Here is the **complete documentation outline** for your system architecture integrating all modules:

**🧠 System Overview**

This system integrates poultry operations, commerce, inventory, infrastructure, analytics, and administrative configuration into a modular architecture designed for scalability, traceability, and automation.

**🧩 Modules and Responsibilities**

**1. Operations**

**Apps**: BirdApp, FeedingApp, WaterApp, CleaningApp, HealthVaccinationApp  
**Purpose**: Manage daily poultry activities  
**Responsibilities**:

* Track flock health and behavior
* Log feeding and watering schedules
* Record cleaning and vaccination activities
* Register egg and meat production

**2. Commerce & Inventory**

**Apps**: CatalogApp, InventoryApp  
**Purpose**: Manage products and stock  
**Responsibilities**:

* Define product categories and details
* Track stock levels and movements
* Deduct inventory based on operations
* Register production outputs (eggs, meat)

**3. Infrastructure & Control**

**Apps**: FarmApp, HousesApp, FlocksApp, SectorApp, AccountApp  
**Purpose**: Define physical and logical infrastructure  
**Responsibilities**:

* Configure farm layout and housing
* Manage flock assignments and movements
* Define operational zones
* Control user access and roles

**4. Analytics & Reporting**

**Apps**: DashboardApp, ReportApp  
**Purpose**: Provide insights and historical data  
**Responsibilities**:

* Visualize feed and water consumption
* Monitor health and vaccination status
* Track production trends
* Generate periodic reports

**5. Admin & Configuration**

**Apps**: SettingsApp, NotificationApp  
**Purpose**: Configure system and send alerts  
**Responsibilities**:

* Configure schedules and thresholds
* Send notifications for low stock, missed operations
* Manage operational parameters
* Trigger alerts based on health and production

**🔗 Integration Points**

| **Source ↔ Target** | **Description** |
| --- | --- |
| Operations ↔ Inventory | FeedingApp and HealthVaccinationApp deduct stock from InventoryApp using products from CatalogApp. |
| Operations ↔ Infrastructure | BirdApp and FlocksApp use housing and sector data from HousesApp and SectorApp. |
| Operations ↔ Analytics | FeedingApp and BirdApp log metrics to DashboardApp and ReportApp. |
| Operations ↔ Admin | Missed operations or low stock trigger alerts via NotificationApp. |
| Commerce ↔ Inventory | CatalogApp defines products used in InventoryApp; InventoryApp tracks stock levels. |
| Infrastructure ↔ Admin | SectorApp and AccountApp define access control and operational zones. |
| Analytics ↔ Admin | DashboardApp triggers alerts based on thresholds set in SettingsApp. |

**🔄 Data Flow Examples**

**🐔 Feeding Process**

1. BirdApp identifies flock and feed requirement
2. FeedingApp checks stock via InventoryApp
3. InventoryApp deducts feed quantity
4. DashboardApp logs feed usage
5. NotificationApp alerts if stock is low

**💉 Vaccination Process**

1. BirdApp schedules vaccination
2. HealthVaccinationApp logs vaccine usage
3. InventoryApp updates vaccine stock
4. ReportApp generates health summary
5. NotificationApp alerts for due vaccinations

**🥚 Production Process**

1. BirdApp logs egg/meat production
2. CatalogApp identifies product type
3. InventoryApp updates stock
4. DashboardApp visualizes output
5. ReportApp tracks production trends

**🌐 API Responsibilities**

| **App** | **Endpoints** |
| --- | --- |
| BirdApp | GET /flocks/, POST /flocks/{id}/feed/, POST /flocks/{id}/vaccinate/ |
| FeedingApp | POST /feed/, GET /feed/logs/ |
| InventoryApp | GET /inventory/, POST /inventory/deduct/, POST /inventory/add/ |
| CatalogApp | GET /products/, GET /categories/ |
| DashboardApp | GET /metrics/feed/, GET /metrics/production/ |
| ReportApp | GET /reports/health/, GET /reports/production/ |
| NotificationApp | POST /notifications/, GET /notifications/ |
| SettingsApp | GET /settings/, POST /settings/update/ |

Would you like this architecture exported as a PDF or formatted into a Markdown or Word document for your project documentation?