High Level Design Document

Introduction

This High Level Design (HLD) document outlines the architecture and core components for **AgroData - Crop Yield Logger**, a backend service enabling farmers to log and retrieve crop yield data. The system is built using Node.js, Express, and MySQL, and supports user authentication and secure data storage.

---

1. System Architecture Overview

Architecture Description:
AgroData is a RESTful backend service. Clients interact via HTTP(S) with the Express API server, which handles authentication, business logic, and data persistence using a MySQL database.

Component	Role/Responsibility
API Server	Handles HTTP requests, authentication, and routing
Authentication	Manages user registration, login, and session tokens
Crop Yield Module	Manages crop yield data CRUD operations
MySQL Database	Stores user and crop yield data

---

2. Component Interactions

Sequence Step	Interaction Description
1	Client sends HTTP request (e.g., login, log yield) to API Server
2	API Server authenticates user (if required)
3	API Server processes request and interacts with MySQL Database via ORM/SQL
4	Database returns data/results to API Server
5	API Server sends HTTP response to client

---

3. Data Flow Overview

Data Flow	Source	Destination	Description
User Registration/Login	Client	API Server	User credentials sent for authentication
Auth Verification	API Server	Database	User data queried for authentication
Log/Retrieve Yield Data	Client	API Server	Crop yield data sent/retrieved via endpoints
Data Persistence	API Server	Database	Yield data stored or fetched as needed

4. Technology Stack

Layer/Function	Technology/Framework
Backend Runtime	Node.js
Web Framework	Express.js
Database	MySQL
Authentication	JWT (JSON Web Tokens)
ORM/DB Access	Sequelize or mysql2
API Protocol	REST (HTTP/HTTPS)

5. Scalability & Reliability

- **Scalability:**
  - Stateless API enables horizontal scaling (multiple server instances).
  - Database can be scaled vertically or via replication if needed.
- **Reliability & Security:**
  - JWT-based authentication secures endpoints.
  - Input validation and error handling to prevent invalid data.
  - Regular database backups recommended.
  - Use HTTPS for secure data transmission.

End of Document