Low Level Design Document

Introduction

This Low Level Design (LLD) document outlines the core implementation details for **AgroVision - Smart Agriculture Dashboard**. AgroVision is a React, TypeScript, and Tailwind CSS-based dashboard for real-time sensor data visualization, advanced crop management forms, and role-based access control, utilizing Redux Toolkit and responsive design.

---

1. System Components

Component	Description / Responsibility
SensorDataPanel	Displays real-time sensor data (charts, tables)
CropManagementForm	Advanced forms for crop data entry/edit
UserAuthModule	Handles authentication and role-based access
Redux Store	Centralized state management (sensor, user, crop data)
API Service Layer	Fetches/sends data to backend APIs
Layout & Navigation	Responsive layout, sidebar, topbar, routing

---

2. Class/Interface Overview

Name	Type	Key Methods/Attributes	Relationships
`ISensorData`	Interface	id, type, value, timestamp	Used in SensorDataPanel
`ICrop`	Interface	id, name, type, status, plantedAt	Used in CropManagementForm
`IUser`	Interface	id, name, role (admin, farmer, viewer)	Used in UserAuthModule
`SensorDataPanel`	Component	render(), useEffect(), handleRefresh()	Uses ISensorData, Redux
`CropManagementForm`	Component	handleSubmit(), validate(), render()	Uses ICrop, Redux
`UserAuthModule`	Component	login(), logout(), checkRole()	Uses IUser, Redux
`apiService`	Module	fetchSensorData(), submitCropData(), etc.	Used by components

Example Interface:

```
interface ISensorData {
  id: string;
  type: 'temperature' | 'humidity' | 'soilMoisture';
```

```
  value: number;
  timestamp: string;
}
```

---

3. Data Structure Overview

Model	Fields
SensorData	id, type, value, timestamp
Crop	id, name, type, status, plantedAt, notes
User	id, name, email, role (enum: admin, farmer, viewer)
Redux State	sensorData: ISensorData[], crops: ICrop[], user: IUser

---

4. Algorithms / Logic

Sensor Data Fetch & Update Flow:

```
On component mount:
  Dispatch fetchSensorData action
  -> apiService.fetchSensorData()
  -> On success: update Redux sensorData state
  -> On error: dispatch error to Redux

On interval (e.g., 30s):
  Repeat fetchSensorData
```

Role-Based Access Logic:

```
On route access:
  If user.role in allowedRoles:
    Render component
  Else:
    Redirect to "Access Denied"
```

---

5. Error Handling

Scenario	Handling Approach
API fetch failure	Show error toast, retry option, log error
Unauthorized access	Redirect to login or “Access Denied” page
Form validation error	Inline error messages, prevent submission
Network timeout	Show retry prompt, fallback to cached data

Redux state inconsistency	Reset state, show error, prompt reload

---

End of Document