Smart Supply Chain for Farm-to-Table Network

Entities:

- **Farmers** (produce suppliers)
- **Distributors** (logistics & warehousing)
- **Restaurants / Markets** (end customers)
- **Health Inspectors** (compliance checks)

Workflow:

- Farmer lists available produce (Product Service)
- Distributor picks up orders (Order Service)
- Restaurants place orders → schedule deliveries
- Health inspector check-ins (Guest check-in)
- Chatbot assists with order status / compliance FAQs

Microservices:

- **Order Management**
- **Chatbot** (delivery status, food safety)
- **Check-in System** for inspectors or deliveries
- **Notification System** for delivery arrival or delays
- **Comment threads** between farms and restaurants for custom orders
- **RabbitMQ** → Order placed → Delivery scheduled → Check-in logged
- **Discovery service** for locating available distributors nearby

Overview:

The Smart Farm-to-Table Supply Chain system connects multiple entities within the agricultural and food supply chain: **Farmers, Distributors, Restaurants, and Health Inspectors**. Each entity operates independently but collaborates through a network of services to ensure that fresh produce is efficiently sourced, tracked, delivered, and compliant with health regulations.

This project will teach concepts such as **service decomposition**, **asynchronous messaging (RabbitMQ)**, **service discovery**, **check-in systems**, and **notifications** in a distributed system.

Entities Involved:

1. **Farmers** (Producers of food)
    - Manages available produce, batches, and supply data.
2. **Distributors** (Logistics & Warehousing)
    - Coordinates transport, storage, and inventory management.
3. **Restaurants/Markets** (End consumers)
    - Places orders for ingredients, schedules deliveries.
4. **Health Inspectors** (Regulatory compliance)
    - Inspects produce for quality and safety.

Core Features of the System:

1. **Farmers:**
    - Can list available produce (quantity, type, quality).
    - Receive order requests for specific produce items from restaurants/markets.
2. **Distributors:**
    - Accept orders from restaurants and forward them to farmers.
      Track delivery schedules, ensure compliance with storage and transport conditions
3. **Restaurants/Markets:**
    - Place orders for fresh produce from farmers.
    - Receive delivery status updates, schedule deliveries.
    - Use a **Chatbot** to track order status, interact with suppliers.
4. **Health Inspectors:**
    - Check produce for health and safety.
    - Perform **check-ins** and provide regulatory feedback.
    - Can leave comments on the quality of specific produce batches.

5. **Notification System:**
    - Sends reminders, order updates, and regulatory compliance alerts (e.g., when the delivery is scheduled, when an inspector completes an inspection).

Microservices Architecture:

1. **User Service** (Authentication & Role Management):
    - Handles user login/registration for Farmers, Distributors, Restaurants, Health Inspectors.
    - Manages roles, permissions, and profiles.
2. **Order Service** (Order Management):
    - Manages customer orders (restaurants/markets) and farmer orders.
    - Handles batch tracking, order statuses, and scheduling.
    - **RabbitMQ** is used to trigger updates (e.g., order placed → confirmation → order dispatched).
3. **Product Service** (Inventory and Produce Management):
    - Handles produce catalog (types, availability, quality).
    - Used by farmers to list available items and by distributors to update inventory levels.
4. **Delivery Service** (Tracking and Logistics):
    - Manages deliveries from farmers to distributors to restaurants.
    - Tracks delivery time, route, and status.
5. **Health Compliance Service** (Inspection & Reports):
    - Manages inspection schedules, reports, and compliance tracking.
    - Provides a **Check-in Service** for health inspectors.
    - Sends regulatory alerts and updates to restaurants.
6. **Notification Service**:
    - Handles all notifications (e.g., order status updates, delivery scheduling, health inspection reminders).
7. **Chatbot Service** (Communication & Assistance):
    - Helps restaurants track order status and interacts with farmers/distributors for common queries (e.g., “When will my delivery arrive?”).
8. **Service Discovery** (e.g., Consul or Eureka):
    - Ensures services can dynamically discover each other and communicate across the network, e.g., **Order Service** communicating with **Product Service**.
9. **API Gateway**:
    - Routes all external requests to the appropriate service. For example, when a restaurant places an order, the **API Gateway** ensures it is routed to the **Order Service**.

Technology Stack:

- **Backend Services**: Java Spring Boot or Node.js
- **Messaging**: RabbitMQ (for asynchronous messaging)
- **Service Discovery**: Consul or Eureka
- **API Gateway**: NGINX or Spring Cloud Gateway
- **Database**: PostgreSQL/MySQL (per service), MongoDB for unstructured data (e.g., chat logs)
- **Docker**: For containerization and isolation
- **Kubernetes**: For orchestration and management of microservices

---

Use Case Flow Example:

1. **Farm Lists Produce:**
    - Farmer logs into the system and adds produce to their catalog (via Product Service).
    - Availability, price, and quality info are saved.
2. **Restaurant Places Order:**
    - A restaurant places an order for vegetables via the **Order Service**.
    - The order is queued in RabbitMQ, and an event is triggered to notify the **Delivery Service** to prepare for transportation.
3. **Distributor Receives Order:**
    - The distributor checks RabbitMQ, sees the new order, and fetches items from the farmer's inventory.
    - Updates order status in **Order Service** (e.g., "Order Dispatched").
4. **Health Inspector Performs Check:**
    - The Health Inspector performs a compliance check for the batch of produce. Updates are logged in the Health **Compliance Service**.
    - Health check results are pushed to the **Notification Service**, which sends an alert to the restaurant that their order passed the inspection.
5. **Notifications Sent:**
    - **Notification Service** sends updates to the restaurant and distributor. For example, “Your order has been dispatched, estimated delivery time: 2 hours.”
6. **Chatbot Assists:**
    - A restaurant queries the **Chatbot Service** for delivery time: “Where’s my delivery?”
    - The chatbot checks **Order Service** and responds with real-time information.