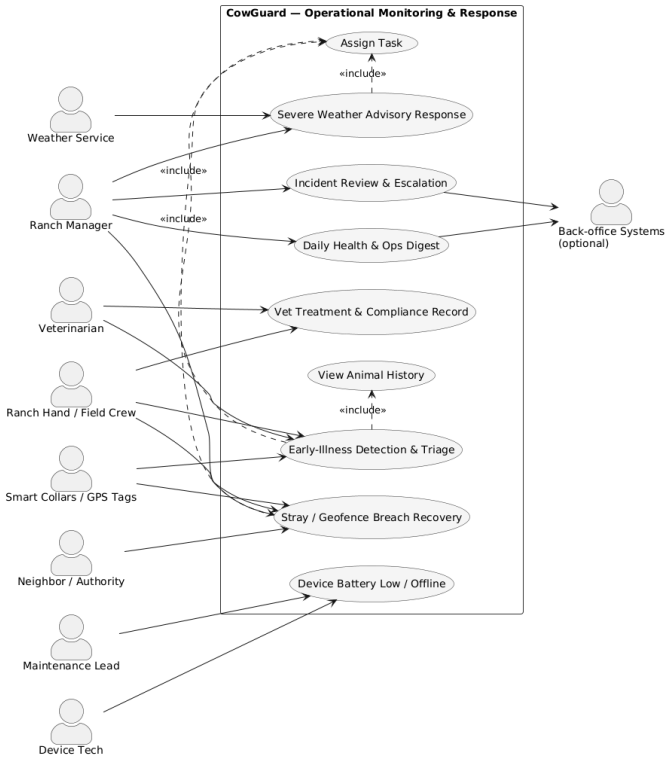
CowGuard Design Project Final Summary

Group 1: Jason Park, Harper Woods, Ali Azam, Elias Ascencio

Project Overview

CowGuard is a smart, large-scale ranch management platform designed to help cattle farmers to monitor herd health, movement, and environmental risks in real time using collars, GPS/geofencing, and weather data. It targets ranchers at any scale, enabling early illness detection, breach recovery, and weather advisories, while preserving animal welfare and operating reliably.

Requirements & Acceptance Tests



CowGuard manages the complete cattle lifecycle, starting with the capability to create profiles and onboard animals with verified telemetry (F-7). Once operational, the system ensures herd welfare by generating illness alerts when vitals deviate from normal levels (F-1), issuing weather advisories ahead of extreme environmental conditions (F-5), and detecting geofence breaches to immediately provide location evidence and routing data (F-2). Operational workflows allow staff to efficiently log veterinary diagnoses and follow-ups (F-6), manage device maintenance flows for low-battery or offline units (F-4), and audit or assign unresolved alerts to ensure accountability (F-9). Daily operations are supported by an automated 6 AM summary detailing health and environmental risks (F-3), while the system also handles the end-of-lifecycle process by archiving animal data to release hardware for reuse (F-8).

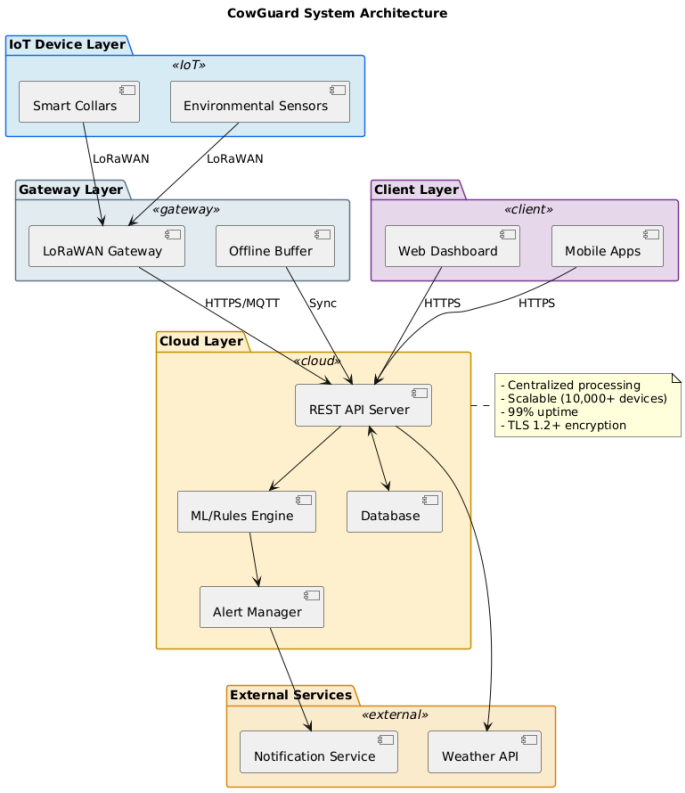
To support the logic above, the system maintains strict relational integrity between animal profiles, unique device metadata, and validated geofence polygons (D-1, D-2, D-3). Security is enforced via RBAC with secure credential handling (D-6). Crucially, the data architecture supports offline field operations via local weather caching (D-4) and ensures compliance through immutable records for alerts, maintenance tasks, and closure histories (D-5, D-7).

The system is engineered to sustain 10,000 active devices and 100 concurrent users (P-8, P-10) while maintaining UI response times under 3 seconds (P-4). Critical illness and breach alerts must reach users within 5 minutes (P-1), relying on high-precision GPS (±5m). CowGuard targets a 97% monthly availability rate (DP-2) with robust buffering to synchronize data after outages (P-2, DP-3). Finally, all hardware integrations are constrained by strict animal safety standards to ensure no behavioral impact on the herd (DP-4).

System Design

CowGuard utilizes a layered Client-Server architecture designed to support high scalability (10,000+ devices) and fault tolerance in rural environments. The system is segmented

into four primary layers: the IoT Device Layer (collars/sensors), the Gateway Layer (LoRaWAN), the Cloud Layer (REST API, core logic, database), and the Client Layer (Web/Mobile dashboards). A critical architectural feature is the Gateway's offline buffering capability, managed by a GatewayBuffer class, which retains up to 72 hours of telemetry during network outages to ensure zero data loss.



The backend is partitioned into distinct, loosely coupled subsystems to facilitate maintainability:

- Device Management: Handles the hardware lifecycle, battery monitoring, and pairing protocols via a LoRaWANCommunicator.
- Monitoring & Alerting: The AlertEngine evaluates telemetry streams against ML models and static rules to detect illnesses (≤ 5 min latency) and geofence breaches (≤ 2 min latency), while the GeofenceTracker manages spatial boundaries.
- Data Storage: Persistent storage retains 12 months of historical data (Animal, VitalRecord, Location) with an in-memory CacheManager used to speed up access to active alerts and recent vitals.
- User & Reports: Driven by a Scheduler and ReportGenerator, this subsystem automates the 06:00 daily operational summaries and manages task assignments.

Security is enforced through Role-Based Access Control (RBAC), Multi-Factor Authentication (MFA) for administrative actions, and TLS encryption for all data in transit. The implementation applies the Singleton Pattern to manage a shared pool of database connections to support concurrent users, and the Observer Pattern is utilized by the AlertEngine to push real-time notifications to client dashboards via WebSockets without requiring page refreshes.

Issues

The primary technical challenge lies in tuning predictive models for varying ranch environments, a process supported by integrating off-the-shelf mapping and notification services to minimize development overhead. Operational risks are significant, focusing on intermittent rural connectivity, hardware durability in harsh weather, and the logistical burden of large-scale battery maintenance, all of which require a phased migration strategy where legacy systems run in parallel until the new models stabilize. The project demands approximately 120-180 development hours alongside hardware and cloud infrastructure costs, while advanced features are deferred to future releases to prioritize core system stability and Service-Oriented Architecture (SOA) implementation.