Inspection Management System: Comprehensive Project Documentation

Version: 4.0**Date**: November 3, 2025

Executive Summary

The **Inspection Management System (IMS) v4.0** is an enterprise-grade, intelligent platform designed to manage the complete lifecycle of facility inspections across diverse operational environments. This system integrates advanced capabilities including offline-first mobile operations, intelligent route optimization, indoor positioning for GPS-denied environments, comprehensive compliance management, and real-time geospatial analytics. The platform supports multi-role workflows with granular permissions, ensuring operational efficiency, regulatory compliance, and complete audit trails for all actions.

Table of Contents

1. Introduction
- 1.1 Purpose
- 1.2 Scope
- 1.3 Key Features
- 1.4 Document Overview

1. User Roles and Permissions
- 2.1 Owner
- 2.2 Project Manager (PM)
- 2.3 Supervisor
- 2.4 Inspector
- 2.5 Auditor
- 2.6 Role Permissions Matrix
- 2.7 Role Assignment and Transfer Logic

1. System Architecture
- 3.1 Architectural Overview
- 3.2 Technology Stack

- 3.3 Data Architecture
- 3.4 Security Architecture

1. Core Modules: Detailed Business Logic

- 4.1 User Management & Security Module
- 4.2 Geography & GIS Module
- 4.3 Facility Management Module
- 4.4 License Management Module
- 4.5 Compliance & Regulations Module
- 4.6 Shift & Availability Management Module
- 4.7 Visit Management Module
- 4.8 Checklist Management Module
- 4.9 Incident Management Module
- 4.10 Pre-Visit Issue Management Module
- 4.11 Smart Route & Navigation Module
- 4.12 Offline Mode & Data Synchronization Module
- 4.13 Indoor Positioning Module
- 4.14 Settings & Configuration Module
- 4.15 Dashboard, Metrics & Reporting Module
- 4.16 Audit Trail & History Module

1. Database Schema

- 5.1 Schema Overview
- 5.2 Core Tables
- 5.3 Extended Tables
- 5.4 Audit and History Tables
- 5.5 Relationship Diagram

1. Comprehensive Workflows

- 6.1 End-to-End System Workflow
- 6.2 Visit Lifecycle Workflow
- 6.3 Visit Re-assignment Workflow
- 6.4 Inspector Unavailability Workflow

- 6.5 Supervisor Unavailability Workflow
- 6.6 Pre-Visit Issue Resolution Workflow
- 6.7 Incident Review Workflow
- 6.8 License Expiry Management Workflow
- 6.9 Visit Auto-Cancellation Workflow
- 6.10 Offline Data Synchronization Workflow

1. Detailed Use Cases

- 7.1 Use Case 1: System Setup and Initial Configuration
- 7.2 Use Case 2: Creating and Managing Facilities
- 7.3 Use Case 3: Scheduling and Assigning Visits
- 7.4 Use Case 4: Inspector Executes Visit with Smart Route
- 7.5 Use Case 5: Handling Underground Facility Inspection
- 7.6 Use Case 6: Recording Multi-Media Incident Evidence
- 7.7 Use Case 7: Offline Visit Execution and Sync
- 7.8 Use Case 8: Emergency Re-assignment
- 7.9 Use Case 9: Supervisor Delegation
- 7.10 Use Case 10: Compliance Dashboard Analysis

1. Business Rules and Conditions

- 8.1 Assignment and Validation Rules
- 8.2 Geo-fencing and Location Rules
- 8.3 Status Transition Rules
- 8.4 Compliance and Regulatory Rules
- 8.5 Notification and Alert Rules
- 8.6 Data Retention and Soft Delete Rules
- 8.7 Time Tracking and Performance Rules

1. Conclusion

1. Introduction

1.1. Purpose

This document serves as the definitive, comprehensive specification for the **Inspection Management System (IMS) Version 4.0**. It consolidates and integrates all system requirements, enhancements, and operational logic developed across multiple iterations to create a complete blueprint for development, implementation, deployment, and ongoing management. The system is designed to streamline facility inspections through intelligent automation, real-time tracking, compliance enforcement, multi-layered approval workflows, and advanced geospatial capabilities that function seamlessly in both connected and offline environments.

1.2. Scope

The Inspection Management System encompasses the complete lifecycle of facility inspections across diverse operational contexts. The system manages user authentication and role-based access control, geographical service area definition with GIS integration, facility and license administration, inspector shift scheduling and availability management, visit creation and intelligent assignment, real-time and indoor location tracking, checklist-based data collection with granular time tracking, incident reporting with multi-media evidence capture, pre-visit issue management, multi-stage review workflows with optional auditor involvement, comprehensive compliance tracking through a regulatory knowledge base, offline-first mobile operations with robust data synchronization, smart route optimization for field efficiency, and extensive reporting and analytics capabilities.

1.3. Key Features

The Inspection Management System v4.0 introduces a comprehensive set of advanced features that distinguish it as an enterprise-grade solution. The platform implements a **five-role RBAC model** (Owner, Project Manager, Supervisor, Inspector, Auditor) with granular permissions ensuring data security and operational accountability. The **offline-first mobile architecture** enables inspectors to execute complete visits without network connectivity, with intelligent data synchronization upon reconnection. The **smart route optimization** feature leverages routing algorithms to generate the most efficient daily inspection routes, prioritizing critical visits and minimizing travel time. The **hybrid positioning system** combines GPS for outdoor locations with Wi-Fi triangulation, BLE beacons, or manual floor plan tagging for indoor and underground facilities, ensuring accurate location capture in all environments. The **multi-media evidence system** supports images, videos, audio recordings, and documents as incident attachments, providing comprehensive documentation capabilities. The **compliance knowledge base** serves as a searchable FAQ database of government regulations, violations, and resolutions, with a dedicated dashboard for compliance analytics. The **comprehensive audit trail** logs every system action with full attribution, supporting forensic analysis and regulatory compliance. The **soft delete architecture** with configurable retention policies ensures data recoverability

while maintaining system performance. The **granular time tracking** captures duration data at both the visit level and individual checklist item level, enabling detailed performance analytics. The **supervisor availability toggle** provides a simple, user-driven mechanism for managing absences and delegations, replacing complex automated detection logic.

1.4. Document Overview

This documentation is structured to provide both strategic understanding for stakeholders and detailed technical specifications for implementation teams. Section 2 defines all user roles with explicit responsibilities and a comprehensive permissions matrix. Section 3 describes the system architecture, including the technology stack, data architecture, and security model. Section 4 provides detailed business logic for all 16 core modules, including entity definitions, validation rules, workflows, and edge case handling. Section 5 presents the complete database schema with SQL definitions for all tables, including core entities, extended features, and audit structures. Section 6 documents 10 comprehensive workflows that illustrate system behavior across various scenarios. Section 7 provides 10 detailed use cases with step-by-step execution, demonstrating real-world application of system features. Section 8 codifies all business rules and conditions governing system operations. Section 9 concludes with implementation recommendations and future enhancement opportunities. This document should be used by project managers, business analysts, system architects, database administrators, software developers, quality assurance teams, and operations personnel throughout the project lifecycle.

---

2. User Roles and Permissions

The system implements a strict **Role-Based Access Control (RBAC)** model with five distinct user roles. Each role has specific permissions and responsibilities designed to ensure operational efficiency, data security, and accountability. The role hierarchy is designed to support both centralized control and distributed operational autonomy.

2.1. Owner

The **Owner** role represents high-level stakeholders who require comprehensive visibility into system operations without direct involvement in day-to-day activities. This role is designed for executive oversight and strategic decision-making based on system-wide analytics.

Permissions: The Owner has view-only access to all dashboards, reports, and system data. This includes access to the GIS map view showing real-time inspector locations, facilities, and incidents. The Owner can view all compliance dashboards, performance metrics, and

historical data. The Owner has no create, update, or delete permissions on any entity, ensuring a strict separation between oversight and operations.

Typical Users: Executive management, board members, external auditors, regulatory oversight personnel, and strategic stakeholders.

2.2. Project Manager (PM)

The **Project Manager** is the system administrator with the highest level of authority and responsibility. The PM has complete control over all system entities and configurations, serving as the ultimate authority for system operations.

Responsibilities: The PM is responsible for onboarding and managing all system users across all roles (Supervisors, Inspectors, Auditors, Owners). The PM creates and manages Supervisors, assigning them to service areas and defining their team compositions. The PM configures all system-wide settings and operational parameters through the Settings & Configuration Module. The PM manages all core geographical entities, including Cities and Service Areas, defining the operational boundaries of the system. The PM has full CRUD permissions on all Facilities and Licenses, though day-to-day management may be delegated to Supervisors. The PM provides final approval for all completed visits, representing the last checkpoint in the review workflow. The PM handles supervisor delegations when a Supervisor marks themselves as unavailable or when a Supervisor account is deactivated. The PM has access to system-wide analytics and reports, including cross-supervisor comparisons, system performance metrics, and compliance summaries. The PM manages the Compliance Knowledge Base, adding and updating regulatory rules, violations, and resolutions.

Permissions: Full CRUD (Create, Read, Update, Delete) operations on all entities without geographical or team restrictions. The PM can override any system state and has access to all audit trail data.

Typical Users: System administrators, operations directors, compliance managers, and senior operational leadership.

2.3. Supervisor

The **Supervisor** is an operational manager responsible for a team of inspectors and specific geographical service areas. Supervisors serve as the primary operational layer, managing day-to-day inspection activities within their assigned domains.

Responsibilities: The Supervisor manages a team of inspectors, with team size configurable by the PM (default range: 1-50 inspectors). The Supervisor creates and manages facilities within their assigned service areas, ensuring facility data is accurate and up-to-date. The Supervisor creates and manages licenses for facilities in their assigned service areas, monitoring expiry dates and renewal requirements. The Supervisor creates

visits for facilities in their assigned service areas, determining inspection schedules based on regulatory requirements and operational priorities. The Supervisor assigns visits to inspectors on their team, balancing workload and considering inspector specializations, skills and availability. The Supervisor manages shift schedules for their team, defining work hours and days off. The Supervisor reviews completed visits from their team, examining checklist responses and incident reports for accuracy and completeness. The Supervisor handles visit re-assignments due to inspector unavailability, whether planned (scheduled time off) or unplanned (illness, emergency). The Supervisor resolves pre-visit issues reported by their team, such as incorrect facility locations or access problems. The Supervisor uses the availability toggle on their dashboard to indicate their own availability status, triggering PM notification when marked unavailable.

Permissions: The Supervisor has CRUD operations on Facilities limited to their assigned service areas. The Supervisor has CRUD operations on Licenses limited to facilities within their assigned service areas. The Supervisor has CRUD operations on Visits limited to their team members. The Supervisor has CRUD operations on Shifts limited to their team. The Supervisor has review and approval permissions on Incidents and Pre-Visit Issues created by their team. The Supervisor has view access to the Compliance Knowledge Base and can link incidents to compliance rules. The Supervisor has access to team-level analytics and reports, including inspector performance metrics and service area compliance scores.

Geographical Scope: A Supervisor can be assigned to multiple Service Areas. All permissions are scoped to the union of all assigned Service Areas. For example, if Supervisor Jane is assigned to "Downtown" and "Harbor District" service areas, she can manage facilities in both areas.

Typical Users: Field managers, regional supervisors, team leads, and operational coordinators.

2.4. Inspector

The **Inspector** is field personnel responsible for executing inspections at facility locations using the mobile application. Inspectors are the primary data collectors in the system, operating in diverse environments including outdoor, indoor, and underground locations.

Responsibilities: The Inspector views assigned visits in their mobile app schedule, organized by scheduled date. The Inspector uses the Smart Route feature to optimize their daily travel path across multiple facility locations. The Inspector navigates to facility locations using integrated turn-by-turn navigation in the mobile app. The Inspector starts visits, with the system performing geo-fence validation to ensure the inspector is physically present at the facility location. The Inspector completes assigned checklists during visits, answering all required questions with appropriate response types. The Inspector records incidents discovered during inspections, including title, description, severity, defect type,

and multi-media evidence (photos, videos, audio, documents). The Inspector reports pre-visit issues when unable to start a visit due to logistical problems such as incorrect location, facility closure, or access denial. The Inspector submits completed visits for supervisor review upon finishing all checklist items and documenting all incidents. The Inspector operates in offline mode when network connectivity is unavailable, with all data saved locally and synchronized upon reconnection.

Permissions:

The Inspector has view access to assigned visits only, with no visibility into other inspectors' schedules.

The Inspector can update visit status limited to their own visits, transitioning from Assigned to In Progress to Submitted.

The Inspector can create incidents during active visits, with automatic geolocation capture.

The Inspector can create pre-visit issues before starting a visit, automatically placing the visit on hold.

The Inspector can fill out checklist responses for visits assigned to them.

The Inspector has view access to facility information for assigned visits.

The Inspector has view access to the Compliance Knowledge Base for reference during inspections.

Typical Users: Field inspectors, compliance officers, safety inspectors, environmental inspectors, and quality assurance field personnel.

2.5. Auditor

The **Auditor** is a compliance specialist responsible for reviewing incidents recorded during inspections. This role is part of an **optional module** that can be enabled or disabled system-wide through the Settings & Configuration Module. When enabled, the Auditor provides an additional review layer between Supervisor review and PM approval, ensuring regulatory compliance and incident documentation quality.

Responsibilities: The Auditor reviews all incidents recorded during visits that have been approved by Supervisors. The Auditor examines incident descriptions, severity classifications, defect types, and attached evidence (photos, videos, audio, documents) for accuracy and compliance with regulatory standards. The Auditor approves incidents that meet documentation standards, allowing the visit to proceed to PM final approval. The Auditor rejects incidents that are inadequately documented or incorrectly classified, returning the visit to the Supervisor with detailed comments for correction. The Auditor ensures that incidents are correctly linked to relevant Compliance Rules from the

knowledge base. The Auditor contributes to the Compliance Dashboard by providing quality-assured incident data.

Permissions: The Auditor has view access to visits assigned for audit review, specifically those in the "Under Approval (Auditor Review)" status. The Auditor has review and approve/reject permissions on incidents, with the ability to add review comments. The Auditor has view access to facilities and licenses (read-only) to understand the context of inspections. The Auditor has full access to the Compliance Knowledge Base and can link incidents to specific compliance rules. The Auditor has access to audit-specific reports showing incident review metrics and compliance trends.

Typical Users: Compliance auditors, quality assurance specialists, regulatory compliance officers, and legal compliance reviewers.

2.6. Role Permissions Matrix

The following table provides a comprehensive overview of permissions for each role across all major system entities and actions.

Entity/Action	Owner	Project Manager	Supervisor	Inspector	Auditor
System Settings	View	**CRUD**	None	None	None
User Management	View	**CRUD**	View (Own Team)	None	None
Cities	View	**CRUD**	View	None	None
Service Areas	View	**CRUD**	View	None	None
Facilities	View	**CRUD**	**CRUD** (Own Areas)	View (Assigned)	View
Licenses	View	**CRUD**	**CRUD** (Own Areas)	View (Assigned)	View
Compliance Rules	View	**CRUD**	View	View	CRUD
Shifts	View	**CRUD**	**CRUD** (Own Team)	View (Own)	None
Visits	View	**CRUD**	**CRUD** (Own Team)	Update Status (Own)	View (For Review)
Checklists	View	**CRUD**	**CRUD**	Fill Out	View
Checklist Responses	View	View	View	**Create/Update** (Own)	View
Incidents	View	Review/Approve	Review	**Create** (Own Visits)	**Review/Approve**
Incident Attachments	View	View	View	**Create** (Own Incidents)	View
Pre-Visit Issues	View	Review/Resolve	Review/Resolve	**Create**	None
GIS Map View	**View**	**View**	View (Own Areas)	None	None
Compliance Dashboard	**View**	**View**	View (Own Areas)	None	View

Reports & Analytics	View (All)	View (All)	View (Own Areas/Team)	View (Own)	View (Assigned)
Audit Trail	View	**View (All)**	View (Own Actions)	View (Own Actions)	View (Own Actions)

2.7. Role Assignment and Transfer Logic

The system implements sophisticated logic for managing role assignments, transfers, and unavailability to ensure operational continuity.

Inspector Transfer Between Supervisors: When an Inspector is transferred from Supervisor A to Supervisor B, the system executes the following logic. All pending visits that are in "Assigned" or "In Progress" status remain assigned to the Inspector and move to Supervisor B's oversight. All visits that have been submitted by the Inspector but are still in Supervisor A's review queue (status "Under Review") are automatically moved to Supervisor B's review queue. Future visit assignments for this Inspector will be created by Supervisor B. Historical data (completed visits, incidents, audit trail) remains unchanged, preserving the record that Supervisor A was the original assigner and reviewer.

Supervisor Account Deactivation: If Supervisor A's account is deactivated (due to termination, long-term leave, or role change) while they have pending responsibilities, the system prevents orphaned workflows through forced re-assignment. All visits in Supervisor A's review queue are automatically re-assigned to Supervisor A's designated delegate (if configured) or to the PM's review queue. All Inspectors on Supervisor A's team are temporarily assigned to the PM, who must then assign them to a new Supervisor. All facilities and licenses in Supervisor A's service areas remain accessible to the PM and any other Supervisors assigned to those same service areas. The system sends an immediate notification to the PM when a Supervisor account is deactivated with pending items.

Supervisor Availability Toggle: Each Supervisor has a prominent "Available" / "Unavailable" toggle on their dashboard. When a Supervisor sets their status to "Unavailable," the system immediately sends a notification to the PM, including the Supervisor's name, team size, and number of pending reviews. The PM can then assign a temporary delegate Supervisor to handle reviews and urgent matters. Visits already in the Supervisor's review queue remain there unless the PM manually re-assigns them. New visits cannot be created by an unavailable Supervisor, but their team's Inspectors can continue executing already-assigned visits. When the Supervisor returns and sets their status back to "Available," normal operations resume.

Inspector Unavailability Handling: Inspector unavailability is managed through two mechanisms: planned absence and unplanned absence. For planned absence (scheduled

time off), the Supervisor manually re-assigns the Inspector's pending visits to other team members before the absence begins. For unplanned absence (illness, emergency, equipment failure), the system detects unavailability through the absence of location updates and lack of visit progress. If an Inspector has not sent a location update or changed any visit status for a configurable period (default: 30 minutes) and the mobile app is not in declared offline mode, the system flags the Inspector as potentially unavailable and sends a notification to the Supervisor. The Supervisor can then contact the Inspector and, if necessary, re-assign pending visits to other team members.

---

3. System Architecture

3.1. Architectural Overview

The Inspection Management System employs a modern, service-oriented, three-tier architecture designed for scalability, resilience, offline-first mobile functionality, and high availability. The architecture is cloud-native with support for on-premises deployment where required by regulatory or security constraints.

Presentation Layer: The presentation layer consists of two primary applications. The web application serves back-office roles (Owner, PM, Supervisor, Auditor) and is built using modern JavaScript frameworks such as React or Vue.js. The web application provides responsive design for desktop and tablet access, featuring interactive dashboards, GIS map views, and comprehensive reporting interfaces. The mobile application serves field personnel (Inspectors) and is built using cross-platform frameworks such as React Native or Flutter. The mobile app is designed with an offline-first architecture, storing all necessary data locally and synchronizing with the server when connectivity is available. The mobile app integrates native device capabilities including GPS, camera, microphone, file system access, and biometric authentication.

Application Layer: The application layer is implemented as a set of microservices, each responsible for specific business logic domains. This includes a User Service for authentication and authorization, a Visit Service for visit lifecycle management, a GIS Service for geospatial operations and indoor positioning, a Routing Service for smart route optimization, a Sync Service for offline data synchronization and conflict resolution, a Notification Service for multi-channel alerts, a Compliance Service for regulatory rule management, a Reporting Service for analytics and dashboard data aggregation, and a Media Service for multi-media file processing and storage. Microservices communicate via RESTful APIs and asynchronous message queues for event-driven operations.

Data Layer: The data layer employs a polyglot persistence strategy, using the most appropriate database technology for each data type. The primary relational database is PostgreSQL with the PostGIS extension for geospatial data types and queries. A time-series

database such as TimescaleDB stores high-frequency location tracking data for inspectors. A document database such as MongoDB stores the Compliance Knowledge Base, allowing for flexible schema evolution as regulatory rules change. A cloud object store such as AWS S3 or Google Cloud Storage stores multi-media files (images, videos, audio, documents) with CDN integration for fast global access. A Redis cache layer provides session management and frequently accessed data caching to reduce database load.

3.2. Technology Stack

The recommended technology stack balances modern capabilities with enterprise stability and long-term support.

Frontend Web Application:

- Framework: React 18+ or Vue.js 3+
- State Management: Redux or Vuex
- UI Component Library: Material-UI or Ant Design
- Mapping: Mapbox GL JS or Leaflet with OpenStreetMap
- Charts: Chart.js or D3.js
- Build Tool: Vite or Webpack

Mobile Application:

- Framework: React Native or Flutter
- State Management: Redux or MobX
- Local Database: SQLite or Realm
- Mapping: Mapbox SDK or Google Maps SDK
- Offline Support: Redux Persist or Hive

Backend Services:

- Runtime: Node.js (Express/NestJS) or Python (FastAPI/Django)
- API Gateway: Kong or AWS API Gateway
- Message Queue: RabbitMQ or AWS SQS
- Task Scheduler: Bull or Celery

Databases:

- Primary: PostgreSQL 14+ with PostGIS 3+
- Time-Series: TimescaleDB
- Document Store: MongoDB 6+

- Cache: Redis 7+
- Object Storage: AWS S3 / Google Cloud Storage / MinIO

DevOps & Infrastructure:

- Containerization: Docker
- Orchestration: Kubernetes or AWS ECS
- CI/CD: GitHub Actions or GitLab CI
- Monitoring: Prometheus + Grafana
- Logging: ELK Stack (Elasticsearch, Logstash, Kibana)
- APM: New Relic or Datadog

Security:

- Authentication: JWT with refresh tokens
- 2FA: TOTP (Time-based One-Time Password) via Google Authenticator
- Biometric: Native device APIs (Touch ID, Face ID, Fingerprint)
- Encryption: TLS 1.3 for transport, AES-256 for data at rest
- Secrets Management: HashiCorp Vault or AWS Secrets Manager

3.3. Data Architecture

The data architecture implements several key patterns to ensure data integrity, performance, and compliance.

Soft Delete Pattern: All major entities implement soft deletion through `deleted_at` (TIMESTAMP) and `deleted_by` (BIGINT) columns. When a record is "deleted," these fields are populated, and the record is excluded from standard queries using a WHERE clause filter. A scheduled job permanently purges soft-deleted records after the configured retention period (default: 90 days). This pattern enables data recovery, supports audit requirements, and maintains referential integrity.

Audit Trail Pattern: Every Create, Update, and Delete operation on major entities automatically generates an entry in the `AuditTrail` table. The audit record captures the entity type, entity ID, action type, old value (JSONB), new value (JSONB), the user who performed the action, and the timestamp. This is implemented via database triggers or application-level hooks. The `performed_by` column is a non-nullable foreign key to the `User` table, ensuring complete attribution.

Geospatial Data Pattern: All entities with location information store coordinates using PostGIS geometry types. Facilities, Incidents, and Inspector location tracking use the `POINT` geometry type with SRID 4326 (WGS 84 coordinate system). Service Areas use the

`POLYGON` or `MULTIPOLYGON` geometry type to define geographical boundaries. PostGIS spatial indexes (GIST) are created on all geometry columns to enable fast spatial queries such as "find all facilities within this service area" or "find all incidents within 500 meters of this location."

Time-Series Data Pattern: Inspector location tracking generates high-frequency data (potentially every 10-30 seconds while on duty). This data is stored in TimescaleDB, which automatically partitions data by time (hypertables). Retention policies automatically delete location data older than a configured period (default: 30 days) to manage storage costs while maintaining recent data for route analysis.

Multi-Tenancy Pattern: While the current specification describes a single-tenant system, the database schema is designed to support future multi-tenancy through the addition of a `tenant_id` column to all major tables. This would enable a SaaS deployment model where multiple organizations share the same infrastructure with complete data isolation.

3.4. Security Architecture

The system implements defense-in-depth security principles across all layers.

Authentication: Users authenticate via username and password, with passwords hashed using bcrypt with a minimum work factor of 12. Upon successful authentication, the system issues a short-lived JWT access token (15-minute expiry) and a long-lived refresh token (7-day expiry). The mobile app stores tokens in secure device storage (iOS Keychain, Android Keystore). Two-factor authentication can be enabled per user, requiring a TOTP code from an authenticator app. Biometric authentication on mobile devices provides a convenient second factor, with the biometric check unlocking the stored credentials.

Authorization: All API endpoints enforce role-based access control through middleware that validates the JWT and checks the user's role and permissions against the requested resource. Supervisors' permissions are further scoped by their assigned service areas, with the database queries automatically filtering results. Inspectors can only access their own assigned visits, enforced at both the API and database levels.

Data Protection: All data in transit is encrypted using TLS 1.3. All data at rest in databases is encrypted using transparent data encryption (TDE). Multi-media files in object storage are encrypted using server-side encryption (SSE). Personally identifiable information (PII) such as user email addresses and phone numbers are encrypted at the application level using AES-256 before storage.

Network Security: The system is deployed within a Virtual Private Cloud (VPC) with private subnets for databases and application services. Only the API Gateway and load balancer are exposed to the public internet. Database access is restricted to application services via security groups. Administrative access to infrastructure requires VPN connection and multi-factor authentication.

Audit and Compliance: The comprehensive audit trail provides a complete record of all system actions for forensic analysis and compliance reporting. Failed authentication attempts are logged and monitored for brute-force attack detection. The system supports data export for compliance with data protection regulations such as GDPR, allowing users to request their personal data. The soft delete pattern with configurable retention supports "right to be forgotten" requirements while maintaining operational data integrity.

---

4. Core Modules: Detailed Business Logic

This section provides comprehensive documentation for all 16 core modules of the Inspection Management System. Each module description includes purpose, key entities, business logic, validation rules, workflows, use case examples, and edge case handling.

4.1. User Management & Security Module

Purpose: Manage user accounts, authentication, authorization, and personal security settings across all roles.

Key Entities:

TypeScript

```
CREATE TABLE User (
    id BIGSERIAL PRIMARY KEY,
    username VARCHAR(100) UNIQUE NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(50) NOT NULL, -- 'owner', 'pm', 'supervisor', 'inspector',
'auditor'
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    phone_number VARCHAR(20),
    is_active BOOLEAN DEFAULT true,
    is_available BOOLEAN DEFAULT true, -- For Supervisors
    supervisor_id BIGINT REFERENCES User(id), -- For Inspectors
    delegate_supervisor_id BIGINT REFERENCES User(id), -- For Supervisors
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE UserConfiguration (
```

```
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT REFERENCES User(id) ON DELETE CASCADE,
    two_factor_enabled BOOLEAN DEFAULT false,
    two_factor_secret VARCHAR(255), -- TOTP secret
    biometric_enabled BOOLEAN DEFAULT false,
    email_notifications BOOLEAN DEFAULT true,
    sms_notifications BOOLEAN DEFAULT false,
    push_notifications BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Business Logic:

User accounts are created exclusively by the PM, with the exception of the **Owner** and first PM account which is created during system initialization. When creating a user, the PM specifies the role, personal information, and initial password. The system sends an email to the new user with login credentials and a mandatory password change requirement on first login. For Inspector accounts, the PM or Supervisor must specify the `supervisor_id`, establishing the reporting relationship. For Supervisor accounts, the PM can optionally specify a `delegate_supervisor_id` who will receive delegated responsibilities when the Supervisor marks themselves as unavailable.

The **Supervisor Availability Toggle** is a prominent UI element on the Supervisor dashboard. When a Supervisor sets `is_available` to false, the system immediately sends a notification to the PM including the Supervisor's name, team size (count of Inspectors with this Supervisor's ID), and count of pending reviews. The PM can then re-assign pending reviews to the delegate Supervisor or handle them directly. When the Supervisor sets `is_available` back to true, normal operations resume.

Inspector Transfer Logic: When an Inspector is transferred from Supervisor A to Supervisor B, the system updates the Inspector's `supervisor_id` from A's ID to B's ID. All visits in "Assigned" or "In Progress" status remain assigned to the Inspector. All visits in "Under Review" status that were submitted by this Inspector are automatically moved from Supervisor A's review queue to Supervisor B's review queue through a status update trigger. Historical audit trail records remain unchanged, preserving the original assignment chain.

User-Level Security Configuration: Each user can access their personal settings to configure security preferences. Two-factor authentication is enabled by displaying a QR code for the user to scan with an authenticator app (Google Authenticator, Authy, etc.). The user enters the 6-digit TOTP code to verify setup, and the system stores the encrypted `two_factor_secret`. On subsequent logins, the user must provide both password and TOTP code. Biometric authentication on mobile devices is enabled through the mobile app settings, which registers the user's fingerprint or face ID with the device's secure enclave.

Once enabled, the user can unlock the app using biometrics instead of entering their password on each use.

Validation Rules:

- Username must be unique and 3-50 characters
- Email must be valid format and unique
- Password must meet complexity requirements (minimum 8 characters, at least one uppercase, one lowercase, one number, one special character)
- Role must be one of the five defined roles
- Inspector accounts must have a valid `supervisor_id`
- Supervisor accounts cannot be their own delegate

Edge Cases:

- If a Supervisor is deactivated (`is_active` set to false) while having pending reviews, the system automatically re-assigns all pending visits to the delegate Supervisor or PM
- If an Inspector's Supervisor is deactivated, the Inspector is temporarily assigned to the PM until a new Supervisor is assigned
- If a user forgets their password, a secure password reset link is sent to their registered email with a 1-hour expiry

---

4.2. Geography & GIS Module

Purpose: Manage geographical hierarchy (Cities and Service Areas), provide GIS map visualization, and support geospatial analysis.

Key Entities:

SQL

```sql
CREATE TABLE City (
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    country VARCHAR(100) NOT NULL,
    state_province VARCHAR(100),
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
```

```
);

CREATE TABLE ServiceArea (
    id BIGSERIAL PRIMARY KEY,
    city_id BIGINT REFERENCES City(id) ON DELETE CASCADE,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    boundary GEOMETRY(POLYGON, 4326), -- PostGIS polygon defining area
boundary
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE SupervisorServiceArea (
    id BIGSERIAL PRIMARY KEY,
    supervisor_id BIGINT REFERENCES User(id) ON DELETE CASCADE,
    service_area_id BIGINT REFERENCES ServiceArea(id) ON DELETE CASCADE,
    assigned_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    assigned_by BIGINT REFERENCES User(id),
    UNIQUE(supervisor_id, service_area_id)
);

CREATE INDEX idx_service_area_boundary ON ServiceArea USING GIST(boundary);
```

Business Logic:

The geographical hierarchy consists of two levels: Cities and Service Areas. A **City** represents a municipality or metropolitan area and serves as a high-level organizational unit. A **Service Area** is a smaller geographical zone within a City, representing a specific operational district such as "Downtown," "Harbor District," or "Industrial Zone." A single City can contain multiple Service Areas, allowing for fine-grained operational management.

Supervisor Assignment to Service Areas: Supervisors are assigned to one or more Service Areas through the `SupervisorServiceArea` junction table. This many-to-many relationship allows flexible operational coverage. For example, Supervisor Jane might be assigned to both "Downtown" and "Harbor District" service areas, while Supervisor Bob is assigned only to "Industrial Zone." All of a Supervisor's permissions (facility management, license management, visit creation) are scoped to the union of their assigned Service Areas. The system enforces this through database-level row-level security or application-level query filters.

GIS Map View: The web dashboard for PM and Owner roles includes an interactive GIS map powered by Mapbox or Leaflet. The map displays three primary data layers. The **Facilities**

Layer plots all facilities from the `Facility` table using their `geolocation` coordinates, with markers color-coded by facility type or license status. The **Incidents Layer** plots all incidents from the `Incident` table using their `latitude` and `longitude` coordinates, with markers color-coded by severity (Critical=red, High=orange, Medium=yellow, Low=green). The **Inspector Location Layer** plots the current location of all active inspectors from the `InspectorLocation` table, with real-time updates every 30 seconds. The map supports filtering by date range, service area, severity, and other attributes.

GIS Analysis: The collected geospatial data enables advanced spatial analysis. **Incident Hotspot Analysis** uses spatial clustering algorithms to identify geographical areas with high incident density, helping to prioritize enforcement efforts. **Route Efficiency Analysis** compares the smart route suggestion with the actual path taken by inspectors (from location tracking data) to identify optimization opportunities. **Service Area Coverage Analysis** calculates the percentage of facilities in each service area that have been inspected within the last 30/60/90 days, identifying coverage gaps.

Validation Rules:

- City name must be unique within the same country and state/province
- Service Area name must be unique within the same City
- Service Area boundary must be a valid polygon geometry
- A Supervisor cannot be assigned to the same Service Area multiple times

Use Case Example:

The PM creates a new City record for "Springfield" in the United States, Illinois. The PM then creates three Service Areas within Springfield: "Downtown" (boundary defined by uploading a GeoJSON polygon), "North Industrial Zone," and "Riverside District." The PM assigns Supervisor Jane to "Downtown" and "Riverside District," and Supervisor Bob to "North Industrial Zone." Jane can now create facilities and visits only for facilities located within Downtown or Riverside District, while Bob manages the North Industrial Zone.

4.3. Facility Management Module

Purpose: Manage facility records including location, type, ownership, and operational details.

Key Entities:

SQL

```sql
CREATE TABLE Facility (
    id BIGSERIAL PRIMARY KEY,
    service_area_id BIGINT REFERENCES ServiceArea(id),
```

```
    name VARCHAR(255) NOT NULL,
    facility_type VARCHAR(100) NOT NULL, -- 'restaurant', 'factory',
'warehouse', etc.
    address VARCHAR(500) NOT NULL,
    geolocation GEOMETRY(POINT, 4326) NOT NULL, -- PostGIS point for outdoor
location
    owner_name VARCHAR(255),
    owner_contact VARCHAR(255),
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE FacilityFloorPlan (
    id BIGSERIAL PRIMARY KEY,
    facility_id BIGINT REFERENCES Facility(id) ON DELETE CASCADE,
    floor_level VARCHAR(50) NOT NULL, -- 'Ground', '1', '2', '-1', '-2', etc.
    floor_name VARCHAR(100), -- 'Basement Level 2', 'Mechanical Room Floor'
    floor_plan_image_path VARCHAR(500), -- Path to uploaded floor plan image
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE INDEX idx_facility_geolocation ON Facility USING GIST(geolocation);
```

Business Logic:

Facilities are the primary inspection targets in the system. Each facility must be assigned to a Service Area, which determines which Supervisors have management permissions. The `geolocation` field stores the facility's outdoor GPS coordinates as a PostGIS POINT geometry, enabling spatial queries such as "find all facilities within 5 km of this location" or "find all facilities within this service area boundary."

Facility Floor Plans: For facilities with indoor or underground areas (multi-story buildings, basements, tunnels), the PM or Supervisor can upload floor plan images through the `FacilityFloorPlan` table. Each floor plan is associated with a specific floor level (e.g., "Ground", "1", "-1" for first basement level, "-2" for second basement level). These floor plans are used by the Indoor Positioning Module to enable manual location tagging when GPS is unavailable.

Soft Delete Implementation: When a facility is "deleted," the system sets the `deleted_at` timestamp and `deleted_by` user ID. The facility is immediately hidden from all standard views and dropdowns. However, the record remains in the database to preserve referential integrity with historical visits, licenses, and incidents. A scheduled job runs daily to permanently delete (hard delete) facilities where `deleted_at` is older than the configured retention period (default: 90 days from system settings).

Validation Rules:

- Facility name must be unique within the same Service Area
- Service Area must exist and not be soft-deleted
- Geolocation must be a valid POINT geometry with latitude between -90 and 90, longitude between -180 and 180
- Geolocation must fall within the boundary of the assigned Service Area (validated using PostGIS ST_Contains function)
- Facility type must be from a predefined list or custom types configured by PM

Use Case Example:

Supervisor Jane creates a new facility record for "Ocean View Restaurant" in the Downtown service area. She enters the address "123 Main Street, Springfield" and uses the map interface to click the facility's location, which captures the coordinates (lat: 39.7817, lon: -89.6501). The system validates that this point falls within the Downtown service area boundary. Jane uploads a floor plan image for the basement level ("-1") where the kitchen and storage areas are located. This floor plan will be used during inspections when the inspector is in the basement and GPS is unavailable.

4.4. License Management Module

Purpose: Manage facility licenses, track expiry dates, and automate renewal alerts.

Key Entities:

SQL

```sql
CREATE TABLE License (
    id BIGSERIAL PRIMARY KEY,
    facility_id BIGINT REFERENCES Facility(id) ON DELETE CASCADE,
    license_number VARCHAR(100) UNIQUE NOT NULL,
    license_type VARCHAR(100) NOT NULL, -- 'health', 'safety',
'environmental', etc.
    issue_date DATE NOT NULL,
    expiry_date DATE NOT NULL,
    issuing_authority VARCHAR(255),
```

```
    status VARCHAR(50) DEFAULT 'active', -- 'active', 'expired',
'suspended', 'revoked'
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE INDEX idx_license_expiry ON License(expiry_date) WHERE deleted_at IS
NULL;
```

Business Logic:

Licenses are associated with facilities and represent regulatory permits required for operation. Each license has an `expiry_date`, and the system implements automated monitoring to alert stakeholders of approaching expiry. A scheduled job runs daily (typically at midnight in the system's configured timezone) to check for licenses approaching expiry.

License Expiry Alert Workflow: The system configuration defines alert thresholds as a comma-separated list of days (default: "90,60,30,7"). When the daily job detects a license with `expiry_date` within one of these thresholds, it sends notifications to the Supervisor responsible for the facility's service area and the PM. For example, if a license expires on December 31, 2025, and today is November 1, 2025 (60 days before expiry), the system sends an alert. The alert includes the facility name, license type, license number, and expiry date. Alerts are sent via email and as in-app notifications.

License Status Management: The license `status` field is automatically updated by the system. When a license is created with an `expiry_date` in the future, the status is set to "active." When the daily job detects that `expiry_date` has passed (expiry_date < current_date), the status is automatically changed to "expired," and an urgent notification is sent to the Supervisor and PM. The PM or Supervisor can manually update the status to "suspended" or "revoked" if regulatory action is taken.

Validation Rules:

- License number must be unique across the entire system
- Facility must exist and not be soft-deleted
- Issue date must be before or equal to expiry date
- Expiry date must be in the future when creating a new license (warning if in the past)
- License type must be from a predefined list or custom types configured by PM

Use Case Example:

Supervisor Jane creates a health license for "Ocean View Restaurant" with license number "HEALTH-2025-12345," issue date January 1, 2025, and expiry date December 31, 2025. On October 2, 2025 (90 days before expiry), the daily job sends the first alert to Jane and the PM. On November 1, 2025 (60 days before expiry), a second alert is sent. On December 1, 2025 (30 days before expiry), a third alert is sent. On December 24, 2025 (7 days before expiry), a final urgent alert is sent. If the license is not renewed by December 31, the status automatically changes to "expired" on January 1, 2026, and an urgent notification is sent.

4.5. Compliance & Regulations Module

Purpose: Provide a searchable knowledge base of government regulations, violations, resolutions, and fines to support compliance operations.

Key Entities:

SQL

```sql
CREATE TABLE ComplianceRule (
    id BIGSERIAL PRIMARY KEY,
    category VARCHAR(100) NOT NULL, -- 'health', 'safety', 'environmental', 'fire', etc.
    rule_code VARCHAR(50) UNIQUE, -- Official regulation code (e.g., 'OSHA-1910.22')
    rule_name VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    resolution TEXT NOT NULL, -- Recommended action or corrective measure
    fine_amount DECIMAL(10, 2), -- Standard fine amount for violation
    severity VARCHAR(50), -- 'critical', 'high', 'medium', 'low'
    reference_url VARCHAR(500), -- Link to official regulation document
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE INDEX idx_compliance_rule_category ON ComplianceRule(category);
CREATE INDEX idx_compliance_rule_severity ON ComplianceRule(severity);
```

Business Logic:

The Compliance & Regulations Module serves as a centralized FAQ database for all stakeholders to reference government regulations, understand violations, and determine appropriate resolutions. The PM is responsible for populating and maintaining this

database, adding new regulations as they are published and updating existing rules when regulations change.

Compliance Rule Structure: Each rule includes a `category` (e.g., "Health," "Safety," "Environmental"), an optional official `rule_code` (e.g., "OSHA-1910.22" for walking-working surfaces), a `rule_name` (e.g., "Slip, Trip, and Fall Hazards"), a detailed `description` explaining the regulation, a `resolution` describing the corrective action or penalty, an optional `fine_amount` for the standard penalty, a `severity` classification, and an optional `reference_url` linking to the official regulation document.

Search and Browse Functionality: All users can search the Compliance Knowledge Base by keyword, category, severity, or rule code. The search interface provides autocomplete suggestions and filters. Users can browse by category to explore all regulations in a specific domain. Each rule detail page displays the full information and allows users to follow the reference URL to the official source.

Linking Incidents to Compliance Rules: When an Auditor or PM reviews an incident, they can link it to one or more relevant Compliance Rules. This creates a relationship between the incident and the regulation, enabling compliance reporting. For example, an incident titled "Broken Safety Railing" might be linked to the OSHA regulation for guardrails. This linkage is stored in a junction table:

SQL

```
CREATE TABLE IncidentComplianceRule (
    id BIGSERIAL PRIMARY KEY,
    incident_id BIGINT REFERENCES Incident(id) ON DELETE CASCADE,
    compliance_rule_id BIGINT REFERENCES ComplianceRule(id) ON DELETE
CASCADE,
    linked_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    linked_by BIGINT REFERENCES User(id),
    UNIQUE(incident_id, compliance_rule_id)
);
```

Compliance Dashboard: The system provides a dedicated Compliance Dashboard accessible to PM, Owner, and Auditor roles. The dashboard displays key metrics derived from the Compliance Knowledge Base and linked incidents. **Total Potential Fines** sums the `fine_amount` of all Compliance Rules linked to open (unresolved) incidents. **Top 5 Violated Rules** shows the Compliance Rules most frequently linked to incidents in the selected time period. **Violations by Category** displays a pie chart showing the distribution of incidents across regulation categories. **Violations by Severity** shows a bar chart of incidents grouped by severity. **Compliance Trend** shows a line chart of incident counts over time, indicating whether compliance is improving or degrading.

Validation Rules:

- Rule code must be unique if provided
- Category must be from a predefined list or custom categories configured by PM
- Description and resolution are required fields
- Fine amount must be a positive number if provided
- Severity must be one of: 'critical', 'high', 'medium', 'low'

Use Case Example:

The PM adds a new Compliance Rule with category "Health," rule code "FDA-2017-001," rule name "Food Temperature Control," description "All perishable food must be stored at temperatures below 40°F (4°C) to prevent bacterial growth," resolution "Facility must install or repair refrigeration equipment and provide temperature logs. Fine: $500 for first offense, $1000 for repeat offense," fine_amount 500.00, severity "high," and reference_url pointing to the FDA regulation page. Later, Inspector Mike records an incident at a restaurant: "Refrigerator temperature at 55°F, spoiled food present." When Auditor Sarah reviews this incident, she searches the Compliance Knowledge Base for "food temperature," finds the FDA-2017-001 rule, and links it to the incident. The Compliance Dashboard now shows this incident contributing to the "Total Potential Fines" ($500) and incrementing the count for "Food Temperature Control" in the "Top 5 Violated Rules" chart.

4.6. Shift & Availability Management Module

Purpose: Manage inspector work schedules, track shift hours, and coordinate availability for visit assignments.

Key Entities:

SQL

```sql
CREATE TABLE Shift (
    id BIGSERIAL PRIMARY KEY,
    inspector_id BIGINT REFERENCES User(id) ON DELETE CASCADE,
    shift_date DATE NOT NULL,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    status VARCHAR(50) DEFAULT 'scheduled', -- 'scheduled', 'active',
'completed', 'cancelled'
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);
```

```
CREATE INDEX idx_shift_inspector_date ON Shift(inspector_id, shift_date);
```

Business Logic:

The Shift Management Module enables Supervisors to define work schedules for their team of inspectors. Each shift record specifies an inspector, a date, start time, and end time. Shifts are typically created in bulk for recurring schedules (e.g., Monday-Friday, 8:00 AM - 5:00 PM) but can be customized for individual inspectors or specific dates.

Flexible Visit Scheduling: Unlike traditional systems that assign visits to specific time slots, this system implements **date-based visit assignment**. Visits are assigned to an inspector for a specific `scheduled_date` without a specific `scheduled_time`. The inspector is responsible for completing all visits assigned for that date during their shift hours. This provides operational flexibility, allowing inspectors to adapt to real-world conditions such as traffic, facility availability, and incident investigation time. The Smart Route Module optimizes the order of visits based on geolocation, not time constraints.

Shift Status Workflow: When a shift is created, its status is "scheduled." When the inspector opens the mobile app on the shift date and their current time is within the shift's start and end time, the app prompts them to "Start Shift," which changes the status to "active." When the inspector completes all assigned visits for the day or manually ends their shift, the status changes to "completed." If a Supervisor cancels a shift (e.g., due to inspector absence or operational changes), the status changes to "cancelled," and any visits assigned for that date are automatically moved to "Unassigned" status, requiring re-assignment.

Validation Rules:

- Inspector must exist and have role 'inspector'
- End time must be after start time
- Shift duration must be at least 1 hour and no more than 12 hours
- An inspector cannot have overlapping shifts on the same date

Use Case Example:

Supervisor Jane creates a recurring shift schedule for Inspector Mike: Monday-Friday, 8:00 AM - 5:00 PM for the entire month of November 2025. On November 5, Mike opens the mobile app at 7:55 AM. At 8:00 AM, the app prompts "Start your shift?" Mike taps "Start," and the shift status changes to "active." Mike completes his 8 assigned visits by 4:30 PM and taps "End Shift." The shift status changes to "completed," and the system logs the actual work duration.

4.7. Visit Management Module

Purpose: Manage the complete lifecycle of facility inspections from creation to final approval.

Key Entities:

SQL

```
CREATE TABLE Visit (
    id BIGSERIAL PRIMARY KEY,
    facility_id BIGINT REFERENCES Facility(id),
    inspector_id BIGINT REFERENCES User(id),
    supervisor_id BIGINT REFERENCES User(id),
    checklist_id BIGINT REFERENCES Checklist(id),
    scheduled_date DATE NOT NULL,
    priority VARCHAR(50) DEFAULT 'normal', -- 'normal', 'critical'
    status VARCHAR(50) DEFAULT 'unassigned',
    -- Status values: 'unassigned', 'assigned', 'in_progress', 'on_hold',
'interrupted',
    --                'submitted', 'under_review', 'under_approval',
'completed', 'cancelled'
    actual_start_time TIMESTAMP,
    actual_end_time TIMESTAMP,
    start_latitude DECIMAL(10, 8),
    start_longitude DECIMAL(11, 8),
    notes TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE INDEX idx_visit_inspector_date ON Visit(inspector_id, scheduled_date);
CREATE INDEX idx_visit_status ON Visit(status);
```

Business Logic:

Visits are the central entity of the inspection workflow. A visit represents a scheduled inspection of a specific facility on a specific date by a specific inspector. The visit lifecycle progresses through multiple statuses, each representing a distinct stage in the workflow.

Visit Priority System: Visits are classified as either "normal" or "critical." **Normal visits** are routine, scheduled inspections based on regulatory requirements or operational plans. **Critical visits** are high-priority inspections triggered by urgent situations such as a complaint, a reported safety hazard, or a follow-up to a previous critical incident. Critical

visits are visually flagged in all user interfaces (red badge, "URGENT" label) and are prioritized by the Smart Route Module to be completed first in the inspector's daily route.

Visit Status Workflow: The visit status progresses through the following states:

1. **Unassigned**: Visit created by Supervisor but not yet assigned to an inspector
2. **Assigned**: Visit assigned to an inspector for a specific date
3. **In Progress**: Inspector has started the visit (geo-fence validated)
4. **On Hold**: Inspector reported a pre-visit issue; awaiting Supervisor resolution
5. **Interrupted**: System detected inactivity; visit flagged for Supervisor review
6. **Submitted**: Inspector completed all checklists and submitted for review
7. **Under Review**: Supervisor is reviewing the visit and incidents
8. **Under Approval**: Visit is in final approval stage (Auditor if enabled, then PM)
9. **Completed**: Visit fully approved and closed
10. **Cancelled**: Visit cancelled by Supervisor or PM

Time Tracking: The system captures `actual_start_time` when the inspector taps "Start Visit" in the mobile app and `actual_end_time` when the inspector taps "Submit Visit." The duration (end_time - start_time) is calculated and stored for performance analytics. Additionally, the system captures `start_latitude` and `start_longitude` from the inspector's device GPS at the moment of visit start, providing verification that the inspector was physically present at the facility location.

Geo-fence Validation: When an inspector attempts to start a visit, the mobile app captures the device's current GPS coordinates. The app sends these coordinates to the server along with the visit ID. The server retrieves the facility's `geolocation` from the database and calculates the distance between the inspector's location and the facility using the PostGIS `ST_Distance` function. If the distance is greater than the configured geo-fence radius (default: 100 meters from system settings), the server rejects the start request and returns an error message: "You are too far from the facility location. Please move closer to start the visit." This prevents inspectors from starting visits remotely.

Re-assignment Workflow: Visits can be re-assigned in several scenarios. **Manual Re-assignment**: A Supervisor can manually re-assign a visit from one inspector to another (e.g., to balance workload or accommodate inspector specializations). The system updates the `inspector_id` and sends notifications to both the old and new inspector. **Unavailability Re-assignment**: If an inspector is unavailable (sick, emergency, equipment failure), the Supervisor can re-assign all of that inspector's pending visits (status "assigned") to other team members. **Automatic Re-assignment on Shift Cancellation**: If a Supervisor cancels

an inspector's shift, all visits assigned for that date automatically change status to "unassigned," requiring the Supervisor to re-assign them.

Validation Rules:

- Facility must exist and not be soft-deleted
- Inspector must exist, have role 'inspector', and be active
- Supervisor must exist, have role 'supervisor', and be active
- Scheduled date must be today or in the future (warning if in the past)
- Inspector must be on the Supervisor's team (inspector.supervisor_id = supervisor.id)
- Facility must be in one of the Supervisor's assigned service areas
- An inspector cannot have more than a configurable maximum number of visits on the same date (default: 15)

Use Case Example:

Supervisor Jane creates a visit for "Ocean View Restaurant" (facility_id=123), assigns it to Inspector Mike (inspector_id=456), sets scheduled_date to November 10, 2025, priority to "normal," and assigns "Standard Health Checklist" (checklist_id=789). The visit status is "assigned." On November 10, Mike opens the mobile app, sees the visit in his list, and taps "Navigate." The Smart Route feature has optimized his route, and this visit is third in the sequence. At 10:30 AM, Mike arrives at the restaurant. He taps "Start Visit." The app captures his GPS coordinates (lat: 39.7817, lon: -89.6501) and sends them to the server. The server calculates the distance to the facility's location (stored as lat: 39.7820, lon: -89.6505) and finds it is 45 meters, which is within the 100-meter geo-fence. The server approves the start request, sets `actual_start_time` to 2025-11-10 10:30:15, `start_latitude` to 39.7817, `start_longitude` to -89.6501, and changes status to "in_progress." Mike completes the checklist and records one incident. At 11:15 AM, he taps "Submit Visit." The server sets `actual_end_time` to 2025-11-10 11:15:42, calculates duration as 45 minutes 27 seconds, and changes status to "submitted." The visit now appears in Supervisor Jane's review queue.

4.8. Checklist Management Module

Purpose: Provide standardized data collection templates for inspections with various question types and validation rules.

Key Entities:

SQL

```sql
CREATE TABLE Checklist (
    id BIGSERIAL PRIMARY KEY,
```

```
    name VARCHAR(255) NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE ChecklistItem (
    id BIGSERIAL PRIMARY KEY,
    checklist_id BIGINT REFERENCES Checklist(id) ON DELETE CASCADE,
    question_text TEXT NOT NULL,
    response_type VARCHAR(50) NOT NULL,
    -- 'yes_no', 'text', 'number', 'rating', 'multiple_choice', 'date',
'photo', 'video', 'audio'
    options JSONB, -- For multiple_choice: ["Option 1", "Option 2", "Option
3"]
    is_required BOOLEAN DEFAULT false,
    display_order INTEGER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE ChecklistResponse (
    id BIGSERIAL PRIMARY KEY,
    visit_id BIGINT REFERENCES Visit(id) ON DELETE CASCADE,
    checklist_item_id BIGINT REFERENCES ChecklistItem(id),
    response_value TEXT, -- Stores the answer (text, number, selected
option, file path)
    start_time TIMESTAMP, -- When the question was presented to the inspector
    end_time TIMESTAMP, -- When the answer was saved
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_checklist_response_visit ON ChecklistResponse(visit_id);
```

Business Logic:

Checklists provide a standardized, reusable framework for data collection during inspections. PMs and Supervisors create checklist templates with multiple items (questions), each having a specific response type. Checklists are assigned to visits, and inspectors must complete all required items before they can submit the visit.

Response Types: The system supports nine response types to accommodate diverse data collection needs. **Yes/No** presents a binary choice (stored as "yes" or "no"). **Text** allows free-

form text entry (stored as string). **Number** requires numeric input (stored as string representation of number, validated as numeric). **Rating** presents a scale (e.g., 1-5 stars, stored as "1", "2", "3", "4", or "5"). **Multiple Choice** presents predefined options stored in the `options` JSONB field (stored as the selected option text). **Date** presents a date picker (stored as ISO date string "YYYY-MM-DD"). **Photo** allows the inspector to capture or upload an image (stored as file path). **Video** allows video capture (stored as file path). **Audio** allows audio recording (stored as file path).

Granular Time Tracking: Each `ChecklistResponse` record includes `start_time` and `end_time` timestamps. When the mobile app displays a checklist item to the inspector, it records the current timestamp as `start_time`. When the inspector saves their answer (by tapping "Next" or "Save"), the app records the current timestamp as `end_time`. The duration (end_time - start_time) represents the time the inspector spent on that specific question. This granular data enables detailed performance analytics, such as "On average, inspectors spend 45 seconds on the 'Number of Fire Extinguishers' question" or "Inspector Mike takes 2x longer than the team average on photo questions."

Configurable Resumability: The system-wide setting `AllowResumableVisits` (boolean, default: true) controls checklist behavior when a visit is interrupted. **If true (default)**: Partially completed checklist responses are saved as drafts. If an inspector starts a visit, answers 5 out of 10 questions, and then exits the app (or the app crashes, or network is lost), those 5 responses are preserved in the database. When the inspector re-opens the visit, the app loads the existing responses and allows the inspector to continue from where they left off. **If false**: If an inspector exits a visit before submission, all checklist responses for that visit are deleted (hard delete). When the inspector re-opens the visit, the checklist is blank, and they must start from the beginning. This enforces a stricter, single-session inspection protocol where partial data is not trusted.

Validation Rules:

- All items marked `is_required = true` must have a non-null, non-empty `response_value` before visit submission
- Response value must match the expected response type (e.g., numeric value for 'number', valid date for 'date')
- For multiple_choice, response_value must be one of the options defined in the `options` JSONB field
- For photo/video/audio, response_value must be a valid file path pointing to an uploaded file

Use Case Example:

The PM creates a checklist named "Standard Health Inspection" with 12 items. Item 1: "Are all food storage areas clean?" (Yes/No, Required, Order 1). Item 2: "Number of hand-

washing stations" (Number, Required, Order 2). Item 3: "Overall cleanliness rating" (Rating 1-5, Required, Order 3). Item 4: "Photo of kitchen area" (Photo, Required, Order 4). Item 5: "Additional notes" (Text, Not Required, Order 5). Supervisor Jane assigns this checklist to a visit for Inspector Mike. Mike starts the visit at 10:30:15. The app displays Item 1 at 10:30:20 (start_time). Mike reads the question, looks around, and taps "Yes" at 10:30:35 (end_time). Duration: 15 seconds. The app displays Item 2 at 10:30:36 (start_time). Mike counts the stations and enters "3" at 10:31:05 (end_time). Duration: 29 seconds. Mike continues through all items. When he attempts to submit the visit, the app validates that all required items have responses. Item 5 (Additional notes) is empty, but it's not required, so submission proceeds.

4.9. Incident Management Module

Purpose: Record, document, and manage defects, violations, or issues discovered during inspections with comprehensive multi-media evidence.

Key Entities:

SQL

```
CREATE TABLE Incident (
    id BIGSERIAL PRIMARY KEY,
    visit_id BIGINT REFERENCES Visit(id) ON DELETE CASCADE,
    inspector_id BIGINT REFERENCES User(id),
    title VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    defect_type VARCHAR(100) NOT NULL,
    -- 'safety_violation', 'equipment_failure', 'structural_damage',
'cleanliness_issue',
    -- 'documentation_missing', 'other'
    severity VARCHAR(50) NOT NULL, -- 'critical', 'high', 'medium', 'low'
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
    location_method VARCHAR(50),
    -- 'gps', 'wifi_triangulation', 'ble_beacon', 'manual',
'facility_default'
    indoor_location_data JSONB,
    -- {"floor_level": "-2", "building_section": "North Wing",
"room_number": "B-204",
    --  "beacon_id": "BEACON-4523", "confidence_score": 0.85}
    accuracy_meters DECIMAL(8, 2), -- GPS accuracy in meters
    recorded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
```

```
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE IncidentAttachment (
    id BIGSERIAL PRIMARY KEY,
    incident_id BIGINT REFERENCES Incident(id) ON DELETE CASCADE,
    file_path VARCHAR(500) NOT NULL,
    file_name VARCHAR(255) NOT NULL,
    file_type VARCHAR(50) NOT NULL, -- 'image', 'video', 'audio', 'document'
    mime_type VARCHAR(100) NOT NULL, -- 'image/jpeg', 'video/mp4',
'audio/mpeg', 'application/pdf'
    file_size_bytes BIGINT NOT NULL,
    duration_seconds INTEGER, -- For video/audio files
    uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    created_by BIGINT REFERENCES User(id),
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id)
);

CREATE INDEX idx_incident_visit ON Incident(visit_id);
CREATE INDEX idx_incident_severity ON Incident(severity);
```

Business Logic:

Incidents are critical findings that require comprehensive documentation and multi-stage review. During a visit, an inspector can create multiple incident records. Each incident must include a title, description, defect type, severity level, and at least one attachment (configurable). The system automatically captures location data using a hybrid positioning strategy.

Multi-Media Evidence System: The `IncidentAttachment` table replaces the previous image-only approach with support for multiple media types. **Images** (JPEG, PNG, HEIC) with maximum size 10 MB per file are used for visual documentation of defects. **Videos** (MP4, MOV, AVI) with maximum size 100 MB per file and maximum duration 5 minutes are used for dynamic evidence such as water leaks or equipment malfunctions. **Audio** recordings (MP3, WAV, M4A) with maximum size 50 MB per file and maximum duration 10 minutes are used for voice notes describing complex issues. **Documents** (PDF, DOCX) with maximum size 20 MB per file are used for attaching related documentation such as manufacturer bulletins or previous inspection reports. The mobile app compresses images and videos before upload to reduce bandwidth usage. In offline mode, attachments are queued locally and uploaded during synchronization.

Hybrid Positioning Strategy: The system implements a tiered approach to capture incident location, adapting to the availability of positioning technologies. **Tier 1 - GPS (Outdoor)**: When the mobile app detects a strong GPS signal (accuracy < 10 meters), it uses

the device's GPS coordinates and sets `location_method` to "gps." The `latitude` and `longitude` fields are populated with the GPS coordinates, and `accuracy_meters` is set to the reported GPS accuracy. **Tier 2 - Indoor Positioning (Indoor/Underground)**: When GPS is unavailable or weak (accuracy > 50 meters), the app attempts to use indoor positioning. If Wi-Fi access points are detected and the facility has configured Wi-Fi positioning, the app queries the positioning service, receives coordinates relative to the facility, and sets `location_method` to "wifi_triangulation." If BLE beacons are detected, the app queries the beacon positioning service and sets `location_method` to "ble_beacon." The `indoor_location_data` JSONB field stores additional context such as floor level, building section, room number, beacon ID, and confidence score. **Tier 3 - Manual Tagging (Fallback)**: If no positioning is available, the app prompts the inspector to manually tag the location. If the facility has uploaded floor plans, the app displays the floor plan image and allows the inspector to tap their location. The app stores the tapped coordinates relative to the floor plan image and sets `location_method` to "manual." The `indoor_location_data` stores the floor level and tapped coordinates. If no floor plan is available, the app uses the facility's default `geolocation` coordinates, sets `location_method` to "facility_default," and flags the incident for manual verification.

Validation Rules:

- Visit must exist and be in "in_progress" status
- Title and description are required
- Defect type must be from predefined list
- Severity must be one of: 'critical', 'high', 'medium', 'low'
- At least one attachment is required (configurable via system setting `MandatoryIncidentAttachments`)
- File size must not exceed limits for each file type
- Video and audio duration must not exceed limits

Use Case Example:

Inspector Mike is conducting a visit at "Underground Water Treatment Facility" in the basement level (-2). He discovers a broken pipe actively leaking water. He taps "Add Incident" in the mobile app. The app attempts GPS positioning but receives no signal (underground). The app detects BLE beacon "BEACON-BASEMENT-2-NORTH" and queries the positioning service, which returns coordinates and confidence score 0.85. Mike enters title "Active Water Leak from Pipe," description "Pipe joint has separated, water spraying onto floor, approximately 5 gallons per minute," defect_type "equipment_failure," severity "critical." Mike records a 30-second video showing the active leak. The app captures the video, compresses it from 45 MB to 18 MB, and queues it for upload. The incident record is saved with `location_method` = "ble_beacon," `indoor_location_data` = {"floor_level": "-2",

"building_section": "North", "beacon_id": "BEACON-BASEMENT-2-NORTH", "confidence_score": 0.85}, and the video attachment is linked. When Mike finishes the visit and returns to an area with network coverage, the app synchronizes, uploading the video to cloud storage and updating the incident record on the server.

4.10. Pre-Visit Issue Management Module

Purpose: Handle logistical problems that prevent an inspection from starting, enabling rapid resolution and operational continuity.

Key Entities:

SQL

```sql
CREATE TABLE PreVisitIssue (
    id BIGSERIAL PRIMARY KEY,
    visit_id BIGINT REFERENCES Visit(id) ON DELETE CASCADE,
    issue_type VARCHAR(100) NOT NULL,
    -- 'incorrect_location', 'facility_closed', 'access_denied',
'safety_concern', 'other'
    description TEXT NOT NULL,
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
    reported_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    resolved_at TIMESTAMP,
    resolution_notes TEXT,
    resolved_by BIGINT REFERENCES User(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id)
);

CREATE TABLE PreVisitIssueAttachment (
    id BIGSERIAL PRIMARY KEY,
    pre_visit_issue_id BIGINT REFERENCES PreVisitIssue(id) ON DELETE CASCADE,
    file_path VARCHAR(500) NOT NULL,
    file_name VARCHAR(255) NOT NULL,
    file_type VARCHAR(50) NOT NULL, -- 'image', 'video', 'audio', 'document'
    mime_type VARCHAR(100) NOT NULL,
    file_size_bytes BIGINT NOT NULL,
    duration_seconds INTEGER,
    uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    created_by BIGINT REFERENCES User(id)
);
```

```
CREATE INDEX idx_pre_visit_issue_visit ON PreVisitIssue(visit_id);
```

Business Logic:

Pre-Visit Issues are distinct from Incidents. They represent situations where an inspector arrives at a facility location but cannot begin the inspection due to logistical failures rather than facility defects. Reporting a Pre-Visit Issue automatically puts the visit on hold and notifies the Supervisor for immediate resolution.

Multi-Media Support for Pre-Visit Issues: Similar to incidents, pre-visit issues now support multi-media attachments through the `PreVisitIssueAttachment` table. This enables inspectors to provide comprehensive evidence of the problem. For example, if the issue type is "incorrect_location," the inspector can attach a photo showing the actual location (empty lot, different building, etc.). If the issue type is "facility_closed," the inspector can attach a photo of the locked door with a "Closed" sign. If the issue type is "access_denied," the inspector can record an audio note describing the interaction with facility personnel.

Pre-Visit Issue Workflow:

1. Inspector arrives at the scheduled facility location
2. Inspector encounters a problem preventing visit start (wrong address, facility closed, access denied, unsafe conditions)
3. Inspector taps "Report Issue" button in the mobile app (available before starting the visit)
4. Inspector selects issue type from dropdown, enters description, and attaches evidence (photos, videos, audio)
5. Mobile app captures inspector's current GPS coordinates
6. System automatically changes visit status from "assigned" to "on_hold"
7. System sends immediate push notification and email to the Supervisor
8. Supervisor reviews the issue and attached evidence
9. Supervisor decides on resolution:

- **Cancel** the visit if the facility no longer exists or is permanently closed
- **Reschedule** the visit for a later date if the facility is temporarily closed
- **Update** facility information (correct address, update coordinates) and re-assign the visit
- **Resolve** the issue by contacting facility management to arrange access

1. Supervisor enters resolution notes and marks the issue as resolved

2. If visit is re-assigned, status changes back to "assigned" and inspector receives notification

Validation Rules:

- Visit must exist and be in "assigned" status (not yet started)
- Issue type must be from predefined list
- Description is required
- At least one attachment is recommended but not required

Use Case Example:

Inspector Mike is assigned a visit to "Springfield Warehouse" at 123 Oak Street for November 10, 2025. He drives to the address and finds an empty lot with no building. He taps "Report Issue," selects issue_type "incorrect_location," enters description "Address shows empty lot, no warehouse building present," and takes a photo of the empty lot. The app captures his GPS coordinates (lat: 39.7850, lon: -89.6550) and submits the pre-visit issue. The system changes the visit status to "on_hold" and sends a notification to Supervisor Jane. Jane receives the notification, reviews the photo, and investigates. She discovers the warehouse moved to 456 Pine Street last month. She updates the facility's address and geolocation in the system, enters resolution_notes "Facility relocated to 456 Pine Street. Facility record updated. Visit re-assigned," marks the issue as resolved, and changes the visit status back to "assigned." Mike receives a notification with the corrected address and navigates to the new location.

4.11. Smart Route & Navigation Module

Purpose: Optimize inspector daily routes using geospatial algorithms and provide turn-by-turn navigation to facility locations.

Business Logic:

The Smart Route Module is a mobile app feature that solves the Traveling Salesperson Problem (TSP) to generate the most efficient route for an inspector's daily visits. This minimizes travel time, reduces fuel costs, and maximizes the number of visits an inspector can complete in a shift.

Smart Route Workflow:

1. Inspector opens the mobile app at the start of their shift
2. App displays all visits assigned for today's date, initially ordered by creation time or facility name
3. Inspector taps "Plan My Route" button

4. App extracts the geolocation coordinates of all assigned facilities
5. App sends coordinates to a routing optimization API (e.g., Google Maps Directions API with waypoint optimization, Mapbox Optimization API, or an internal service using OR-Tools)
6. App includes the inspector's current location as the starting point
7. App specifies priority handling: critical visits must be completed before normal visits
8. Routing API solves the TSP and returns the optimal visit order and estimated travel times
9. App re-orders the visit list according to the optimized sequence
10. App displays total estimated travel time and total estimated route distance
11. Inspector taps on the first visit in the optimized list
12. App launches turn-by-turn navigation using the device's native maps app or an integrated navigation SDK
13. Inspector completes the visit and returns to the app
14. App automatically highlights the next visit in the sequence and offers to start navigation

Priority Handling: Critical visits are always prioritized. The routing algorithm first sequences all critical visits in the most efficient order, then sequences all normal visits. For example, if an inspector has 2 critical visits and 8 normal visits, the route will be: Current Location → Critical Visit 1 → Critical Visit 2 → Normal Visit A → Normal Visit B → ... → Normal Visit H.

Offline Support: The Smart Route feature requires network connectivity to call the routing API. However, once the route is calculated, the app caches the optimized sequence locally. If the inspector goes offline during the day, they can still follow the cached route sequence. For turn-by-turn navigation in offline mode, the app can use pre-downloaded offline map tiles (e.g., Mapbox offline maps, OpenStreetMap tiles stored locally).

Use Case Example:

Inspector Mike starts his shift on November 10, 2025, at 8:00 AM. He has 10 visits assigned for today: 2 critical and 8 normal. He opens the app and taps "Plan My Route." The app sends his current location (lat: 39.7800, lon: -89.6500) and the 10 facility locations to the Mapbox Optimization API. The API returns the optimal sequence: Critical Visit at Facility A (5 km north) → Critical Visit at Facility B (3 km east) → Normal Visit at Facility C (2 km south) → ... → Normal Visit at Facility J (4 km west of Facility I). The app displays "Total Distance: 45 km, Estimated Travel Time: 1 hour 15 minutes." Mike taps on Facility A, and the app launches turn-by-turn navigation. He arrives, completes the visit, and returns to the app.

The app highlights Facility B and shows "Next Stop: 3 km, 8 minutes." Mike taps "Navigate" and proceeds.

4.12. Offline Mode & Data Synchronization Module

Purpose: Enable full mobile app functionality without network connectivity and ensure robust data synchronization when connectivity is restored.

Business Logic:

The mobile app is designed with an **offline-first architecture**, recognizing that inspectors frequently work in areas with poor or no network coverage (basements, remote facilities, underground locations). All critical app functions must work offline, with data stored locally and synchronized when connectivity is restored.

Offline Data Storage: When an inspector logs into the mobile app with network connectivity (typically at the start of their shift), the app performs a **full data sync**, downloading all necessary data to the device's local database (SQLite or Realm). This includes all visits assigned to the inspector for the current date and upcoming dates (configurable, default: next 7 days), all facility records for those visits (name, address, geolocation, floor plans), all checklists assigned to those visits (checklist items, question text, response types, options), all compliance rules from the knowledge base (for reference), and the inspector's user profile and configuration settings.

Offline Operations: While offline, the inspector can perform all core functions. **Start Visit**: The app validates geo-fence using the locally cached facility geolocation. If validation passes, the app creates a local visit start record with timestamp and GPS coordinates. **Complete Checklist**: The app presents checklist items from the local database. The inspector's responses are saved to the local database with timestamps. **Record Incidents**: The app creates incident records in the local database. Photos, videos, and audio are saved to the device's local storage. **Report Pre-Visit Issues**: The app creates pre-visit issue records locally with attachments. **Submit Visit**: The app marks the visit as "submitted" in the local database and queues it for synchronization.

Data Synchronization Workflow: When network connectivity is restored (detected by the app through periodic connectivity checks), the app initiates an automatic sync process:

1. **Upload Phase**: The app identifies all locally created or modified records (visits started, checklist responses, incidents, attachments) that have not been synchronized. The app uploads these records to the server in order of creation timestamp. For large files (videos, high-resolution images), the app uploads them in chunks to handle intermittent connectivity. The app marks each record as "synced" in the local database upon successful server acknowledgment.

2. **Download Phase**: The app requests the latest state from the server for all visits assigned to the inspector. If any visit has been cancelled or re-assigned while the inspector was offline, the server returns the updated status. The app updates its local database to reflect the server state.
3. **Conflict Resolution**: The server is the single source of truth. If the app attempts to sync a completed visit that was cancelled on the server while the inspector was offline, the server rejects the submission and returns an error. The app displays a message to the inspector: "Visit was cancelled while you were offline. Your work has been discarded." The app deletes the local visit data. This prevents inspectors from wasting time on cancelled visits, though it is a rare edge case.

Impact on Unavailability Detection: The system tracks a `last_seen` timestamp for each inspector, updated every time the mobile app sends data to the server (location update, sync request, API call). If `last_seen` is older than the configured threshold (default: 30 minutes), the system checks if the inspector's app has declared itself in "offline mode" (a flag sent by the app when it detects no connectivity). If the app is in offline mode, the system does not flag the inspector as unavailable, recognizing that they are working in a no-coverage area. If the app is not in offline mode (meaning it should have connectivity), the system flags the visit as "interrupted" and notifies the Supervisor.

Use Case Example:

Inspector Mike starts his shift at 8:00 AM with full network connectivity. The app syncs, downloading 8 visits for today. At 9:00 AM, Mike arrives at an underground facility. He descends to the basement level and loses all network signal. The app detects no connectivity and enters offline mode. Mike starts the visit (geo-fence validated using cached facility location), completes the checklist (responses saved locally), records 2 incidents with photos (saved to local storage), and submits the visit (marked as "submitted" locally). At 10:30 AM, Mike exits the facility and returns to ground level. The app detects network connectivity and automatically initiates sync. The app uploads the visit start record, 15 checklist responses, 2 incident records, and 4 photos (total 12 MB). The server acknowledges receipt, and the app marks all records as synced. Mike's next visit appears in the app with the latest data from the server.

4.13. Indoor Positioning Module

Purpose: Provide accurate location capture for incidents and visit starts in GPS-denied environments such as indoor and underground facilities.

Business Logic:

The Indoor Positioning Module implements a hybrid, tiered strategy to capture location data when GPS is unavailable or unreliable. This ensures that all incidents and visit starts

have associated location data, even in challenging environments.

Positioning Technologies:

The system supports multiple indoor positioning technologies, with the choice determined by facility infrastructure and organizational budget. **Wi-Fi Triangulation** uses existing Wi-Fi access points with known locations to triangulate the mobile device's position, providing accuracy of 5-15 meters with low infrastructure cost. **Bluetooth Low Energy (BLE) Beacons** are small, battery-powered devices placed throughout a facility that broadcast their location, providing accuracy of 1-5 meters with medium infrastructure cost. **Ultra-Wideband (UWB)** is a high-precision radio technology requiring specialized anchors and tags, providing accuracy of 10-30 cm with high infrastructure cost. **Esri ArcGIS Indoors** is an enterprise indoor mapping and positioning platform that integrates with existing Esri infrastructure, providing accuracy of 3-5 meters with high licensing cost. **Manual Floor Plan Tagging** is a zero-infrastructure fallback where inspectors manually select their location on uploaded floor plan images.

Tiered Positioning Logic:

The mobile app implements the following decision tree when capturing location for an incident or visit start:

1. **Attempt GPS**: The app requests the device's GPS coordinates. If GPS accuracy is better than 10 meters, use GPS coordinates, set `location_method` = "gps", and set `accuracy_meters` to the reported GPS accuracy.
2. **Attempt Indoor Positioning**: If GPS accuracy is worse than 50 meters or GPS is unavailable, check if indoor positioning is enabled for this facility (configured in the facility record). If Wi-Fi positioning is configured, scan for Wi-Fi access points, query the positioning service with detected SSIDs and signal strengths, receive coordinates, set `location_method` = "wifi_triangulation", and store additional data in `indoor_location_data` JSONB field. If BLE positioning is configured, scan for BLE beacons, query the positioning service with detected beacon IDs and signal strengths, receive coordinates and floor level, set `location_method` = "ble_beacon", and store beacon ID, floor level, and confidence score in `indoor_location_data`.
3. **Manual Tagging**: If no indoor positioning is available, check if the facility has uploaded floor plans. If floor plans exist, display the floor plan image for the inspector to select their current floor level, allow the inspector to tap their location on the floor plan image, capture the tapped coordinates relative to the image, set `location_method` = "manual", and store floor level and relative coordinates in `indoor_location_data`. If no floor plans exist, use the facility's default `geolocation` coordinates (outdoor location), set `location_method` = "facility_default", flag the incident/visit for manual verification, and notify the Supervisor.

Facility Configuration for Indoor Positioning:

The PM or Supervisor configures indoor positioning for each facility through the facility management interface. Configuration options include: Indoor Positioning Enabled (yes/no), Positioning Technology (Wi-Fi, BLE, Esri, Manual), Wi-Fi Access Point List (if Wi-Fi selected): SSID and coordinates of each AP, BLE Beacon List (if BLE selected): Beacon ID, floor level, and coordinates of each beacon, and Esri Indoor Map ID (if Esri selected).

Use Case Example:

Inspector Mike is assigned a visit to "City Hospital" which has a 5-story building with 2 basement levels. The PM has configured BLE positioning for this facility, with 20 beacons deployed across all floors. Mike enters the building and descends to basement level -2 to inspect the mechanical room. He discovers a critical incident: "Steam pipe leaking." He taps "Add Incident" in the app. The app attempts GPS but receives no signal. The app scans for BLE beacons and detects "BEACON-HOSPITAL-B2-MECH-01" with strong signal strength. The app queries the positioning service, which returns coordinates (relative to the building's origin point), floor level "-2", building section "Mechanical Room", and confidence score 0.92. The app sets `location_method` = "ble_beacon" and `indoor_location_data` = {"floor_level": "-2", "building_section": "Mechanical Room", "beacon_id": "BEACON-HOSPITAL-B2-MECH-01", "confidence_score": 0.92}. Mike completes the incident with photos and description. Later, when the PM views the incident on the GIS map, the system plots it on the hospital's floor plan for basement level -2 in the mechanical room area, providing precise location context.

4.14. Settings & Configuration Module

Purpose: Provide centralized administrative control over system-wide operational parameters and user-level preferences.

Key Entities:

SQL

```
CREATE TABLE SystemConfiguration (
    id BIGSERIAL PRIMARY KEY,
    config_key VARCHAR(100) UNIQUE NOT NULL,
    config_value TEXT NOT NULL,
    config_type VARCHAR(50) NOT NULL, -- 'boolean', 'integer', 'string',
'json'
    description TEXT,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_by BIGINT REFERENCES User(id)
);
```

Business Logic:

The Settings & Configuration Module is a PM-only interface for managing system-wide operational parameters. All configuration values are stored in the `SystemConfiguration` table as key-value pairs. Changes to settings take effect immediately and apply to all future operations.

System-Wide Configuration Categories:

Compliance Settings:

- `AuditorModuleEnabled` (boolean, default: false): Enable or disable the Auditor review step in the visit workflow
- `LicenseExpiryAlertThresholds` (string, default: "90,60,30,7"): Comma-separated list of days before expiry to send alerts
- `MandatoryIncidentAttachments` (boolean, default: true): Require at least one attachment per incident

Operational Settings:

- `SupervisorInspectorMinRatio` (integer, default: 5): Minimum number of inspectors per supervisor
- `SupervisorInspectorMaxRatio` (integer, default: 10): Maximum number of inspectors per supervisor
- `VisitAutoInterruptionTime` (integer, default: 30): Minutes of inactivity before flagging visit as interrupted
- `GeoFenceRadius` (integer, default: 100): Meters around facility for location validation
- `MaxVisitsPerInspectorPerDay` (integer, default: 15): Maximum visits assignable to one inspector on one date

Data Management Settings:

- `DataRetentionPeriod` (integer, default: 90): Days before permanently purging soft-deleted records
- `AllowResumableVisits` (boolean, default: true): Allow inspectors to resume partially completed checklists
- `LocationTrackingInterval` (integer, default: 30): Seconds between location updates from mobile app
- `LocationDataRetentionDays` (integer, default: 30): Days to retain inspector location tracking data

Notification Settings:

- `EmailNotificationsEnabled` (boolean, default: true): Enable email notifications

- `SMSNotificationsEnabled` (boolean, default: false): Enable SMS notifications (requires integration)
- `PushNotificationsEnabled` (boolean, default: true): Enable mobile push notifications

File Upload Settings:

- `MaxAttachmentSizeImage` (integer, default: 10): Maximum image file size in MB
- `MaxAttachmentSizeVideo` (integer, default: 100): Maximum video file size in MB
- `MaxAttachmentSizeAudio` (integer, default: 50): Maximum audio file size in MB
- `MaxAttachmentSizeDocument` (integer, default: 20): Maximum document file size in MB
- `MaxVideoDurationSeconds` (integer, default: 300): Maximum video duration (5 minutes)
- `MaxAudioDurationSeconds` (integer, default: 600): Maximum audio duration (10 minutes)

Indoor Positioning Settings:

- `IndoorPositioningEnabled` (boolean, default: false): Enable indoor positioning features
- `IndoorPositioningProvider` (string, default: "manual"): Options: 'wifi', 'ble', 'esri', 'manual'

Use Case Example:

The PM accesses the Settings & Configuration module and changes `VisitAutoInterruptionTime` from 30 to 45 minutes, recognizing that inspectors in their region often work in areas with intermittent connectivity. The PM also enables the Auditor Module by setting `AuditorModuleEnabled` to true. The PM sets `MandatoryIncidentAttachments` to true to enforce evidence requirements. All these changes take effect immediately. Future visit interruption checks use the 45-minute threshold, future completed visits enter the Auditor review queue, and future incident submissions without attachments are rejected by the mobile app.

4.15. Dashboard, Metrics & Reporting Module

Purpose: Provide comprehensive analytics, performance metrics, and compliance reporting across all system roles.

Business Logic:

The Dashboard & Reporting Module aggregates data from all system entities to provide actionable insights for decision-making. Different roles have access to different dashboard views and reports based on their permissions.

Dashboard Views by Role:

Owner Dashboard:

- System-wide visit completion rate (completed visits / total visits)

- Total facilities under management
- Total active licenses and count of expired licenses
- Total incidents by severity (pie chart)
- GIS map view showing all facilities, incidents, and inspector locations
- Compliance dashboard showing total potential fines and top violated rules
- Trend charts showing visits, incidents, and compliance over time (last 30/60/90 days)

PM Dashboard:

- All metrics from Owner dashboard
- Supervisor performance comparison (completion rate, average review time, team incident rate)
- Inspector performance ranking (visits completed, average visit duration, incidents recorded)
- Service area coverage analysis (percentage of facilities inspected in last 30/60/90 days)
- License expiry alerts (upcoming expiries in next 7/30/90 days)
- System health metrics (average sync time, offline mode usage, failed uploads)

Supervisor Dashboard:

- Team performance metrics (completion rate, average visit duration, total incidents)
- Individual inspector performance (visits completed, average time per visit, incidents per visit)
- Pending reviews queue (count of visits awaiting supervisor review)
- Service area compliance score (calculated from incident severity and frequency)
- Availability toggle (prominent UI element to set available/unavailable status)
- Team schedule view (calendar showing all inspector shifts and assigned visits)

Inspector Dashboard:

- Personal performance metrics (visits completed this week/month, average visit duration)
- Assigned visits for today and upcoming days (list and calendar views)
- Pending visits (assigned but not started)
- In-progress visits (started but not submitted)
- Personal time tracking (time spent per checklist item, compared to team average)

Auditor Dashboard:

- Pending incident reviews queue (count of visits awaiting auditor review)
- Incidents reviewed this week/month (approved vs. rejected)
- Compliance dashboard (incidents by rule category, severity distribution)
- Average review time per incident

Standard Reports:

The system provides pre-built reports that can be generated on-demand or scheduled for automatic delivery via email. **Visit Report by Inspector** lists all visits for a selected inspector in a date range, showing facility name, date, duration, incidents count, and status. **Visit Report by Supervisor** lists all visits for a supervisor's team in a date range, aggregated by inspector. **Visit Report by Service Area** lists all visits in a selected service area in a date range, aggregated by facility. **Inspector Performance Report** calculates average visit duration, average time per checklist item, incidents recorded vs. team average, and completion rate for each inspector. **Supervisor Performance Report** calculates team completion rate, average review time, team incident rate, and service area coverage for each supervisor. **Facility Compliance Score Report** calculates a compliance score for each facility based on incident count, severity, and recency, highlighting high-risk facilities. **Route Efficiency Report** compares the smart route suggestion with the actual path taken by inspectors (from location tracking data), calculating deviation percentage and identifying optimization opportunities. **License Expiry Report** lists all licenses expiring in the next 30/60/90 days, grouped by service area and supervisor. **Incident Summary Report** aggregates incidents by severity, defect type, facility type, and service area for a selected date range. **Compliance Violations Report** lists all incidents linked to compliance rules, showing potential fines and resolution status.

Use Case Example:

The PM accesses the Dashboard and views the "Supervisor Performance Comparison" widget. The widget shows that Supervisor Jane's team has a 95% completion rate with an average review time of 12 minutes, while Supervisor Bob's team has an 87% completion rate with an average review time of 22 minutes. The PM clicks on Bob's metrics to drill down and discovers that Bob has 15 pending reviews, some over 5 days old. The PM contacts Bob to discuss workload and offers to assign a temporary delegate. The PM then generates the "Inspector Performance Report" for Jane's team and discovers that Inspector Mike has an average visit duration of 45 minutes, while the team average is 60 minutes. The PM recognizes Mike as a high-performer and considers him for a mentorship role.

4.16. Audit Trail & History Module

Purpose: Provide a comprehensive, immutable log of all system actions for accountability, compliance, and forensic analysis.

Key Entities:

SQL

```sql
CREATE TABLE AuditTrail (
    id BIGSERIAL PRIMARY KEY,
    entity_type VARCHAR(100) NOT NULL, -- 'User', 'Facility', 'Visit',
'Incident', etc.
    entity_id BIGINT NOT NULL, -- ID of the affected record
    action VARCHAR(50) NOT NULL, -- 'CREATE', 'UPDATE', 'DELETE', 'APPROVE',
'REJECT'
    old_value JSONB, -- Previous state of the record (for UPDATE and DELETE)
    new_value JSONB, -- New state of the record (for CREATE and UPDATE)
    performed_by BIGINT NOT NULL REFERENCES User(id), -- User who performed
the action
    performed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ip_address VARCHAR(45), -- IP address of the request
    user_agent TEXT -- Browser/app user agent string
);

CREATE INDEX idx_audit_trail_entity ON AuditTrail(entity_type, entity_id);
CREATE INDEX idx_audit_trail_user ON AuditTrail(performed_by);
CREATE INDEX idx_audit_trail_timestamp ON AuditTrail(performed_at);
```

Business Logic:

The Audit Trail Module automatically logs every Create, Update, and Delete operation on all major system entities. This provides a complete, tamper-proof history of all system actions, supporting accountability, compliance requirements, and forensic investigations.

Automatic Audit Logging: Audit logging is implemented via database triggers or application-level hooks (depending on the chosen architecture). When any record in a major table (User, Facility, License, Visit, Incident, etc.) is created, updated, or deleted, an audit trail entry is automatically generated. The entry captures the entity type (table name), entity ID (primary key), action type (CREATE, UPDATE, DELETE), old value (for UPDATE and DELETE, the complete record state before the change, stored as JSONB), new value (for CREATE and UPDATE, the complete record state after the change, stored as JSONB), the user who performed the action (from the authenticated session), the timestamp, the IP address of the request, and the user agent string.

Non-Nullable Attribution: The `performed_by` column is a non-nullable foreign key to the `User` table. This ensures that every action is attributable to a specific user account. System-initiated actions (e.g., automated license expiry status updates) are attributed to a special system user account (e.g., user_id = 1, username = "SYSTEM"). The column type is `BIGINT` to be future-proof for large user bases.

Audit Trail Viewing: The PM has full access to the audit trail for all entities and all users. Supervisors can view the audit trail for actions they performed or actions on entities within their scope (facilities in their service areas, visits for their team). Inspectors can view the audit trail for their own actions. The audit trail interface provides filtering by entity type, entity ID, action type, user, and date range. The interface displays the old and new values in a diff view, highlighting changes.

Use Case Example:

On November 10, 2025, at 10:30 AM, Supervisor Jane updates the address of "Ocean View Restaurant" from "123 Main Street" to "456 Ocean Avenue." The system automatically creates an audit trail entry: entity_type = "Facility", entity_id = 123, action = "UPDATE", old_value = {"id": 123, "name": "Ocean View Restaurant", "address": "123 Main Street", "geolocation": "POINT(-89.6501 39.7817)", ...}, new_value = {"id": 123, "name": "Ocean View Restaurant", "address": "456 Ocean Avenue", "geolocation": "POINT(-89.6505 39.7820)", ...}, performed_by = 5 (Jane's user ID), performed_at = "2025-11-10 10:30:15", ip_address = "192.168.1.50", user_agent = "Mozilla/5.0...". Later, the PM reviews the audit trail for this facility and sees the complete history of changes, including who changed the address and when.

---

5. Database Schema

5.1. Schema Overview

The Inspection Management System database is implemented using PostgreSQL 14+ with the PostGIS extension for geospatial data types and operations. The schema consists of 25 core tables organized into logical groups: User Management, Geography, Facility Management, Compliance, Scheduling, Visit Management, Checklist Management, Incident Management, Configuration, and Audit. All tables implement soft delete patterns with `deleted_at` and `deleted_by` columns. All tables include audit fields (`created_at` , `updated_at` , `created_by` , `updated_by`) for complete traceability.

5.2. Core Tables

The following SQL definitions provide the complete database schema. All tables use `BIGSERIAL` for primary keys to support large-scale deployments. Foreign key constraints enforce referential integrity with appropriate cascade rules.

TypeScript

```
-- ======================
-- USER MANAGEMENT TABLES
```

```
-- ======================

CREATE TABLE User (
    id BIGSERIAL PRIMARY KEY,
    username VARCHAR(100) UNIQUE NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(50) NOT NULL CHECK (role IN ('owner', 'pm', 'supervisor',
'inspector', 'auditor')),
    first_name VARCHAR(100) NOT NULL,
    last_name VARCHAR(100) NOT NULL,
    phone_number VARCHAR(20),
    is_active BOOLEAN DEFAULT true,
    is_available BOOLEAN DEFAULT true,
    supervisor_id BIGINT REFERENCES User(id) ON DELETE SET NULL,
    delegate_supervisor_id BIGINT REFERENCES User(id) ON DELETE SET NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE UserConfiguration (
    id BIGSERIAL PRIMARY KEY,
    user_id BIGINT UNIQUE REFERENCES User(id) ON DELETE CASCADE,
    two_factor_enabled BOOLEAN DEFAULT false,
    two_factor_secret VARCHAR(255),
    biometric_enabled BOOLEAN DEFAULT false,
    email_notifications BOOLEAN DEFAULT true,
    sms_notifications BOOLEAN DEFAULT false,
    push_notifications BOOLEAN DEFAULT true,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- ======================
-- GEOGRAPHY TABLES
-- ======================

CREATE TABLE City (
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    country VARCHAR(100) NOT NULL,
    state_province VARCHAR(100),
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id),
    UNIQUE(name, country, state_province)
);

CREATE TABLE ServiceArea (
    id BIGSERIAL PRIMARY KEY,
    city_id BIGINT REFERENCES City(id) ON DELETE CASCADE,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    boundary GEOMETRY(POLYGON, 4326),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE SupervisorServiceArea (
    id BIGSERIAL PRIMARY KEY,
    supervisor_id BIGINT REFERENCES User(id) ON DELETE CASCADE,
    service_area_id BIGINT REFERENCES ServiceArea(id) ON DELETE CASCADE,
    assigned_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    assigned_by BIGINT REFERENCES User(id),
    UNIQUE(supervisor_id, service_area_id)
);

-- ======================
-- FACILITY MANAGEMENT TABLES
-- ======================

CREATE TABLE Facility (
    id BIGSERIAL PRIMARY KEY,
    service_area_id BIGINT REFERENCES ServiceArea(id),
    name VARCHAR(255) NOT NULL,
    facility_type VARCHAR(100) NOT NULL,
    address VARCHAR(500) NOT NULL,
    geolocation GEOMETRY(POINT, 4326) NOT NULL,
    owner_name VARCHAR(255),
    owner_contact VARCHAR(255),
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE FacilityFloorPlan (
    id BIGSERIAL PRIMARY KEY,
    facility_id BIGINT REFERENCES Facility(id) ON DELETE CASCADE,
    floor_level VARCHAR(50) NOT NULL,
    floor_name VARCHAR(100),
    floor_plan_image_path VARCHAR(500),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE License (
    id BIGSERIAL PRIMARY KEY,
    facility_id BIGINT REFERENCES Facility(id) ON DELETE CASCADE,
    license_number VARCHAR(100) UNIQUE NOT NULL,
    license_type VARCHAR(100) NOT NULL,
    issue_date DATE NOT NULL,
    expiry_date DATE NOT NULL,
    issuing_authority VARCHAR(255),
    status VARCHAR(50) DEFAULT 'active' CHECK (status IN ('active',
'expired', 'suspended', 'revoked')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

-- ======================
-- COMPLIANCE TABLES
-- ======================

CREATE TABLE ComplianceRule (
    id BIGSERIAL PRIMARY KEY,
    category VARCHAR(100) NOT NULL,
    rule_code VARCHAR(50) UNIQUE,
    rule_name VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
```

```
    resolution TEXT NOT NULL,
    fine_amount DECIMAL(10, 2),
    severity VARCHAR(50) CHECK (severity IN ('critical', 'high', 'medium',
'low')),
    reference_url VARCHAR(500),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

-- ======================
-- SCHEDULING TABLES
-- ======================

CREATE TABLE Shift (
    id BIGSERIAL PRIMARY KEY,
    inspector_id BIGINT REFERENCES User(id) ON DELETE CASCADE,
    shift_date DATE NOT NULL,
    start_time TIME NOT NULL,
    end_time TIME NOT NULL,
    status VARCHAR(50) DEFAULT 'scheduled' CHECK (status IN ('scheduled',
'active', 'completed', 'cancelled')),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

-- ======================
-- VISIT MANAGEMENT TABLES
-- ======================

CREATE TABLE Visit (
    id BIGSERIAL PRIMARY KEY,
    facility_id BIGINT REFERENCES Facility(id),
    inspector_id BIGINT REFERENCES User(id),
    supervisor_id BIGINT REFERENCES User(id),
    checklist_id BIGINT REFERENCES Checklist(id),
    scheduled_date DATE NOT NULL,
    priority VARCHAR(50) DEFAULT 'normal' CHECK (priority IN ('normal',
'critical')),
    status VARCHAR(50) DEFAULT 'unassigned' CHECK (status IN ('unassigned',
'assigned', 'in_progress', 'on_hold', 'interrupted', 'submitted',
```

```
'under_review', 'under_approval', 'completed', 'cancelled')),
    actual_start_time TIMESTAMP,
    actual_end_time TIMESTAMP,
    start_latitude DECIMAL(10, 8),
    start_longitude DECIMAL(11, 8),
    notes TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

-- ======================
-- CHECKLIST TABLES
-- ======================

CREATE TABLE Checklist (
    id BIGSERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE ChecklistItem (
    id BIGSERIAL PRIMARY KEY,
    checklist_id BIGINT REFERENCES Checklist(id) ON DELETE CASCADE,
    question_text TEXT NOT NULL,
    response_type VARCHAR(50) NOT NULL CHECK (response_type IN ('yes_no',
'text', 'number', 'rating', 'multiple_choice', 'date', 'photo', 'video',
'audio')),
    options JSONB,
    is_required BOOLEAN DEFAULT false,
    display_order INTEGER NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE ChecklistResponse (
    id BIGSERIAL PRIMARY KEY,
    visit_id BIGINT REFERENCES Visit(id) ON DELETE CASCADE,
    checklist_item_id BIGINT REFERENCES ChecklistItem(id),
```

```
    response_value TEXT,
    start_time TIMESTAMP,
    end_time TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- ======================
-- INCIDENT TABLES
-- ======================

CREATE TABLE Incident (
    id BIGSERIAL PRIMARY KEY,
    visit_id BIGINT REFERENCES Visit(id) ON DELETE CASCADE,
    inspector_id BIGINT REFERENCES User(id),
    title VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    defect_type VARCHAR(100) NOT NULL,
    severity VARCHAR(50) NOT NULL CHECK (severity IN ('critical', 'high',
'medium', 'low')),
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
    location_method VARCHAR(50),
    indoor_location_data JSONB,
    accuracy_meters DECIMAL(8, 2),
    recorded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id),
    created_by BIGINT REFERENCES User(id),
    updated_by BIGINT REFERENCES User(id)
);

CREATE TABLE IncidentAttachment (
    id BIGSERIAL PRIMARY KEY,
    incident_id BIGINT REFERENCES Incident(id) ON DELETE CASCADE,
    file_path VARCHAR(500) NOT NULL,
    file_name VARCHAR(255) NOT NULL,
    file_type VARCHAR(50) NOT NULL CHECK (file_type IN ('image', 'video',
'audio', 'document')),
    mime_type VARCHAR(100) NOT NULL,
    file_size_bytes BIGINT NOT NULL,
    duration_seconds INTEGER,
    uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    created_by BIGINT REFERENCES User(id),
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id)
```

```
);

CREATE TABLE IncidentComplianceRule (
    id BIGSERIAL PRIMARY KEY,
    incident_id BIGINT REFERENCES Incident(id) ON DELETE CASCADE,
    compliance_rule_id BIGINT REFERENCES ComplianceRule(id) ON DELETE
CASCADE,
    linked_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    linked_by BIGINT REFERENCES User(id),
    UNIQUE(incident_id, compliance_rule_id)
);

-- ======================
-- PRE-VISIT ISSUE TABLES
-- ======================

CREATE TABLE PreVisitIssue (
    id BIGSERIAL PRIMARY KEY,
    visit_id BIGINT REFERENCES Visit(id) ON DELETE CASCADE,
    issue_type VARCHAR(100) NOT NULL,
    description TEXT NOT NULL,
    latitude DECIMAL(10, 8),
    longitude DECIMAL(11, 8),
    reported_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    resolved_at TIMESTAMP,
    resolution_notes TEXT,
    resolved_by BIGINT REFERENCES User(id),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    deleted_at TIMESTAMP,
    deleted_by BIGINT REFERENCES User(id)
);

CREATE TABLE PreVisitIssueAttachment (
    id BIGSERIAL PRIMARY KEY,
    pre_visit_issue_id BIGINT REFERENCES PreVisitIssue(id) ON DELETE CASCADE,
    file_path VARCHAR(500) NOT NULL,
    file_name VARCHAR(255) NOT NULL,
    file_type VARCHAR(50) NOT NULL CHECK (file_type IN ('image', 'video',
'audio', 'document')),
    mime_type VARCHAR(100) NOT NULL,
    file_size_bytes BIGINT NOT NULL,
    duration_seconds INTEGER,
    uploaded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    created_by BIGINT REFERENCES User(id)
);

-- ======================
```

```sql
-- LOCATION TRACKING TABLE
-- ======================

CREATE TABLE InspectorLocation (
    id BIGSERIAL PRIMARY KEY,
    inspector_id BIGINT REFERENCES User(id) ON DELETE CASCADE,
    latitude DECIMAL(10, 8) NOT NULL,
    longitude DECIMAL(11, 8) NOT NULL,
    accuracy_meters DECIMAL(8, 2),
    recorded_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- ======================
-- CONFIGURATION TABLE
-- ======================

CREATE TABLE SystemConfiguration (
    id BIGSERIAL PRIMARY KEY,
    config_key VARCHAR(100) UNIQUE NOT NULL,
    config_value TEXT NOT NULL,
    config_type VARCHAR(50) NOT NULL CHECK (config_type IN ('boolean',
'integer', 'string', 'json')),
    description TEXT,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_by BIGINT REFERENCES User(id)
);

-- ======================
-- AUDIT TRAIL TABLE
-- ======================

CREATE TABLE AuditTrail (
    id BIGSERIAL PRIMARY KEY,
    entity_type VARCHAR(100) NOT NULL,
    entity_id BIGINT NOT NULL,
    action VARCHAR(50) NOT NULL CHECK (action IN ('CREATE', 'UPDATE',
'DELETE', 'APPROVE', 'REJECT')),
    old_value JSONB,
    new_value JSONB,
    performed_by BIGINT NOT NULL REFERENCES User(id),
    performed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    ip_address VARCHAR(45),
    user_agent TEXT
);

-- ======================
-- INDEXES
-- ======================
```

```sql
-- Geospatial indexes
CREATE INDEX idx_service_area_boundary ON ServiceArea USING GIST(boundary);
CREATE INDEX idx_facility_geolocation ON Facility USING GIST(geolocation);

-- Performance indexes
CREATE INDEX idx_visit_inspector_date ON Visit(inspector_id, scheduled_date);
CREATE INDEX idx_visit_status ON Visit(status);
CREATE INDEX idx_shift_inspector_date ON Shift(inspector_id, shift_date);
CREATE INDEX idx_license_expiry ON License(expiry_date) WHERE deleted_at IS
NULL;
CREATE INDEX idx_checklist_response_visit ON ChecklistResponse(visit_id);
CREATE INDEX idx_incident_visit ON Incident(visit_id);
CREATE INDEX idx_incident_severity ON Incident(severity);
CREATE INDEX idx_pre_visit_issue_visit ON PreVisitIssue(visit_id);
CREATE INDEX idx_compliance_rule_category ON ComplianceRule(category);
CREATE INDEX idx_compliance_rule_severity ON ComplianceRule(severity);
CREATE INDEX idx_audit_trail_entity ON AuditTrail(entity_type, entity_id);
CREATE INDEX idx_audit_trail_user ON AuditTrail(performed_by);
CREATE INDEX idx_audit_trail_timestamp ON AuditTrail(performed_at);
CREATE INDEX idx_inspector_location_inspector_time ON
InspectorLocation(inspector_id, recorded_at DESC);
```

5.3. Extended Tables

The schema includes several extended tables that support advanced features beyond the core inspection workflow.

InspectorLocation Table (Time-Series): This table is implemented using TimescaleDB for efficient storage and querying of high-frequency location data. The table is converted to a hypertable partitioned by `recorded_at` timestamp. A retention policy automatically deletes data older than the configured period (default: 30 days).

SQL

```sql
-- Convert to TimescaleDB hypertable (if using TimescaleDB)
SELECT create_hypertable('InspectorLocation', 'recorded_at');

-- Set retention policy
SELECT add_retention_policy('InspectorLocation', INTERVAL '30 days');
```

SystemConfiguration Initial Data: The system is initialized with default configuration values.

SQL

```
INSERT INTO SystemConfiguration (config_key, config_value, config_type,
description) VALUES
('AuditorModuleEnabled', 'false', 'boolean', 'Enable or disable the Auditor
review step'),
('LicenseExpiryAlertThresholds', '90,60,30,7', 'string', 'Days before expiry
to send alerts'),
('MandatoryIncidentAttachments', 'true', 'boolean', 'Require at least one
attachment per incident'),
('SupervisorInspectorMinRatio', '5', 'integer', 'Minimum inspectors per
supervisor'),
('SupervisorInspectorMaxRatio', '10', 'integer', 'Maximum inspectors per
supervisor'),
('VisitAutoInterruptionTime', '30', 'integer', 'Minutes of inactivity before
flagging as interrupted'),
('GeoFenceRadius', '100', 'integer', 'Meters around facility for location
validation'),
('MaxVisitsPerInspectorPerDay', '15', 'integer', 'Maximum visits per
inspector per day'),
('DataRetentionPeriod', '90', 'integer', 'Days before purging soft-deleted
records'),
('AllowResumableVisits', 'true', 'boolean', 'Allow resuming partially
completed checklists'),
('LocationTrackingInterval', '30', 'integer', 'Seconds between location
updates'),
('LocationDataRetentionDays', '30', 'integer', 'Days to retain location
tracking data'),
('EmailNotificationsEnabled', 'true', 'boolean', 'Enable email
notifications'),
('SMSNotificationsEnabled', 'false', 'boolean', 'Enable SMS notifications'),
('PushNotificationsEnabled', 'true', 'boolean', 'Enable mobile push
notifications'),
('MaxAttachmentSizeImage', '10', 'integer', 'Maximum image size in MB'),
('MaxAttachmentSizeVideo', '100', 'integer', 'Maximum video size in MB'),
('MaxAttachmentSizeAudio', '50', 'integer', 'Maximum audio size in MB'),
('MaxAttachmentSizeDocument', '20', 'integer', 'Maximum document size in
MB'),
('MaxVideoDurationSeconds', '300', 'integer', 'Maximum video duration in
seconds'),
('MaxAudioDurationSeconds', '600', 'integer', 'Maximum audio duration in
seconds'),
('IndoorPositioningEnabled', 'false', 'boolean', 'Enable indoor positioning
features'),
('IndoorPositioningProvider', 'manual', 'string', 'Indoor positioning
provider');
```

5.4. Audit and History Tables

The `AuditTrail` table provides comprehensive logging. Additionally, all major tables include standard audit fields that are automatically populated:

- `created_at`: Timestamp when the record was created
- `updated_at`: Timestamp when the record was last modified
- `deleted_at`: Timestamp when the record was soft-deleted (NULL if active)
- `created_by`: User ID who created the record
- `updated_by`: User ID who last modified the record
- `deleted_by`: User ID who soft-deleted the record

These fields are managed through database triggers or application-level middleware.

5.5. Relationship Diagram

The database schema implements the following key relationships:

- **User → User**: Self-referencing for supervisor hierarchy (inspector.supervisor_id → supervisor.id)
- **User → SupervisorServiceArea → ServiceArea**: Many-to-many relationship for supervisor assignments
- **ServiceArea → City**: Many-to-one relationship for geographical hierarchy
- **Facility → ServiceArea**: Many-to-one relationship for facility location
- **License → Facility**: Many-to-one relationship for facility licensing
- **Visit → Facility, Inspector, Supervisor, Checklist**: Many-to-one relationships for visit context
- **ChecklistResponse → Visit, ChecklistItem**: Many-to-one relationships for response data
- **Incident → Visit, Inspector**: Many-to-one relationships for incident context
- **IncidentAttachment → Incident**: One-to-many relationship for multi-media evidence
- **IncidentComplianceRule → Incident, ComplianceRule**: Many-to-many relationship for compliance linkage
- **PreVisitIssue → Visit**: One-to-many relationship for issue tracking
- **InspectorLocation → Inspector**: One-to-many relationship for location history

---

6. Comprehensive Workflows

This section documents 10 comprehensive workflows that illustrate system behavior across various operational scenarios. Each workflow includes step-by-step execution, system actions, user interactions, and state transitions.

6.1. End-to-End System Workflow

Scenario: Complete system lifecycle from initial setup through visit execution, review, and completion.

Actors: PM (Alice), Supervisor (Jane), Inspector (Mike), Auditor (Sarah), Owner (Bob)

Preconditions: System is deployed and initialized with default configuration.

Workflow Steps:

1. **System Setup (Day 1):**

- PM Alice logs in as the first user (created during system initialization)
- Alice creates a City: "Springfield, Illinois, USA"
- Alice creates three Service Areas within Springfield: "Downtown" (with polygon boundary), "Industrial Zone," and "Riverside"
- Alice creates Supervisor account for Jane, assigns her to "Downtown" and "Riverside" service areas
- Alice creates Supervisor account for Tom, assigns him to "Industrial Zone"
- Alice creates Inspector accounts for Mike, Sarah, and John, assigns them to Supervisor Jane
- Alice creates Inspector accounts for Dave and Emma, assigns them to Supervisor Tom
- Alice enables the Auditor Module by setting `AuditorModuleEnabled` to true
- Alice creates Auditor account for Sarah (compliance specialist)
- Alice creates Owner account for Bob (executive stakeholder)

1. **Facility and License Setup (Day 2):**

- Supervisor Jane logs in and creates a facility: "Ocean View Restaurant," Downtown service area, address "123 Main St," geolocation (39.7817, -89.6501)
- Jane uploads a floor plan for the basement level (-1) of the restaurant
- Jane creates a health license for Ocean View Restaurant: license number "HEALTH-2025-001," issue date Jan 1, 2025, expiry date Dec 31, 2025
- Jane creates a checklist: "Standard Health Inspection" with 10 items (mix of yes/no, number, photo, and text questions)

1. **Visit Scheduling (Day 3):**

- Jane creates a visit for Ocean View Restaurant, scheduled for November 10, 2025, priority "normal," assigns checklist "Standard Health Inspection"
- Jane assigns the visit to Inspector Mike
- Visit status changes from "unassigned" to "assigned"
- Mike receives a push notification: "New visit assigned for Nov 10: Ocean View Restaurant"

1. **Shift and Route Planning (November 10, 8:00 AM):**

- Mike opens the mobile app and sees his shift: 8:00 AM - 5:00 PM
- Mike taps "Start Shift," changing shift status to "active"
- Mike sees 8 visits assigned for today (including Ocean View Restaurant)
- Mike taps "Plan My Route," and the app optimizes the sequence based on geolocation
- Ocean View Restaurant is third in the optimized route

1. **Visit Execution (November 10, 10:30 AM):**

- Mike arrives at Ocean View Restaurant and taps "Start Visit"
- App captures GPS coordinates (39.7818, -89.6502), calculates distance to facility (12 meters), validates within 100-meter geo-fence
- Server approves visit start, sets `actual_start_time` to 2025-11-10 10:30:15, status changes to "in_progress"
- Mike begins answering checklist items, with start_time and end_time recorded for each
- Mike discovers a broken refrigerator and taps "Add Incident"
- Mike enters title "Refrigerator Not Cooling," description "Walk-in refrigerator temperature at 55°F, food spoiling," defect_type "equipment_failure," severity "high"
- Mike takes 3 photos and records a 20-second video showing the temperature gauge
- App captures GPS coordinates for the incident (same as visit start location)
- Mike completes all 10 checklist items
- Mike taps "Submit Visit" at 11:15 AM
- Server sets `actual_end_time`, calculates duration (45 minutes), changes status to "submitted"
- Supervisor Jane receives notification: "Visit submitted by Mike: Ocean View Restaurant"

1. **Supervisor Review (November 10, 2:00 PM):**

- Jane logs into the web dashboard and sees the visit in her "Pending Reviews" queue
- Jane reviews all 10 checklist responses, examining the time spent on each question

- Jane reviews the incident, viewing all 3 photos and the video
- Jane links the incident to Compliance Rule "FDA-2017-001: Food Temperature Control"
- Jane approves the visit, adding notes: "Incident documented thoroughly. Facility must repair refrigerator within 7 days."
- Visit status changes to "under_approval"
- Since Auditor Module is enabled, the visit enters Auditor Sarah's queue

1. **Auditor Review (November 10, 4:00 PM):**

- Auditor Sarah logs in and sees the visit in her "Pending Audits" queue
- Sarah reviews the incident linked to FDA-2017-001
- Sarah verifies that the photos clearly show the temperature gauge at 55°F
- Sarah approves the incident
- Visit proceeds to PM Alice's final approval queue

1. **PM Final Approval (November 11, 9:00 AM):**

- PM Alice logs in and sees the visit in her "Final Approval" queue
- Alice reviews the entire visit: checklist, incident, supervisor notes, auditor approval
- Alice approves the visit
- Visit status changes to "completed"
- The Compliance Dashboard updates, adding $500 (fine amount from FDA-2017-001) to "Total Potential Fines"

1. **Owner Oversight (November 11, 10:00 AM):**

- Owner Bob logs into the dashboard and views the GIS map
- Bob sees Ocean View Restaurant marked with a red incident marker (high severity)
- Bob clicks on the marker and views the incident details
- Bob views the Compliance Dashboard and sees the $500 potential fine
- Bob reviews the "Inspector Performance Report" and sees that Mike completed 8 visits on November 10 with an average duration of 50 minutes

Postconditions: Visit is completed and archived. Incident is documented and linked to compliance rule. All stakeholders have visibility into the inspection outcome. Performance data is captured for analytics.

6.2. Visit Lifecycle Workflow

Scenario: Detailed state transitions for a visit from creation to completion.

Visit Status State Machine:

```
Plain Text

unassigned → assigned → in_progress → submitted → under_review →
under_approval → completed
                ↓           ↓             ↓            ↓              ↓
            cancelled   on_hold      cancelled    cancelled      cancelled
                        interrupted
```

Status Transition Rules:

1. **unassigned → assigned**: Supervisor assigns visit to an inspector
2. **assigned → in_progress**: Inspector starts visit (geo-fence validated)
3. **assigned → on_hold**: Inspector reports pre-visit issue
4. **in_progress → interrupted**: System detects inactivity (no updates for configured threshold)
5. **in_progress → submitted**: Inspector completes checklist and submits
6. **submitted → under_review**: Automatic transition upon submission
7. **under_review → under_approval**: Supervisor approves (if Auditor Module disabled, goes directly to PM approval)
8. **under_approval → under_approval**: Auditor approves, forwarding to PM (if Auditor Module enabled)
9. **under_approval → completed**: PM gives final approval
10. **Any status → cancelled**: Supervisor or PM cancels visit
11. **on_hold → assigned**: Supervisor resolves pre-visit issue
12. **interrupted → assigned**: Supervisor re-assigns visit after resolving interruption

---

6.3. Visit Re-assignment Workflow

Scenario: Three common re-assignment scenarios.

Scenario A: Manual Workload Balancing

1. Supervisor Jane reviews her team's schedule for November 10
2. Jane notices Inspector Mike has 10 visits assigned, while Inspector Sarah has only 5
3. Jane selects 2 visits from Mike's schedule and re-assigns them to Sarah

4. System updates `inspector_id` for both visits from Mike's ID to Sarah's ID
5. Mike receives notification: "2 visits have been re-assigned from your schedule"
6. Sarah receives notification: "2 new visits assigned for Nov 10"
7. Both inspectors' mobile apps sync and update their visit lists

Scenario B: Inspector Unavailability (Unplanned)

1. On November 10 at 8:30 AM, Inspector Mike calls Supervisor Jane: "I'm sick, can't work today"
2. Jane logs into the web dashboard and views Mike's schedule: 8 visits assigned for today, all in "assigned" status
3. Jane selects all 8 visits and uses "Bulk Re-assign" feature
4. Jane distributes the visits among Sarah (4 visits) and John (4 visits)
5. System updates `inspector_id` for all 8 visits
6. Sarah and John receive notifications with updated schedules
7. Their mobile apps recalculate smart routes with the new visits included

Scenario C: Shift Cancellation

1. Supervisor Jane realizes she scheduled Inspector Mike for November 10, but it's a public holiday
2. Jane cancels Mike's shift for November 10
3. System detects that 8 visits are assigned to Mike for November 10
4. System automatically changes all 8 visits from "assigned" to "unassigned"
5. Jane receives notification: "8 visits have been unassigned due to shift cancellation"
6. Jane must manually re-assign these visits to other inspectors or reschedule them for different dates

---

6.4. Inspector Unavailability Workflow

Scenario: Handling inspector absence through automated detection and manual intervention.

Automated Detection (Offline Mode Exception):

1. Inspector Mike starts his shift at 8:00 AM and begins his first visit at an underground facility
2. Mike's mobile app loses network connectivity and enters offline mode
3. Mike completes the visit offline, with all data saved locally

4. System tracks Mike's `last_seen` timestamp (last API call was at 8:05 AM when he started the visit)
5. At 8:35 AM (30 minutes later), the system's background job checks for inactive inspectors
6. System queries Mike's record and finds `last_seen` is 30 minutes old
7. System checks if Mike's app is in offline mode (flag sent by app before losing connectivity)
8. Since offline mode is true, system does NOT flag Mike as unavailable
9. At 9:00 AM, Mike exits the facility, regains connectivity, and the app syncs
10. System updates `last_seen` to 9:00 AM, and Mike is considered active

Manual Intervention (Actual Unavailability):

1. Inspector Mike is scheduled to work November 10 but has a car accident on the way to work
2. Mike calls Supervisor Jane at 8:15 AM to report he cannot work
3. Jane logs into the dashboard and views Mike's schedule: 8 visits assigned
4. Jane uses the "Inspector Unavailable" workflow:

- Jane marks Mike as unavailable for today (optional feature to prevent future assignments)
- Jane selects all 8 visits and re-assigns them using the bulk re-assign feature
- Jane distributes visits to Sarah (4) and John (4)

1. System sends notifications to Sarah and John
2. Sarah and John's mobile apps update their schedules and recalculate routes

---

6.5. Supervisor Unavailability Workflow

Scenario: Supervisor uses availability toggle to manage planned absence.

Planned Absence (Availability Toggle):

1. Supervisor Jane plans to take vacation from November 10-15
2. On November 9, Jane logs into the web dashboard
3. Jane clicks the "Available" toggle, changing it to "Unavailable"
4. System prompts: "You have 12 pending reviews. Please assign a delegate or they will be forwarded to the PM."
5. Jane selects Supervisor Tom as her temporary delegate

6. System updates Jane's `is_available` to false and `delegate_supervisor_id` to Tom's ID
7. System sends notification to PM Alice: "Supervisor Jane is unavailable Nov 9-15. Delegate: Tom. Pending reviews: 12."
8. System sends notification to Tom: "You are now the delegate for Supervisor Jane. 12 reviews have been assigned to you."
9. All visits in Jane's review queue are moved to Tom's review queue
10. During Jane's absence (Nov 10-15):

- Jane cannot create new visits or facilities
- Jane's team (Mike, Sarah, John) continue executing assigned visits
- Completed visits from Jane's team go to Tom's review queue

1. On November 16, Jane returns and sets her status back to "Available"
2. System clears the delegate assignment
3. Future visits from Jane's team return to Jane's review queue

Unplanned Absence (Account Deactivation):

1. Supervisor Jane resigns from the company on November 10
2. PM Alice deactivates Jane's account by setting `is_active` to false
3. System detects that Jane has 15 pending reviews and a team of 3 inspectors
4. System automatically:

- Moves all 15 pending reviews to PM Alice's review queue
- Temporarily assigns all 3 inspectors (Mike, Sarah, John) to PM Alice

1. System sends notification to Alice: "Supervisor Jane's account deactivated. 15 reviews and 3 inspectors require re-assignment."
2. Alice assigns the 3 inspectors to Supervisor Tom
3. System updates `supervisor_id` for Mike, Sarah, and John to Tom's ID
4. All future visits submitted by these inspectors go to Tom's review queue

---

6.6. Pre-Visit Issue Resolution Workflow

Scenario: Inspector encounters logistical problem preventing visit start.

1. Inspector Mike is assigned a visit to "Springfield Warehouse" at 123 Oak Street for November 10
2. Mike arrives at the address at 10:00 AM and finds an empty lot with no building

3. Mike taps "Report Issue" in the mobile app (before starting the visit)
4. Mike selects issue_type "incorrect_location"
5. Mike enters description: "Address shows empty lot, no warehouse building present"
6. Mike takes 2 photos of the empty lot showing the street sign and empty land
7. App captures Mike's GPS coordinates (39.7850, -89.6550)
8. System automatically changes visit status from "assigned" to "on_hold"
9. System sends immediate notification to Supervisor Jane: "Pre-visit issue reported by Mike for Springfield Warehouse"
10. Jane logs in and reviews the issue, viewing the 2 photos
11. Jane investigates and discovers the warehouse moved to 456 Pine Street last month
12. Jane updates the facility record: changes address to "456 Pine Street" and updates geolocation to (39.7855, -89.6560)
13. Jane enters resolution_notes: "Facility relocated to 456 Pine Street. Facility record updated. Visit re-assigned."
14. Jane marks the issue as resolved (`resolved_at` timestamp set, `resolved_by` set to Jane's ID)
15. Jane changes visit status from "on_hold" back to "assigned"
16. Mike receives notification: "Pre-visit issue resolved. Updated location: 456 Pine Street. Please proceed with visit."
17. Mike navigates to the new address and successfully starts the visit

---

6.7. Incident Review Workflow

Scenario: Multi-stage review of a critical incident with Auditor Module enabled.

1. Inspector Mike records a critical incident during a visit: "Gas Leak Detected"
2. Mike attaches 1 photo and 1 video showing the gas meter and audible hissing sound
3. Mike links the incident to Compliance Rule "OSHA-1910.146: Permit-Required Confined Spaces"
4. Mike submits the visit
5. Visit enters Supervisor Jane's review queue
6. Jane reviews the incident and verifies the evidence is clear and complete
7. Jane approves the visit, adding supervisor notes: "Critical safety hazard. Facility must be evacuated and gas company notified immediately."

8. Visit enters Auditor Sarah's review queue (Auditor Module enabled)
9. Sarah reviews the incident against OSHA-1910.146 compliance requirements
10. Sarah verifies that the video clearly demonstrates the gas leak
11. Sarah approves the incident
12. Visit enters PM Alice's final approval queue
13. Alice reviews the entire workflow and approves
14. Visit status changes to "completed"
15. Compliance Dashboard updates, adding the fine amount from OSHA-1910.146 to potential fines
16. System generates an automatic alert to facility management (if configured)

6.8. License Expiry Management Workflow

Scenario: Automated license expiry monitoring and alerting.

1. Facility "Ocean View Restaurant" has a health license expiring on December 31, 2025
2. System configuration: `LicenseExpiryAlertThresholds` = "90,60,30,7"
3. **October 2, 2025 (90 days before expiry):**

- Daily scheduled job runs at midnight
- Job queries all licenses where `expiry_date` is between today and today + 90 days
- Job finds Ocean View Restaurant's license (expires in 90 days)
- System sends email and push notification to Supervisor Jane and PM Alice
- Notification: "License expiring in 90 days: Ocean View Restaurant, Health License HEALTH-2025-001, Expires Dec 31, 2025"

1. **November 1, 2025 (60 days before expiry):**

- Daily job runs and finds the license (expires in 60 days)
- System sends second alert to Jane and Alice

1. **December 1, 2025 (30 days before expiry):**

- Daily job runs and finds the license (expires in 30 days)
- System sends third alert with increased urgency

1. **December 24, 2025 (7 days before expiry):**

- Daily job runs and finds the license (expires in 7 days)
- System sends final urgent alert

1. **December 28, 2025:**

- Jane renews the license with the issuing authority
- Jane updates the license record: new `expiry_date` = December 31, 2026
- License status remains "active"

1. **Alternative: January 1, 2026 (license not renewed):**

- Daily job runs and finds the license has `expiry_date` < current_date
- System automatically changes license `status` from "active" to "expired"
- System sends urgent notification to Jane and Alice: "License EXPIRED: Ocean View Restaurant, Health License HEALTH-2025-001"
- Dashboard displays the facility with a red "EXPIRED LICENSE" badge

---

6.9. Visit Auto-Cancellation Workflow

Scenario: System detects prolonged visit inactivity and flags for supervisor intervention.

1. Inspector Mike starts a visit at "City Hospital" at 10:00 AM
2. Visit status changes to "in_progress"
3. Mike begins answering checklist items
4. At 10:15 AM, Mike receives an emergency phone call and leaves the facility, forgetting to close the app
5. Mike's mobile app remains open but inactive (no checklist responses, no incidents created)
6. System configuration: `VisitAutoInterruptionTime` = 30 minutes
7. At 10:45 AM (30 minutes after visit start), the system's background job runs
8. Job queries all visits in "in_progress" status where `actual_start_time` is more than 30 minutes ago and no updates have occurred
9. Job finds Mike's visit (started at 10:00 AM, last update at 10:15 AM, now 10:45 AM)
10. Job checks if Mike's app is in offline mode (by checking `last_seen` timestamp and offline flag)
11. Mike's app is not in offline mode (last_seen is recent, but no visit activity)
12. System changes visit status from "in_progress" to "interrupted"
13. System sends notification to Supervisor Jane: "Visit interrupted: City Hospital, Inspector Mike, started 10:00 AM, no activity for 30 minutes"
14. Jane contacts Mike to determine the issue

15. Mike explains he had an emergency and forgot about the visit
16. Jane decides to re-assign the visit to Inspector Sarah
17. Jane changes visit status from "interrupted" to "assigned" and updates `inspector_id` to Sarah's ID
18. Sarah receives notification and completes the visit

Alternative Resolution (Resumable Visits Enabled):

1. Steps 1-13 same as above
2. Jane contacts Mike, who explains he can return to the facility in 1 hour
3. Jane changes visit status from "interrupted" back to "assigned"
4. Mike returns to City Hospital at 11:45 AM
5. Mike taps "Resume Visit" in the app
6. Since `AllowResumableVisits` = true, Mike's previous checklist responses are loaded
7. Mike continues from where he left off and completes the visit

---

6.10. Offline Data Synchronization Workflow

Scenario: Inspector works offline and synchronizes data upon reconnection.

1. Inspector Mike starts his shift at 8:00 AM with full network connectivity
2. Mobile app performs full sync, downloading 8 visits for today, all facility data, all checklists, and compliance rules
3. At 9:00 AM, Mike arrives at "Underground Metro Station" for an inspection
4. Mike descends to the platform level and loses all network signal
5. App detects no connectivity and displays "Offline Mode" banner
6. Mike taps "Start Visit"
7. App validates geo-fence using locally cached facility geolocation (successful)
8. App creates visit start record in local SQLite database with timestamp 9:05 AM
9. Mike completes the checklist, with all responses saved to local database
10. Mike records 2 incidents with 5 photos (total 15 MB), saved to local device storage
11. Mike taps "Submit Visit" at 10:00 AM
12. App marks visit as "submitted" in local database and adds it to the sync queue
13. At 10:15 AM, Mike exits the station and returns to street level

14. App detects network connectivity and displays "Syncing..." notification
15. **Upload Phase:**
- App uploads visit start record (actual_start_time, coordinates)
- App uploads 15 checklist responses with start_time and end_time for each
- App uploads 2 incident records with descriptions, severity, coordinates
- App uploads 5 photos in chunks (3 MB, 4 MB, 2 MB, 3 MB, 3 MB)
- Server acknowledges each upload and marks records as synced
1. **Download Phase:**
- App requests latest state for all assigned visits
- Server returns updated visit statuses (one visit was cancelled while Mike was offline)
- App updates local database
1. **Conflict Resolution:**
- App displays notification: "Visit to 'Old Library' was cancelled while you were offline"
- App removes the cancelled visit from Mike's local database
1. Sync completes successfully
2. Mike's next visit appears with the latest data from the server

---

7. Detailed Use Cases

This section provides 10 comprehensive use cases demonstrating real-world application of system features. Each use case includes actors, preconditions, main flow, alternative flows, and postconditions.

7.1. Use Case 1: System Setup and Initial Configuration

Actors: Project Manager (Alice)

Preconditions: System is deployed, database is initialized, Alice has PM credentials.

Main Flow:

1. Alice logs into the web application
2. Alice navigates to Settings & Configuration
3. Alice reviews and updates system-wide settings:
- Sets `GeoFenceRadius` to 150 meters (facilities are large industrial sites)
- Sets `VisitAutoInterruptionTime` to 45 minutes (inspections are lengthy)

- Enables Auditor Module: `AuditorModuleEnabled` = true
- Sets `LicenseExpiryAlertThresholds` to "120,90,60,30,7" (earlier warnings needed)

1. Alice creates geographical structure:

- Creates City: "Springfield, Illinois, USA"
- Creates 3 Service Areas: "Downtown," "Industrial Zone," "Riverside"
- Uploads polygon boundaries for each service area using GeoJSON files

1. Alice creates user accounts:

- Creates 2 Supervisors: Jane (Downtown, Riverside), Tom (Industrial Zone)
- Creates 5 Inspectors: Mike, Sarah, John (assigned to Jane), Dave, Emma (assigned to Tom)
- Creates 1 Auditor: Sarah (compliance specialist)
- Creates 1 Owner: Bob (executive)

1. Alice populates Compliance Knowledge Base:

- Imports 50 compliance rules from a CSV file (category, rule_code, rule_name, description, resolution, fine_amount, severity)
- Reviews and verifies each rule

1. Alice creates checklist templates:

- Creates "Standard Health Inspection" with 12 items
- Creates "Safety Inspection" with 15 items
- Creates "Environmental Inspection" with 10 items

Postconditions: System is fully configured and ready for operational use. All users can log in and access their respective interfaces.

7.2. Use Case 2: Creating and Managing Facilities

Actors: Supervisor (Jane)

Preconditions: Jane is logged in, service areas are configured.

Main Flow:

1. Jane navigates to Facilities → Create New Facility
2. Jane enters facility details:

- Name: "Ocean View Restaurant"
- Service Area: "Downtown" (selected from dropdown)

- Facility Type: "Restaurant" (selected from predefined list)
- Address: "123 Main Street, Springfield, IL"

1. Jane uses the map interface to set geolocation:

- Map displays the Downtown service area boundary
- Jane searches for "123 Main Street" and the map zooms to the location
- Jane clicks on the building, capturing coordinates (39.7817, -89.6501)
- System validates that the point falls within the Downtown boundary (success)

1. Jane enters additional details:

- Owner Name: "John Smith"
- Owner Contact: "john@oceanview.com, (555) 123-4567"
- Description: "Full-service restaurant with basement kitchen"

1. Jane uploads floor plan:

- Clicks "Add Floor Plan"
- Selects Floor Level: "-1" (basement)
- Floor Name: "Basement Kitchen and Storage"
- Uploads floor plan image (basement_plan.png, 2 MB)

1. Jane saves the facility
2. System creates facility record and displays success message
3. Jane creates a license for the facility:

- License Number: "HEALTH-2025-001"
- License Type: "Health Permit"
- Issue Date: January 1, 2025
- Expiry Date: December 31, 2025
- Issuing Authority: "Springfield Health Department"

1. Jane saves the license

Alternative Flow A: Geolocation Outside Service Area

- At step 3, Jane clicks a location outside the Downtown boundary
- System displays error: "Selected location is outside your assigned service areas. Please select a location within Downtown or Riverside."
- Jane corrects the location

Postconditions: Facility and license are created and visible to Jane, PM Alice, and Owner Bob. Facility is available for visit assignment.

7.3. Use Case 3: Scheduling and Assigning Visits

Actors: Supervisor (Jane)

Preconditions: Facilities and inspectors exist, checklists are created.

Main Flow:

1. Jane navigates to Visits → Create New Visit
2. Jane selects facility: "Ocean View Restaurant" (from dropdown filtered to her service areas)
3. Jane sets scheduled date: November 10, 2025
4. Jane sets priority: "Normal"
5. Jane assigns checklist: "Standard Health Inspection"
6. Jane assigns inspector: "Mike" (from dropdown showing her team members)
7. Jane adds notes: "Focus on kitchen cleanliness and food storage temperatures"
8. Jane saves the visit
9. System creates visit with status "assigned"
10. System sends push notification to Mike: "New visit assigned for Nov 10: Ocean View Restaurant"
11. Mike's mobile app syncs and displays the new visit in his schedule

Alternative Flow A: Inspector Overloaded

- At step 6, Jane selects Mike
- System checks Mike's schedule for November 10 and finds he already has 15 visits (maximum per day)
- System displays warning: "Inspector Mike has 15 visits on Nov 10 (maximum). Consider assigning to another inspector."
- Jane selects Sarah instead

Alternative Flow B: Critical Visit

- At step 4, Jane sets priority to "Critical"
- Visit is flagged as "URGENT" in Mike's mobile app
- Smart Route Module will prioritize this visit first in the daily route

Postconditions: Visit is created and assigned. Inspector is notified and can view the visit in the mobile app.

7.4. Use Case 4: Inspector Executes Visit with Smart Route

Actors: Inspector (Mike)

Preconditions: Mike has 8 visits assigned for today (November 10, 2025), including 1 critical visit.

Main Flow:

1. Mike opens the mobile app at 8:00 AM
2. App displays shift: 8:00 AM - 5:00 PM
3. Mike taps "Start Shift"
4. App displays 8 visits in default order (by facility name)
5. Mike taps "Plan My Route"
6. App extracts geolocation for all 8 facilities
7. App sends coordinates to Mapbox Optimization API with Mike's current location as start point
8. App specifies that the critical visit must be first
9. API returns optimized route: Critical Visit (Facility A) → Normal Visit (Facility C) → Normal Visit (Facility B) → ... → Normal Visit (Facility H)
10. App re-orders Mike's visit list and displays: "Total Distance: 52 km, Estimated Travel Time: 1 hour 20 minutes"
11. Mike taps on the first visit (Critical Visit at Facility A)
12. App launches turn-by-turn navigation to Facility A
13. Mike arrives at Facility A at 8:45 AM
14. Mike taps "Start Visit"
15. App validates geo-fence (Mike is 25 meters from facility, within 150-meter radius)
16. Visit status changes to "in_progress"
17. Mike completes the checklist and submits the visit
18. App returns to the visit list and highlights the next visit (Facility C)
19. Mike taps "Navigate" and proceeds to Facility C
20. Mike repeats the process for all 8 visits

21. Mike completes the last visit at 4:30 PM
22. Mike taps "End Shift"
23. Shift status changes to "completed"

Postconditions: All 8 visits are completed and submitted. Mike's performance metrics are updated with total distance traveled, total time, and average visit duration.

7.5. Use Case 5: Handling Underground Facility Inspection

Actors: Inspector (Mike)

Preconditions: Mike is assigned a visit to "Underground Metro Station," facility has BLE beacons configured.

Main Flow:

1. Mike arrives at the metro station entrance at 10:00 AM
2. Mike descends to the platform level (underground, no GPS signal)
3. Mike taps "Start Visit"
4. App attempts GPS positioning but receives no signal
5. App checks facility configuration and finds BLE positioning is enabled
6. App scans for BLE beacons and detects "BEACON-METRO-PLATFORM-A-01"
7. App queries the positioning service with beacon ID and signal strength
8. Positioning service returns: coordinates (relative to station entrance), floor_level "-2", building_section "Platform A", confidence_score 0.88
9. App validates that Mike is within the facility boundary (using relative coordinates)
10. Visit start is approved with `location_method` = "ble_beacon"
11. Mike completes the checklist
12. Mike discovers a critical incident: "Broken Emergency Exit Light"
13. Mike taps "Add Incident"
14. Mike enters title, description, defect_type "safety_violation", severity "high"
15. Mike takes 2 photos of the broken light
16. App attempts to capture incident location
17. App uses the same BLE beacon positioning (BEACON-METRO-PLATFORM-A-01)
18. Incident is saved with `location_method` = "ble_beacon" and `indoor_location_data` = {"floor_level": "-2", "building_section": "Platform A", "beacon_id": "BEACON-METRO-

PLATFORM-A-01", "confidence_score": 0.88}

19. Mike submits the visit
20. Mike exits the station and returns to street level
21. App detects network connectivity and syncs all data (visit start, checklist responses, incident, photos)

Postconditions: Visit is completed with accurate indoor location data. Incident is plotted on the facility's floor plan in the GIS map view.

7.6. Use Case 6: Recording Multi-Media Incident Evidence

Actors: Inspector (Mike)

Preconditions: Mike is conducting a visit at "Industrial Factory," discovers a safety violation.

Main Flow:

1. Mike is inspecting the factory floor and discovers an active water leak from a pipe
2. Mike taps "Add Incident"
3. Mike enters title: "Active Water Leak from Overhead Pipe"
4. Mike enters description: "Pipe joint has separated, water spraying onto floor at approximately 10 gallons per minute. Floor is slippery and poses slip hazard."
5. Mike selects defect_type: "Safety Violation"
6. Mike selects severity: "Critical"
7. Mike taps "Add Evidence" and selects "Photo"
8. Mike takes 3 photos showing the leak from different angles
9. Mike taps "Add Evidence" and selects "Video"
10. Mike records a 45-second video showing the active water spray and the wet floor
11. App compresses the video from 65 MB to 28 MB
12. Mike taps "Add Evidence" and selects "Audio Note"
13. Mike records a 2-minute audio note: "This leak is in the northwest corner of the factory floor, approximately 20 feet from the main entrance. The pipe appears to be a water supply line for the cooling system. Facility manager has been notified and is shutting off water supply."
14. Mike taps "Save Incident"
15. App captures GPS coordinates (outdoor facility, GPS available)
16. Incident is saved locally with 3 photos (total 8 MB), 1 video (28 MB), 1 audio (1.5 MB)

17. App queues all attachments for upload
18. Mike completes the visit and submits
19. App syncs, uploading all attachments to cloud storage
20. Server creates incident record and 5 attachment records (3 IncidentAttachment for photos, 1 for video, 1 for audio)

Postconditions: Incident is comprehensively documented with multi-media evidence. Supervisor can review all evidence types during visit review.

7.7. Use Case 7: Supervisor Reviews and Approves Visit

Actors: Supervisor (Jane), Inspector (Mike)

Preconditions: Mike has submitted a visit for "Ocean View Restaurant."

Main Flow:

1. Jane receives notification: "Visit submitted by Mike: Ocean View Restaurant"
2. Jane logs into the web dashboard
3. Jane navigates to Reviews → Pending Reviews
4. Jane sees the visit in her queue with status "submitted"
5. Jane clicks on the visit to open the review interface
6. Review interface displays:

- Facility name, address, and map location
- Inspector name and visit duration (45 minutes)
- All 12 checklist responses with time spent on each question
- 1 incident: "Refrigerator Not Cooling" with 3 photos and 1 video

1. Jane reviews each checklist response:

- Question 1: "Are all food storage areas clean?" Answer: "Yes" (Time: 15 seconds)
- Question 2: "Number of hand-washing stations" Answer: "3" (Time: 29 seconds)
- ... (reviews all 12 responses)

1. Jane reviews the incident:

- Views all 3 photos showing the temperature gauge at 55°F
- Plays the 20-second video showing the refrigerator interior and audible alarm
- Reads the description: "Walk-in refrigerator temperature at 55°F, food spoiling"

1. Jane searches the Compliance Knowledge Base for "food temperature"

2. Jane finds rule "FDA-2017-001: Food Temperature Control"
3. Jane links the incident to the compliance rule
4. Jane adds supervisor notes: "Incident documented thoroughly. Facility must repair refrigerator within 7 days. Follow-up visit required."
5. Jane approves the visit
6. Visit status changes to "under_approval" (enters Auditor queue)
7. Auditor Sarah receives notification

Alternative Flow A: Supervisor Requests Clarification

- At step 8, Jane finds the incident description unclear
- Jane clicks "Request Clarification" and enters a message: "Please provide the exact temperature reading and the time you observed it."
- Visit status changes to "under_review (clarification requested)"
- Mike receives notification and adds a comment: "Temperature was 55°F at 10:45 AM, observed for 5 minutes."
- Visit returns to Jane's queue
- Jane reviews the clarification and approves

Alternative Flow B: Supervisor Rejects Visit

- At step 7, Jane notices that Question 5 (a required photo question) has no response
- Jane clicks "Reject Visit" and enters reason: "Missing required photo for Question 5: Kitchen area"
- Visit status changes to "assigned" (returns to Mike)
- Mike receives notification and must re-execute the visit

Postconditions: Visit is approved and forwarded to the next review stage. Incident is linked to compliance rule for reporting.

7.8. Use Case 8: Auditor Reviews Compliance

Actors: Auditor (Sarah), Supervisor (Jane), PM (Alice)

Preconditions: Auditor Module is enabled, Jane has approved a visit with a critical incident.

Main Flow:

1. Sarah receives notification: "New visit for audit: Ocean View Restaurant (Critical Incident)"
2. Sarah logs into the web dashboard

3. Sarah navigates to Audits → Pending Audits
4. Sarah sees the visit in her queue with status "under_approval"
5. Sarah clicks on the visit to open the audit interface
6. Audit interface displays:

- All checklist responses
- All incidents with linked compliance rules
- Supervisor notes

1. Sarah focuses on the critical incident: "Refrigerator Not Cooling"
2. Sarah reviews the linked compliance rule: "FDA-2017-001: Food Temperature Control"
3. Sarah verifies the evidence:

- Photos clearly show temperature gauge at 55°F
- Video demonstrates the refrigerator alarm and warm interior
- Description matches the compliance violation criteria

1. Sarah checks the compliance rule resolution: "Facility must install or repair refrigeration equipment. Fine: $500 for first offense."
2. Sarah approves the incident
3. Sarah adds audit notes: "Compliance violation confirmed. Recommend $500 fine and 7-day repair deadline."
4. Sarah approves the visit
5. Visit status remains "under_approval" but is forwarded to PM Alice's final approval queue
6. Alice receives notification

Alternative Flow A: Auditor Rejects Incident

- At step 9, Sarah finds the evidence insufficient (photos are blurry, video doesn't show temperature gauge)
- Sarah clicks "Reject Incident" and enters reason: "Evidence quality insufficient for compliance enforcement. Please re-inspect with clear photos of temperature gauge."
- Visit is returned to Jane's review queue
- Jane re-assigns the visit to Mike for re-inspection

Postconditions: Visit is audited for compliance and forwarded to PM for final approval. Compliance Dashboard is updated with potential fine amount.

7.9. Use Case 9: PM Generates Performance Reports

Actors: Project Manager (Alice)

Preconditions: System has been operational for 1 month with completed visits.

Main Flow:

1. Alice logs into the web dashboard
2. Alice navigates to Reports → Inspector Performance Report
3. Alice selects date range: November 1-30, 2025
4. Alice selects all inspectors (Mike, Sarah, John, Dave, Emma)
5. Alice clicks "Generate Report"
6. System queries the database:

- Counts completed visits per inspector
- Calculates average visit duration per inspector
- Calculates average time per checklist item per inspector
- Counts incidents recorded per inspector
- Calculates completion rate (completed / assigned) per inspector

1. System generates a table:

Inspector	Visits Completed	Avg Duration	Avg Time/Question	Incidents Recorded	Comple Rate
Mike	120	45 min	2.5 min	35	98%
Sarah	115	52 min	3.1 min	42	96%
John	105	58 min	3.8 min	38	94%
Dave	110	50 min	3.0 min	40	97%
Emma	108	49 min	2.9 min	36	95%

1. Alice reviews the report and identifies insights:

- Mike is the fastest inspector (45 min average) with the highest completion rate (98%)
- John is the slowest (58 min average) and may need additional training

1. Alice exports the report to PDF
2. Alice shares the report with Owner Bob via email

Alternative Flow A: Route Efficiency Report

- At step 2, Alice selects Reports → Route Efficiency Report
- Alice selects inspector: Mike, date range: November 1-30
- System compares smart route suggestions with actual paths taken (from location tracking data)
- System calculates: Average route deviation: 8%, Average extra distance: 4.2 km per day
- Report shows that Mike generally follows the optimized route with minimal deviation

Postconditions: PM has actionable insights into inspector performance. High performers can be recognized, and low performers can receive targeted support.

7.10. Use Case 10: Owner Views GIS Map and Compliance Dashboard

Actors: Owner (Bob)

Preconditions: System has been operational for 3 months with completed visits and incidents.

Main Flow:

1. Bob logs into the web dashboard
2. Bob navigates to Dashboard → GIS Map View
3. Map displays:

- All 150 facilities in Springfield (color-coded by facility type)
- All 87 incidents recorded in the last 30 days (color-coded by severity: critical=red, high=orange, medium=yellow, low=green)
- Real-time locations of all 5 active inspectors (blue markers with inspector names)

1. Bob zooms into the Downtown service area
2. Bob clicks on a red incident marker at "Ocean View Restaurant"
3. Popup displays:

- Incident title: "Refrigerator Not Cooling"
- Severity: High
- Date: November 10, 2025
- Inspector: Mike
- Linked compliance rule: FDA-2017-001
- Potential fine: $500

1. Bob clicks "View Details" and sees the full incident with photos and video
2. Bob navigates to Dashboard → Compliance Dashboard
3. Compliance Dashboard displays:

- Total Potential Fines: $12,500 (from 25 unresolved incidents)
    - Top 5 Violated Rules:
        1. FDA-2017-001: Food Temperature Control (8 incidents)
        2. OSHA-1910.22: Slip, Trip, and Fall Hazards (5 incidents)
        3. EPA-2020-003: Improper Waste Disposal (4 incidents)
        4. OSHA-1910.146: Confined Spaces (3 incidents)
        5. FDA-2018-005: Cross-Contamination (3 incidents)
- Violations by Category (pie chart): Health (45%), Safety (35%), Environmental (20%)
- Violations by Severity (bar chart): Critical (10), High (25), Medium (35), Low (17)
- Compliance Trend (line chart): Incidents per week over last 12 weeks (showing a declining trend, indicating improving compliance)

1. Bob reviews the trend and notes that compliance is improving
2. Bob clicks on "FDA-2017-001: Food Temperature Control" to drill down
3. System displays all 8 incidents linked to this rule, showing facility names, dates, and resolution status
4. Bob exports the Compliance Dashboard to PDF
5. Bob shares the report with the executive team

Postconditions: Owner has comprehensive visibility into system-wide compliance status and geographical distribution of issues. Data supports strategic decision-making.

---

8. Business Rules and Validation Logic

This section consolidates all business rules, validation logic, and operational constraints that govern system behavior.

8.1. User Management Rules

1. **Username Uniqueness**: Usernames must be unique across the entire system (case-insensitive).
2. **Email Uniqueness**: Email addresses must be unique across the entire system (case-insensitive).

3. **Password Complexity**: Passwords must be at least 8 characters with at least one uppercase letter, one lowercase letter, one number, and one special character.
4. **Role Assignment**: A user can have only one role. Role changes require PM approval and trigger a full permission re-evaluation.
5. **Supervisor-Inspector Ratio**: A supervisor must have between 5 and 10 inspectors (configurable via `SupervisorInspectorMinRatio` and `SupervisorInspectorMaxRatio`).
6. **Inspector Supervisor Assignment**: Every inspector must be assigned to exactly one supervisor at all times.
7. **Supervisor Delegation**: A supervisor can designate one delegate supervisor. The delegate cannot be themselves.
8. **Account Deactivation**: Deactivating a supervisor automatically re-assigns their team and pending reviews to the PM.
9. **Two-Factor Authentication**: Once enabled, 2FA cannot be disabled for 24 hours (security cooldown period).

8.2. Geography and Service Area Rules

1. **City Uniqueness**: City name must be unique within the same country and state/province combination.
2. **Service Area Uniqueness**: Service Area name must be unique within the same city.
3. **Service Area Boundary**: Service Area boundary must be a valid polygon geometry (no self-intersections, at least 3 vertices).
4. **Service Area Assignment**: A supervisor can be assigned to multiple service areas, but each assignment must be unique (no duplicate assignments).
5. **Facility Geolocation Validation**: Facility geolocation must fall within the boundary of the assigned service area (validated using PostGIS ST_Contains function).
6. **Supervisor Scope**: Supervisors can only create and manage facilities within their assigned service areas.

8.3. Facility and License Rules

1. **Facility Name Uniqueness**: Facility name must be unique within the same service area (different service areas can have facilities with the same name).
2. **Geolocation Coordinates**: Latitude must be between -90 and 90, longitude must be between -180 and 180.
3. **License Number Uniqueness**: License number must be unique across the entire system.

4. **License Date Validation**: Issue date must be before or equal to expiry date.
5. **License Expiry Warning**: Creating a license with an expiry date in the past triggers a warning but does not prevent creation.
6. **Automatic License Status Update**: When expiry_date < current_date, license status automatically changes from "active" to "expired" (daily job).
7. **Floor Plan Validation**: Floor plan images must be in JPEG or PNG format, maximum size 10 MB.

8.4. Visit Management Rules

1. **Visit Assignment Scope**: Supervisors can only create visits for facilities within their assigned service areas.
2. **Inspector Team Validation**: Supervisors can only assign visits to inspectors on their team (inspector.supervisor_id = supervisor.id).
3. **Maximum Visits Per Day**: An inspector cannot be assigned more than a configurable maximum number of visits on the same date (default: 15, configurable via `MaxVisitsPerInspectorPerDay`).
4. **Geo-Fence Validation**: Inspector must be within a configurable radius of the facility to start a visit (default: 100 meters, configurable via `GeoFenceRadius`).
5. **Visit Status Transitions**: Visit status must follow the defined state machine (see Section 6.2). Invalid transitions are rejected.
6. **Visit Interruption Detection**: If a visit is in "in_progress" status with no updates for a configurable period (default: 30 minutes, configurable via `VisitAutoInterruptionTime`), and the app is not in offline mode, the visit is automatically flagged as "interrupted."
7. **Checklist Completion Requirement**: All checklist items marked as `is_required = true` must have a non-null, non-empty response before visit submission.
8. **Visit Resumability**: If `AllowResumableVisits` = false, exiting a visit before submission deletes all partial checklist responses.

8.5. Incident Management Rules

1. **Incident Attachment Requirement**: If `MandatoryIncidentAttachments` = true, each incident must have at least one attachment.
2. **File Size Limits**: Image files max 10 MB, video files max 100 MB, audio files max 50 MB, document files max 20 MB (configurable).
3. **File Duration Limits**: Video files max 5 minutes (300 seconds), audio files max 10 minutes (600 seconds) (configurable).

4. **File Type Validation**: Only allowed file types can be uploaded (images: JPEG, PNG, HEIC; videos: MP4, MOV, AVI; audio: MP3, WAV, M4A; documents: PDF, DOCX).
5. **Location Capture**: Incidents must have location data captured using one of the supported methods (GPS, Wi-Fi, BLE, manual, facility_default).
6. **Severity Classification**: Incident severity must be one of: critical, high, medium, low.
7. **Compliance Rule Linking**: An incident can be linked to multiple compliance rules, but each linkage must be unique (no duplicate links).

8.6. Checklist Management Rules

1. **Response Type Validation**: Checklist response value must match the expected response type (e.g., numeric value for 'number' type, valid date for 'date' type).
2. **Multiple Choice Validation**: For multiple_choice response type, the response value must be one of the options defined in the `options` JSONB field.
3. **Required Item Validation**: All items with `is_required = true` must have a response before visit submission.
4. **Display Order**: Checklist items are presented to inspectors in ascending order of `display_order`.
5. **Time Tracking**: Each checklist response must have `start_time` and `end_time` timestamps for analytics.

8.7. Shift and Availability Rules

1. **Shift Duration**: Shift duration must be at least 1 hour and no more than 12 hours.
2. **Shift Time Validation**: End time must be after start time.
3. **Shift Overlap Prevention**: An inspector cannot have overlapping shifts on the same date.
4. **Shift Cancellation Impact**: Cancelling a shift automatically changes all visits assigned for that date to "unassigned" status.
5. **Supervisor Availability**: When a supervisor sets `is_available` to false, the PM is immediately notified.
6. **Delegate Assignment**: When a supervisor is unavailable, pending reviews can be delegated to another supervisor.

8.8. Compliance and Configuration Rules

1. **Compliance Rule Code Uniqueness**: If provided, `rule_code` must be unique across all compliance rules.

2. **Fine Amount Validation**: Fine amount must be a positive number if provided.
3. **Severity Classification**: Compliance rule severity must be one of: critical, high, medium, low.
4. **Configuration Value Type**: Configuration values must match the specified `config_type` (boolean values must be "true" or "false", integer values must be numeric strings).
5. **Configuration Update Audit**: All configuration changes are logged in the audit trail with the PM's user ID.

8.9. Offline Mode and Synchronization Rules

1. **Offline Mode Declaration**: Mobile app must declare offline mode status when losing connectivity to prevent false unavailability alerts.
2. **Sync Priority**: Data is synchronized in order of creation timestamp (oldest first).
3. **Conflict Resolution**: Server is the single source of truth. If a visit is cancelled on the server while the inspector is offline, the inspector's work is discarded upon sync.
4. **File Upload Chunking**: Large files (>10 MB) are uploaded in chunks to handle intermittent connectivity.
5. **Sync Queue Management**: Failed uploads are retried up to 3 times with exponential backoff before alerting the inspector.

8.10. Audit Trail and Data Retention Rules

1. **Audit Trail Immutability**: Audit trail records cannot be modified or deleted (append-only table).
2. **Audit Trail Attribution**: Every action must be attributed to a specific user ID. System-initiated actions are attributed to a special "SYSTEM" user.
3. **Soft Delete Implementation**: All major entities use soft delete (setting `deleted_at` timestamp) rather than hard delete.
4. **Data Retention Period**: Soft-deleted records are permanently purged after a configurable retention period (default: 90 days, configurable via `DataRetentionPeriod`).
5. **Location Data Retention**: Inspector location tracking data is retained for a configurable period (default: 30 days, configurable via `LocationDataRetentionDays`) and then automatically purged.

8.11. Notification Rules

1. **Email Notifications**: Sent for critical events (visit submitted, license expiring, supervisor unavailable) if `EmailNotificationsEnabled` = true.

2. **Push Notifications**: Sent for real-time events (visit assigned, visit re-assigned, pre-visit issue reported) if `PushNotificationsEnabled` = true.
3. **SMS Notifications**: Sent for urgent events (critical incident recorded, visit interrupted) if `SMSNotificationsEnabled` = true (requires SMS gateway integration).
4. **User-Level Notification Preferences**: Users can override system-wide notification settings in their personal configuration.
5. **Notification Delivery Guarantee**: System retries failed notifications up to 3 times before logging the failure.

8.12. Security and Access Control Rules

1. **Role-Based Access Control (RBAC)**: All API endpoints and UI elements enforce role-based permissions.
2. **Data Scope Filtering**: Supervisors can only access data within their assigned service areas. Inspectors can only access their own assigned visits.
3. **Password Reset Expiry**: Password reset links expire after 1 hour.
4. **Session Timeout**: Web sessions expire after 8 hours of inactivity (configurable).
5. **Mobile Token Expiry**: Mobile app authentication tokens expire after 30 days and must be refreshed.
6. **Audit Trail Access**: Only PM and Owner roles can access the full audit trail. Supervisors and Inspectors can only view their own actions.

---

9. Conclusion and Future Enhancements

9.1. System Summary

The Inspection Management System Version 4.0 is a comprehensive, production-ready platform designed to streamline facility inspection operations across multiple geographical service areas. The system integrates advanced features including offline-first mobile architecture, hybrid indoor/outdoor positioning, multi-media incident documentation, smart route optimization, real-time geospatial tracking, and multi-stage compliance review workflows.

The system supports five distinct user roles (Owner, Project Manager, Supervisor, Inspector, Auditor) with granular permissions and flexible organizational hierarchies. The database schema consists of 25 core tables with full audit trail support, soft delete implementation, and geospatial indexing for high-performance queries.

Key operational capabilities include:

- **Flexible Visit Scheduling**: Date-based assignments with critical priority handling
- **Offline Operations**: Full mobile app functionality without network connectivity
- **Hybrid Positioning**: GPS, Wi-Fi, BLE, and manual location capture for all environments
- **Multi-Media Evidence**: Support for images, videos, audio, and documents
- **Smart Route Optimization**: TSP-based route planning to minimize travel time
- **Real-Time Tracking**: Live inspector location monitoring with GIS map visualization
- **Compliance Knowledge Base**: Searchable database of regulations with incident linking
- **Comprehensive Analytics**: Performance reports, compliance dashboards, and route efficiency analysis
- **Configurable Workflows**: System-wide settings for operational flexibility
- **Complete Audit Trail**: Immutable logging of all system actions

9.2. Implementation Recommendations

Technology Stack:

- **Backend**: Node.js with Express or Python with FastAPI
- **Database**: PostgreSQL 14+ with PostGIS extension, TimescaleDB for location tracking
- **Mobile App**: React Native or Flutter for cross-platform iOS/Android support
- **Web Frontend**: React or Vue.js with Mapbox GL JS for GIS mapping
- **File Storage**: AWS S3 or Azure Blob Storage for multi-media attachments
- **Offline Storage**: SQLite or Realm for mobile app local database
- **Routing API**: Mapbox Optimization API or Google Maps Directions API
- **Indoor Positioning**: Esri ArcGIS Indoors (enterprise) or custom BLE solution (budget-friendly)
- **Authentication**: JWT tokens with refresh token rotation
- **Notifications**: Firebase Cloud Messaging (push), SendGrid (email), Twilio (SMS)

Deployment Architecture:

- **Cloud Provider**: AWS, Azure, or Google Cloud Platform
- **Container Orchestration**: Kubernetes or AWS ECS
- **Load Balancing**: Application Load Balancer with auto-scaling
- **Database**: Managed PostgreSQL service (AWS RDS, Azure Database, Google Cloud SQL)
- **CDN**: CloudFront or Azure CDN for file delivery

- **Monitoring**: Prometheus + Grafana or cloud-native monitoring (CloudWatch, Azure Monitor)

9.3. Future Enhancement Opportunities

Phase 2 Enhancements:

1. **AI-Powered Incident Detection**: Use computer vision to automatically detect defects in uploaded photos (e.g., cracks, leaks, rust) and suggest incident severity.
2. **Predictive Maintenance**: Analyze historical incident data to predict which facilities are likely to have issues, enabling proactive inspections.
3. **Voice-to-Text Checklist**: Allow inspectors to answer checklist questions using voice commands, with automatic speech-to-text conversion.
4. **Automated Report Generation**: Generate facility compliance reports automatically from visit data, with customizable templates.
5. **Integration with Facility Management Systems**: Sync facility data with external systems (e.g., building management systems, CMMS platforms).
6. **Mobile Offline Maps**: Pre-download map tiles for offline navigation in areas with poor connectivity.
7. **Wearable Device Support**: Integrate with smart glasses or smartwatches for hands-free inspection data entry.
8. **Blockchain-Based Audit Trail**: Use blockchain for tamper-proof audit logging to meet regulatory requirements.
9. **Multi-Language Support**: Internationalize the system for deployment in non-English speaking regions.
10. **Advanced Analytics Dashboard**: Machine learning-based insights such as anomaly detection, trend forecasting, and inspector performance clustering.

Phase 3 Enhancements:

1. **Public Transparency Portal**: Allow the public to view facility compliance scores and inspection history (with sensitive data redacted).
2. **Facility Self-Service Portal**: Enable facility owners to view their inspection history, submit corrective action reports, and request re-inspections.
3. **Third-Party Auditor Integration**: Allow external auditing firms to access specific visits for independent verification.
4. **IoT Sensor Integration**: Integrate with IoT sensors (temperature, air quality, water flow) for automated data collection during inspections.

5. **Augmented Reality (AR) Inspection**: Use AR to overlay facility information, past incidents, and compliance requirements on the inspector's camera view.

9.4. Final Remarks

This documentation represents the complete specification for the Inspection Management System Version 4.0. It integrates all requirements, enhancements, and business logic discussed throughout the design process. The system is designed to be scalable, maintainable, and adaptable to evolving operational needs.

All stakeholders should review this document thoroughly before development begins. Any questions, clarifications, or additional requirements should be documented and incorporated into future versions of this specification.

Document Version: 4.0**Last Updated**: November 3, 2025**Total Pages**: Approximately 80 pages (2,500+ lines)**Status**: Final for Development

---

End of Document