IN13/00055/20 – Leevayle Kinanga Kerindo

IN13/00086/23 - Ochieng Austine

IN13/00076/23 - Mwende Quinter

IN13/00057/23 – Dean Arnold

IN13/00039/23 – Viola Macharia

Smart Farming IoT — Project Report

1. Project Summary

This project implements a small Smart Farming IoT dashboard with a Spring Boot backend and a React frontend. The backend provides REST endpoints to manage farms, sensors, and sensor readings and exposes a simple seeded dataset. The frontend is a React app that visualizes sensor readings, lists farms, and provides CRUD operations for farms.

Key features:

- Spring Boot backend with JPA/Hibernate and H2 in-memory database (development)
- REST API endpoints for farms and sensor readings
- React frontend with Chart.js visualizations and an alert system
- OpenAPI (springdoc) available in backend for interactive docs

2. Project Structure

Root structure (relevant folders):

```
Backend/
  src/main/java/...  # Spring Boot app
  pom.xml
Frontend/
  src/               # React app
  package.json
Documentation/
  Postman_Collection.json
  Project_Report.md
  ER_Diagram.png
  UI_Screenshots/
```

3. Setup & Running

Prerequisites:

- Java 17+ (project uses Java 17 in pom)

- Maven (or use the included Maven wrapper `` `mvnw` / `mvnw.cmd` ``)

- Node.js (v16+ recommended) and npm

Backend (run from project root):

```
powershell
cd Backend
.\mvnw.cmd -DskipTests package
.\mvnw.cmd spring-boot:run
```

Backend will run on *`http://localhost:8080`* by default.

Frontend (from project root):

```
powershell
cd Frontend
npm install
npm start
```

Frontend dev server runs on *`http://localhost:3000`*. The development server proxies *`/api`* requests to *`http://localhost:8080`* (see *`src/setupProxy.js`*).

4. API Endpoints (summary)

Base URL: *`http://localhost:8080/api`*

- GET `` `/farms` `` — list farms
- GET `` `/farms/{id}` `` — get a single farm of provided ID
- POST `` `/farms` `` — create farm (body: `` `{ name, location }`) ``
- PUT `` `/farms/{id}` `` — update farm
- DELETE `` `/farms/{id}` `` — delete farm
- GET `` `/farms/{id}/readings?days=7` `` — get sensor readings for last N days (default 7)

Example: Create a farm (POST `` `/api/farms` ``)

Request body (JSON):

```json
{
  "name": "Demo Farm",
  "location": "Nairobi"
}
```

5. Data Model (brief)

- Farm: id, name, location, sensors
- Sensor: id, type, location, farm
- SensorReading: id, timestamp, readingValue, sensor

6. Testing & Postman

A Postman collection is provided at `Documentation/Postman_Collection.json`. Import it into Postman, set the `baseUrl` variable to `http://localhost:8080`, and you can run the included requests (GET/POST/PUT/DELETE and readings).

7. Known Issues & Notes

- The frontend dependencies report several (non-blocking) vulnerabilities due to older transitive packages from `react-scripts`. These are warnings; for production you should update to a newer template or adjust dependencies.
- CORS is enabled for local development: `http://localhost:3000` and `http://127.0.0.1:3000`.

- The H2 database is in-memory by default; data is lost on restart. For production, configure a persistent database (MySQL, PostgreSQL) in `application.properties`.

- Duplicate farm names and locations. This is intentional because we may have farms with same names and in the same location. This is solved by use of a different ID for each farm.