Product Requirements and Specification Document

Project Name

AgroData - Crop Yield Logger

Description

AgroData is an open-source backend service enabling farmers to securely log and retrieve crop yield data. The service provides user authentication and persistent data storage, built with Node.js, Express, and MySQL.

---

1. Goals & Objectives

Goal	Description
Secure Data Logging	Allow authenticated users to log crop yield data
Data Retrieval	Enable users to retrieve their historical yield records
User Authentication	Ensure only authorized access to data
Open-Source	Codebase is publicly available and well-documented

---

2. Functional Requirements

ID	Requirement
FR1	Users can register and log in with email and password
FR2	Authenticated users can create, read, update, and delete (CRUD) yield logs
FR3	Each yield log includes crop type, quantity, unit, date, and location
FR4	Users can retrieve only their own data
FR5	API provides clear error messages and status codes

---

3. Non-Functional Requirements

ID	Requirement
NFR1	RESTful API design
NFR2	Secure password storage (hashing)
NFR3	Input validation and sanitization
NFR4	Basic rate limiting to prevent abuse
NFR5	Open-source license and documentation

4. User Stories

ID	As a…	I want to…	So that…
US1	Farmer	Register and log in	My data is secure
US2	Farmer	Log new crop yield data	I can track my harvests
US3	Farmer	View, update, or delete my yield records	I can manage my data
US4	Developer	Access API documentation	I can integrate or contribute

5. API Endpoints

Method	Endpoint	Description	Auth Required	Request Body / Params
POST	/api/register	Register new user	No	`{ email, password }`
POST	/api/login	Authenticate user	No	`{ email, password }`
GET	/api/yields	List all yield logs (user only)	Yes	-
POST	/api/yields	Create new yield log	Yes	`{ cropType, quantity, unit, date, location }`
GET	/api/yields/:id	Get specific yield log	Yes	`:id`
PUT	/api/yields/:id	Update yield log	Yes	`{ ...fields }`
DELETE	/api/yields/:id	Delete yield log	Yes	`:id`

6. Data Model

```
-- Users Table
CREATE TABLE users (
  id INT PRIMARY KEY AUTO_INCREMENT,
  email VARCHAR(255) UNIQUE NOT NULL,
  password_hash VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Yields Table
CREATE TABLE yields (
  id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT NOT NULL,
  crop_type VARCHAR(100) NOT NULL,
  quantity DECIMAL(10,2) NOT NULL,
  unit VARCHAR(20) NOT NULL,
  date DATE NOT NULL,
  location VARCHAR(255),
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
```

```
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

---

7. Security & Privacy

- Passwords stored using strong hashing (e.g., bcrypt)
- JWT-based authentication for API access
- Input validation to prevent SQL injection and XSS
- Users can access only their own data

---

8. Technology Stack

Component	Technology
Language	Node.js
Framework	Express
Database	MySQL
Auth	JWT, bcrypt
License	MIT (Open-source)

---

9. Open-Source & Documentation

- Public GitHub repository with README and API docs
- Example `.env` and setup instructions
- Contribution guidelines

---

10. Acceptance Criteria

ID	Criteria
AC1	Users can register, log in, and receive JWT tokens
AC2	Authenticated users can perform CRUD on their yield logs
AC3	API returns appropriate status codes and error messages
AC4	All sensitive data is securely stored and transmitted
AC5	Project is documented and installable via provided scripts

---

11. Out of Scope

- No frontend/UI
- No advanced analytics or reporting

- No third-party integrations

---

12. Milestones

Milestone	Description
M1: Project Setup	Repo, dependencies, base structure
M2: Auth Implementation	Register, login, JWT
M3: Yield Log CRUD	Endpoints, DB integration
M4: Security & Validation	Hashing, validation, rate limiting
M5: Documentation	README, API docs

End of Document