Thanks for sharing the architecture diagram of your **Poultry Management System** 🐓—it's well-structured and modular!

To help you with the **Web App**, here’s a step-by-step guide to get started based on your architecture:

**🧱 Web App Development Plan**

**1. Tech Stack Recommendation**

* **Frontend**: React.js (for modular UI and fast rendering)
* **Backend**: Django REST Framework (already aligned with your API Layer)
* **Authentication**: Token-based RBAC (Role-Based Access Control)
* **UI Framework**: Tailwind CSS or Material UI for clean design

**2. Core Pages to Build**

Based on your diagram, here are the key screens/modules for the Web App:

**🔹 Dashboard**

* Real-time metrics (flock count, feed usage, water levels)
* Charts (e.g., feed consumption trends)
* Notifications

**🔹 Inventory Management**

* View current stock
* Add/remove items
* Alerts for low stock

**🔹 Health & Vaccination**

* Record bird health status
* Schedule and log vaccinations
* Filter by flock/date/type

**🔹 Operations**

* Bird tracking
* Feeding and cleaning logs
* Water usage

**🔹 Admin Console**

* User roles and permissions
* System settings

**3. Frontend Structure (React)**

src/

├── components/

│ ├── Dashboard.js

│ ├── Inventory.js

│ ├── HealthVaccination.js

│ ├── BirdManagement.js

│ ├── AdminConsole.js

│ └── Navbar.js

├── App.js

├── index.js

**4. API Integration**

Use axios or fetch to connect to your Django REST API:

axios.get('/api/inventory/')

.then(response =&gt; setInventory(response.data))

.catch(error =&gt; console.error(error));

**5. Authentication (RBAC)**

* Login page with token-based auth
* Role-based routing (e.g., only Admins can access AdminConsole)

Would you like me to:

* Scaffold the React project for you?
* Help design the UI for a specific module (e.g., Dashboard or Inventory)?
* Generate API endpoints for Django backend?

Let me know how you'd like to proceed!