COSMOS

A Ubiquitous Automation Solution

CSC 591 (059) INTERNET OF THINGS FALL 2018

CLASS PROJECT DESIGN DOCUMENT

DECEMBER 2ND, 2018

Jubeen Shah
Computer Science
North Carolina State University
jnshah2@ncsu.edu

Giang Nguyen
Computer Engineering
North Carolina State University
ghnguye2@ncsu.edu

Content

List of Figures

1. Introduction

Internet of Things (IoT) is a burgeoning field which is the next big thing in the world of automation. IoT devices usually have a universal application as they are not constrained to any one particular domain. With more advances in technology and the impact it can have on our daily lives, IoT can have various applications in our home that allow improving the way we live and carry out our daily chores. Smart homes are no longer an unexplored territory, and new devices are being invented to make our lives easy and make us feel secure. However, it has been known how expensive these individual devices tend to get. A single household has multiple lamps, and the cost tends to exponentially increase while the willingness to pay for each bulbs by the consumers tends to decrease with increment in the number of devices being bought.

Our project is a design and architectural extension of the project[1] undertaken by one of the authors (Jubeen Shah) in his undergraduate study. The idea of COSMOS, in brief, is to make affordable custom IoT end-points, that can be managed via a single device or custom designed interface. These devices could be any iOS, Android or any smart device which can access the internet in the desired automation environment. Few examples of the managed end-points that will be developed are intrusion sensors, fire sensors, connected RGB LED Lights, Connected Switches & Switchboards, integration of – temperature & humidity sensors, Air Quality Sensor, UV Sensor, Air Pressure Sensor and so on. In a related trend, and keeping the user experience into consideration, other forms of digital interactions including personal digital assistants – like Siri, Alexa, and Google Assistant, are also considered for integration into the project.

This document outlines the business and technical requirement of COSMOS in Sections 2 and 3. The elements of the IoT system architecture including the things, people, sensors and actuators, involved are described in Section 4. Questions pertaining to the architecture of the project are addressed in Section 5, while alternate design methodologies considered with their limitations are discussed in Section 6. A detailed overview of the IoT-A architecture with the related diagrams and feedback model are discussed in Section 7 and 8. The type of analytics, computing models, and open source standards used in the project are discussed in Section 9 and 10. Details about the management tasks and virtualization are deliberated in Section 11. Finally, we wrap with some future work in Section 12.

[1] https://github.com/jubeenshah/COSMOS

2. Business Requirements

The *COSMOS* project should be designed to help the users/owner of the house in the following ways:

- **BR 1 –** Monitor various aspects of the environment of home to give the user a detailed overview of the conditions in the house.

- **BR 2 –** Record the values from sensors positioned in the house and store them in a persistent and reliable environment.

- **BR 3 –** Build a system that is scalable to allow for the addition of other devices, that were previously not considered for the implementation or add the same type of device.

- **BR 4 –** Visualize the data collected in the form of graphs to allow for easy interpretation by the user of the system.

- **BR 5 –** Develop a machine learning model from the collected data to allow for automated decision taking by the system.

- **BR 6 –** Manage the states of Switches, Irrigation system, and ambient lights in the environment.

- **BR 7 –** Manage the states of the devices in the environment automatically, manually and using a schedule from both, within the network and remotely.

- **BR 8 –** Manage the updates to the particular device automatically, manually, and using a schedule to ensure the security of the devices and vulnerabilities are sought after.

- **BR 9 –** Control and monitor the states of the things in the environment using a mobile application, and a web application and through a behavioral model.

- **BR 10 –** Integrate with voice recognition services such as Alexa, Siri, and Google Assistant to allow an additional layer of user experience and interaction with the system under consideration.

- **BR 11 –** Ensure durability of data in case of a disaster to allow for analysis of data

- **BR 12 –** Store the recorded data into a persistent data store.

- **BR 13 –** Make the model react to the predictions from the machine learning model.

- **BR 14 –** Monitor the state of doors and windows in the house which can be a point of entry into the house.

- **BR 15 –** Classify the type of plant the user has in his house in each room.

As a consequence of helping the user of the COSMOS in these ways, we would expect an increase in the productivity of the user for daily chores. We would also expect the users to save up on electricity costs. For this reason, we believe that the this type of system could have significant commercial potential.

3. Technical Requirements

In order to meet the business requirements discussed in Section 2, the system must meet the technical requirements **TR 1-6** set out in this section. Even though we focus on these 6 technical requirements there are several more that the system would need as a part of a complete implementation.

- **TR 1 –** Custom designed hardware product. For example, sensors which compliment each other (Temperature, Humidity, Air Pressure, Light Intensity) can be put together on a single circuit board. This would enable the system to monitor the environment in a more complete fashion and thus satisfy **BR 1** and **BR 14**.
  - **TR 1.1 –** Obtain information about the environmental conditions
  - **TR 1.2 –** Obtain the status of the devices in the environment
  - **TR 1.3 –** Reacting to signals sent over from the cloud environment

- **TR 2 –** A database cloud service (DynamoDB) would be needed to store the data collected into a NoSQL database. The high availability and reliability of the cloud service should satisfy the need for persistent storage for **BR 2** and **BR 3**.
  - **TR 2.1 –** Couple with AWS Machine Learning Service, **BR 5** can be satisfied.
  - **TR 2.2 –**The database solution with the machine learning service should help solve **BR 4** for visualization of data through IoT Analytics.

- **TR 3 –** With a streaming service already in place, a service such as AWS Simple Notification Service would enable to send notifications (email, SMS) to the registered users there yet satisfying **BR 6**.
  - **TR 3.1–** Fire off events such as lambda function in an AWS environment, **BR 7**, **BR 8**, and **BR 9** can be satisfied.

- **TR 4 –** A Mobile/Web application with API connection to the AWS services should allow for interaction between the devices in the house and the User. This would help satisfy **BR 10**.
  - **TR 4.1 –** For **BR 11** Depending on the operating system of the mobile being used; Siri or Google Assistant can be incorporated in the project
  - **TR 4.2 –** Integrate with the local digital personal assistant available – Siri, or Google Assistant.
  - **TR 4.3 –** Provide Alexa integration if a digital personal assistant is not available for integration.

- **TR 5 –** Transfer of data from DynamoDB to an object-based storage (S3) with policies for lifecycle management and archival should help with **BR 12**.
  - **TR 5.1 –** Transfer the data from Standard S3 to Infrequently-Accessed (IA) S3 data store after 30 days
  - **TR 5.2 –** Transfer the data from IA-S3 to Glacier after 60 days.

- **TR 6 –** Use of a pre built machine learning model for classification of the plants and using inputs from the devices outlined in TR 1, should help with **BR 13** and **BR 15**.
  - **TR 6.1 –** Obtain information plant in the environment using the Amazon Recognition service
  - **TR 6.2 –** Obtain the water requirement for the plant under consideration using a lambda function trigger from a DynamoDB
  - **TR 6.3 –** Trigger AWS Lambda functions to take necessary actions to turn ON/OFF the irrigation system based on the soil moisture reading from the plant under consideration.

4. The elements of the IoT system

4.1.Things

This sub-section describes the things in the project, and their functionality in minor details.

- **Soil –** The soil moisture sensor would monitor the soil to record the moisture levels which satisfies BR 1.

- **Plants –** The water sprinklers would have a camera setup that would monitor the type of plant in the user's home which satisfies **BR 15**

- **Rooms –** Environment station, with all the different sensors, would monitor the rooms for temperature, humidity, air quality, UV index, light intensity which satisfies **BR 1** There would be smart switches deployed in the room that would control the states of the switches and ambient lights in the room

- **Door and Windows –** Intrusion sensor and motion sensor would be used to monitor the doors and windows of the house which satisfies **BR 14**

- **Switch –** The switches would make use of the relays to switch on/off a device. The smart switches would simply make use of multiple relays to switch on/off a device.

Like the irrigation system, the smart switches would be given three triggering mechanisms. The first would be an alert signal from any of the sensors in the environment which would trigger the switch being turned on/off, and the remaining two are discussed in the mobile app.

- **Mobile Device –** GPS would be used to monitor the location of the user and make predictions about the users' movement. The mobile device would be used to give an overview of all the connected things in the environment and also be used to manage and monitor other things in the environment. One of the ways it can manage other things in the environment would be through manual control of the devices by sending an HTTP signal to the server, which in turn sends an MQTT signal/command to the ESP8266 to turn on or off certain devices in the environment – Smart Switches or irrigation system. Another set of information that would be transferred between the mobile app and the server would be the Red, Blue, and Green values to be set in the Ambient Lights. Finally, there could also a scheduler in terms of how frequently certain devices are to be turned on/off. For example, the irrigation system might be configured to turn on every day at 9 am, and turn off at 9:30 am. This can be done using the mobile application. Another, functionality that would be a part of the mobile app would be to allow integration with the in-house operating system's voice assistant – Siri in case of iOS and google assistant in case of Android. For any other operating system, Alexa can be used since, it is an open source assistant provided by Amazon, thus satisfying **BR 11**.

4.2. People

This sub-section outlines the people directly involved in interacting with the project. It also outlines the roles of the people involved. This is not an exhaustive list of the people involved in the project, but the ones that would be directly interacting with the system either to use it on a daily basis (Users), or to solve any problem associated with the devices or services being used a part of the system as whole (AWS Administrator and Local Administrator).

- **Owner/User –** The first type of people would be the direct consumers of the information being generated in the house – the owners of the house and the system. The owner would monitor and manage the states of the things in the environment, and take any action in case of an anomaly.

- **Administrator –** The administrator would be responsible for maintenance of the devices installed in the environment and take actions if the devices are malfunctioning or not functioning. They would also be responsible for managing this information for a group of users across houses.

- **AWS administrator –** The AWS administrator would be responsible for registration and de-registration of any services that the user subscribes through either manually, or automatically through the use of the mobile application or other devices.

4.3. Processes

This section describes in detail few processes that would be a part of the project. It describes the process for registration of a device with the environment under consideration for a particular home. It also provides the series of steps that would be performed whenever there is a need to transfer information between the things in the project and the AWS environment. Finally, we also discuss a few more processes that would be a part of the system but are not described in detail.

4.3.1. Registration of device

It would help with the **BR 1, 2, 7, 8, 9,** and **10** for each of the devices in the project as this would be the starting point for each of the aforementioned BRs. Also, it would make use of **TR 4**, since a mobile application or web application would be needed for interaction with the device.

I. Switch on the device

A. Once on, the device would look for a wireless network to connect to.
B. Once connected, the device would be available on the mobile device, the web application, or the command line interface for the people involved in the project to interact with, for the registration process.

II. Get device information

A. Scan the barcode of the device using a smartphone application, to get the information such as the Unique identifier (UID) and the Name of the thing (device).
B. In case of command line interface registration of the devices, the unique ID from the device itself can be copied into the CLI for registration
C. This information would be saved on the mobile application/ web application, to keep a track of the registered devices and then this information would be

sent to AWS for registration with the IoT core service using the AWS IoT management service.

III. Generate a unique certificate

A. Generate a unique certificate for the thing on the AWS platform using the mobile application/ web application.

IV. Attach the policy for the devices to the generated certificate

A. These policies would allow communicating with other AWS services in the environment

V. Register thing with AWS

A. Use the private key of the thing along with the device certificate to register the thing with AWS environment

VI. Configure the device in the AWS environment

A. Use the publish and subscribe parameters from the devices to generate as many numbers of events in the AWS environment.
B. If the thing is configured to only publish values (for example the environment monitor), as many publishing topics would be generated as there are sensors publishing the values
C. If the thing is configured to be both publish and subscribe, respective topics should be created in the AWS environment

---

4.3.2. Transfer of information between things and AWS

*I. The thing would collect data from the environment as required and described in **BR 1**, and **BR 2**.*

A. This data could in digital or analog form. Such as temperature, humidity, soil moisture, etc.
B. If the data is in analog format, it is converted into digital format.
C. This process would be repeated for all the sensors mounted on the thing.
D. If multiple sensors are connected to the same pins, a multiplexer or a demultiplexer would be used to switch between the sensors connected to avoid interference amongst sensors.

II. The thing would subscribe to events.

A. These events can be manually triggered by the user, for example in the case of switches.

B. These events can be automatically triggered by lambda function from other things in the environment.
C. For example, the soil moisture sensor sending a lambda event to the irrigation system in case of reduced moisture levels in the soil.

III. Publish the data to the publishing topic.

A. Publishing topic would be a unique address to which the things would be sending the MQTT messages too.
B. Based on the publishing topic for the particular thing, the AWS message broker would route the formatted data from the publishing client to the subscribing client.

IV. AWS IoT Core responds to the event by creating a lambda function.

A. Once the subscribing client has received the messages, it would trigger off a lambda function.
B. Using the lambda function; the received data can be processed to carry out the following functionalities
    1. Storing the received data in DynamoDB which satisfies **BR 12**.
    2. Based on the threshold values for the processed data, another lambda would be triggered to react with notifications (SMS, Emails, Push notifications) or triggering actuators in the environment.

4.3.3.Other processes

- Generating graphs from the data collected from the sensors as mentioned in **BR 4**.
- Classification of plants from the image capture as mentioned in **BR 15**.
- Generation of behavioral analytic model based on the use of devices like smart switches, smart irrigation system, and ambient lights as mentioned in **BR 9**.
- Creation of redundant copy of data as described in **BR 11**.

4.4.Data

This sub-sections outlines the type of data that would be collected in the automation environment.

- The data being generated by the sensors as required in **BR 1**, **BR 2**, and **BR 14**, are as follows :
  - Temperature
  - Humidity
  - Air Quality
    - CO2 , CO, LPG, Propane and hydrogen
  - Light intensity
  - Air pressure
  - Soil moisture
  - UV radiation
  - Motion
  - Magnetic sensor status

- The data from the actuators as required in BR 2 are as follows
  - Status of the switches
  - Status of the irrigation system
  - Status of fire alarm
  - Red, Blue, Green values form the Ambient lights

4.5. Sensors and actuators

This sub-section outlines the sensors and actuators that would be a part of the project. The first sub section describes the list of actuators that would be needed, while the second sub-section outlines the list of sensors that would be needed as a part of the Business and Technical requirements. This is not a comprehensive list and more sensors and actuators might be needed for implementation purposes.

4.5.1. Actuators

As required by **BR 7**, **BR 9**, and **BR 15** the following list of actuators are identified

- **Micro-controller (ESP8266) –** The ESP8266 would be the brains of the individual sensors that would have the compute capability to wirelessly transfer information to the cloud server. This would also have the power supply connected to it; either in the form of a battery or any other source of wall outlet (5V).

- **Valves –** The valves would be part of the irrigation system which would be used to control the flow of water to the plants
- **Relays –** The relays would be a part of the smart switches, and irrigation system which would be used to switch ON/OFF the devices in which it is embedded
- **Temperature Controller –** The temperature controller would be a part of the smart thermostat system which would be responsible for setting the temperature of the environment

---

4.5.2.Sensors

This sub-section lists and describes in some detail the sensors that would be needed as a part of the project. This is not an exhaustive list and more/less sensors might be needed as a part of the implementation when considering each house.

- **Temperature and Humidity Sensor (DHT22) –** The temperature and humidity sensor would help record the temperature and humidity of the environment (room/house) with an accuracy unto *±2%* for relative humidity and unto *±0.5 °C (±0.9 °F)*. The primary purpose of the DHT22 sensor is to send updates to the user and other things (actuators) in the environment to take appropriate action based on the readings from the sensor. For example, if the temperature shoots above a threshold value, an alert can be sent to the user that the air conditioner has been turned on, while concurrently sending a signal to the air conditioner to turn on. Similarly, in case of humidity readings, a dehumidifier can be turned on.

- **Air Pressure Sensor (BMP180) –** Air Pressure Sensor can give relative height from the sea level, and this information along with the temperature reading either from BMP180 sensor itself, or the DHT22 sensor, can help control the temperature of the environment more accurately in hilly regions to provide more energy efficiency. According to the datasheet, the BMP 180 sensor has an accuracy of *±0.12 hPa*.

- **Light Intensity Sensor (BH1750) –** Light Intensity sensor would help record the amount luminosity in the environment. This would then be translated into other signals for other actuators in the environment primarily the switches for lights and window blinds. It has a very high resolution that gives values ranging from *1 lux to 65535 lux*.

- **Air Quality Sensor (MQ2) –** The MQ2 sensor would record the concentration of different gases in the environment. The concentration of the gases is calculated on the basis of electrical conductivity. MQ2 Gas sensor has high sensitivity to LPG,

Propane, and hydrogen, and also other combustible steam. When different gasses are present in the higher concentration, the conductivity of the sensor increase.

- **Motion Sensor (PIR) –** The motion sensor is a Passive Infrared sensor that makes use of anomalies in IR generated to detect motion in the environment. This sensor would be able to detect motion, and then based on a certain set of predefined rules and conditions, ESP8266 would be used to send a signal other devices in the environment to generate an alarm.

- **Soil Moisture Sensor, (FC-28) –** Soil Moisture sensor makes use of conductance between two electrodes, to determine the percentage amount of water in the soil. This sensor makes use of an Analog pinto detect the difference in voltage level, which can then be used to translated into digital data.

- **UV Sensor (ML8511) –** ML8511 works on a similar principle such as FC-28. However, it is to be noted that both FC-28 and the ML8511 use the Analog pin to communicate with the micro-controller. What this means, is that only one of them would be able to communicate with the micro-controller at any one time since ESP8266 has only one Analog pin, a Demultiplexer such as the IC 4051 would be needed to switch between the Analog pins between two or more sensors using it.

- **Magnetic sensor –** A magnetic sensor would be used to monitor and sense the state of the door and the windows on which it is installed. If the magnetic sensor is in contact with the reader, then a "closed" message would be sent to the AWS environment, whereas if the sensor is not in contact with the reader, then an "open" message would be sent to AWS environment.

- **Camera –** The camera would be used to classify the type of plant as required in **BR 15.** It would make use of the AWS Rekognition service to classify the plant type.

4.6. Community of things

This sub-section outlines an arbitrary collection of things, as seen in Section 4.1, which are useful from a management perspective . There are several community of things that are possible with the project, however, we limit ourselves to four.

- **Community of rooms –** This would be a collection of the rooms in the house, with sub-communities defined for different types of rooms such as the living room, bathroom, and study room.

- **Community of plants –** This would be a collection of the different plants that would be identified in the house of the user.

- **Community of doors and windows –** This would be a collection of the doors and windows in the house that are required to be monitored by **BR 14**.

- **Community of mobile devices –** this would be a collection of the mobile devices being used by the user to manage and monitor the state of the devices in the smart house.

4.7. Federation of communities

This sub-section outlines the communication. Between Heterogeneous sets of communities as discussed in Section 4.6. There are several federation of communities possible in the project, but we limit ourselves to the three that are discussed below.

- **Community of rooms communicating with community of mobile devices**
  - The different sensors deployed in the rooms would send values to the mobile devices via the AWS environment as mentioned in **BR 1**.
  - The mobile device would also send signals to the devices such as the smart switches, smart thermostat, irrigation system etch to control the states of the devices via the AWS environment as expected in **BR 9**.

- **Community of doors and windows communicating with the community of mobile devices**
  - The magnetic sensor fitted on the doors and windows would send its status when they are changed to the mobile devices which are described in **BR 14**.
  - These status messages would be sent to the mobile devices either whenever the state changes or as described by the user.

- **Community of plants communicating with community of rooms**
  - The plants would communicate with the room, which would help identify what plants are available in which room which is described in **BR 15**.
  - This information would be necessary to identify which rooms have which plant

4.8.Conversations

This sub-section outlines the different types of conversations that are possible between the sets and sub-sets of things, people, and data as discussed in Section 4.1, 4.2, and 4.4 respectively. A few of the many conversations possible are listed below.

- **Things to Data –** Rooms would be monitored for different data points including temperature, humidity, air pressure, the light intensity which would be collected by different sensors in the room. For example, when the temperature sensor records some value from the room it would generate data that would then be sent to the AWS environment for further processing or storage.

- **Thing to thing –** The mobile device is used to manage the states of the devices in the environment. For example, when the mobile device is used by the user of the system to manage the state of switches in the environment, the user would toggle the switch on the mobile device via the application which would then be sent to the AWS environment for processing. This would then trigger a Lambda function which would then cause the switch in the house of the user to change states.

- **Thing to People –** Notification being sent by the device to the user. For example when the fire alarm is triggered, it would send a MQTT message to the AWS environment which would then trigger a Lambda function to trigger the AWS SNS to send a notification to the user of the system on the mobile device associated with the automation environment.

- **Data to Things –** Trigger generated based on events from the database to control the states of things in the environment. For example, soil moisture reading, triggering the activation of the water sprinklers.

- **Data to People –** Visualization of the past information of the data being stored in the database. The data being collected by the sensors, when sent to the AWS environment would trigger several Lambda functions , one of which would be responsible for the visualization of the data being collected for the user of the system to understand.

- **Data to data –** When data is used to extract data from other databases. for example, when we use the flower name from the classification model to get the water requirements from another database.

- **People to Data –** People manually changing the status of a particular device in the environment. For example when the user is using the mobile device to manage the state of a particular device in the environment, the user is also adding data points for the states of the devices that he is managing.

4.9.Enabling technologies

- **Visualization –** Visualization of data would be an enabling technology in project, as it would give the user a glimpse of all the information that the system is trying to accumulate with all the sensors in the smart home environment. The visualization, mainly can compose a time series display of the readings from the sensors. Open source tools such as influxDB (Time Series Database) and Grafana (Open source Visualization tool)

- **Sensing Technologies –** The sensors as described in **Section 4.5.2** would be a part of the enabling technologies.

5. Architectural questions

5.1. How to identify things?

In this sub-section we describe two ways that things can be identified – RFID and Barcode. The RFID method would be used by professionals setting up the environment for the user, whereas the method of barcode would be used by the user to setup the automation environment themselves as required by **BR 3**.

- *RFID*

  - Using tags for each of the devices we can uniquely identify a thing.

  - Program the tag to set a unique identifier, and attach the tag to the device under consideration.

  - When the need for identifying the thing arises, an RFID reader can be used to read the tags to uniquely identify a particular device, by reading the hexadecimal value from the tag and comparing the hex value, to the values stored in the database of the reader.

- *Barcode*

  - Barcodes are used to help the user identify the tags of the device faster in a more convenient manner

  - The user can scan the barcode using their smartphone or a barcode reader to identify the device and connect the device to AWS through the mobile/web application.

5.2. How to discover things?

Here, we briefly outline the AWS service that would be used for discovery of things. This allows for setup of a scalable environment as needed by **BR 3**.

AWS IoT Management Service would allow bulk registration of things

- Whenever thing is to be discovered in the environment a simple API call can be made to the AWS servers with the following parameters – region name, zip code, street address, apartment number, floor, room, thing type, and ID.

- If any parameter(s) is/are not provided, then the rest of the parameters would be used to discover the thing.
  - For example, if floor, room, thing type, and id are not provided, the API call would return the set of things in a particular apartment.
  - A JSON file for registration of several devices with attributes defined like Wattage, type, model, name, ID, etc. would be returned.
  - This JSON file would also help ensure the **BR 8** to control the deployment speed of the over-the-air updates to the consumer devices already in the field.

5.3.How to connect to things?

- Our project would be using Publish-Subscribe methodology for things to publish values to AWS message broker from the publishing clients, and then the subscribing clients at the AWS end would listen to the events from the things in the smart home environment.

- There would be certain things that would be subscribed to events from the AWS end, such as the Irrigation system or the smart switches that would be listening to the events from the AWS Lambda service (essentially the MQTT messages) that would be to switch them ON/OFF.

- So things in the environment would be connected through the AWS message broker, followed by a layer of lambda function which would be responsible for reacting to the messages sent from one thing in the environment to another thing in the environment.

5.4. How to connect things to people?

Here, we describe two ways of connecting the things to people. One would be the way that the user would be interacting with the system. This would be to check the status of the devices, and/or monitor the states of the devices deployed in the house. The second would be for the administrators to interact with the environment for debugging problems encountered by the users, or serve important security patches, or any other managerial functions that the administrator would have.

- *Via the mobile Application/ Web Application*
  - User interaction with their home environment would happen through a mobile/ web application. The mobile/web application would connect to the user's environment using a generated unique identifier to his/her environment. This would be set up at the time the devices are set up at the home environment.
  - The user can access the readings of the devices such as temperature, humidity, air quality, moisture level etc through the mobile/web application.
  - The user can change the statuses of devices such as smart switches, the brightness of the light, the temperature of the room etc… through the mobile/web application at any place in time.
  - The user will receive notifications about the configured events that were detected from the environment and any changes were made in the environment by our system following a detected event.

- *Administrator interaction*
  - Administrator interaction would be enabled once the AWS command line interface environment has been set up on the laptop/ desktop of the administrator.
  - This would first require the AWS administrator, to create a role specific to the administrator based on the location of the smart-home environment under consideration and then associate the respective policies to his account.
  - The AWS administrator would then have to create a set of Access Key ID and secret access key for the local administrator while giving him CLI access to the smart home environments for a particular location.
  - The local administrator would then use the secret access key and access key ID to set up the AWS environment up on his laptop/ desktop which would then enable him to connect with the things in the smart home environment under consideration.