**AgriWealth Project - Requirements Document**

**1. Project Overview**

**1.1 Project Title :- AgriWealth - Smart Farm Management System**

**1.2 Project Description**

A comprehensive farm management system that enables efficient management of farms, polyhouses, zones, and human resources through role-based access control.

**1.3 Objectives**

* Streamline farm operations and resource management
* Provide real-time monitoring of farm assets
* Enable multi-level user management with proper authorization
* Optimize resource allocation and area utilization
* Generate revenue through subscription-based services

**2. System Architecture**

**2.1 Technology Stack**

* **Backend**: Spring Boot, Java 17+
* **Security**: Spring Security, JWT Authentication
* **Database**: MySQL
* **Build Tool**: Maven
* **API Documentation**: OpenAPI/Swagger

**2.2 System Components**

* Authentication & Authorization Module
* User Management Module
* Farm Management Module
* Subscription Management Module
* Reporting & Analytics Module

**3. Functional Requirements**

**3.1 Authentication & Authorization**

**3.1.1 User Authentication**

* User login with phone number and password
* JWT token-based authentication
* Password change functionality
* Token expiration and refresh mechanism

**3.1.2 Role-Based Access Control**

**Supported Roles:**

* **ADMIN**: System administrator with full access
* **OWNER**: Farm business owner
* **MANAGER**: Farm manager assigned by owner
* **WORKER**: Field worker assigned by manager/owner

**3.2 User Management**

**3.2.1 Admin User Management**

* Create farm owners with subscription plans
* View all farm owners and their details
* Monitor system revenue

**3.2.2 Owner User Management**

* Create managers and workers
* Assign managers to specific farms
* View all users created by the owner
* Enable/disable user accounts

**3.2.3 Manager User Management**

* Create workers (assigned to manager's farms)
* View workers under their supervision

**3.3 Farm Management Hierarchy**

Farm Owner

↓

Farm(s)

↓

Polyhouse(s) (Max: Based on farm area)

↓

Zone(s) (Max: 4 per polyhouse)

↓

Reservoir(s) (Water management)

**3.4 Farm Operations**

**3.4.1 Farm Management (Owner/Manager)**

* Create, read, update, delete farms
* Track farm area utilization
* Monitor remaining available area
* Manage farm-specific devices (IO Devices)

**3.4.2 Polyhouse Management (Owner/Manager)**

* Create polyhouses within farm area limits
* Configure polyhouse specifications:
  + Structure life expectancy
  + Plastic life expectancy
  + Polyhouse type (NVPH, Fan Pad, Shade Net, Polytunnel)
  + Dimensions (gutter height, top height)
  + Equipment (ACF number, exhaust fans)
  + Growing type

**3.4.3 Zone Management (Owner/Manager)**

* Create zones within polyhouses (max 4 per polyhouse)
* Configure zone specifications:
  + System type (NFT, Soilless, Soil, DWC)
  + Cultivation area and parameters
  + Crop information
  + Planting configuration (rows, spacing, plants)
  + Nutrient dosing system
  + Weather/irrigation type

**3.4.4 Reservoir Management (Owner/Manager)**

* Create water reservoirs for farms
* Configure capacity and reservoir type
* Track water resource allocation

**3.5 Subscription Management**

**3.5.1 Subscription Plans**

* **BASIC**: Monthly subscription ($100/month)
* **QUARTERLY**: 3-month subscription ($250/quarter)
* **YEARLY**: Annual subscription ($900/year)

**3.5.2 Features**

* Automatic subscription calculation
* End date calculation based on plan type
* Revenue tracking for admin

**3.6 Data Visibility & Access Control**

**3.6.1 Admin Visibility**

* Access to all system data
* View all farms, owners, and subscriptions
* System-wide revenue reports

**3.6.2 Owner Visibility**

* Access to all own farms and their data
* View all polyhouses, zones, and reservoirs
* Access to users created by the owner
* Area utilization reports

**3.6.3 Manager Visibility**

* View all farms assigned by the owner
* View all polyhouses in assigned farms
* View all zones in assigned polyhouses
* View reservoir information for assigned farms
* Access to workers under their supervision
* Restricted to only assigned farms (cannot access other owner farms)

**3.6.4 Worker Visibility**

* View assigned tasks
* Update task status
* Restricted to task-related operations only

**3.7 Business Rules & Validation**

**3.7.1 Area Management**

* Farm area cannot be exceeded by polyhouses
* Real-time remaining area calculation
* Validation for area constraints

**3.7.2 Hierarchy Constraints**

* Maximum 4 zones per polyhouse
* Role-based user creation permissions:
  + Admin can create Owners
  + Owners can create Managers and Workers
  + Managers can create Workers

**3.7.3 Data Integrity**

* Unique phone number validation
* Unique email validation
* Foreign key constraints
* Orphan removal for deleted entities

**4. Non-Functional Requirements**

**4.1 Performance Requirements**

* Response time < 2 seconds for all API calls
* Support up to 1000 concurrent users
* Database queries optimized with proper indexing

**4.2 Security Requirements**

* Password encryption using BCrypt
* JWT token expiration (configurable)
* Role-based endpoint protection
* SQL injection prevention
* XSS protection

**4.3 Reliability Requirements**

* Transaction management for data consistency
* Proper error handling and logging
* Data backup and recovery procedures

**4.4 Usability Requirements**

* RESTful API design conventions
* Consistent error response format
* Comprehensive API documentation
* Clear authentication flow

**5. API Endpoints Specification**

**5.1 Authentication Endpoints**

text

POST /api/auth/login

POST /api/auth/change-password

**5.2 Admin Endpoints**

text

POST /api/admin/owners - Create farm owner

GET /api/admin/owners - Get all farm owners

GET /api/admin/revenue - Get total revenue

**5.3 Owner Endpoints**

text

# Farm Management

POST /api/owner/farms - Create farm

GET /api/owner/farms - Get all farms

GET /api/owner/farms/{id} - Get farm by ID

PUT /api/owner/farms/{id} - Update farm

DELETE /api/owner/farms/{id} - Delete farm

# Polyhouse Management

POST /api/owner/farms/{id}/polyhouses - Create polyhouse

GET /api/owner/farms/{id}/polyhouses - Get polyhouses by farm

GET /api/owner/polyhouses/{id}/with-zones - Get polyhouse with zones

# Zone Management

POST /api/owner/polyhouses/{id}/zones - Create zone

GET /api/owner/polyhouses/{id}/zones - Get zones by polyhouse

GET /api/owner/polyhouses/{id}/zones/{zoneId} - Get specific zone

# Reservoir Management

POST /api/owner/farms/{id}/reservoirs - Create reservoir

GET /api/owner/farms/{id}/reservoirs - Get reservoirs by farm

# User Management

POST /api/owner/managers - Create manager

POST /api/owner/workers - Create worker

GET /api/owner/managers - Get all managers (to be implemented)

GET /api/owner/workers - Get all workers (to be implemented)

# Utilities

GET /api/owner/farms/{id}/remaining-area - Get remaining area

**5.4 Manager Endpoints**

text

# Farm Visibility (Assigned Farms)

GET /api/manager/farms - View all assigned farms

GET /api/manager/farms/{id} - View specific assigned farm

# Polyhouse Visibility (in Assigned Farms)

GET /api/manager/farms/{id}/polyhouses - View polyhouses in assigned farm

GET /api/manager/polyhouses/{id} - View specific polyhouse

# Zone Visibility (in Assigned Polyhouses)

GET /api/manager/polyhouses/{id}/zones - View zones in assigned polyhouse

GET /api/manager/zones/{id} - View specific zone

# Reservoir Visibility

GET /api/manager/farms/{id}/reservoirs - View reservoirs in assigned farm

# User Management

POST /api/manager/workers - Create worker

GET /api/manager/workers - View assigned workers

**5.5 Worker Endpoints**

text

GET /api/worker/tasks - View assigned tasks

POST /api/worker/tasks/complete - Mark task as complete

**5. Data Models**

**5.1. User**

* id (Long)
* name (String)
* email (String)
* phoneNumber (String)
* password (String)
* role (Role: ADMIN, OWNER, MANAGER, WORKER)
* enabled (boolean)
* createdAt (LocalDateTime)
* updatedAt (LocalDateTime)
* createdBy (Long) // who created this user

**5.2. FarmOwner**

* id (Long)
* user (User) - one-to-one
* farms (List<Farm>) - one-to-many
* subscription (Subscription) - one-to-one
* totalUsedArea (Double)

**5.3. Farm**

* id (Long)
* farmName (String)
* farmArea (Double)
* location (String)
* farmType (String)
* ioDevices (String)
* owner (FarmOwner) - many-to-one
* polyhouses (List<Polyhouse>) - one-to-many
* reservoirs (List<Reservoir>) - one-to-many

**5.4. Polyhouse**

* id (Long)
* polyhouseName (String)
* polyhouseArea (Double)
* structureLife (Integer)
* plasticLife (Integer)
* polyhouseType (String)
* gutterHeight (Double)
* topHeight (Double)
* acfNumber (Integer)
* exhaustFanNumber (Integer)
* growingType (String)
* farm (Farm) - many-to-one
* zones (List<Zone>) - one-to-many

**5.5. Zone**

* id (Long)
* zoneName (String)
* systemType (String)
* zoneArea (Double)
* cultivationArea (Double)
* cropInfo (String)
* rowCount (Integer)
* rowSpacing (Double)
* plantSpacing (Double)
* plantsPerRow (Integer)
* nutrientDosing (String)
* weatherType (String)
* customWeatherType (String)
* polyhouse (Polyhouse) - many-to-one

**5.6. Reservoir**

* id (Long)
* reservoirName (String)
* capacity (Double)
* reservoirType (String)
* customReservoirType (String)
* farm (Farm) - many-to-one

**5.7. Subscription**

* id (Long)
* planType (String) - BASIC, QUARTERLY, YEARLY
* startDate (LocalDate)
* endDate (LocalDate)
* price (Double)