Integrated Smart Farming System

A Full-Stack Oracle + Node.js + React Platform for Farm Operations, Telemetry & Analytics

Rohan Gaurav Singh

Department of Computer Science

October 29, 2025

Outline

Problem Statement

- Farm data scattered across paper, spreadsheets, and manual tools.
- Limited telemetry integration (sensors) and late feedback loops.
- Manual SQL/reporting increases errors and effort.
- No unified dashboard for operations and alerts.

Goal: Deliver a single, database-centric platform for operations, telemetry, alerts, and analytics.

System Overview

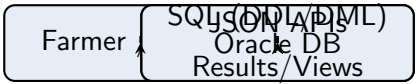
Key Features

- Farmer onboarding & JWT authentication
- CRUD: Farms, Crops, Labour, Equipment, Fertilizers, Sales
- Sensor readings: Temperature, Humidity, Soil Moisture/PH, Light
- Thresholds per farmer & automated alerts (in-app; optional SMS/Email)
- Admin analytics & CSV export

Technology Stack

- Frontend: React
- Backend: Node.js (Express)
- Database: Oracle (sequences, triggers, views)
- Schedulers: Cron for weather/ops tasks

System Architecture



Data Flow: User → Frontend → Backend → Database → Response

Core Modules

Operations

- Farms, Crops, Labour, Labour Work, Equipment, Fertilizers, Sales
- Constraints, validation, referential integrity

Telemetry

- Ingest sensor readings (`/api/sensors/reading`)
- Compute status: NORMAL / WARNING / CRITICAL

Alerts & Analytics

- Thresholds per farmer/sensor; automated `SENSOR_ALERTS`
- Admin stats, distributions, CSV export

Database Design (Oracle)

- **Sequences**: FARMER_SEQ, FARM_SEQ, CROP_SEQ, SENSOR_DATA_SEQ, SENSOR_ALERT_SEQ, ...
- **Triggers**: TRG_FARM_INSERT/UPDATE/DELETE maintain `TOTAL_FARMS`, `TOTAL_AREA`
- **Views**: FARMER_DASHBOARD, FARM_PERFORMANCE, CROP_ANALYTICS, MONTHLY_REVENUE
- **Telemetry Tables**: `SENSOR_DATA`, `SENSOR_THRESHOLDS`, `SENSOR_ALERTS`

Why Oracle? Sequences for scalable inserts, strong constraints, mature SQL features, analytics via views.

Sensor & Alert Flow

1. Farmer submits reading (farmId, sensorType, value, unit?, notes?).
2. Backend stores row (`SENSOR_DATA_SEQ.NEXTVAL`); checks thresholds (custom or defaults).
3. If CRITICAL: create `SENSOR_ALERTS` + notification; log delivery flags (in-app; optional SMS/Email).
4. Admin queries alerts by farm/sensorType/date; exports CSV as needed.

Defaults: SOIL_MOISTURE 20–80%, SOIL_PH 5.0–8.5, TEMPERATURE 5–40°C, HUMIDITY 30–95%, LIGHT 1000–100000 lux.

Key API Endpoints

Auth

- `POST /api/auth/login` → { token, farmerId, name }
- `POST /api/admin/login` → { token }

Sensors (Farmer)

- `POST /api/sensors/reading`
- `GET /api/sensors/readings` (filters: farmId, sensorType, dates, status)
- `PUT /api/sensors/thresholds`

Sensors (Admin)

- `GET /api/admin/sensors/alerts`, `GET /api/admin/sensors/stats`, `GET /api/admin/sensors/export`

Frontend UX

- Role-based navigation (farmer/admin), protected routes (JWT).
- Quick actions: Add Farm, Add Crop, Add Sensor (thresholds), Submit Reading.
- Thresholds editor (defaults or custom min/max).
- Notifications panel for critical events.

Goal: Low-friction, mobile-friendly forms; instant feedback on actions.

Results

Functional

- End-to-end alerting verified for out-of-range readings.
- Admin stats match aggregated SQL; CSV exports validated.

Performance

- Indexed joins on farmId/farmerId/dates keep queries responsive.
- Views reduce client processing; live recompute acceptable for current loads.

Validation & Testing

API Tests

- Positive: auth, CRUD, thresholds, alert flows.
- Negative: ownership violations, invalid sensor types, range errors.

DB Tests

- Sequence increments; trigger updates for farmer totals.
- Views return expected KPIs; CSV column mapping verified.

Conclusion

Key Achievements

- Unified operations & telemetry in one Oracle-backed platform.
- Reliable IDs via sequences; integrity via constraints/triggers.
- Practical dashboards, alerts, and exports for real-world use.

Impact

- Faster decisions, fewer errors, improved traceability.
- Easier onboarding for non-expert users.

Future Work

1. Multi-DB support (PostgreSQL/MySQL dialects)
2. Rule-based anomaly detection, seasonality profiles
3. Realtime dashboards and map overlays
4. Multi-tenant org hierarchy, cooperative views
5. Rich exports (PDF, SVG), scheduled reports

Thank You!

Questions?

Rohan & Gaurav Singh — Department of Computer Science