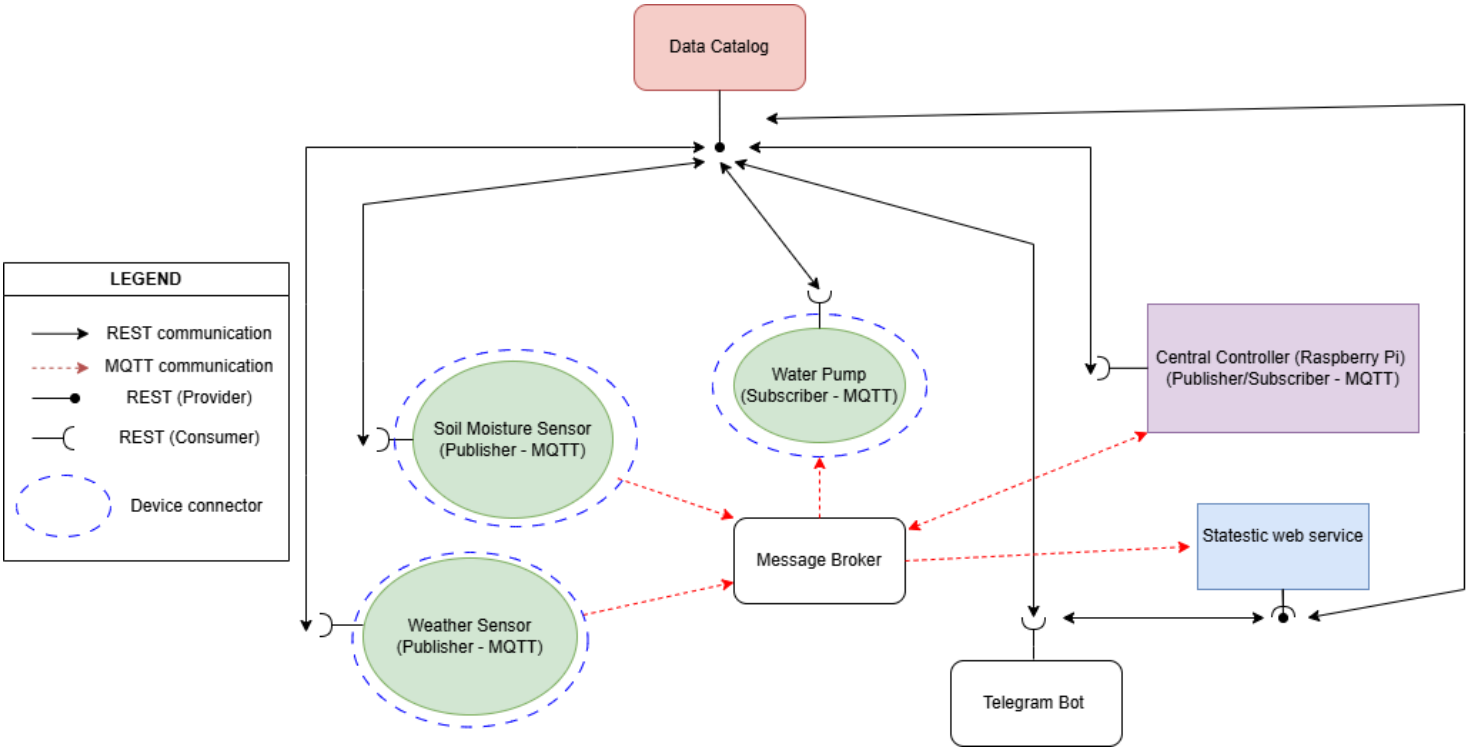
1.Name of Use Case

Name of the Use Case	**Smart Garden IoT System**
Version No.	**v1.0**
Date	**26/11/2024**
Team Members (with student ids)	**Group 9** **Masoud Momeni S324829 – Setareh Ghorbani S327772–** **Alireza Soleiman S327773– Niloofar Karbalaeiahmadharati S328771**

2.Scope and Objectives of Function

Scope and Objectives of Use Case	
Scope	The Smart Garden IoT System automates irrigation using sensors and cloud-based components. It integrates soil and weather data for real-time decision-making and provides user interaction through a Telegram Bot, with data aggregation and insights facilitated by a Statistics Web Server.
Objective(s)	• Automate irrigation using real-time data from soil moisture and weather sensors. • Aggregate and analyze data for system insights via a Statistics Web Server. • Provide user interaction through a Telegram Bot. • Optimize water usage and enhance garden care efficiency.
Domain(s)	IoT, Smart Home (Garden) Automation, Smart Agriculture
Stakeholder(s)	• Homeowners and garden owners • Developers of smart home systems
Short description	The Smart Garden IoT System automates irrigation by integrating multiple components. **Soil Moisture Sensors** and **Weather Sensors** provide real-time data, which is processed by the **Central Controller**. Communication between components is managed by the **Message Broker** using MQTT. Configurations and device details are stored in the **Data Catalog**. A **Statistics Web Server** aggregates data and provides REST APIs to the **Telegram Bot**, which enables user interaction and notifications. The **Water Pump** executes commands to irrigate the garden efficiently.

3.Diagram of Use Case



4. Complete Description of the System

The Smart Garden IoT System integrates multiple components to ensure efficient irrigation management. Each part plays a vital role and interacts with other components using **REST** and **MQTT** protocols.

1. Data Catalog

- **What it Does:**
    - Centralized repository for storing system configurations, MQTT topics, and device information.
    - Acts as a REST provider for all devices during initialization and dynamic updates.
- **How it Works:**
    - When the system starts, components fetch configuration details such as MQTT topics and operational thresholds from the Data Catalog.
    - If new devices are added, the catalog updates its registry dynamically.

- **Connections:**
    - **REST Connection** to the **all microservises** for retrieving configuration settings.

2. Soil Moisture Sensor (MQTT Publisher)

- **What it Does:**
    - Continuously measures the soil moisture level in the garden.
    - Sends real-time moisture data to the Message Broker using MQTT.
- **How it Works:**
    - The sensor is configured during initialization by fetching MQTT topics and thresholds from the Data Catalog via REST.
    - Periodically publishes soil moisture data to a specific MQTT topic.
- **Connections:**
    - **REST Connection** to the **Data Catalog** for initial configuration.
    - **MQTT Connection** to the **Message Broker** for publishing real-time soil moisture data.

3. Weather Sensor (MQTT Publisher)

- **What it Does:**
    - Measures environmental conditions such as temperature, humidity, and rainfall.
    - Publishes real-time weather data to the Message Broker.
- **How it Works:**
    - Configures MQTT topics and thresholds by retrieving settings from the Data Catalog via REST.
    - Continuously monitors weather conditions and sends updates to a designated MQTT topic.
- **Connections:**
    - **REST Connection** to the **Data Catalog** for initial setup.
    - **MQTT Connection** to the **Message Broker** for publishing weather data.

4. Central Controller (Raspberry Pi)

- **What it Does:**
    - Core of the system that processes sensor data, applies decision-making logic, and sends irrigation commands.
    - Acts as the main consumer of data from the Message Broker .
- **How it Works:**
    - Subscribes to soil moisture and weather data from the Message Broker.
    - Fetches device settings, operational thresholds, and configurations from the Data Catalog.
    - Processes data to decide when and how much to irrigate, then publishes commands to the Message Broker.
- **Connections:**
    - **REST Connection** to the **Data Catalog** for fetching configuration data.
    - **MQTT Connection** to the **Message Broker** to subscribe to sensor data and publish commands.

5. Message Broker (MQTT)

- **What it Does:**
    - Facilitates communication between all MQTT-enabled components by managing publish/subscribe messaging.
- **How it Works:**
    - Receives real-time data from publishers (Soil Moisture Sensor, Weather Sensor).
    - Delivers this data to subscribers (Central Controller, Statistics Web Server).
    - Sends irrigation commands from the Central Controller to the Water Pump.
- **Connections:**
    - **MQTT Connection** to the **Soil Moisture Sensor** and **Weather Sensor** (publishers).
    - **MQTT Connection** to the **Central Controller** and **Statistics Web Server** (subscribers).
    - **MQTT Connection** to the **Water Pump** to deliver irrigation commands.

6. Water Pump (MQTT Subscriber)

- **What it Does:**
  - Actuator that executes irrigation commands from the Central Controller.
- **How it Works:**
  - Configures its MQTT topic during initialization by retrieving details from the Data Catalog.
  - Subscribes to the relevant MQTT topic to listen for irrigation commands.
  - Executes commands such as starting or stopping irrigation based on these messages.
- **Connections:**
  - **REST Connection** to the **Data Catalog** for configuration setup.
  - **MQTT Connection** to the **Message Broker** to receive irrigation commands.

7. Statistics Web Server

- **What it Does:**
  - Aggregates data from the Message Broker for analysis and provides REST APIs for sharing insights.
- **How it Works:**
  - Subscribes to soil moisture and weather data topics on the Message Broker.
  - Logs and analyzes data to generate insights such as average soil moisture or irrigation trends.
  - Shares aggregated data with the Telegram Bot through REST APIs.
- **Connections:**
  - **MQTT Connection** to the **Message Broker** to subscribe to sensor data.
  - **REST Connection** to the **Telegram Bot** for serving data and notifications.
  - **REST Connection** to the **Data Catalog** for configuration setup.

8. Telegram Bot (REST Consumer)

- **What it Does:**
    - Provides user interaction with system.
- **How it Works:**
    - Fetches system insights and data (e.g., soil moisture, irrigation status) from the Statistics Web Server via REST.
    - showing to user the data of garden with some buttons.
- **Connections:**
    - **REST Connection** to the **Statistics Web Server** for data exchange.

5. Desired Hardware components

Device Name	**Quantity**	**Needed for…**
Soil Moisture Sensor	**1**	**Measuring soil humidity.**
Weather Sensor	**1**	**Measuring local weather conditions.**
Central Controller (Raspberry Pi)	**1**	**Central Controller.**
Water Pump	**1**	**Automated irrigation.**
Statistics Web Server	**1**	**Data aggregation and REST API.**
MQTT Message Broker	**1**	**Managing publish/subscribe messaging.**
Telegram Bot Integration	**1**	**Telegram User interaction**