Requirement Document for the IoT System Development Project

Software Engineering for the Internet of Things course

Objective

Students are tasked with designing and implementing a complete IoT system capable of collecting, processing, storing, and visualizing data from various domains. The system should demonstrate key IoT concepts, flexibility, and applicability across multiple application areas.

Project Scope

The system must:

1. Simulate or integrate real sensors for data collection.
2. Process the collected data using middleware such as Node-RED.
3. Store data in a time-series database (e.g., InfluxDB) for analysis.
4. Visualize data on an interactive dashboard (e.g., Grafana).
5. Implement alerting mechanisms based on configurable thresholds.
6. Containerize all components for seamless deployment using Docker.

Applicable Domains

The system should be designed for one or more of the following domains:

1. **Smart Agriculture**:
    - Monitor environmental factors (e.g., temperature, soil moisture) or livestock health.
2. **Smart Cities**:
    - Measure air quality, traffic conditions, or utility usage (e.g., electricity, water).
3. **Industrial IoT (IIoT)**:
    - Track equipment health, vibration analysis, or production metrics.
4. **Environmental Monitoring**:
    - Detect seismic activity, weather patterns, or pollution levels.
5. **Healthcare**:
    - Monitor vital signs like heart rate, oxygen levels, and temperature.
6. **Smart Homes**:
    - Automate lighting, heating, and security systems based on sensor data.

Functional Requirements

1. Sensor Integration

- **Simulated or Real Sensors**:
    - Simulate sensor data using software, or use hardware for real-world applications.
    - Examples:
        - Smart Agriculture: Soil moisture, temperature, and livestock heart rate.
        - Smart Cities: Air quality (CO2, PM2.5), noise levels, and traffic speed.
        - IIoT: Vibration, pressure, and temperature sensors.
- **Data Characteristics**:
    - Simulations should produce realistic data ranges.
    - Sensors should publish data periodically with timestamps and unique identifiers.

2. Communication Protocols

- Implement MQTT as the primary protocol for communication:
    - Define well-structured topics, such as:
        - /<domain>/<location>/<sensor> (e.g., /agriculture/field1/temperature).
        - /iot/<device-type>/<data-type>.
- Configure sensors and applications to publish and subscribe to these topics via an MQTT broker (e.g., Eclipse Mosquitto).
- Allow for dynamic configuration, such as adjusting sensor thresholds or requesting specific data.

3. Data Processing

- **Middleware**:
    - Use Node-RED or similar tools to process sensor data:
        - Aggregate data from multiple sensors.
        - Transform data into JSON for easier storage and analysis.
        - Implement calculations, such as:
            - Environmental indices (e.g., Air Quality Index).
            - Alerts for threshold violations.
- **Data Forwarding**:
    - Send processed data to storage and visualization platforms.

4. Data Storage

- **Database**:
    - Use a time-series database (e.g., InfluxDB) to store sensor readings.
    - Tag data with:
        - Timestamps.
        - Sensor type, location, and domain.

5. Visualization

- **Dashboard**:
    - Develop dashboards to:
        - Display real-time data (e.g., line graphs, gauges, and maps).
        - Provide filtering options (e.g., by sensor, location, or time).
    - Include actionable insights:
        - Threshold violations highlighted in red.
        - Summary statistics (e.g., average, max, and min values).

6. Alerting Mechanisms

- Configure alerts for critical conditions:
    - Define customizable thresholds (e.g., temperature > 40°C or vibration > 10g).
    - Notify users through:
        - Messaging platforms (e.g., Telegram, Slack).
        - Email or HTTP callbacks to external systems.
- Include structured alert messages:
    - Domain: "Industrial IoT."
    - Sensor: "Vibration Sensor 3."
    - Location: "Line 1 - Factory A."
    - Condition: "Exceeds vibration threshold: 12g."

Non-Functional Requirements

1. **Portability**:
    - Fully containerize the system using Docker Compose for deployment across diverse environments.
2. **Scalability**:
    - Ensure the system supports additional sensors and locations without significant reconfiguration.
3. **Resilience**:
    - Implement error handling for communication failures and ensure data consistency.
4. **User-Friendly Design**:
    - Ensure intuitive dashboards and clear system documentation.
5. **Security**:
    - Include basic security features like authentication for MQTT brokers and databases.