To design a reactive system that fulfills the provided requirements for the **Warehouse Service** and **Central Monitoring Service**, we can use a system based on event-driven architecture.

**High-level design**:

### **1. Warehouse Service**

This service is responsible for receiving data from sensors and forwarding it to the Central Monitoring Service. It is designed to handle sensor data asynchronously and has the following components:

#### **Components:**

* **UDP Listener**: Listens for UDP packets from temperature and humidity sensors. Each sensor sends periodic updates, which the listener receives and processes.
* **Sensor Data Parser**: Parses the incoming UDP data into structured events (e.g., JSON).
* **Data Publisher**: Publishes the parsed sensor data to the Central Monitoring Service using a message broker (such as Kafka).

#### **Workflow:**

* **Step 1**: The UDP Listener receives data from the sensors.
* **Step 2**: The Sensor Data Parser processes the UDP packets and converts them into structured messages.
* **Step 3**: The Data Publisher sends the data to the Central Monitoring Service for analysis.

### **2. Central Monitoring Service**

This service handles the analysis of sensor data and triggers an alarm when the temperature or humidity crosses configured thresholds. It needs to be highly responsive to incoming data and able to handle multiple warehouses simultaneously.

#### **Components:**

* **Data Ingestor**: This component subscribes to sensor data, i.e. from a message broker (e.g., Kafka). It will listen to incoming data from all warehouse services.
* **Threshold Observer**: This component compares the received sensor data against defined thresholds for temperature and humidity.
* **Alarm Logger**: If any sensor data exceeds the threshold, the system logs an alarm to the console or stores it in persistent storage (logs).

#### **Workflow:**

* **Step 1**: The Data Ingestor receives the sensor data (thru a messaging-queue)
* **Step 2**: The Threshold Checker compares the incoming values against the defined thresholds.
* **Step 3**: If any value exceeds the threshold, the Alarm Logger outputs an alarm message to the logs/console indicating the issue.

### **3. Logging Alarm Mechanism**

When the **Alarm Logger** raises an alarm, the system should log a message in a structured format. The log message should include:

* **SensorId: Id of the sensor.**
* **Actual Value**: The temperature or humidity reading.

Example log format:

ALARM: Humidity threshold exceeded! Sensor ID: {}, Value: {}

### **4. Technological Stack**

To implement this system, we can use a reactive programming model, which efficiently manages asynchronous data flows.

* **Java**: For the Warehouse Service and Central Monitoring Service.
* **Kafka**: For event-driven communication between services
* **SLF4J** (Java): For structured logging of alarms.