create governance tasks and update runbooks stored alongside integration metadata.■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L59-L118■■F:docs/03-systems/06-audit-logging-and-monitoring/readme.md†L7-L160■

Release & Compliance Readiness

- Coordinate releases with Admin feature flags, ensuring connectors pass automated contract tests and manual smoke validations before toggling to production tenants.■F:docs/03-systems/05-admin-and-configuration-system/readme.md†L59-L118■
- Export OpenAPI specs, schema migrations, and audit evidence for compliance reviews; attach to Evidence Management records for traceability.■F:docs/02-technical-specifications/02-backend-architecture-and-apis.md†L196-L248■■F:docs/03-systems/11-evidence-management-system/readme.md†L51-L101■
- Confirm alignment with platform security posture (encryption, least privilege, monitoring) ahead of external audits or regulator attestations.■F:docs/01-about/04-security-and-data-protection.md†L74-L199■

Related Documentation

- — alert routing for integration failures and acknowledgment flows.
- — remediation tracking for connector incidents and SLA enforcement.
- — evidence ingestion from external sources and retention policies.
- — credential governance, rollout controls, and secret rotation workflows.
- — scheduler coordination, heartbeat expectations, and shared monitoring patterns.

Story: Integrations Backend Layout

Summary

Describe modules managing partner connectors, webhooks, schedulers, and mapping utilities.

As a…

As a backend engineer

I want to…

I want to find integration code

So that…

So that I can build new connectors

Acceptance Criteria

- [ ] Layout covers adapters, controllers, services, schedulers, and mapping helpers for partner integrations.
- [ ] Notes explain how integrations interact with authentication, governance, and audit subsystems.

Story: Integrations Core Responsibilities

Summary

Summarize responsibilities for onboarding partners, normalizing payloads, retrying deliveries, and auditing activity.

As a…

As a platform architect

I want to…

I want integration duties documented

So that…

So that external connections remain reliable

Acceptance Criteria

- [ ] Core modules handle credential management, payload translation, retry orchestration, and audit logging.
- [ ] Responsibilities call out alignment with governance policies, SLAs, and compliance requirements.

Story: Integrations Configuration & Secrets

Summary

Explain secrets governance, environment variables, and configuration patterns for connectors.

As a…

As a DevOps engineer

I want to…

I want integration configuration clarity

So that…

So that deployments manage partner credentials securely

Acceptance Criteria

- [ ] Configuration lists API keys, OAuth credentials, webhooks, and retry settings sourced from secure vaults.
- [ ] Guidance covers rotation policies, tenant scoping, and approval workflows for secret updates.

Story: Integrations REST & Webhook Endpoints

Summary

Detail REST endpoints and webhook listeners exposed for partner interactions.

As a…

As an API consumer

I want to…

I want integration endpoint specs

So that…

So that partners can exchange data with governance services

Acceptance Criteria

- [ ] Endpoints describe onboarding flows, credential provisioning, inbound webhook handlers, and status queries.
- [ ] Contracts document authentication, retry semantics, and idempotency expectations for partner calls.

Story: Integration Schedulers & Data Flow

Summary

Explain background jobs, polling schedules, and streaming routes used to sync external systems.

As a…

As an operations engineer

I want to…

I want data flow visibility

So that…

So that partner sync stays timely

Acceptance Criteria

- [ ] Schedulers manage polling intervals, delta syncs, and reconciliation loops for each integration.
- [ ] Data flows cover queue usage, batching, and failure handling across connectors.

Story: Integration Data Contracts

Summary

Summarize mapping rules, payload schemas, and transformation utilities shared across connectors.

As a…

As a data engineer

I want to…

I want contract documentation

So that…

So that partner data maps cleanly into governance models

Acceptance Criteria

- [ ] Data contracts define canonical payload shapes, field mappings, and validation layers.
- [ ] Utilities support transformation, versioning, and compatibility checks across partner APIs.

Story: Integration Observability & Testing

Summary

Detail monitoring, logging, automated tests, and quality gates for integrations.

As a…

As a QA lead

I want to…

I want integration quality standards

So that…

So that connectors remain stable

Acceptance Criteria

- [ ] Observability captures metrics for throughput, failure rates, and latency with alert thresholds.
- [ ] Testing includes contract suites, sandbox simulations, and CI gates before promoting connector changes.

Story: Integrations Frontend Structure

Summary

Map UI directories for managing connectors, monitoring status, and configuring credentials.

As a…

As a frontend developer

I want to…

I want to locate integration UI code

So that…

So that I can enhance partner management tools

Acceptance Criteria

- [ ] Structure references client/src/features/integrations pages, components, and hooks.
- [ ] Shared UI integrates with admin navigation, RBAC guards, and status indicators.

Story: Integrations UI State & Data Patterns

Summary

Explain state management, data fetching, and caching strategies used by the integrations console.

As a…

As a frontend architect

I want to…

I want consistent data patterns

So that…

So that UI stays responsive and accurate

Acceptance Criteria

- [ ] State patterns leverage query caching, polling, and websocket updates for connector health.
- [ ] Data access patterns handle pagination, error states, and optimistic updates for configuration changes.

Story: Integrations UI Flows

Summary

Describe onboarding wizards, credential rotation screens, and health dashboards for integrations.

As a…

As a UX lead

I want to…

I want comprehensive integration flows

So that…

So that operators manage partners confidently

Acceptance Criteria

- [ ] Flows include connector discovery, approval workflows, credential editors, and webhook health monitors.
- [ ] UI interactions surface alerts, documentation links, and remediation actions for failed syncs.

Story: Integration Environment Configuration

Summary

Summarize deployment topologies, regional footprints, and environment-specific overrides.

As a…

As an infrastructure planner

I want to…

I want environment guidance

So that…

So that integrations scale across regions

Acceptance Criteria

- [ ] Configuration outlines regional endpoints, failover strategies, and environment-specific secrets.
- [ ] Guidance covers feature flags, rollout strategies, and dependency provisioning per environment.

Story: Integration Daily Checks & Incident Response

Summary

Document daily health checks, runbooks, and escalation procedures for integration incidents.

As a…

As an on-call engineer

I want to…

I want integration runbooks

So that…

So that I can respond quickly to partner outages

Acceptance Criteria

- [ ] Daily routines verify job backlogs, webhook deliveries, credential status, and alert dashboards.
- [ ] Incident response outlines escalation tiers, communication templates, and recovery verification steps.

Story: Integration Release & Compliance Readiness

Summary

Explain release checklists, compliance reviews, and stakeholder sign-offs required before deploying integration changes.

As a…

As a release manager

I want to…

I want integration release guidance

So that…

So that launches satisfy contractual and regulatory expectations

Acceptance Criteria

- [ ] Release steps include sandbox validation, partner coordination, legal review, and documentation updates.
- [ ] Compliance readiness requires evidence collection, audit log verification, and SLA confirmation prior to go-live.