IoT-Based Smart Home Automation using AWS IoT Core

A Course Project Report Submitted in partial fulfillment of the course requirements for the award of grades in the subject of

CLOUD BASED AIML SPECIALITY
(22SDCS07A)

by

Karpuram Tejaswini

(2210030434)

Under the esteemed guidance of

Ms. P. Sree Lakshmi
Assistant Professor,
Department of Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

K L Deemed to be UNIVERSITY

Aziznagar, Moinabad, Hyderabad,
Telangana, Pincode: 500075

April 2025

K L Deemed to be UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Certificate

This is Certified that the project entitled **"IoT-Based Smart Home Automation using AWS IoT Core"** which is a experimental &/ theoretical &/ Simulation&/ hardware work carried out by Karpuram Tejaswini (2210030434), in partial fulfillment of the course requirements for the award of grades in the subject of **CLOUD BASED AIML SPECIALITY**, during the year **2024-2025**. The project has been approved as it satisfies the academic requirements.

Ms.P.Sree Lakshmi

Course Coordinator

Dr. Arpita Gupta

Head of the Department

Ms. P. Sree Lakshmi

Course Instructor

CONTENTS

	Page No.
1. Introduction	4
2. AWS Services Used as part of the project	5
3. Steps involved in solving project problem statement	6
4. Stepwise Screenshots with brief description	7
5. Learning Outcomes	16
6. Conclusion	18
7. Future Enchancement	19
8. References	20

1. INTRODUCTION

Home automation has transformed the way people interact with their living spaces, offering convenience, energy efficiency, and enhanced security. By integrating IoT with cloud computing, users can remotely monitor and control devices in real-time [1]. This project demonstrates the capabilities of AWS IoT Core in establishing a smart automation system without the need for physical hardware. Instead of using an actual sensor, a Python-based simulation is employed to generate temperature data, which is published to an MQTT topic in AWS IoT Core [2].

The platform processes this data and evaluates conditions through an IoT rule, determining whether predefined criteria, such as a temperature threshold, are met. When the temperature surpasses the limit, an IoT rule triggers an AWS Lambda function, which in turn invokes Amazon Simple Notification Service (SNS) to send an alert email to the user [4].

This project showcases the effectiveness of AWS cloud services in handling real-time data, automating responses, and providing seamless communication between IoT components. The integration of AWS IoT Core, Lambda, and SNS forms a scalable and event-driven architecture that can be applied to various real-world scenarios [3]. Industries such as smart home automation, industrial equipment monitoring, environmental tracking, and precision agriculture rely on similar IoT-based implementations to ensure timely decision-making and operational efficiency [2].

By simulating the IoT ecosystem without requiring specialized hardware, this project enables easy experimentation and learning, offering a practical approach to understanding IoT-based automation in a cloud-driven environment. As IoT technology continues to evolve, such frameworks can be extended to include advanced analytics, predictive maintenance, and AI-driven decision-making, making them invaluable in modern automation systems [5].

2. AWS Services Used as part of the project

AWS IoT Core

AWS IoT Core serves as the central platform for connecting and managing IoT devices. In this project, it is used to simulate a temperature sensor by publishing data to an MQTT topic [2]. IoT Core enables secure communication and real-time data processing, allowing the system to respond to sensor readings dynamically.

AWS Lambda

AWS Lambda provides serverless computing capabilities, enabling automatic execution of code when a trigger event occurs. In this implementation, Lambda is triggered by an IoT rule whenever the temperature reading exceeds a specified threshold [3]. The function processes the data and interacts with other AWS services to automate responses.

Amazon Simple Notification Service (SNS)

Amazon SNS is used for real-time alerting and notifications. When the Lambda function detects an abnormal temperature reading, it sends an email notification to the user through SNS [4]. This ensures timely alerts, allowing users to take necessary action in response to critical conditions.

Amazon CloudWatch

Amazon CloudWatch is employed for monitoring and logging system activities. It records events from AWS IoT Core and Lambda, providing valuable insights into system performance. CloudWatch logs assist in debugging and optimizing workflows by tracking message flows and function executions [5].

3. Steps involved in solving project problem statement

Step 1: Setting Up AWS IoT Core

- Started by setting up AWS IoT Core via the AWS Management Console.
- Created an IoT Thing to represent the virtual device.
- Downloaded the Connection Kit with certificates, private keys, and a sample script.
- These credentials ensure secure, authorized communication with AWS IoT Core.

Step 2: Connecting the Device and Testing MQTT Communication

- After setting up AWS IoT Core, the next step was testing device communication using MQTT.
- MQTT is a lightweight protocol for reliable device-to-cloud messaging.
- Opened the MQTT Test Client in AWS IoT Core.
- Subscribed to a topic to listen for incoming messages.
- Published a test message to the same topic.
- Successful message reception confirmed the communication was working properly.

Step 3: Configuring Message Routing with AWS IoT Rules

- Configured AWS IoT Core rules to process incoming messages from the virtual device.
- Created an IoT Rule with a unique name for easy identification.
- Used an SQL query in the rule to filter relevant data (e.g., temperature).
- Set the rule's action to invoke a Lambda function.
- The Lambda function processed the data and triggered alerts if temperature crossed a set threshold.
- This ensured only meaningful data was handled and appropriate actions were taken.

3.4 Creating and Configuring AWS Lambda

- AWS Lambda was used to process temperature data and trigger alerts.
- A Lambda function was created using Python as the runtime.
- The function extracted temperature values and compared them to a set threshold.
- If the threshold was exceeded, an alert was triggered.
- Permissions were set to allow AWS IoT Core to invoke the function.
- After testing, the function was deployed and linked to the IoT Rule.
- This enabled automatic responses to temperature changes in real-time.

3.5 Monitoring and Debugging with CloudWatch Logs

- After deploying the Lambda function, Amazon CloudWatch was used for monitoring and debugging.
- CloudWatch Logs were enabled to capture detailed execution records.
- Logs helped track issues and verify if temperature alerts were triggered correctly.
- The CloudWatch console provided real-time insights into system behavior.
- This ensured smooth operation by quickly identifying and fixing any errors.

3.6 Setting Up SNS Alerts for Email Notifications

- Amazon SNS was set up to send real-time email alerts for high-temperature events.
- Created an SNS Topic and subscribed an email address to it.
- Verified the email subscription via a confirmation link.
- Updated the Lambda function to publish alerts to the SNS Topic when the temperature exceeded the threshold.
- Tested the setup by simulating a high-temperature event and confirming email receipt.
- This completed the system's ability to send real-time notifications effectively.

4. Stepwise Screenshots with brief description

Step 1: Connect Device to AWS IoT Core

Begin by accessing AWS IoT Core and connecting it to your device. Download the Connection Kit needed for secure communication.

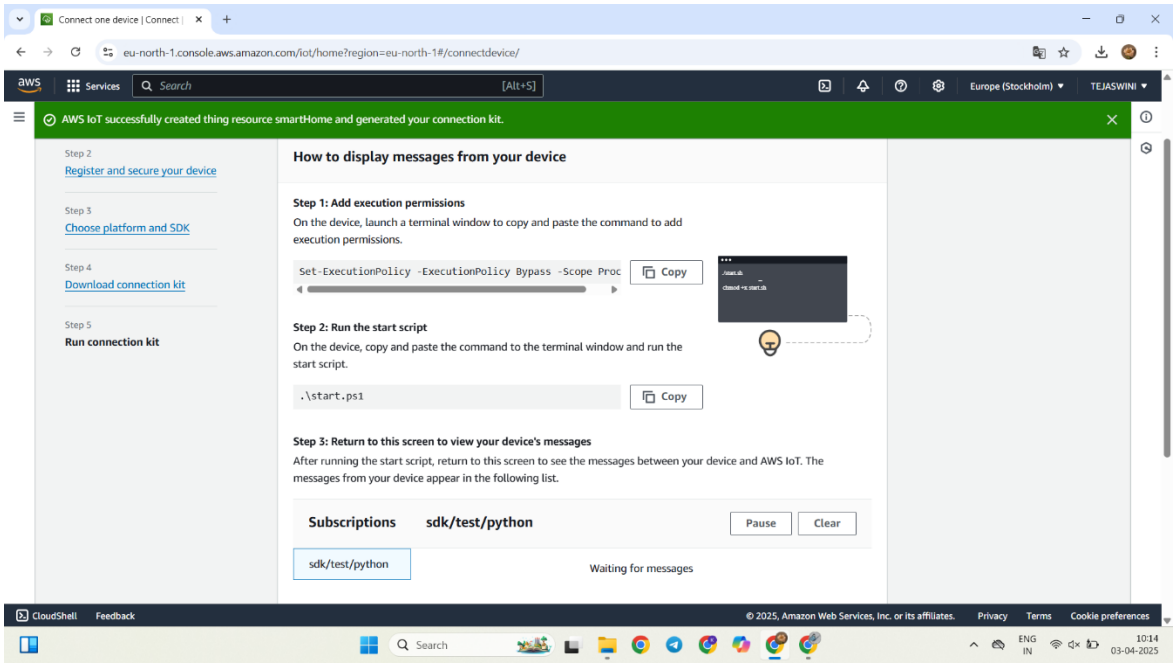

Fig 4.1: Opening AWS IoT Core and initiating the connection process

Step 2: Start the Connection Script

Unzip the downloaded file and run the provided script in the command prompt to establish a secure connection.

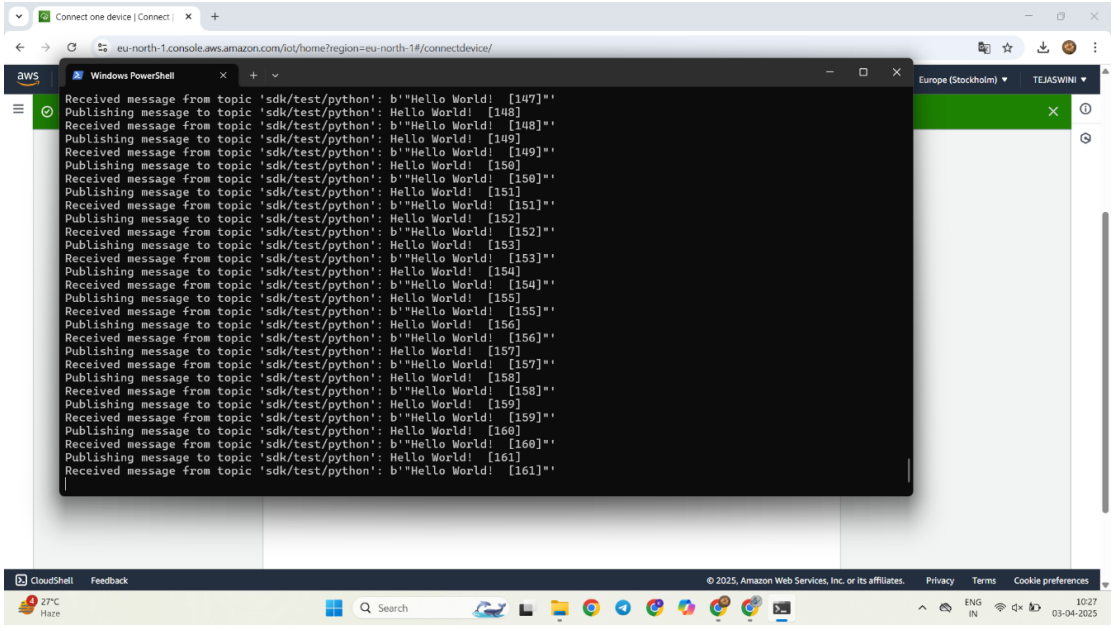

Fig 4.2: Running the start script from the unzipped Connection Kit

Step 3: Subscribe to MQTT Topic

Use the MQTT Test Client to subscribe to a topic where your virtual device will publish messages.

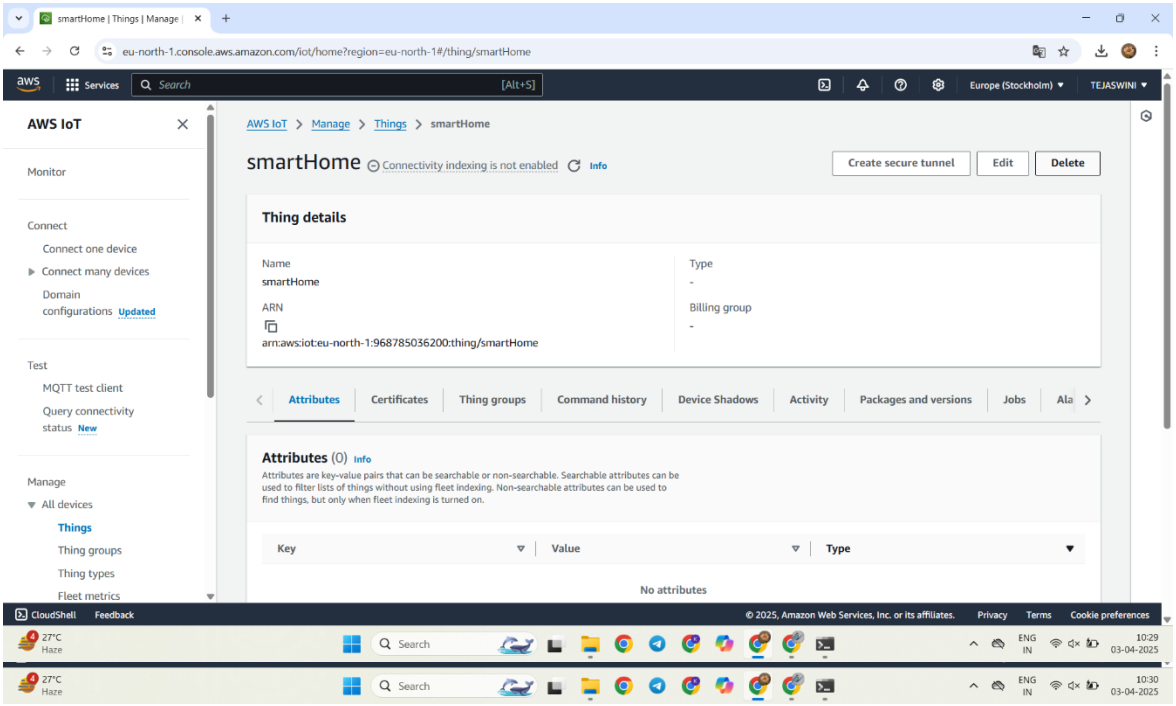

Fig 4.3: Subscribing to a topic in the MQTT Test Client

Step 4: Publish a Message to MQTT Topic

Publish a test payload to verify if the device is successfully sending data to the topic.

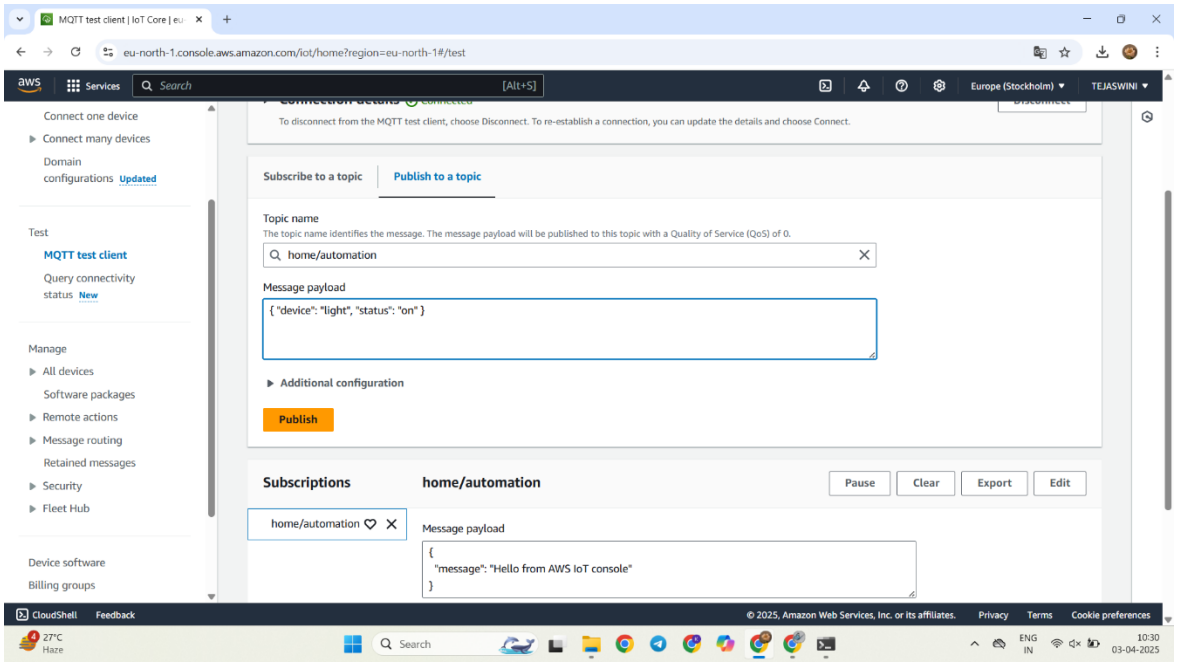

Fig 4.4: Publishing test data to the same topic

Step 5: Create a Message Routing Rule

Open the Message Routing section and create a new rule to process incoming data.

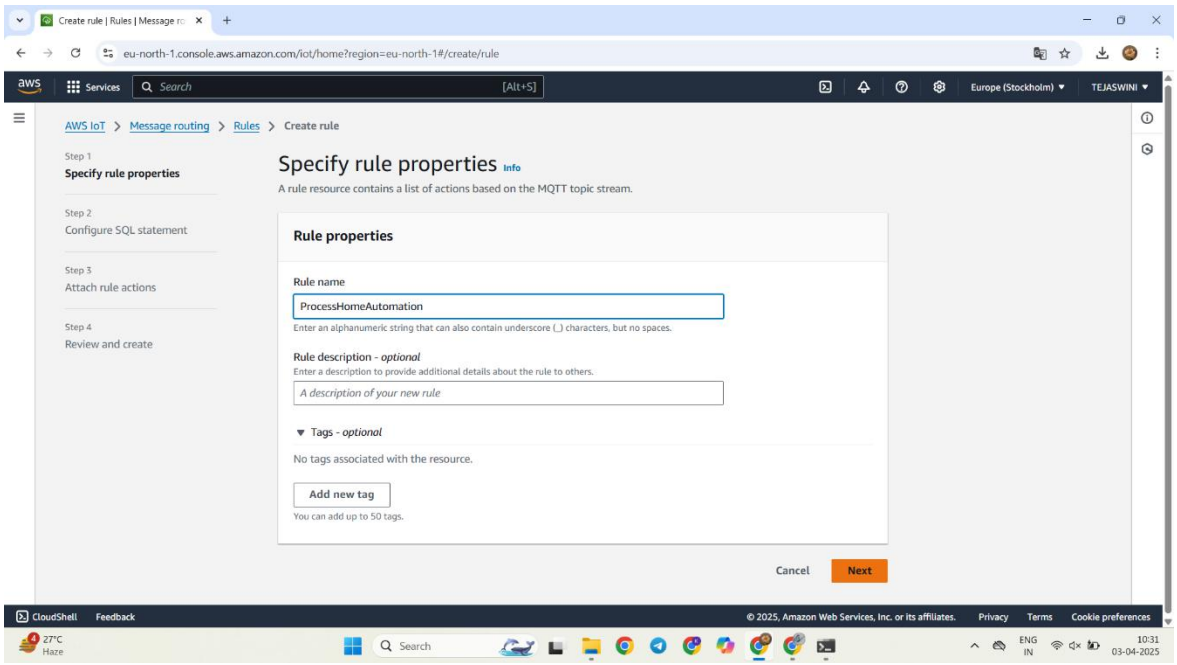

Fig 4.5: Creating a rule in AWS IoT Core for message routing

Step 6: Add Filtering Logic with SQL

Specify SQL conditions in the rule to filter relevant data like temperature readings.

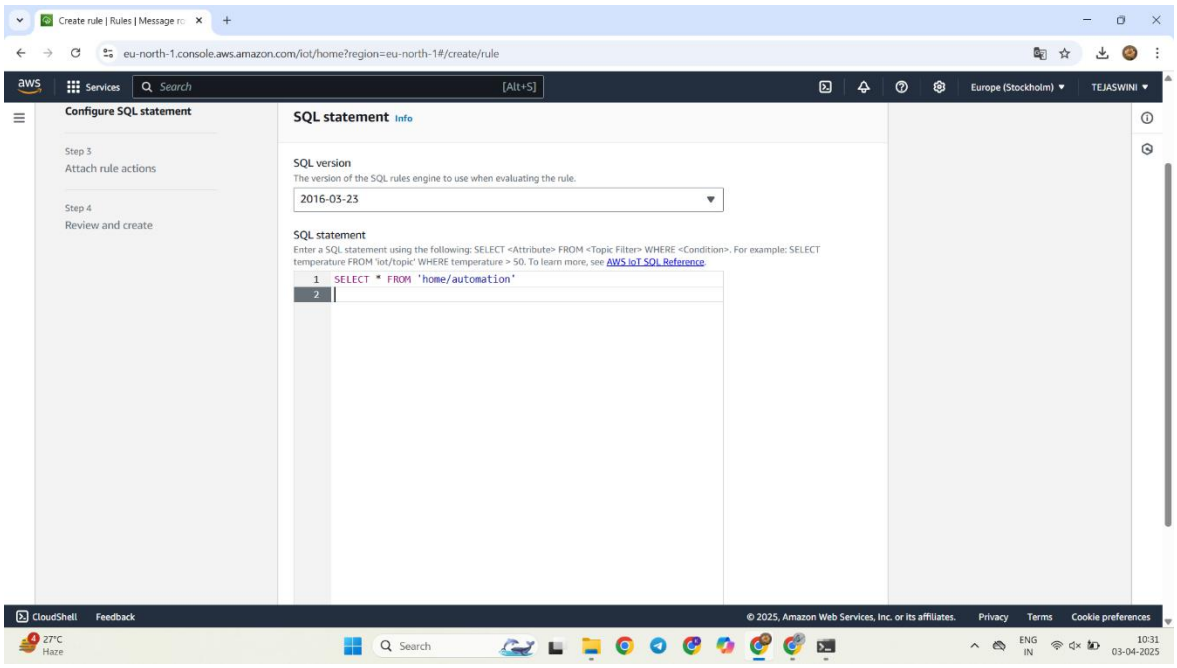

Fig 4.6: Using SQL to filter messages

Step 7: Create a Lambda Function

Open AWS Lambda and create a function that will handle data processing and alert logic.

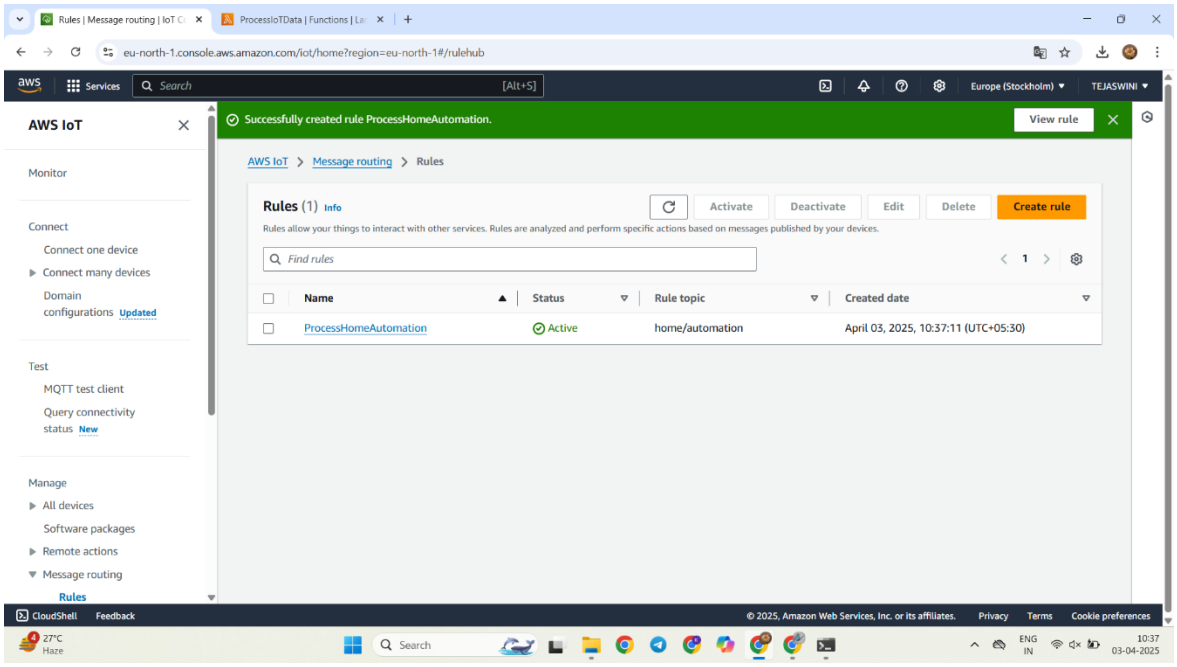

Fig 4.7: Creating a new AWS Lambda function

Step 8: Monitor Logs in CloudWatch

Use CloudWatch to monitor logs and check for data being received and processed correctly.

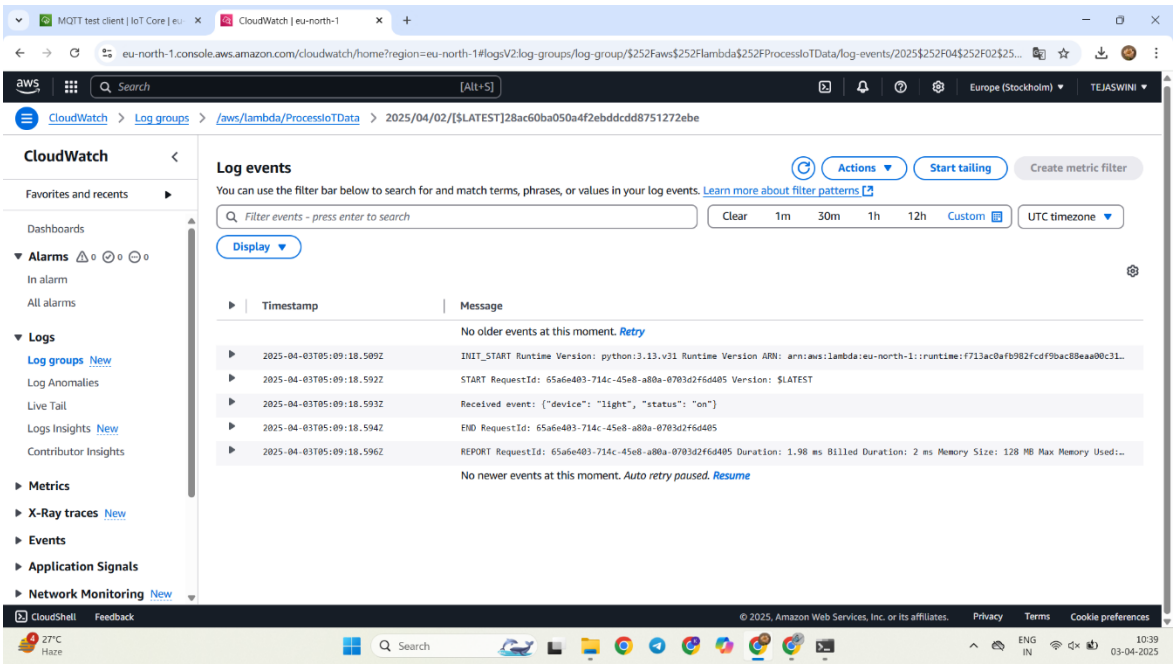

Fig 4.8: Viewing logs in Amazon CloudWatch

Step 9: Track Log Events
View updated logs for each temperature message, helping with debugging and performance checks.

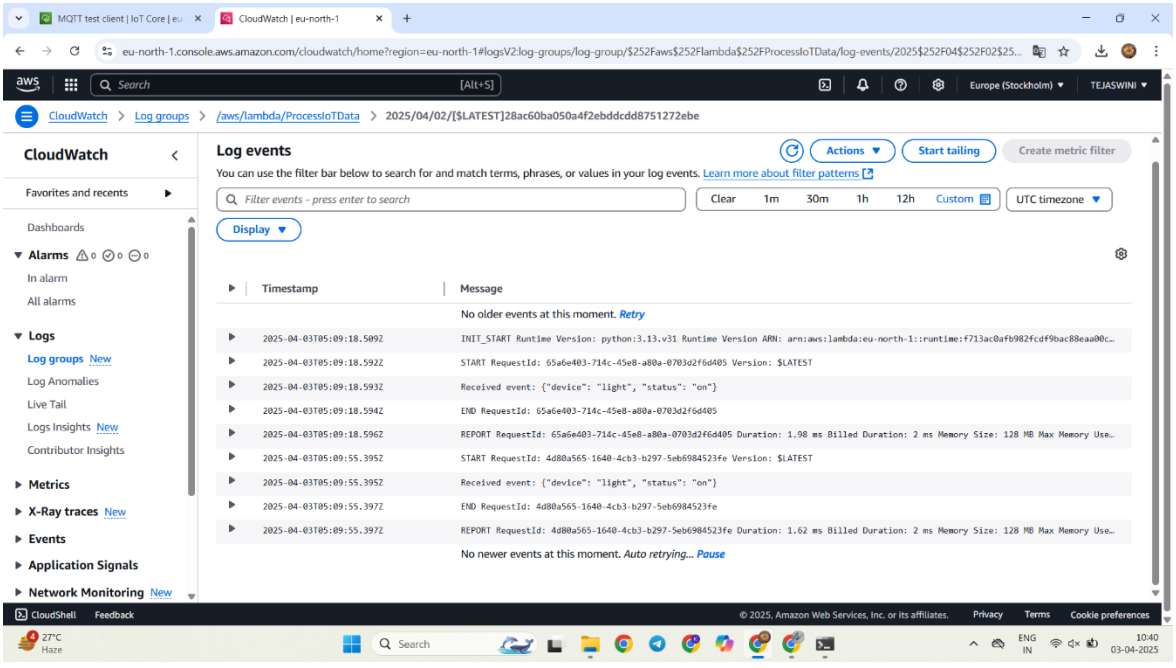

Fig 4.9: Updated log events with each function execution

Step 10: Set Up SNS Topic for Alerts

Set up Amazon SNS and create a topic to send email or SMS alerts when high temperatures are detected.

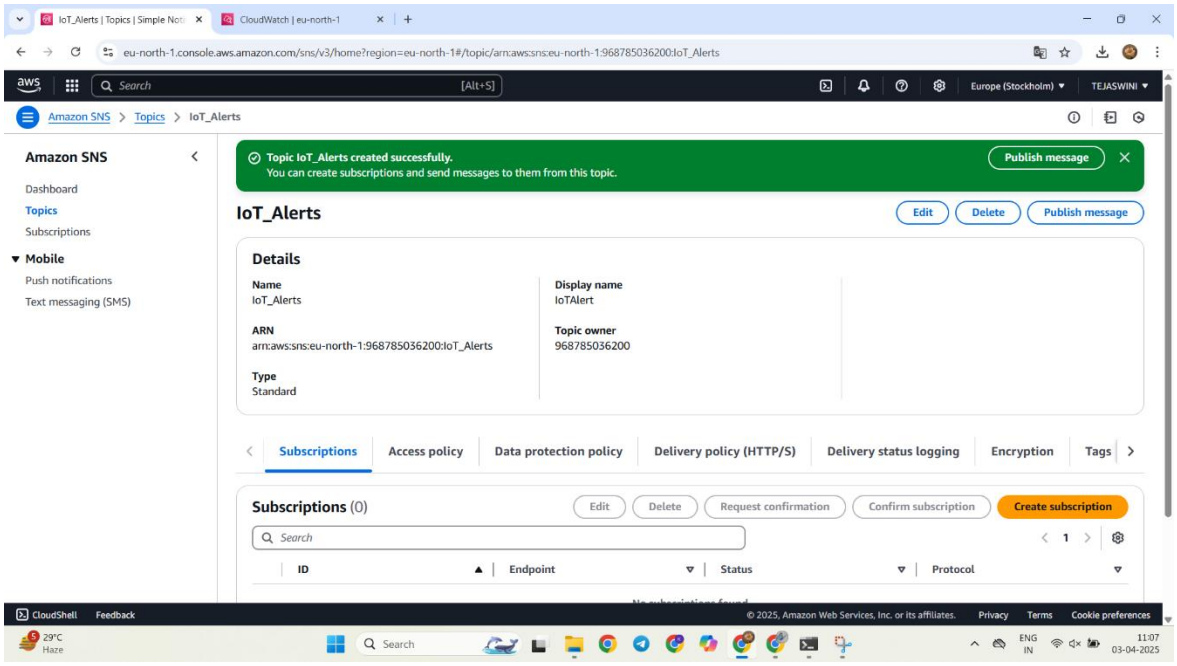

Fig 4.10: Creating an SNS topic for alert notifications

Step 11: Confirm SNS Subscription

After entering your email or number, verify it through the confirmation link sent by SNS.

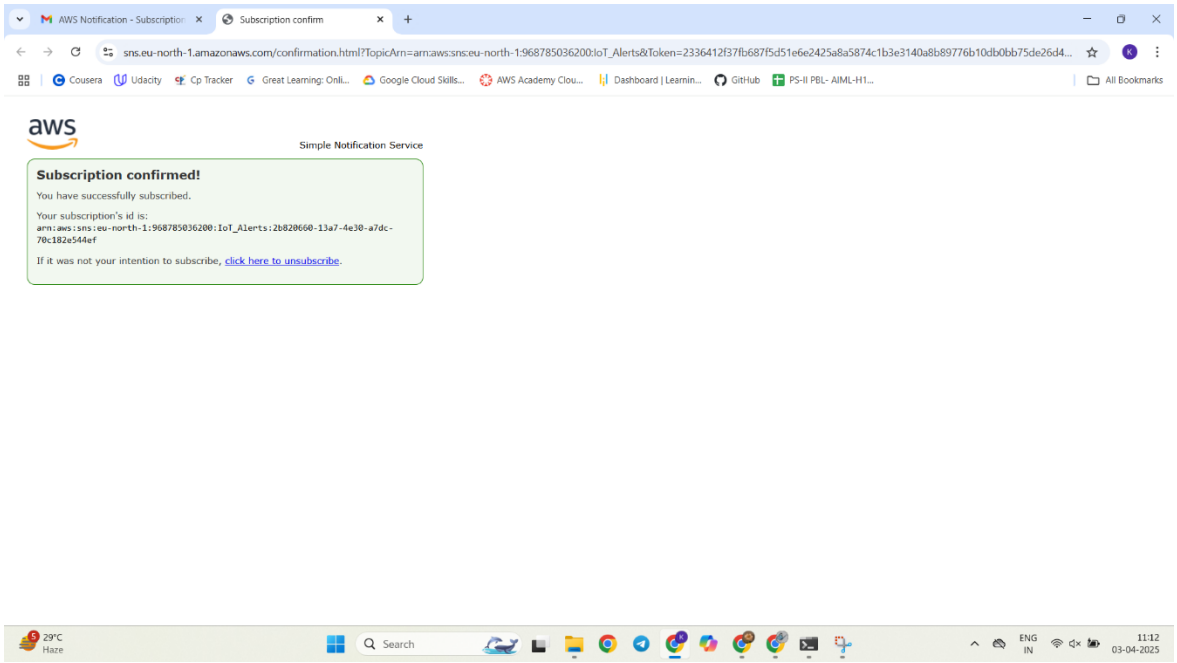

Fig 4.11: Confirming email/SMS subscription for SNS

Step 12: Check Subscription Status

Once verified, the status changes from “Pending” to “Confirmed” in the SNS console.

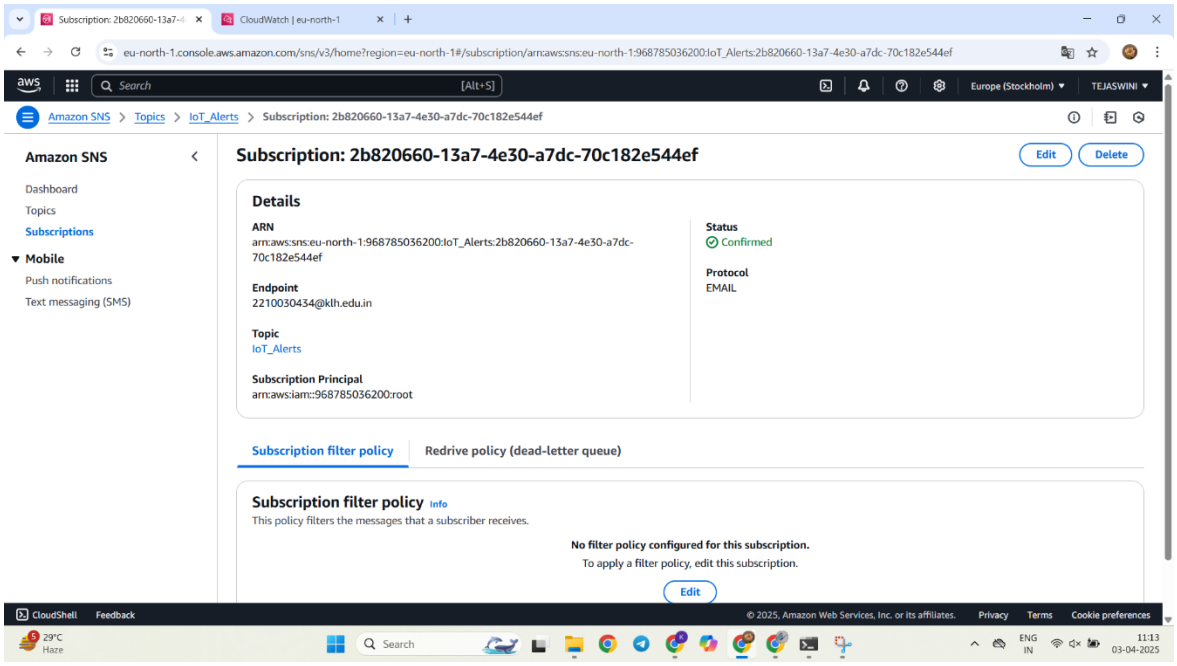

Fig 4.12: Verifying confirmed subscription status

Step 13: Link Rule to SNS for Notifications

Create or modify an IoT rule to publish alerts to SNS when certain conditions (e.g., temperature > 35°C) are met.

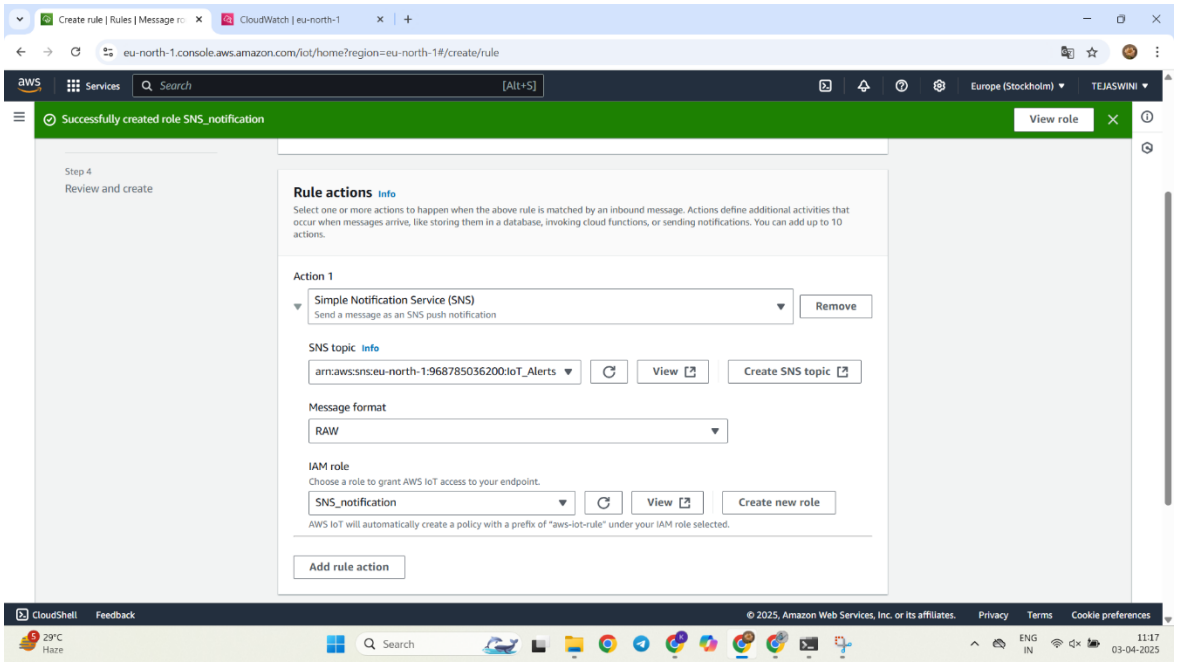

Fig 4.13: Creating a rule to trigger SNS notifications

Step 14: View All Created Rules

All configured rules can be viewed and managed under the “Message Routing” section in AWS IoT Core.

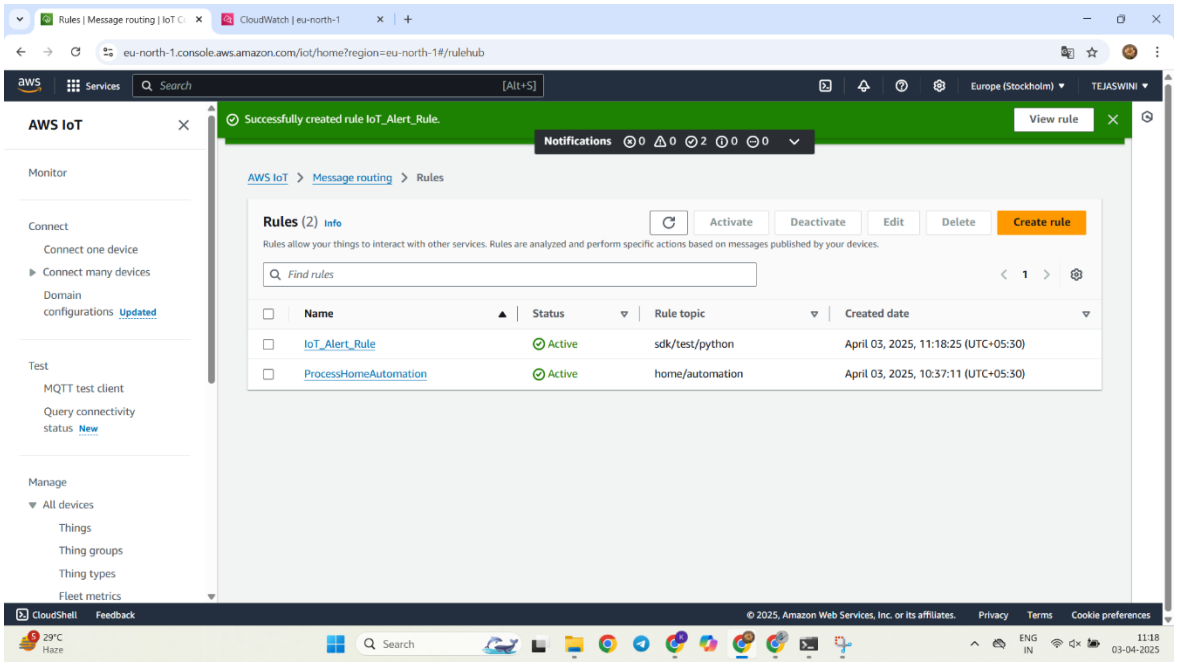

Fig 4.14: List of rules in the Message Routing section

Step 15: Receive Alert Notifications
Check your email or SMS to confirm that alerts are being received as expected when the rule conditions are triggered.

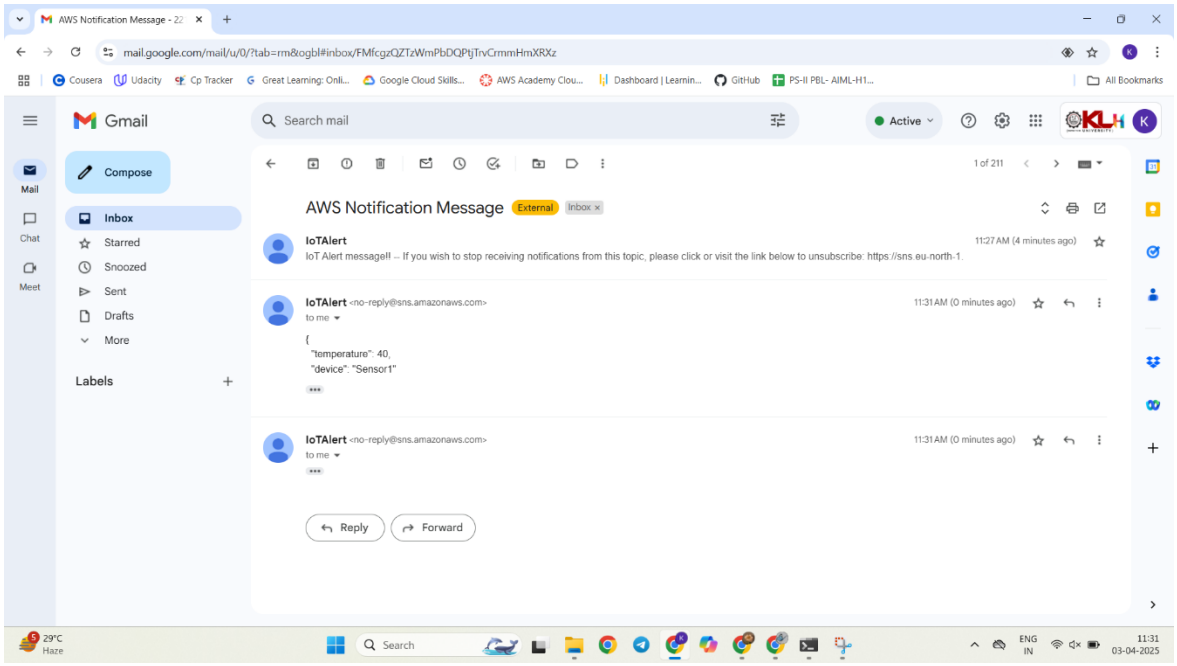

Fig 4.15: Email alert received when temperature exceeds threshold

5. Learning Outcomes

Understanding AWS IoT Core

Configuration and management of virtual IoT devices were explored using AWS IoT Core. Devices, known as "things", were set up with certificates and secure policies to establish trust. MQTT communication was enabled between devices and the cloud, facilitating bi-directional messaging. AWS IoT Core was used as the central platform for managing device connectivity and routing data to other AWS services for further processing.

Hands-on Experience with MQTT Protocol

The MQTT protocol was utilized for real-time, lightweight communication between devices and the cloud. Key operations such as publishing messages to topics and subscribing to them were practiced. The topic hierarchy and wildcard support were studied to manage message flow effectively. Different Quality of Service (QoS) levels were evaluated to understand message delivery reliability and efficiency in constrained networks.

Creating and Managing AWS Lambda Functions

AWS Lambda was employed to process IoT data in a serverless manner. Functions were written to transform and forward incoming messages based on defined logic. Lambda was triggered by IoT Rules, allowing automatic execution in response to specific conditions. This serverless approach enabled dynamic data processing without managing any underlying infrastructure, enhancing scalability and responsiveness.

Implementing Message Routing and Rule-Based Automation

AWS IoT Rules were used to filter and route data from MQTT topics. SQL-based rules were created to extract specific information from messages and initiate actions such as storing data in DynamoDB or invoking Lambda functions. Conditional logic enabled precise control over automation workflows, supporting efficient, event-driven operations in the IoT environment.

Integrating AWS Services for End-to-End Automation

Integration of AWS IoT Core with Lambda and SNS enabled seamless, end-to-end automation. Sensor data triggered predefined rules, which initiated Lambda functions and notifications through SNS. This setup demonstrated the power of AWS services working together to support responsive and automated systems, reducing the need for manual intervention and improving system efficiency.

Monitoring and Debugging with CloudWatch

Amazon CloudWatch was used to monitor system performance and log execution details of Lambda functions. Logs provided visibility into data flow and helped identify and resolve issues. Custom metrics and alarms were configured to track key parameters such as error rates and execution time, ensuring smooth operation and quick debugging of the IoT system.

Implementing AWS SNS for Alerts and Notifications

Amazon SNS was configured to send real-time alerts based on IoT-triggered events. Email notifications were sent when sensor data exceeded predefined thresholds. SNS topics and subscriptions enabled the distribution of alerts to multiple recipients, ensuring timely communication of critical information and supporting prompt action.

Security and Permissions Management

Security was maintained using AWS Identity and Access Management (IAM). Roles and permissions were defined to control access between AWS services securely. X.509 certificates were used for device authentication. Best practices were followed to implement least-privilege access and ensure that all service interactions were authorized and compliant with security policies.

6. Conclusion

This project demonstrated how AWS IoT Core and other AWS services can be used to simulate a smart home system without physical devices. By integrating MQTT, Lambda, SNS, and CloudWatch, the system efficiently monitored virtual temperature data, triggered alerts based on thresholds and sent real-time email notifications. AWS Lambda enabled serverless, on-demand data processing, while SNS and CloudWatch ensured timely alerts and system monitoring. IoT Rules helped filter relevant data, and built-in security features protected against unauthorized access. Overall, the project highlighted the efficiency, scalability, and security of cloud-based IoT solutions, offering hands-on experience with real-time data processing and cloud automation.

7.Future Enhancement

For future enhancements, the IoT system could benefit from integrating real-time analytics using Amazon Kinesis, enabling faster and more efficient data processing. AWS IoT Device Defender would improve security by continuously monitoring and addressing potential vulnerabilities in devices. To further optimize performance, predictive maintenance capabilities could be introduced using Amazon SageMaker, allowing the system to predict and prevent device failures. Edge computing through AWS IoT Greengrass would reduce latency by processing data locally, which is especially useful for remote areas with limited connectivity. Automation of workflows using AWS Step Functions could streamline the entire process, making it more efficient and less prone to human error. Voice command integration with Alexa would enhance user interaction with the system, while long-term data storage in Amazon S3 would facilitate extensive data analysis over time. Finally, leveraging Amazon QuickSight for visual reporting would provide valuable insights into system performance, trends, and patterns, enabling data-driven decisions. These enhancements would collectively improve scalability, security, automation, and overall system efficiency.

8. References

[1]Amazon Web Services (AWS). (2023). AWS IoT Core: Securely Connect Internet of Things (IoT) Devices to the Cloud. Retrieved from https://aws.amazon.com/iot-core/

[2]Amazon Web Services (AWS). (2023). AWS IoT Core Security: Device Authentication, Encryption, and Access Control. Retrieved from https://docs.aws.amazon.com/iot/latest/developerguide/iot security.html

[3] Amazon Web Services (AWS). (2023). AWS IoT Core Rules Engine: Automate Device Data Routing and Trigger Actions. Retrieved from https://docs.aws.amazon.com/iot/latest/developerguide/iot rules.html

[4] Amazon Web Services (AWS). (2023). Amazon SNS message delivery. Retrieved from https://docs.aws.amazon.com/sns/latest/dg/sns-getting-started.html

[5] Amazon Web Services (AWS). (2023). Amazon CloudWatch. Retrieved from https://aws.amazon.com/cloudwatch/