An IOT-based Smart Street Infrastructure Management System on Cloud

CMPE 281-02: Cloud Technologies

Final Project Report

Submitted To

Dr. Jerry Gao

Date of Submission

11th December 2018

Submitted By

Team Member	Student ID
Amruta Saraf	**011456744**
Kashika Jain	**012505649**
Manisha Shivshette	**012560353**
Shraddha Jamadade	**012462281**
Vishwanath Patil	**012526410**

Table of Contents

Project Demo URL - (Please download and view video)
https://drive.google.com/open?id=1zPSC-IFzsR9f5ErEDmne2JMcVIQ0p5ju

Alternate Link to video:

https://drive.google.com/open?id=1aaSxBD8C7Uben9jSw8_rrniiTjbZByHo

1. Introduction

1.1 Objective

Smart City is not only one of the most promising, prominent and challenging Internet of Things (IoT) applications but also the future of the world we live in. The world is moving more and more towards a smart world and internet of things is the next big thing. In the last few years, the smart city concept has played an important role in academic and industry fields, with the development and deployment of various middleware platforms and IoT-based infrastructures. The vision behind developing this project is developing, implementing, and validating an IOT-based cloud infrastructure system as a SaaS for Smart Streets in a smart city.

In our project for developing a smart street we have collected data from the different sensors which are installed on the utility poles in the streets. As shown in Figure 1 below, a smart street consists of several smart nodes installed on a utility pole on a street. Each node could be equipped with a set of sensors, a camera, humidity sensor, temperature, seismic sensors and so on. A smart cluster controller among smart nodes is used to control the connected smart nodes and support the communications with the back-end server to send the collected sensor data for all nodes. In addition, each smart node has wireless communication capability, which supports node-to-cluster communications. Each cluster node has internet connectivity. In our project, we have used three types of sensors i) Temperature, ii) Humidity and iii) Wind

Fig: Smart Node on Smart Street

1.2 Motivation

The motivation behind developing a smart city project was inspired by the fast growing impact of IOT where the Internet of things is taking over all the devices and is supposed to take over 80% of the devices by the next 2 years. Although the primary reason for the smart city revolution comes from cost and energy saving requirements, there is an increased shift of cities moving from traditional way to incorporating sensors and networked control lights as a means of their infrastructure. These networked street sensors provide the perfect platform leading to innovation and various smart city applications. With a wide range of sensors deployed both for power and communication available on the street poles distributed throughout the city, many additional benefits can be expected for the residents of the city.

1.3 Advantages

The future of the world is mega-cities. By 2030, all the lives will depend on the important choices we make today. The invention of smart cities is going to impact the planning, decision-making, efficiency and sustainability of citizens. Some of the most obvious advantages of moving towards a smarter city are-

- ➔ There will be tremendous benefits in terms of cost and efficiency. City efficiency will be increased at lower costs
- ➔ The real time energy monitoring helps in identifying opportunities and leads to environmental improvement
- ➔ The first and foremost benefit of developing smarter cities is enhancing the quality of life of the residents
- ➔ With the advancement in technology, the business opportunities of the city will improve because of greater connectivity.

2. Related Work

Smart city applications are implemented to galvanize innovations and create an infrastructure that is commercially fulfilling to the residents of the city. There are many startups and companies, specially in the bay area working to make the transition of new ideas and innovations based on smart city into high growing ventures whose objective is

to make the cities much more smarter, safer and a lot more resilient. Currently the innovations in making a green city is being focussed on the following areas-

- ➔ A robust model building exercise to optimize technology along with its business plans, different operations and growth.
- ➔ Customization in developing the city plan based on the region, area or zip code to tailor the specific needs of the city per se.
- ➔ Focussing investments on mature companies and inviting investors by marketing and social events
- ➔ Providing opportunities for Investments in many companies and access to investors during and after the program
- ➔ Providing forums and discussion groups for discussion of possible steps and measures to keep the motion forward
- ➔ Building state-of-the-art business processes including management, marketing and CRM tools
- ➔ Migration with AWS platforms and architecture for deployment and control.
- ➔ Availability of technical support from various networks like universities, laboratories etc.

3. Cloud-Based System Infrastructure and Components

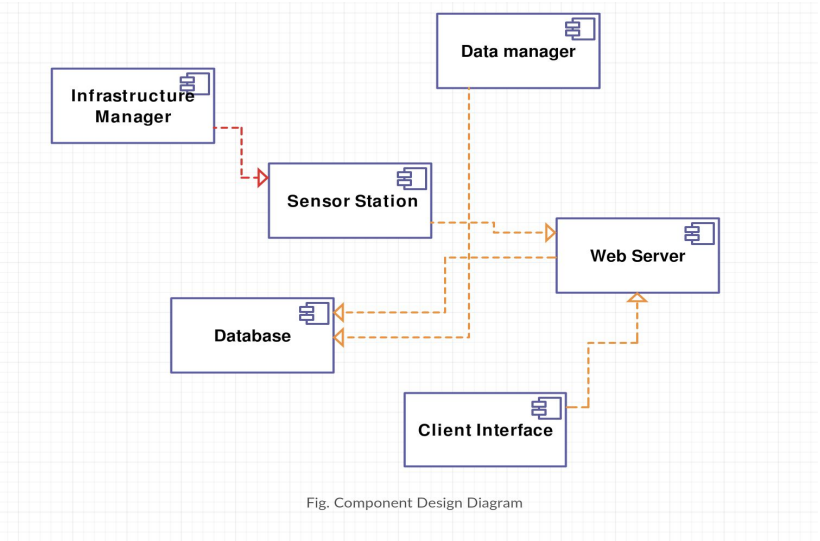
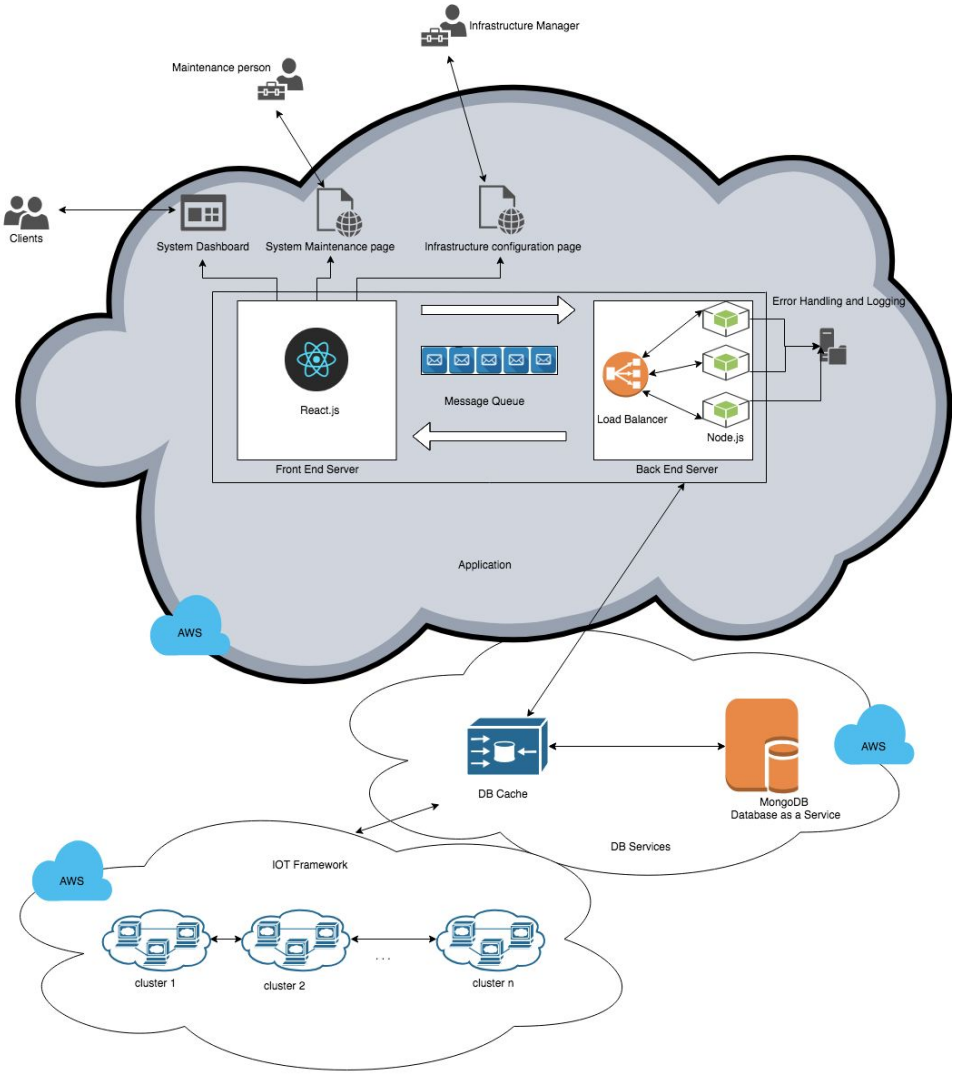


Fig: Component Design of IoT Smart Street

System Architecture



The system Architecture for this application can be roughly divided into five major components-

1) **IOT Infrastructure**

The Internet of Things infrastructure is the system's crucial component. It contains of a number of clusters of smart nodes, smart cluster controller and sensor devices. All these nodes are interconnected and communicate the collected sensor data to the cluster nodes and then to the back end server. various sensors on the smart nodes collect specific type of data like temperature changes, humidity level, particulate matter level, etc.

2) **Visualization and dashboard**

This component is responsible for handling and displaying the various data from the database. Graphical representations are shown to the system users and clients who access the system. AWS Quicksight service is used to help with the easy implementations of these dashboards. Data analytics are provided in the form of graphs and statistics. The sensor data collected can be viewed in the form of different graphs. Clients can view these reports for any particular time stamp or a range of time periods. They can also view a particular sensor data sorting by region, etc.

3) **Back end server**

The back end server communicates with the Front end server, dashboards and the Database on cloud. It contains the API calls and the actual functionality implementations to process these sensor data. It is implemented basically using Node.js

4) **Front end server**

Mainly responsible for parsing data at the UI Layer. It is the interface between the users and the backend services. This component in mainly implemented using React.js

5) **Database**

The database component of the application is mainly used to store the data collected from various sensor devices (e.g. temperature data, camera images, audio sensor sounds, humidity, O2, CO2 level date, etc.).
For this project, the database is hosted on Amazon Cloud with the use of some of its services like Amazon RDS (Relational Database System) and AWS Kinesis. The Amazon Kinesis is used to collect and process large streams of real time data to gain quick insights to be able to react proactively to new information. The Amazon RDS is a typical relational database system use to store and manage data effectively on cloud using the Mysql engine.

3.1 Data Manager System Infrastructure Design

This component is responsible for:

1) Add,delete and view a sensor reading data
2) Retrieve sensor data for a smart node based a given time range
3) Retrieve sensor data for a cluster node based a given time range cloud
4) Display connectivity between all of stations and the back-end server

Major Design Challenges :

1) Collected data per unit time is huge as sensors continuously send data
2) Quick query performance

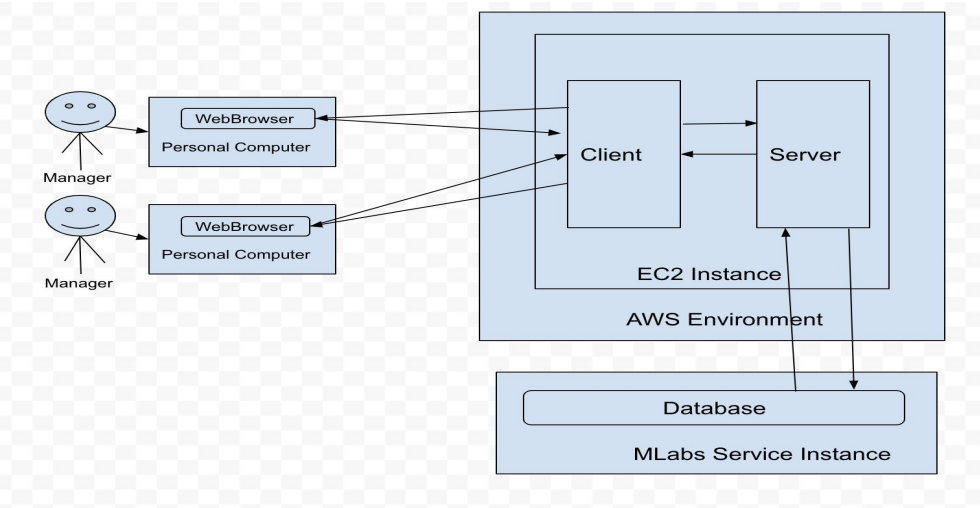


Fig: Data Manager Component Design

3.2 IoT-based Smart Street Infrastructure Management System on Cloud

This component is basically a management program that supports IoT based Smart Street Management Functions.

Component Functions:

1. Cluster Node Management

The IoT Infrastructure Manager can manage cluster nodes by performing the following functions:

- Add a Cluster Node on a Smart Street
- Update a Cluster Node
- Delete a Cluster Node from a Smart Street

- View a Cluster Node
- Track status of the Cluster Node

2. Smart Node Management

The IoT Infrastructure Manager can manage smart nodes which are controlled by cluster nodes. It performs the following functions:

- Add a Smart Node controlled by a cluster
- Update a Smart Node
- Delete a Smart Node controlled by a cluster
- View a Smart Node
- Track status of the Smart Node

3. Sensor Management

The IoT Infrastructure Manager can manage Sensors from a selected Smart Node. It performs the following functions:

- Add a Sensor on a selected Smart Node
- Update a Sensor
- Delete a Sensor from a selected Smart Node
- View a Sensor
- Track Sensor Status

4. Status Report

A status report in general refers to any document, graphical or pictorial representation of how the system is functioning. It contains periodical updates of important elements in the system (here, monitoring the status of sensors, smart nodes and cluster nodes). This can be divided into three functions:

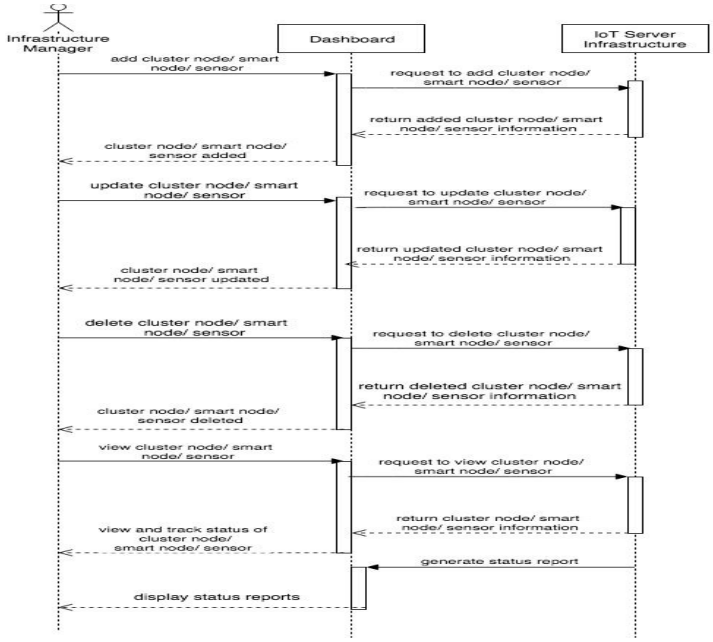
- Keep record of all the tracked statuses of the sensors, smart nodes and cluster nodes
- Generate a status report on an existing IoT smart street infrastructure
- Display the status report

5. Map View

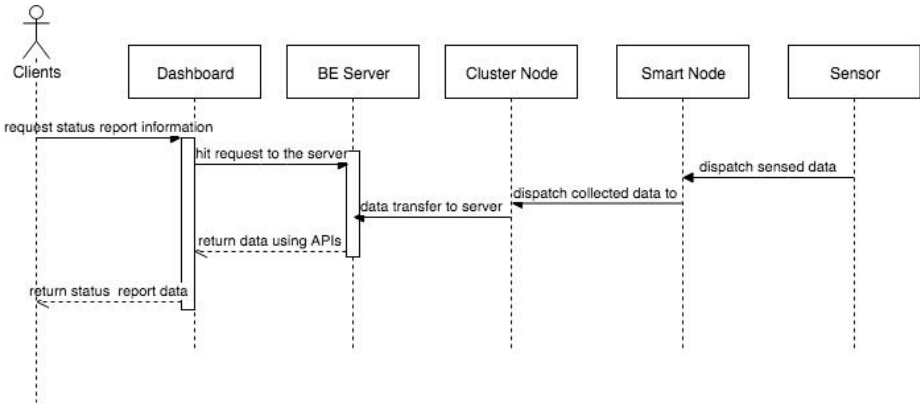
Including a map view in the component will make it very efficient and it will be able to perform all its functions smartly and with ease. Sub-functions:

- Generate a map view about one selected smart street's IoT infrastructure
- Display the map view

The figure below shows how the infrastructure manager module is implemented to work

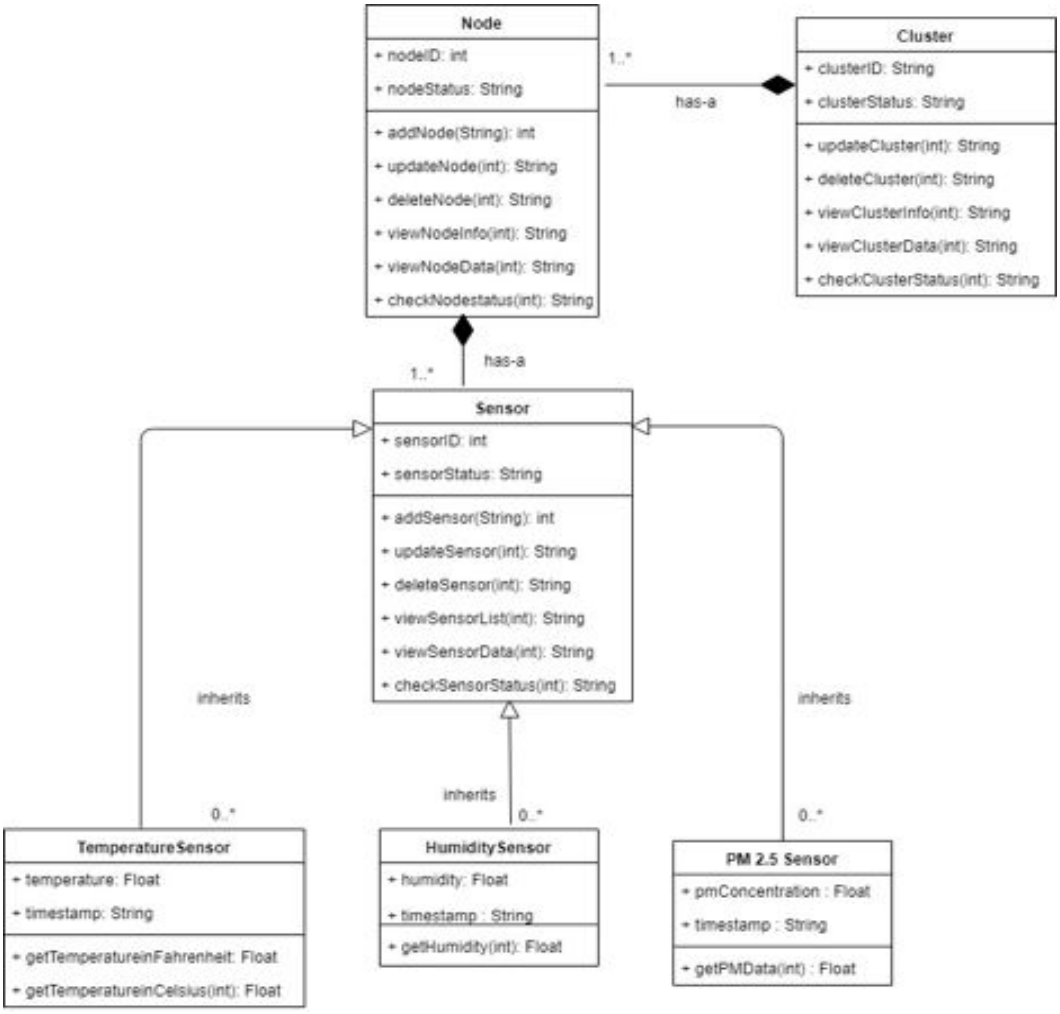


IOT sensor infrastructure data communication and data transfer



3.3 Components for Sensor Station

Classes and their interrelations -



3.4 System Dashboard

The objective of this assignment is to build a Detailed component design for the IOT based Smart Street Infrastructure Management System on a cloud. The idea is to provide the different design views of the project for the Dashboarding component of the project.

3.4.1 System Dashboard Features -

3.4.1.1 Display report – The dashboard will have a feature to display the report where in the client can view all the sensor data, smart node data, cluster data for a given region and locality.

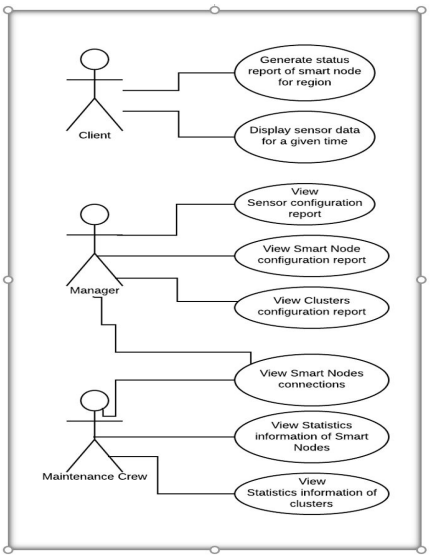
3.4.1.2. Display Map based Sensor ports – This functionality will be present in the dashboard to view all the sensor ports in a particular entered region based on the city and zip code.

3.4.1.3. Configure Server- This feature is responsible for displaying all the configuration information about the smart node for the selected region. We will

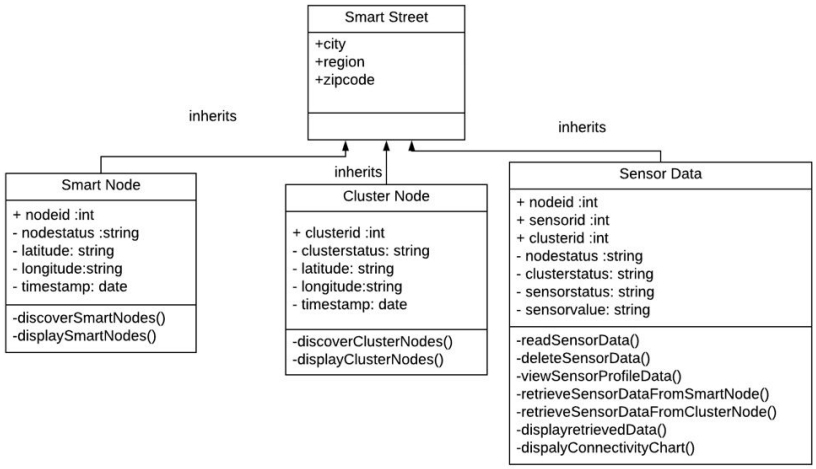
come to know how many nodes were added, deleted, updated and configured in that region.

3.4.1.4. <u>Display node connectivity</u>- This part of the dashboard will focus on displaying the connectivity of the nodes in the selected region. It will show the cluster level data for the nodes.

3.4.2. <u>Class Diagram</u> - The different diagram displayed below shows the roles of the 3 users of the dashboard and what roles will avail for each user.



<u>Fig: Use case diagram</u>



<u>Fig:Class Diagram</u>

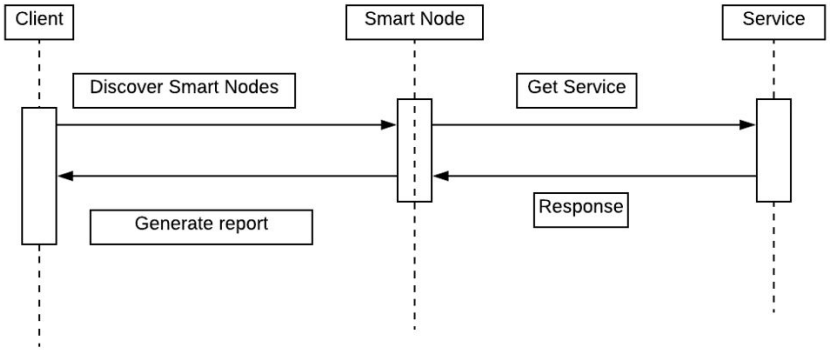


Fig: Sequence Diagram for client

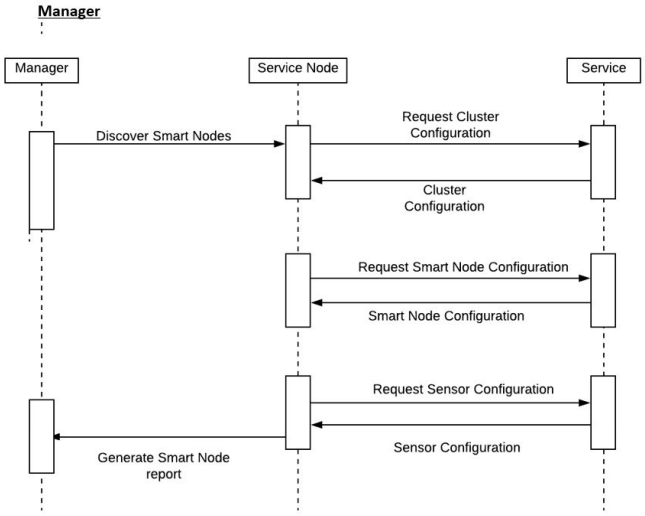


Fig: Sequence Diagram for Manager

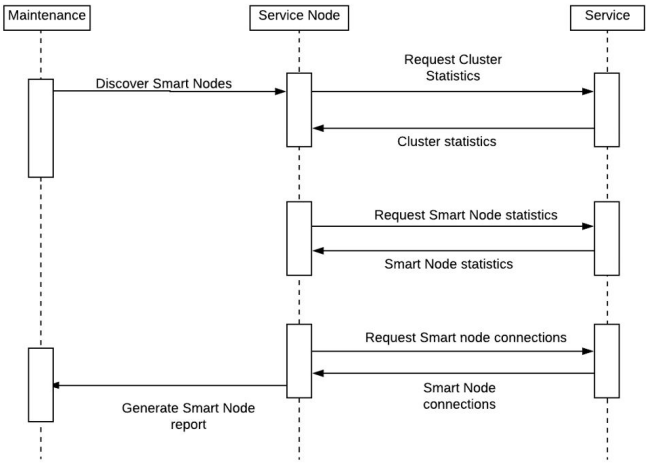


Fig: Sequence Diagram for Maintenance crew

3.4.3 Database Connection:

The Mongodb database is connected to Tableau Dashboard interface using ODBC connection as shown below. The Tableau dashboard is used for showing the views for the 3 use cases.

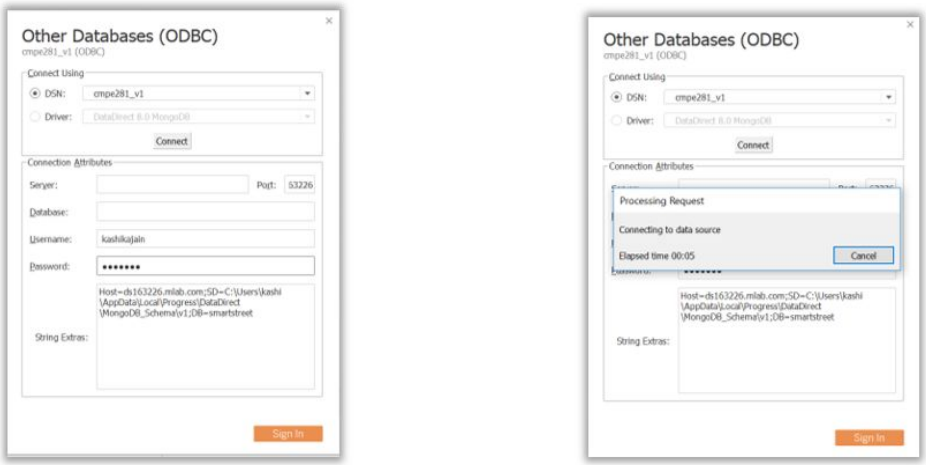
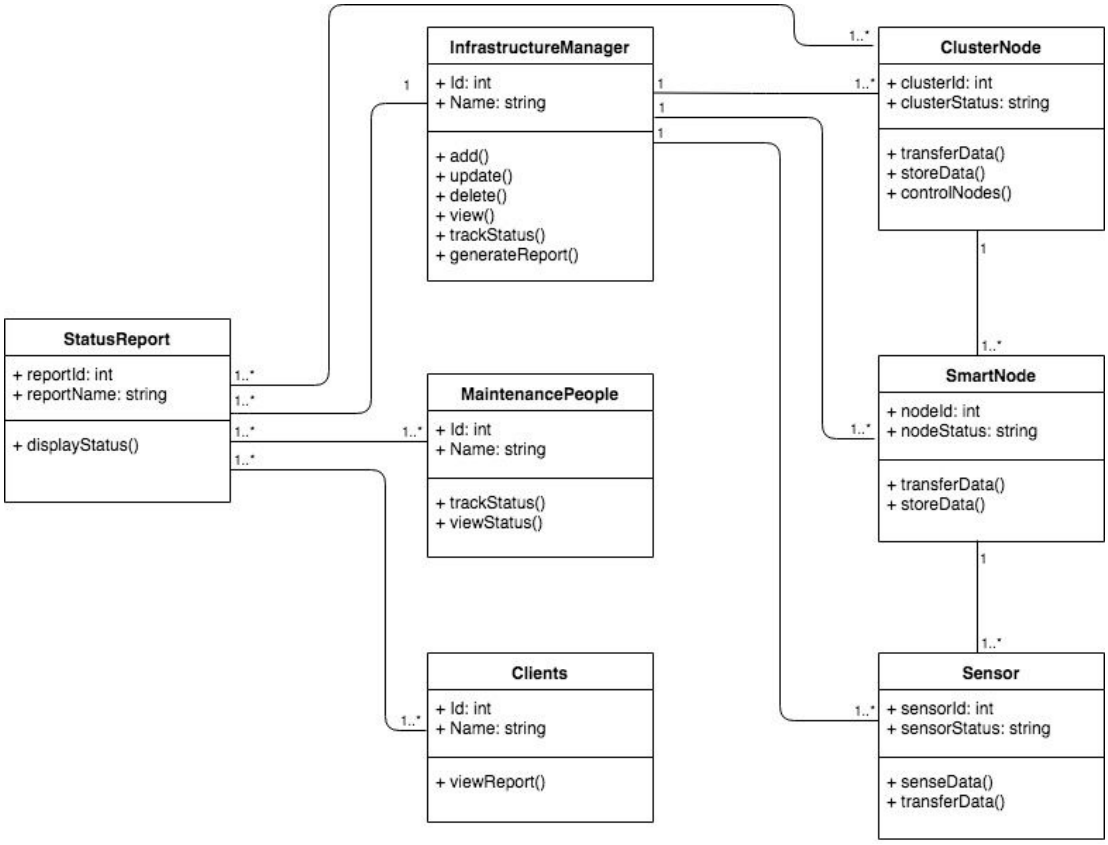


Fig: Database Connection in Tableau

4. Large-Scale IOT Data Design and Implementation



Flg;Overall system Class Diagram

To design CRUD services for large scale IoT data, we have implemented following solutions for high performance:

1.Indexing

When we search by unindexed fields, we have to discover the full path to the value, with no shortcuts.Hence indexing is applied on cluster id and smart node id for faster retrieval of sensor data

2.Limiting Query Scope

Running a query using * (all) can potentially lock up your database while it retrieves points,hence can query scope is limited by using time range while querying data

3.Pagination

Even after using indexing and query scope , large volume of data will get fetched for given query.Hence server side pagination was implemented to limit the fetched data.

4.1 IOT infrastructure DB design and implementation

1. Sensor Data

Column	Data Type
SensorID	INT
SensorStatus	VARCHAR
SensorType	VARCHAR
SensorLalitude	INT
SensorLongitude	INT
NodeID	INT
ClusterID	INT

2. Smart Node Data

Column	Data Type
NodeID	INT
NodeStatus	VARCHAR
NodeLalitude	INT
NodeLongitude	INT
ClusterID	INT

3. Cluster Node Data

Column	Data Type
ClusterID	INT
ClusterStatus	VARCHAR
ClusterLalitude	INT
ClusterLongitude	INT

4. Infrastructure Manager

Database collection for the login page

Column	Data Type
ManagerID	INT
ManagerName	VARCHAR
Email	VARCHAR
Username	VARCHAR
Password	VARCHAR

The below diagram shows the database collections are structured. The field names are not exactly like in the tables but are elaborated for understanding the structure easily.

Cluster Node

Cluster Node Id <int>	Cluster Node Status <string>	Latitude <float>	Longitude <float>	No. of Smart Nodes communicating with the cluster <int>

Smart Node

Smart Node Id <int>	Smart Node Status <string>	Latitude <float>	Longitude <float>	No. of sensors communicating with the smart node <int>	Within the cluster (cluster Id) <int>

Sensor

Sensor Id <int>	Sensor Status <string>	Sensor Type <string>	Latitude <float>	Longitude <float>	Communicating with the smart node (Node Id) <int>	Within the cluster (cluster Id) <int>

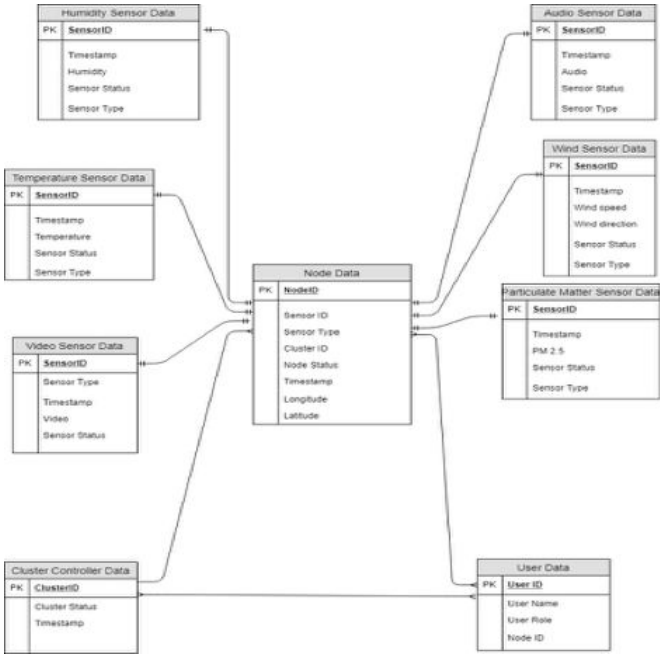
4.2 IOT sensor DB design and implementation

Data generated by the sensor simulator is transferred to the backend at periodic intervals and stored in mLab, a cloud hosted database-as-a-service to be used by other modules.

Sample document in IOT sensor collection in MongoDB -

```
{
    "_id": {
        "$oid": "5c1084044f8f1f0e48256c6c"
    },
    "sensorNodeId": "1",
    "sensorName": "Humidity sensor",
    "sensorType": "Humidity",
    "sensorValue": "76",
    "timestamp1": "12/12/2018 3:42:31",
    "timestamp2": "2018-12-12T03:44:04.940Z",
    "latitude": "37.3382",
    "longitude": "-121.8864",
    "smartNodeId": "1111",
    "clusterId": "1111",
    "__v": 0
}
```


Relationship between Entities:



5. IOT-based cloud system design and services

5.1 API Services

5.1.1 Data Manager services APIs

Following are API specification for to get and edit sensor data:
1.Get Sensor Profile Data

Title	Get Sensor Node
URL	/sensorid/
Method	GET
URL Params	sensorid
Data Params	

Success Response	{
	“Cluster ID”:<cluster id >
	“Smart Node ID”:<node id >
	” Sensor ID”: “<sensor id >”
	” Sensor type”: “<sensor type>”
	” Sensor name”: “<sensor name>”
	” Sensorvalue”: “<sensor value>”
	“status”: “<status>”
	“ Latitude”:”<Latitude>”
	“Longitude”: “<Longitude>”
	}

Error Response	{
	Statuscode: 400,
	Message: "Sensor Profile not found",
	}

2.Get sensor data for given smart node data within given date range

Title	Get Sensor data for Smart Node
URL	/smartnode/

Method	GET
URL Params	Smartnode id

Data Params	{
	“Start date” : <start date>
	“End date” : <end date>
	
	}

Success Response	{
	“Cluster ID”:<cluster id >
	“Smart Node ID”:<node id >
	” Sensor ID”: “<sensor id >”
	” Sensor type”: “<sensor type>”
	” Sensor name”: “<sensor name>”
	” Sensorvalue”: “<sensor value>”
	“timestamp”:”<timestamp>”
	
	}

Error Response	{
	Statuscode: 400,
	Message: "No data found given Smart Node",
	}

3.Get sensor data for given cluster node data within given date range

Title	Get Sensor data for Cluster Node
URL	/clusternode/
Method	GET
URL Params	clusternode id

Data Params	{
	“Start date” : <start date>
	“End date” : <end date>
	
	}

Success Response	{
	“Status Code” : 200
	“Cluster ID”:<cluster id >
	“Smart Node ID”:<node id >
	” Sensor ID”: “<sensor id >”
	“status”: “<status>”
	“ Latitude”:”<Latitude>”
	“Longitude”: “<Longitude>”
	}

Error Response	{
	Statuscode: 400,
	Message: "Cluster Node Not found",
	}

3.Get all smart nodes and cluster nodes connected to server

Title	Get Connected Nodes to server
URL	/connectednodes/
Method	GET
URL Params	
Data Params	

Success Response	{
	“Status Code” : 200
	“Cluster ID”:<cluster id list>
	“Smart Node ID”:<node id list >
	“status”: “<status>”
	“ Latitude”:”<Latitude>”
	“Longitude”: “<Longitude>”
	}

Error Response	{
	Statuscode: 400,
	Message: " Not found",
	}

5.1.2 IoT Infrastructure Manager services APIs

1. Add cluster Node

API specification for addition of cluster node
addClusterNode()

Title	addClusterNode
URL	/add/clusterNode
Method	POST
URL Params	Null

Data Params	{
	clusterId: [integer],
	clusterStatus: [string],
	clusterLatitude: [integer],
	clusterLongitude: [integer]
	}

Success Response	{
	Status Code: 201,
	Message: "Cluster Node added successfully"
	}

Error Response	{
	Status Code: 400,
	Message: "Error in adding the Cluster Node"
	}

2. Update cluster Node

API specification for updating cluster node

updateClusterNode()

Title	updateClusterNode
URL	/update/clusterNode
Method	PUT
URL Params	Null

Data Params	{
	clusterId: [integer],
	clusterStatus: [string],
	clusterLatitude: [integer],
	clusterLongitude: [integer]
	}

Success Response	{
	Status Code: 200,
	Message: "Cluster Node updated successfully"
	}

Error Response	{
	Status Code: 405,
	Message: "Error in updating the Cluster Node"
	}

3. Delete cluster Node

API specification for deletion of cluster node
deleteClusterNode()

Title	deleteClusterNode
URL	/delete/clusterNode
Method	DELETE
URL Params	Null

Data Params	{
	clusterId: [integer],
	clusterLatitude: [integer],
	clusterLongitude: [integer]
	}

Success Response	{
	Status Code: 200,
	Message: "Cluster Node deleted successfully"
	}

Error Response	{
	Status Code: 405,
	Message: "Error in deleting the Cluster Node"
	}

4. View Cluster Node

API specification for displaying cluster node details
viewClusterNode()

Title	viewClusterNode

URL	/get/clusterNode
Method	GET
URL Params	Null

Data Params	{
	clusterId: [integer],
	clusterLatitude: [integer],
	clusterLongitude: [integer]
	}

Success Response	{
	Status Code: 200,
	Message: "Cluster Node details displayed successfully"
	}

Error Response	{
	Status Code: 404,
	Message: "Error in getting the Cluster Node details"
	}

5. Add Smart Node

API specification for addition of smart node
addSmartNode()

Title	addSmartNode
URL	/add/smartNode
Method	POST
URL Params	Null

Data Params	{
	nodeId: [integer],
	nodeStatus: [string],
	clusterId: [integer],
	nodeLatitude: [integer],
	nodeLongitude: [integer]
	}

Success Response	{
	Status Code: 201,
	Message: "Smart Node added successfully"
	}

Error Response	{
	Status Code: 404,
	Message: "Error in adding the Smart Node"
	}

6. Update Smart Node

API specification for updating smart node
updateSmartNode()

Title	updateClusterNode
URL	/update/smartNode
Method	PUT
URL Params	Null

Data Params	{
	nodeId: [integer],
	nodeStatus: [string],
	clusterId: [integer],
	nodeLatitude: [integer],
	nodeLongitude: [integer]
	}

Success Response	{
	Status Code: 200,
	Message: "Smart Node updated successfully"
	}

Error Response	{
	Status Code: 405,
	Message: "Error in updating the Smart Node"
	}

7. Delete Smart Node

API specification for deletion of smart node
deleteSmartNode()

Title	deleteSmartNode
URL	/delete/smartNode
Method	DELETE

URL Params	Null

Data Params	{
	nodeId: [integer],
	nodeLatitude: [integer],
	nodeLongitude: [integer]
	}

Success Response	{
	Status Code: 200,
	Message: "Smart Node deleted successfully"
	}

Error Response	{
	Status Code: 404,
	Message: "Error in deleting the Smart Node"
	}

8. View Smart Node

API specification for displaying smart node details
viewSmartNode()

Title	viewSmartNode
URL	/get/smartNode
Method	GET
URL Params	Null

Data Params	{
	nodeId: [integer],
	nodeLatitude: [integer],
	nodeLongitude: [integer]
	}

Success Response	{
	Status Code: 200,
	Message: "Smart Node details displayed successfully"
	}

Error Response	{
	Status Code: 404,
	Message: "Error in getting Smart Node details"
	}

9. Add Sensor

API specification for addition of sensors
addSensor()

Title	addSensor
URL	/add/Sensor
Method	POST
URL Params	Null

Data Params	{
	sensorId: [integer],
	sensorStatus: [string],
	sensorType: [string],
	nodeId: [integer],
	clusterId: [integer],
	sensorLatitude: [integer],
	sensorLongitude: [integer]
	}

Success Response	{
	Status Code: 201,
	Message: "Sensor added successfully"
	}

Error Response	{
	Status Code: 404,
	Message: "Error in adding the Sensor"
	}

10. Update Sensor

API specification for updating sensor information
updateSensor()

Title	updateSensor
URL	/update/Sensor
Method	PUT
URL Params	Null

Data Params	{
	sensorId: [integer],
	sensorStatus: [string],
	sensorType: [string],
	nodeId: [integer],
	clusterId: [integer],
	sensorLatitude: [integer],
	sensorLongitude: [integer]
	}

Success Response	{
	Status Code: 200,
	Message: "Sensor updated successfully"
	}

Error Response	{
	Status Code: 404,
	Message: "Error in updating the Sensor"
	}

11. Delete Sensor

API specification for deleting the sensor
deleteSensor()

Title	deleteSensor
URL	/delete/Sensor
Method	DELETE
URL Params	Null

Data Params	{
	sensorId: [integer],
	sensorLatitude: [integer],
	sensorLongitude: [integer]
	}

Success Response	{
	Status Code: 200,
	Message: "Sensor deleted successfully"
	}

Error Response	{
	Status Code: 404,
	Message: "Error in deleting the Sensor"
	}

12. View Sensor

API specification for displaying sensor information
viewSensor()

Title	viewSensor
URL	/get/Sensor
Method	GET
URL Params	Null

Data Params	{
	sensorId: [integer],
	sensorLatitude: [integer],
	sensorLongitude: [integer]
	}

Success Response	{
	Status Code: 200,
	Message: "Sensor details displayed successfully"
	}

Error Response	{
	Status Code: 404,
	Message: "Error in getting the Sensor details"
	}

13. Generate Status Report

API specification for generation of status report
generateStatusReport()

Title	generateStatusReport
URL	get/statusReport
Method	GET
URL Params	Null

Data Params	{
	clusterId: [integer],
	nodeId: [integer],
	sensorId: [integer],
	sensorType: [string],
	clusterStatus: [string],
	nodeStatus: [string],
	sensorStatus: [string],
	clusterLatitude: [integer],
	clusterLongitude: [integer],
	nodeLatitude: [integer],
	nodeLongitude: [integer],
	sensorLatitude: [integer],
	sensorLongitude: [integer],
	}

Success Response	{
	Status Code: 200,
	Message: "Successfully generated status reports."
	}

Error Response	{
	Status Code: 404,
	Message: "Error in generating status reports"
	}

14. Generate Map

API specification for getting location values (latitude and longitude) of all the sensing and transmitting resources
getNodeLocation()

Title	getNodeLocation
URL	/get/sensorMapLocations
Method	GET
URL Params	Null

Data Params	{
	clusterId: [integer],
	nodeId: [integer],
	sensorId: [integer]
	}

Success Response	{
	Status Code: 200,
	Message: "Map view generated successfully"
	}

Error Response	{
	Status Code: 404,
	Message: "Error in getting locations"
	}

5.2.3 Sensor Simulator APIs

Title	Create a new Node
URL	/createnode
Method	POST
URL Params	null

Data Params	{
	SensorID : [int],
	sensortype: [String],
	timestamp:[String],
	noofsensors:[String],
	freqofpolling:[String]
	}

Success Response	{
	statusCode: 201,
	message: "Node Created Successfully",
	Result: {}
	}

Error Response	{
	statusCode: 400,
	message: "Error while creating node",
	result: {}
	}

· View Node

Following is sample API specification for viewing smart node:

Title	View Node Information
URL	/getnode
Method	GET
URL Params	Null

Data Params	{
	NodeID : [String]
	}

Success Response	{
	statusCode: 201,
	message: "Node data retrieved successfully",
	Result: {}
	}

Error Response	{
	statusCode: 400,
	message: "Error in retrieving node information",
	result: {}
	}

· Update Node

Following is sample API specification for updating smart node:

Title	Update Node Configurations
URL	/updatenode
Method	POST
URL Params	null
Data Params	{ SensorID : [int], sensortype: [String], timestamp:[String], noofsensors:[String], freqofpolling:[String]

	}

Success Response	{
	statusCode: 201,
	message: "Node Updated Successfully",
	Result: {}
	}

Error Response	{
	statusCode: 400,
	message: "Error while updating node",
	result: {}
	}

· Delete Node

Following is sample API specification for deleting smart node:

Title	Delete a Node
URL	/deletenode
Method	DELETE
URL Params	Null

Data Params	{
	nodeID : [int]
	}

Success Response	{
	statusCode: 201,
	message: "Node deleted successfully",
	Result: {}
	}

Error Response	{
	statusCode: 400,
	message: "Error while deleting node",
	result: {}
	}

· Fetch Node Data from Server

Following is sample API specification for fetching node data:

Title	Retrieve data collected by sensors from the Node
URL	/getnodedata
Method	POST
URL Params	Null

Data Params	{
	nodeID: [string],
	timeperiod : [string]
	}

Success Response	{
	statusCode: 201,
	message: "Data retrieved Successfully for Node",
	Result: {
	SensorID: [101, 102]
	}
	}

Error Response	{
	statusCode: 400,
	message: "Error while retrieving node data",
	result: {}
	}

· Fetch Cluster Data from Server

Title	Retrieve data collected by nodes in the cluster
URL	/getclusterdata
Method	POST
URL Params	Null

Data Params	{
	clusterID: [string],
	timeperiod : [string]
	}

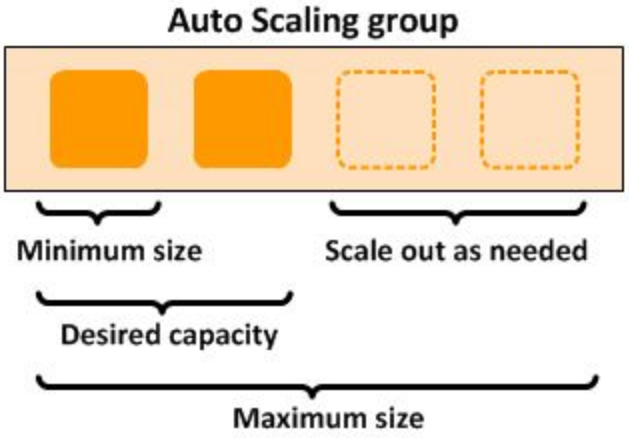
Success Response	{
	statusCode: 201,
	message: "Data retrieved Successfully for Node",
	Result: {}
	}

Error Response	{
	statusCode: 400,
	message: "Error while retrieving node data",
	result: {}
	}

5.2 System Scalability Design and Implementation

Auto Scaling

- Auto Scaling helps in ensuring that we have the correct number of Amazon EC2 instances available to handle the load for the running smart street application.
- Collections of auto scaling groups can be made by selecting the the different EC2 instances.
- Minimum size of the scaling groups ensures that the number of instances never goes below a specified number.
- In short Auto scaling helps in creating new instances for the load balancer as soon as there is an increase in the incoming data from the smart street application.
- We have also deployed load balancing for the different sensor data generated from the simulated sensor station, since the amount of data generated can become too large at times.
- Thus in this way we have been able to manage the load balancing at both the ends.(i.e for the application and the backend)



5.3. System Load Balance Design and Implementation

The project uses AWS services for the purpose of load balancing and auto-scaling and is mainly targeted for two types of users as mentioned below –

1. System Admins
2. System Users

Use Cases for load balancing:
1. Applications are provided with better fault tolerance.
2. Containerized applications are automatically load balanced.
3. Scales applications automatically.
4. Hybrid load balancing with Elastic Load Balancing Component Design for load balancer.

Load balancing policy used :

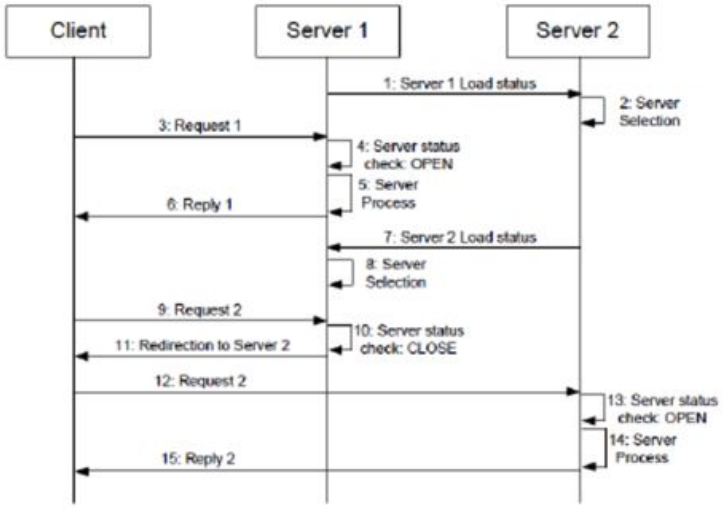
Round Robin
For this project we have used the round robin load balancing policy which is available by default in the AWS services. The working of this algorithm is pretty easy different servers in different regions are configured to provide the dedicated service. Every server uses the same internet domain name, every server will have a different IP address. A DNS server will keep a record of all the unique IP addresses associated with the internet domain name. When requests are received with the IP address associated, the addresses are returned in a sequential manner.

Below is the code for round robin load balancing.

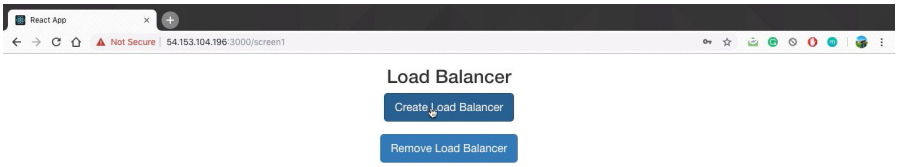
```
private static int[] servers = new int[]{0,1,2};


private static int SERVER_INDEX = 0;

//not fast enough?
public synchronized static int getServer() {
  SERVER_INDEX++;
  if (SERVER_INDEX >= servers.length - 1) {
    SERVER_INDEX = 0;
  }
  return servers[SERVER_INDEX];
}


private static AtomicInteger SERVER_INDEX_2 = new AtomicInteger(0);

//not thread-safe and will get Exception
public static int getServer2() {
  int index = SERVER_INDEX_2.getAndIncrement();
  if (index >= servers.length - 1) {
    SERVER_INDEX_2.set(0);
  }
  return servers[index]; //ERROR! arrayIndexOutOfRange
}



private static AtomicLong SERVER_INDEX_3 = new AtomicLong(0);

//thread-safe but...
public static int getServer3() {
  long longIndex = SERVER_INDEX_3.getAndIncrement();
  long index = longIndex % servers.length;
  int intIndex = (int)index;
  return servers[intIndex];
}
```

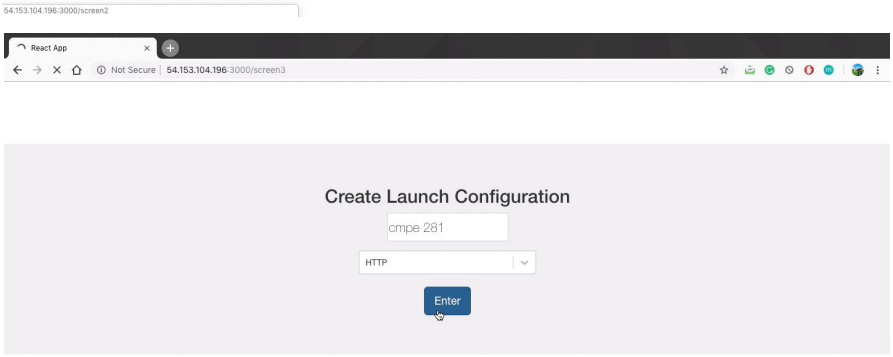
5.3.1 Class Diagrams and Sequence diagrams for Load Balancers

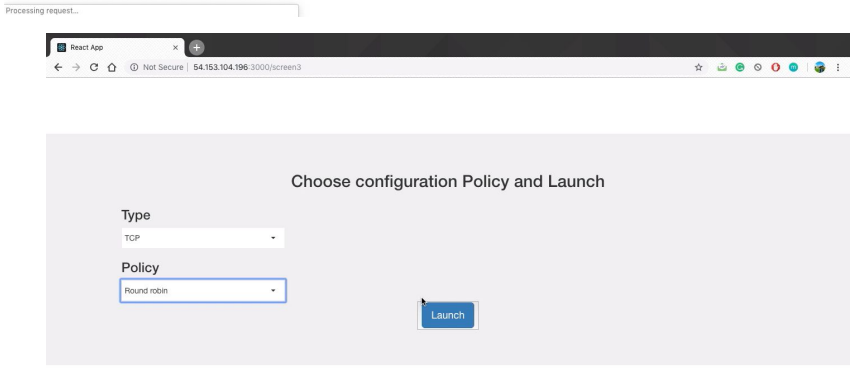
1. Figure below represents the sequence diagram between a server and a client using load balancing.



Below is the GUI which redirects to AWS console for creating a load balancer.







- For this project we have used the classic load balancer for the purpose of balancing incoming traffic.
- A Classic Load Balancer makes routing decisions at either the transport layer (TCP/SSL) or the application layer (HTTP/HTTPS).
- Basically a classic load balancer spins up new instances with the help of auto scalers in either the same zone or different zones as soon as the number of incoming requests increase.
- If the incoming traffic is low and does not require many instances then the load balancer and auto scaler automatically brings down the instances as per the requirement.
- Below is a diagrammatic representation of how a load balancer works

5.4 Application Deployment on AWS

Deploying the Smart street application on cloud involves the following the steps -

(a) Setting up the ec2 instances.

1. Launch the instance from the ec2 dashboard.
2. Choose the AMI as per the requirement. For the project we have chosen a windows AMI.
3. After this step choose the instance type. we have selected the t.micro instance since this is the free type available.
4. Now add the security groups. we have added ssh, https, http for the project instances.
5. Finally launch the instance by selecting the key pair already existing or by making a new key pair.

(b) Deploying the smart street application on the instances created.

1. First do a yum update -y. This command will help in updating all the packages in the instances by pulling up the latest updates from the repository.
2. Now execute the below command to pull the latest packages of node.

   curl --silent --location https://rpm.nodesource.com/setup_6.x | bash
3. Now install the node packages downloaded using the command

   yum -y install nodejs.
4. Now install pm2 in order to restart node app using the command

   npm i -g pm2@2.4.3
5. Get the source code from the GitHub repository using the curl command.

6. Now locate the index.js file in the pulled repository and follow the

   below commands in the path.

   sudo chmod 755 ./index.js

   pm2 start ./index.js -i 0 --name “node-app”

   sudo yum install git -y

7. Finally restart the node application and the pm2 using the below

   commands.

   crontab -l | { cat; echo "@reboot pm2 start index.js -i 0 --name \"node-app\""; } | crontab

   -sudo reboot

(c) Load balancing and auto-scaling on the instances deployed.

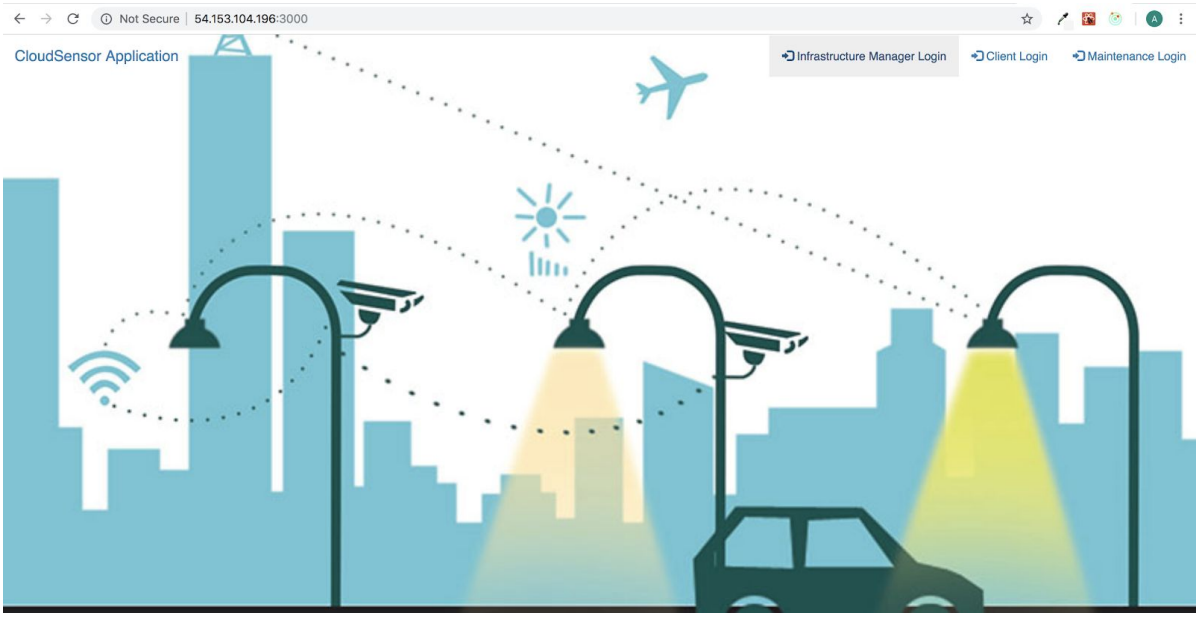
Load balancing :

1. Firstly select a load balancer type from the different available load balancers. we have chosen application load balancer for the purpose.
2. Define your load balancer by selecting the different security groups. we have opened up all the ports selecting all traffic configuration.
3. Select the default VPC and the subnets available.
4. Configure health checks on the ec2 instances.
5. Select all the instances on which the smart street application has been deployed which are to be connected to the load balancer.
6. Create and start your load balancer.

Auto-Scaling :

1. Create auto scaling groups using the launch configurations by specifying the AMI name , instance type and other details.
2. Specify the name of the auto scaling group and the minimum size of the group.
3. Now add the load balancer to the auto scaling groups.
4. Finally configure the scaling policies to the the groups(i.e between how many instances the groups need to be scaled)
5. Once you are done with this, we can see new instances being created as per the requirement of the application.

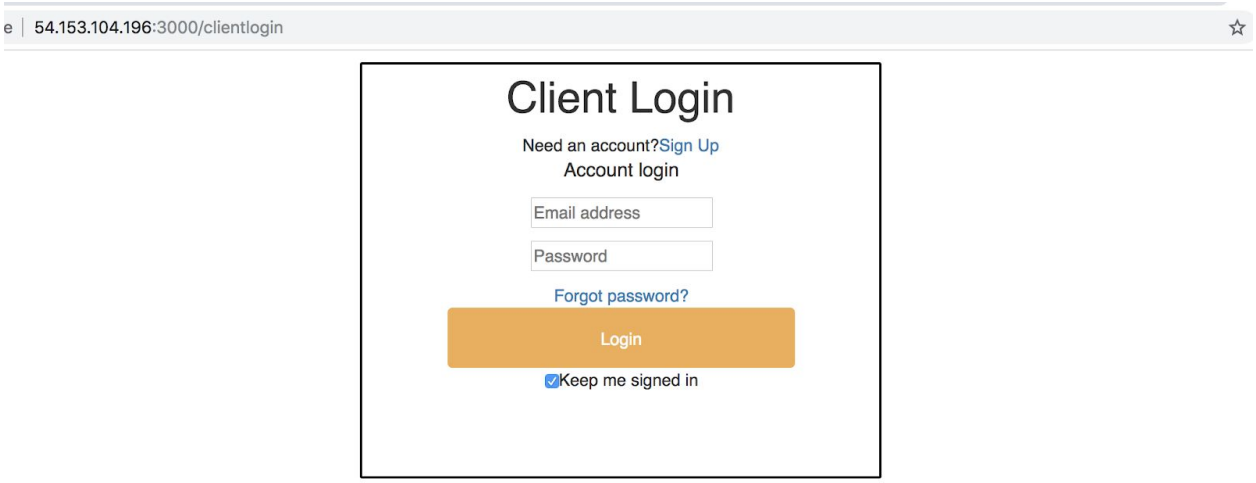
6. System GUI Design and Implementation



The system has the following 3 types of users and there is a separate login for each type of person-

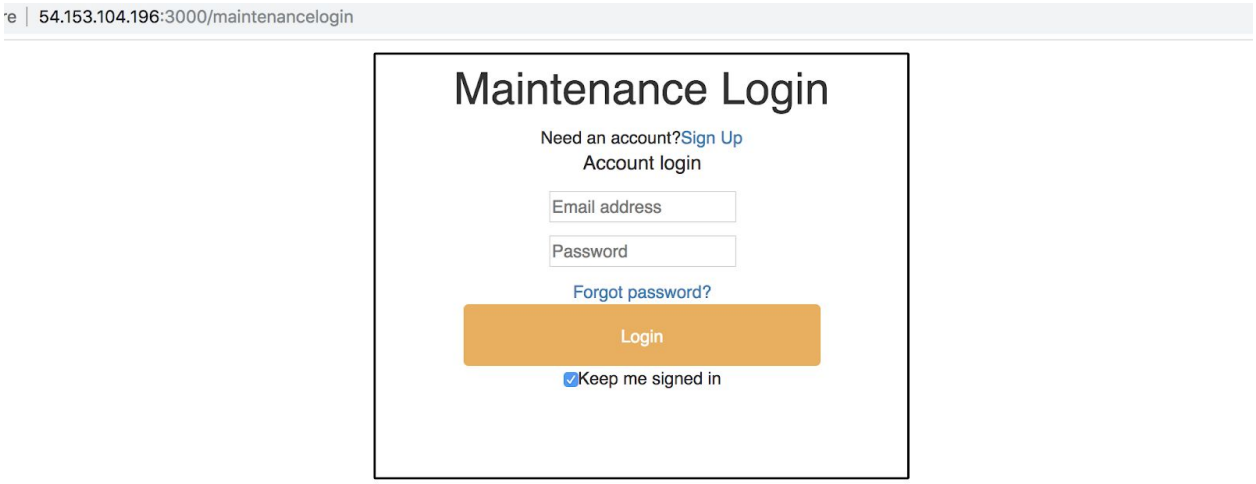
Clients -

People who access the system to collect sensor data and view different reports.

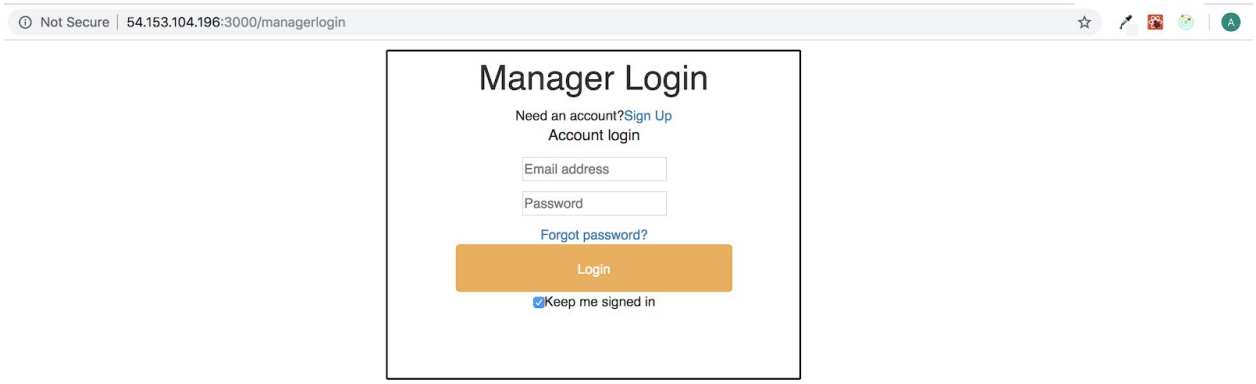


Maintenance people -

People who access the system to find out the status of smart nodes and sensors.

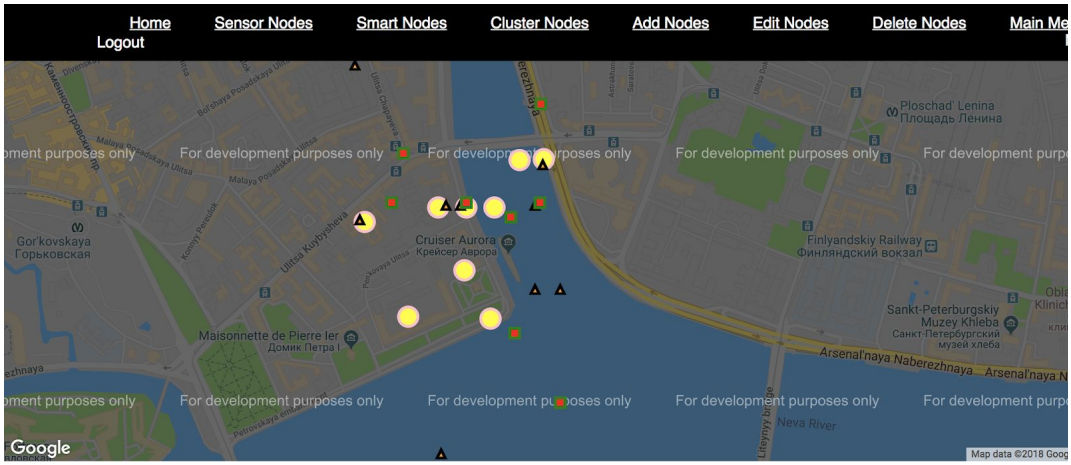


Infrastructure manager -
People who set up, configure, and manage smart nodes, smart cluster nodes, and smart sensor nodes and their inter-connections.



6.1 Infrastructure Manager

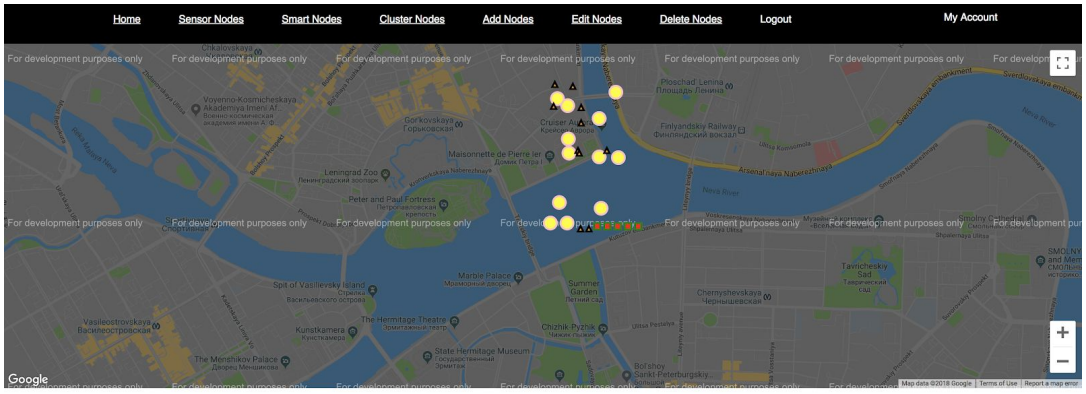
On successful login from the main page, the infrastructure manager user gains access to the infrastructure manager main page where he can view different nodes currently installed in a map view.
The infrastructure manager can see a map view of the streets where the sensors are installed or are to be installed. The user will be able to search or zoom the map.



Legend

Cluster node :
Smart node:
Sensors:

Infrastructure manager main page

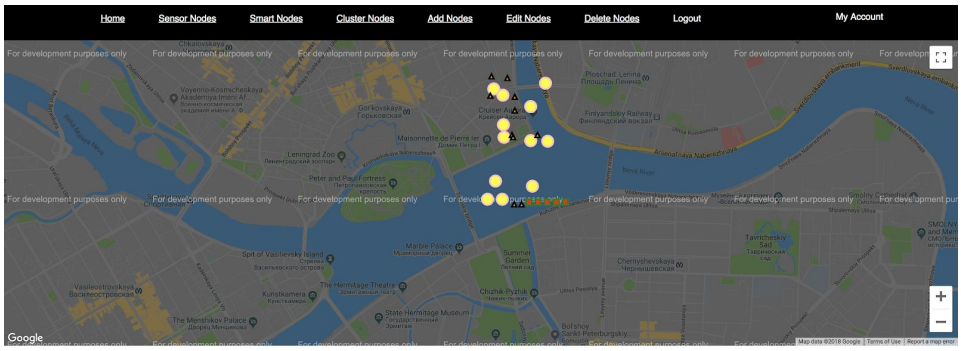


Legend

Cluster node :
Smart node:
Sensors:

Cluster Node

Property	Value
Cluster Id	4
Status	Broken
Latitude	59.951367
Longitude	30.337890
Number of smart nodes communicating	3

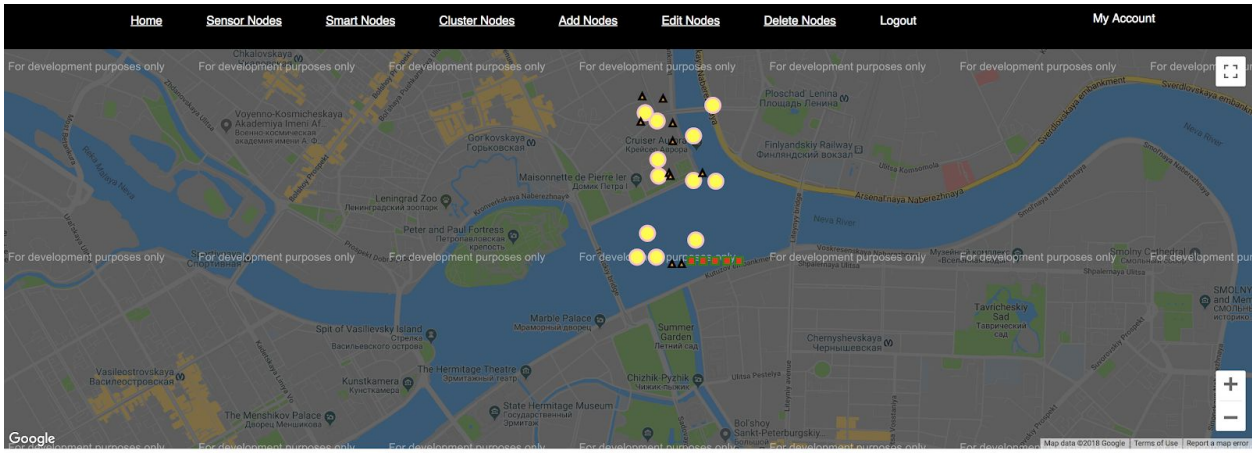


Legend

Cluster node :
Smart node:
Sensors:

Smart Node

Property	Value
Smart Node Id	31
Status	Paused
Latitude	59.956789
Longitude	30.330985
Number of sensors communicating	4
Within the cluster	2



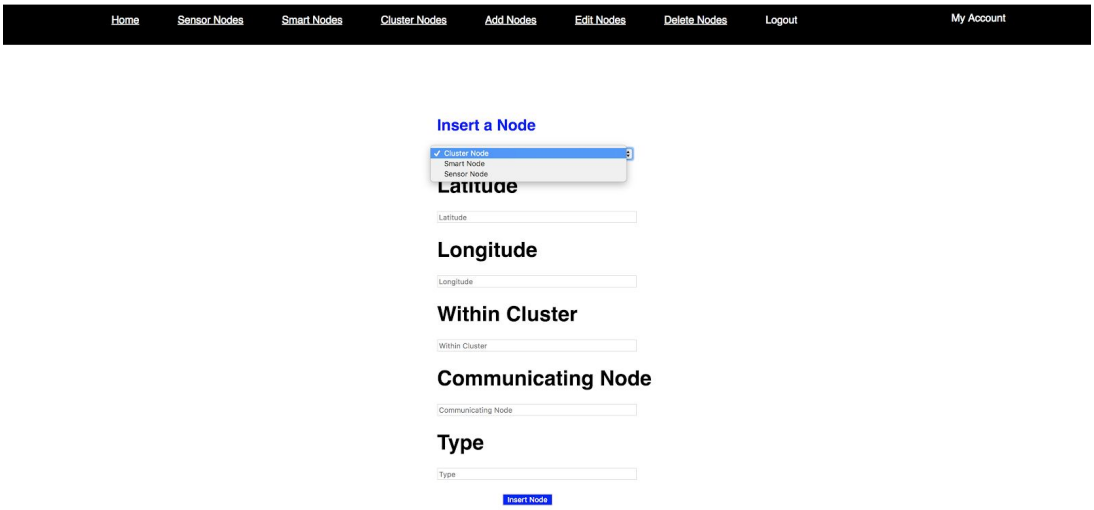
Legend

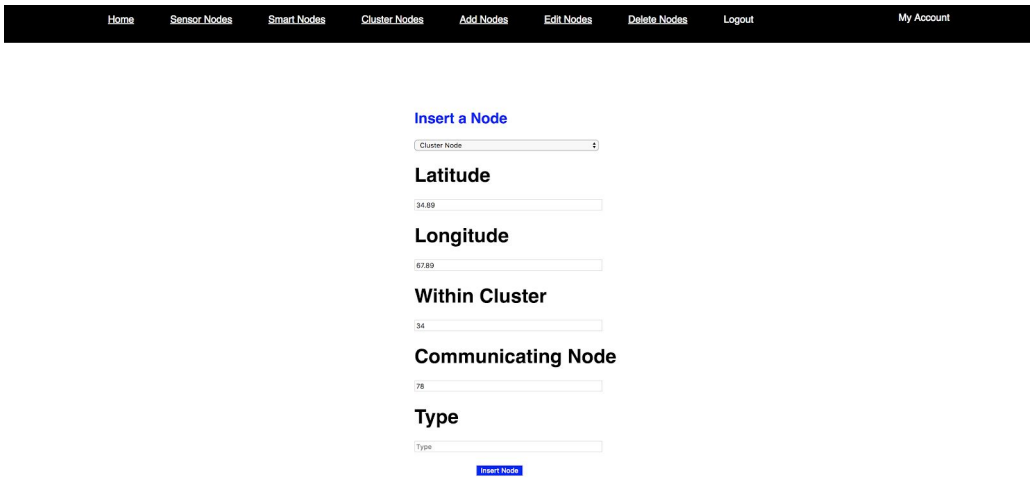
Cluster node :
Smart node:
Sensors:

Sensor Node

Property	Value
Sensor Id	90
Status	Active
Sensor Type	temperature
Latitude	59.956789
Longitude	30.332333
Communicating with the Smart Node	39
Within the Cluster	6

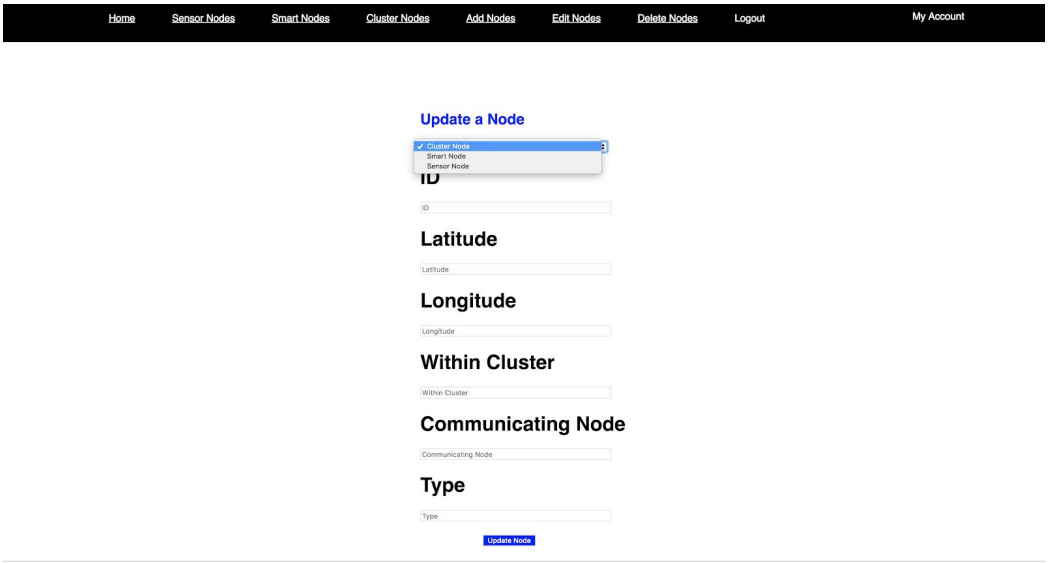
Insert - Cluster Node/ Smart Node/ Sensor
When the infrastructure manager clicks the Add Nodes link, an option to insert new nodes on the map appears.
There is a dropdown to select which resource is to be inserted amongst cluster node, smart node and sensor.
The infrastructure manager should input required details for the selected node and click on Insert Node button.





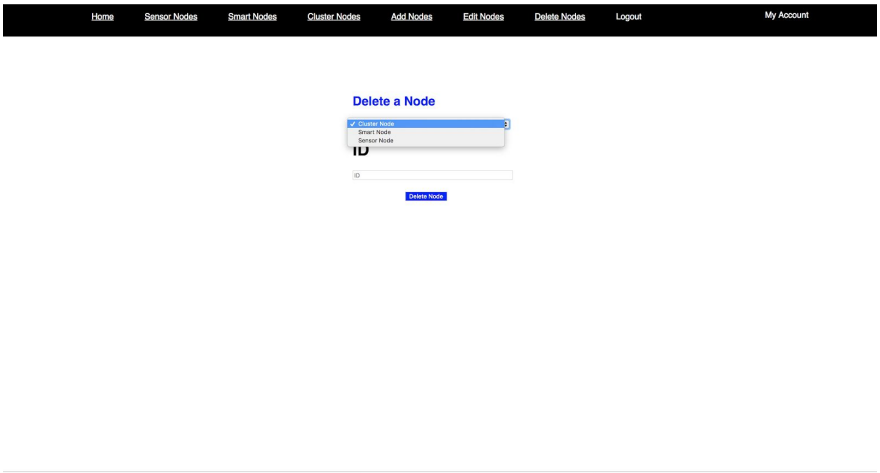
Update Node details - Cluster Node/ Smart Node/ Sensor

When the infrastructure manager clicks the Update Nodes link, an option to update the existing nodes on the map appears.
There is a dropdown to select which resource is to be updated amongst cluster node, smart node and sensor.
The infrastructure manager should input required changes for the selected node and click on Update Node button.



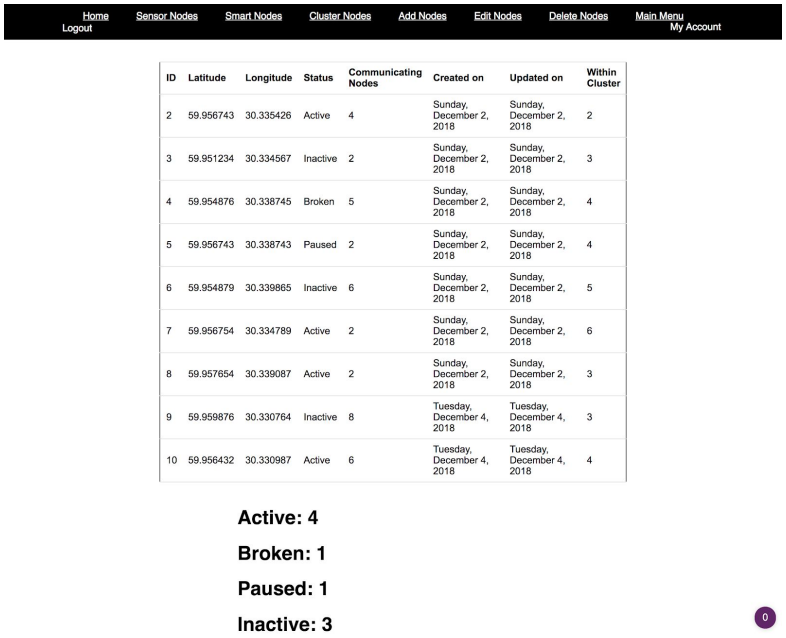
Delete Node - Cluster Node/ Smart Node/ Sensor

When the infrastructure manager clicks the Delete Nodes link, an option to delete the existing nodes on the map appears.
There is a dropdown to select which resource is to be deleted amongst cluster node, smart node and sensor.
The infrastructure manager should input required Id for the selected node and click on Delete Node button.



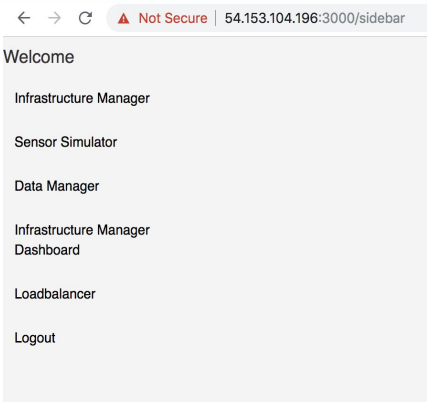
Generation of Status Reports

This status reports will be generated by this IoT infrastructure management component, which will show sensor, smart nodes and cluster nodes information like their statuses, sensor types number of resources.
These reports will display the required information in an organized way. Some examples are shown in the below mockup screens.
Using these status reports, the user (here the infrastructure manager) can get specific information like- How many sensors are broken? Or how many of them are in Active or Inactive state?, etc.



6.2 Sensor Simulator

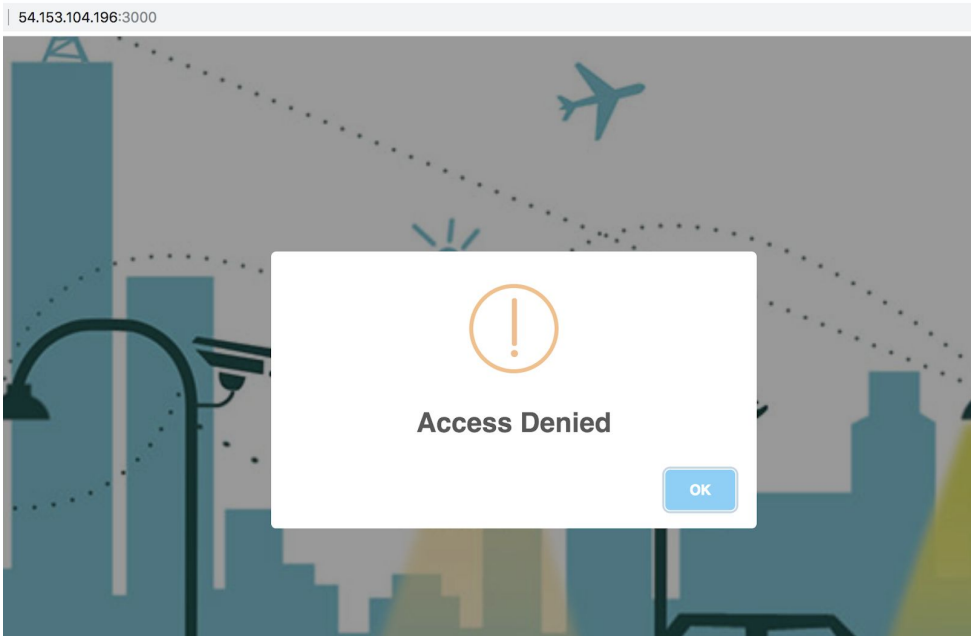
On successful login, the user is redirected to the following menu from which he can choose the next action that he wants to perform.



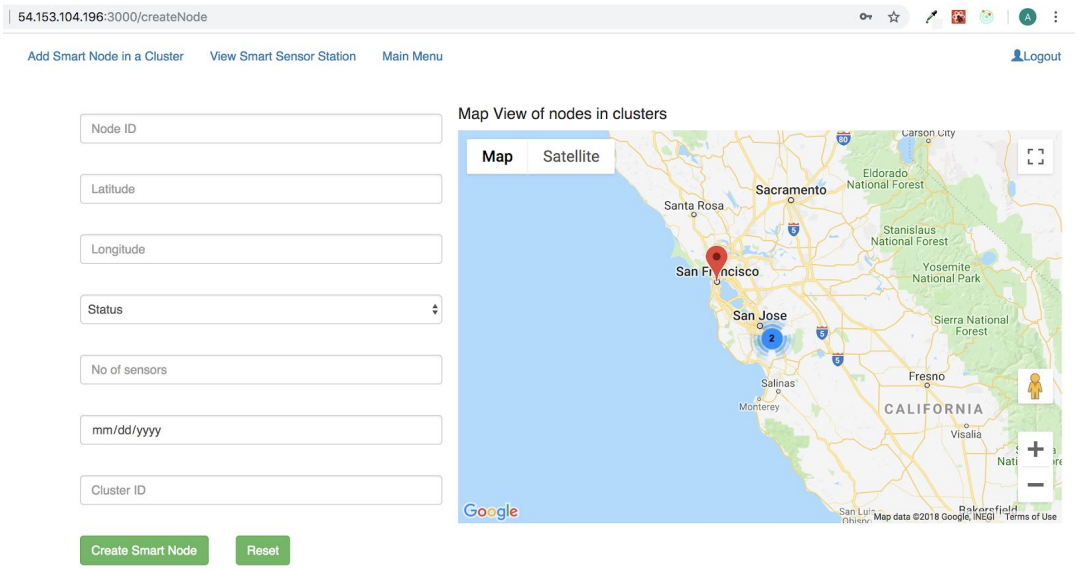
Only the people with right privileges, it. Infrastructure manager or Maintenance people can access the Sensor Simulator.

If a client tries to access the sensor simulator, he will be redirected to the Login screen and prompted to login as a privileged user.

For instance -



Logging in with the right privileges will take you to the landing page of Sensor Simulator module which is a form to fill out for creating a node within some cluster -



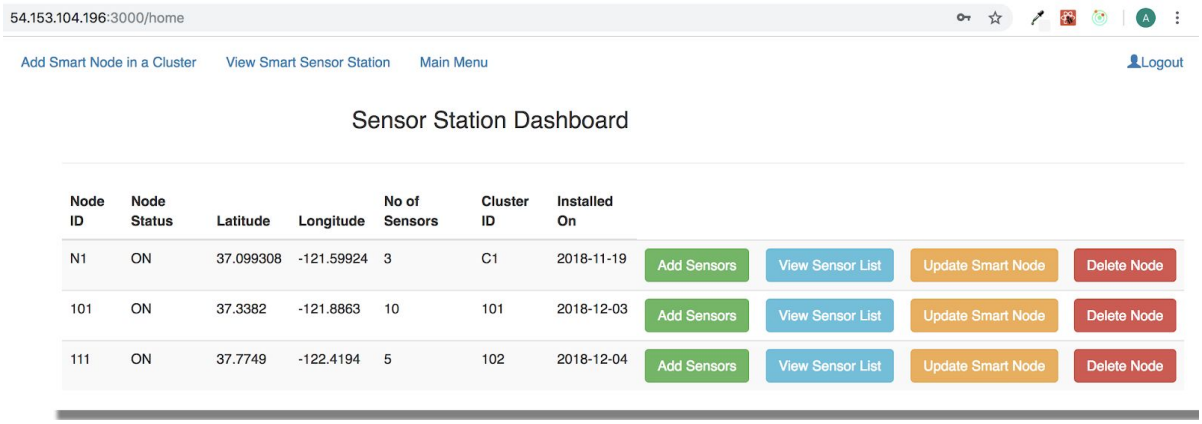
In the above map, the red marker denotes a node and the blue marker denotes a cluster with number of nodes written on the blue marker. Once a node is added, it will be reflected on this map.

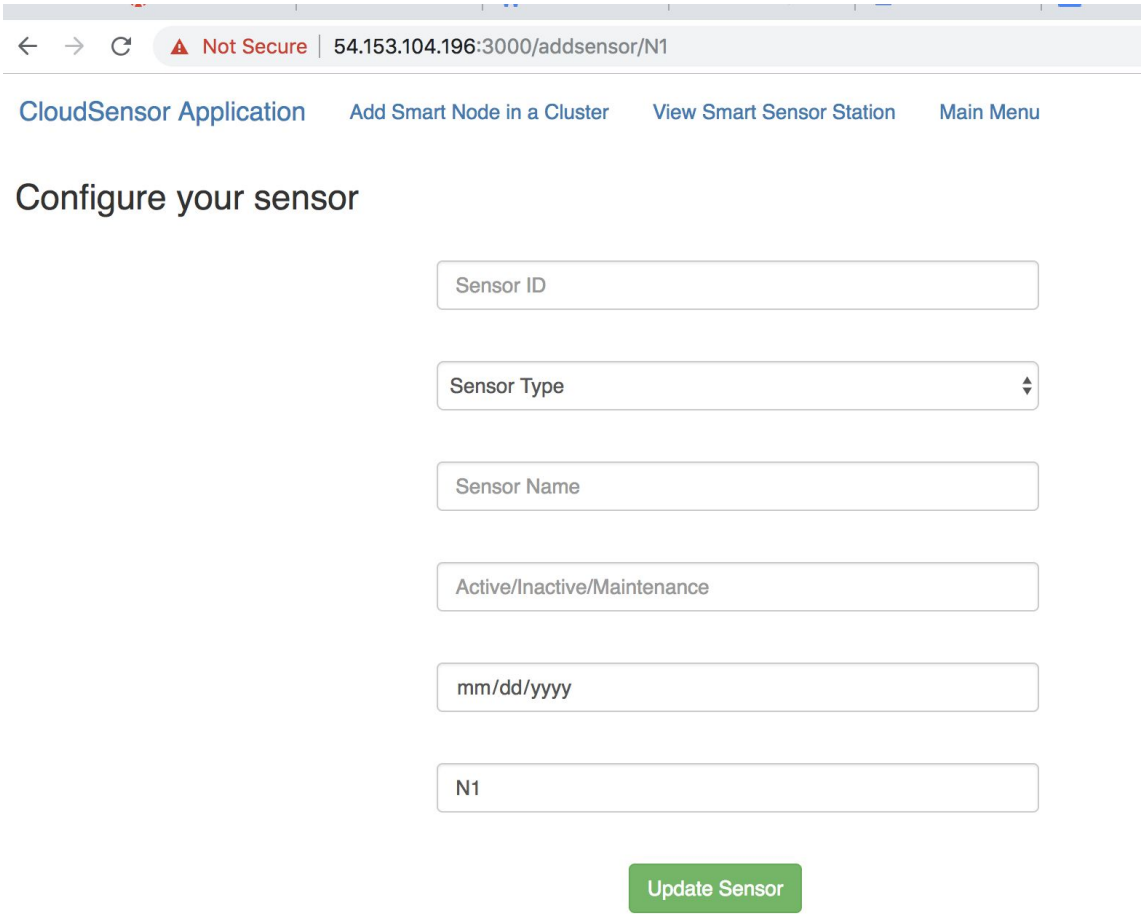
Clicking on View Smart Sensor Station on top will take the user to the following landing screen where following things are displayed on the dashboard on a Node level -

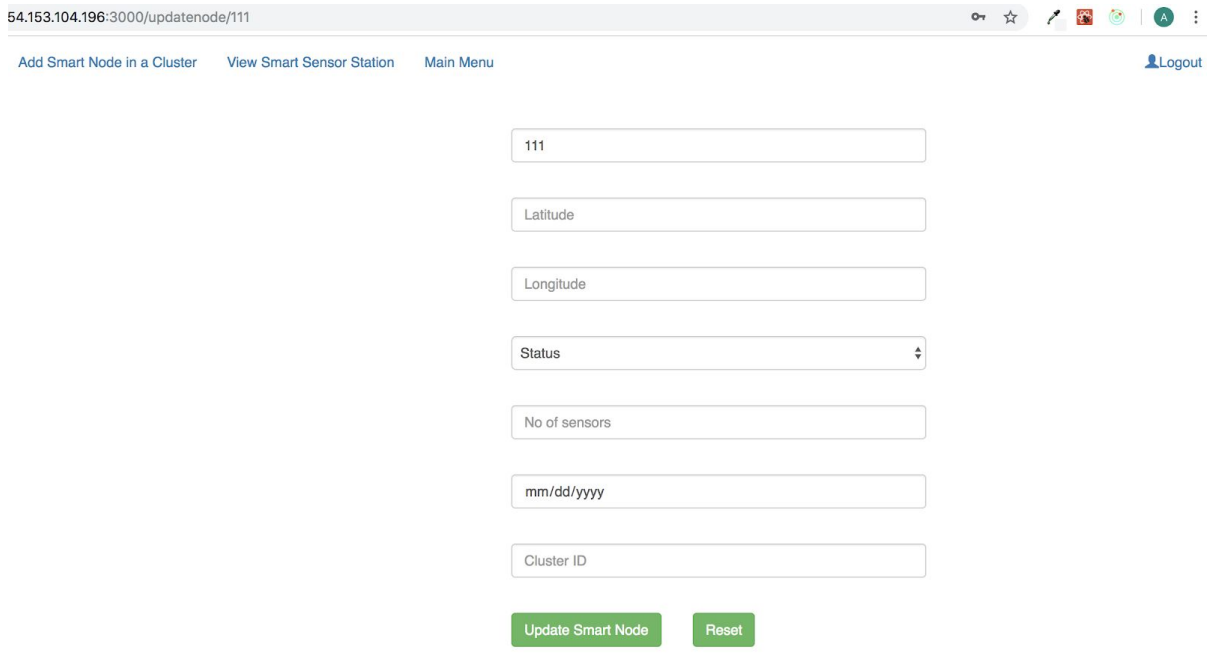
1. Node ID
2. Node Status
3. Latitude
4. Longitude
5. Number of sensors within the node
6. Date Installed On

   Each node will have following options -

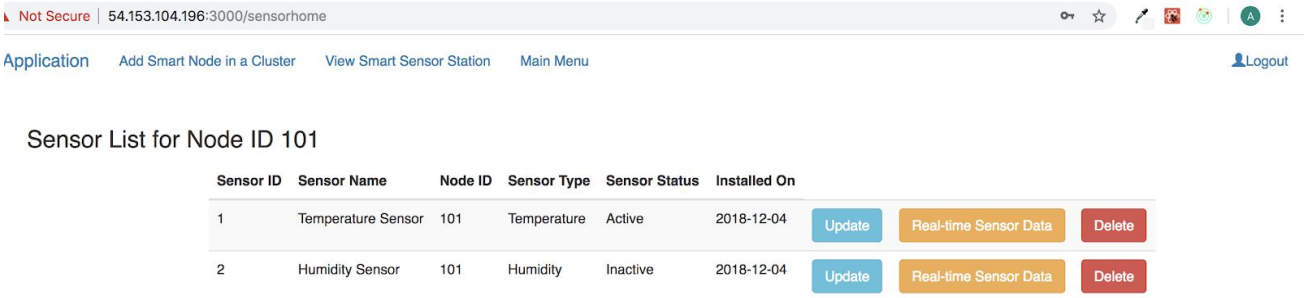
   1. Adding new sensors
   2. Viewing a list of already configured sensors
   3. Update the smart node
   4. Delete the smart node


54.153.104.196:3000/home
Add Smart Node in a Cluster
View Smart Sensor Station
Main Menu
Logout
Sensor Station Dashboard
Node ID
Node Status
Latitude
Longitude
No of Sensors
Cluster ID
Installed On
N1 ON 37.099308 -121.59924 3 C1 2018-11-19 Add Sensors View Sensor List Update Smart Node Delete Node
101 ON 37.3382 -121.8863 10 101 2018-12-03 Add Sensors View Sensor List Update Smart Node Delete Node
111 ON 37.7749 -122.4194 5 102 2018-12-04 Add Sensors View Sensor List Update Smart Node Delete Node

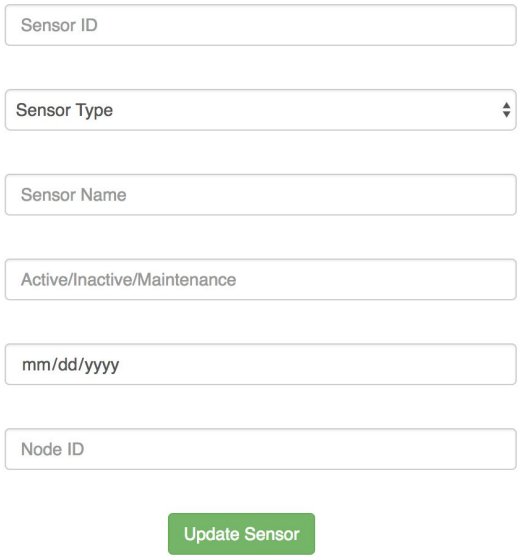

Not Secure
54.153.104.196:3000/addsensor/N1
CloudSensor Application
Add Smart Node in a Cluster
View Smart Sensor Station
Main Menu
Configure your sensor
Sensor ID
Sensor Type
Sensor Name
Active/Inactive/Maintenance
mm/dd/yyyy
N1
Update Sensor



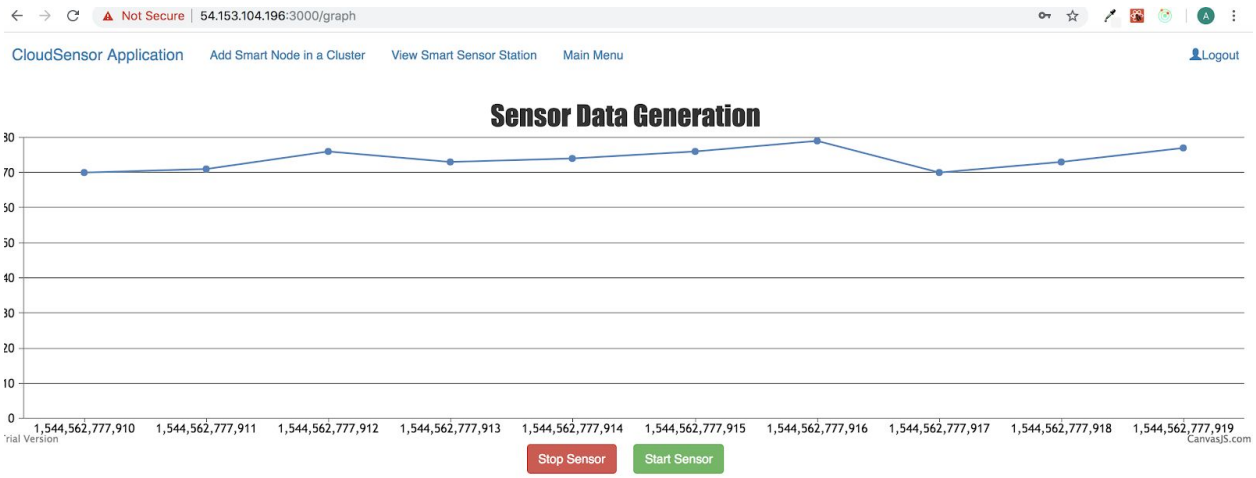
You can view list of sensors configured within the the node on the following screen -



Clicking on update sensor will take you to following screen -



You can view real time data being generated -

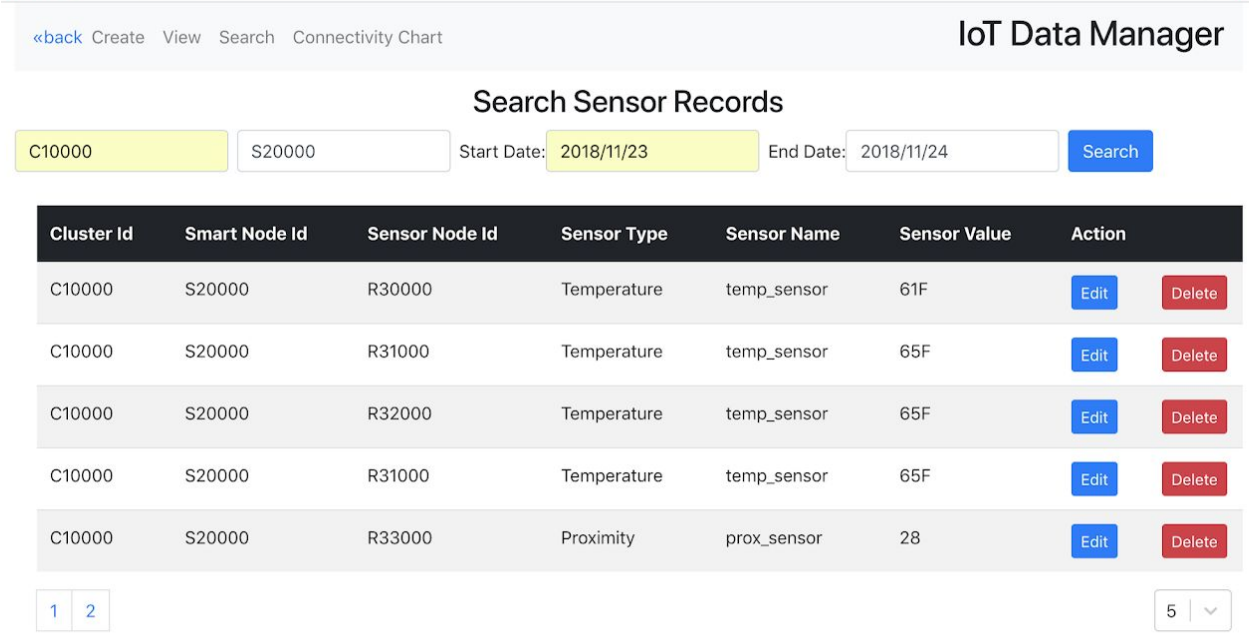


You can control the generation of data using Start and Stop buttons.

6.3 Data Manager

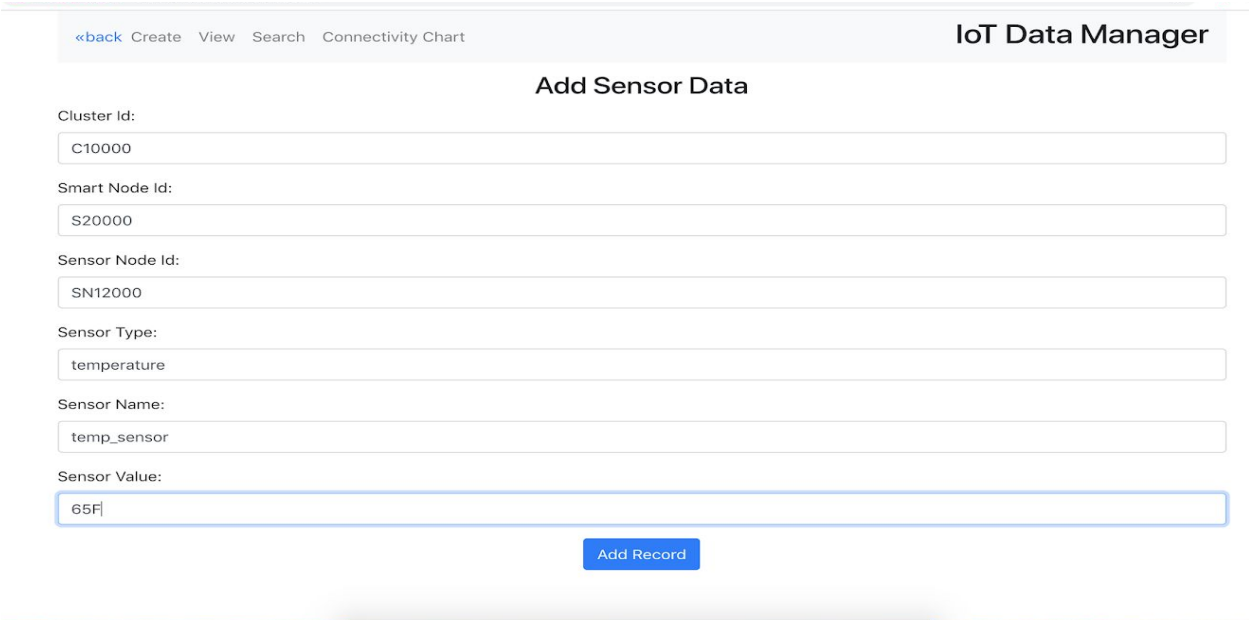
1.Search Sensor Data / Delete Selected Sensor Data / Edit Selected Sensor Data

With this functionality , for a given a range of dates you can retrieve the sensor data for required smart node or cluster node



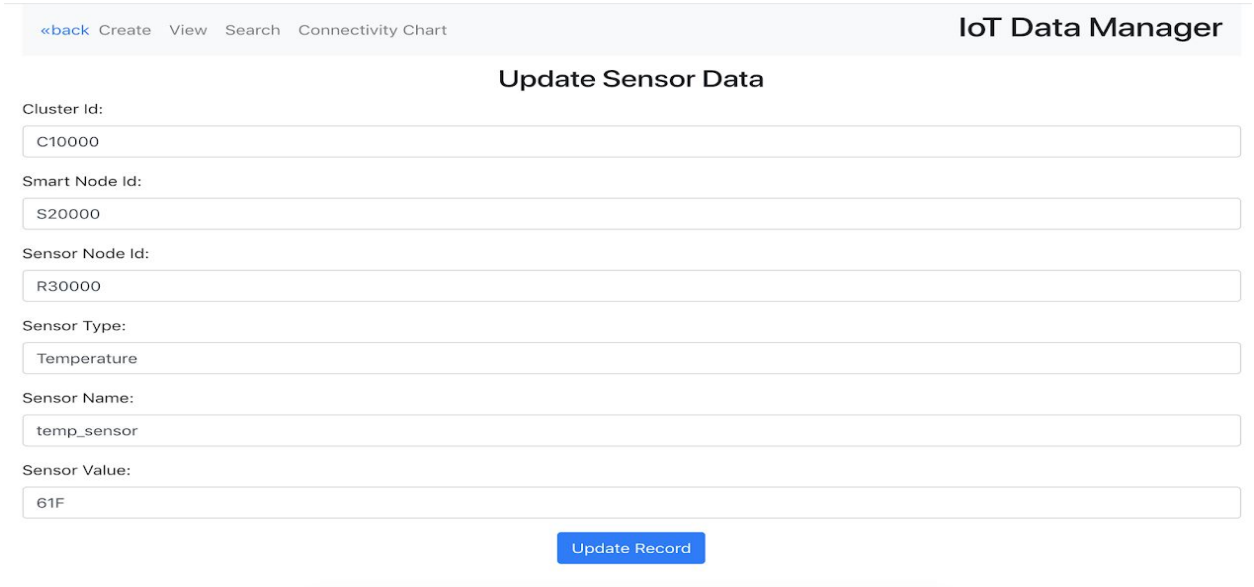
2. Add Sensor Data

With this functionality you can add new sensor data.



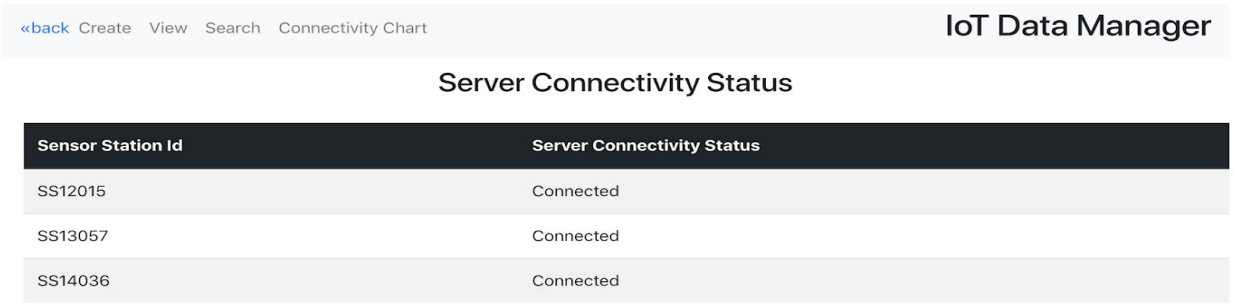
3. Edit Sensor Data

With this functionality you can edit intended sensor data



4.Connectivity chart

Displays connected sensor station to server



6.4 System dashboard

- The objective of this dashboard is to provide periodic insights about the iot based smart city information to the users.
- We have created a geospatial view of the Cluster, Smartnode & SensorNode using latitude and longitude.

Dashboarding Tool - Tableau

We have 3 views for our dashboard-

- **The dashboard view for Client** - Details about the sensors in the smart node like Temperature, Humidity and Wind
- **The dashboard view for Maintenance crew-** Details about the status of the nodes - Active, Inactive, Paused and Broken
- **The dashboard view for Infrastructure Manager-** Details about nodes added, deleted and updated

3.4.1. Dashboard view for Client

Key Points:

- Functionality - Interactive Dashboard

Data Points:

- Daily Max Min Humidity
- Daily Max Min Temperature
- Daily Max Min Wind

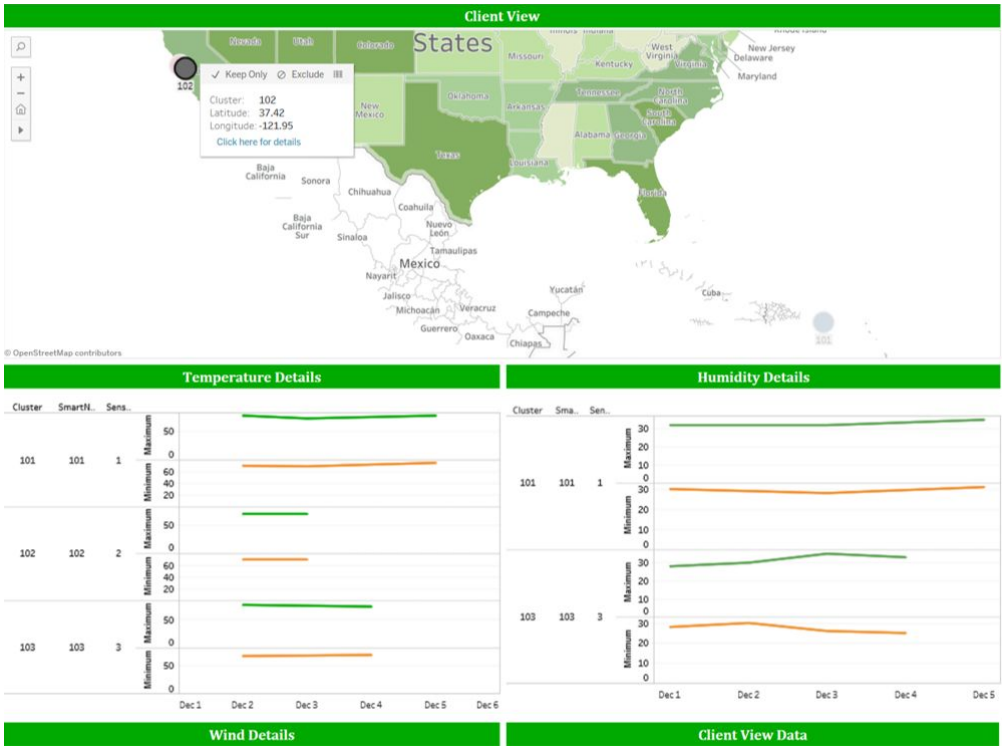


Fig: Interactive view for client

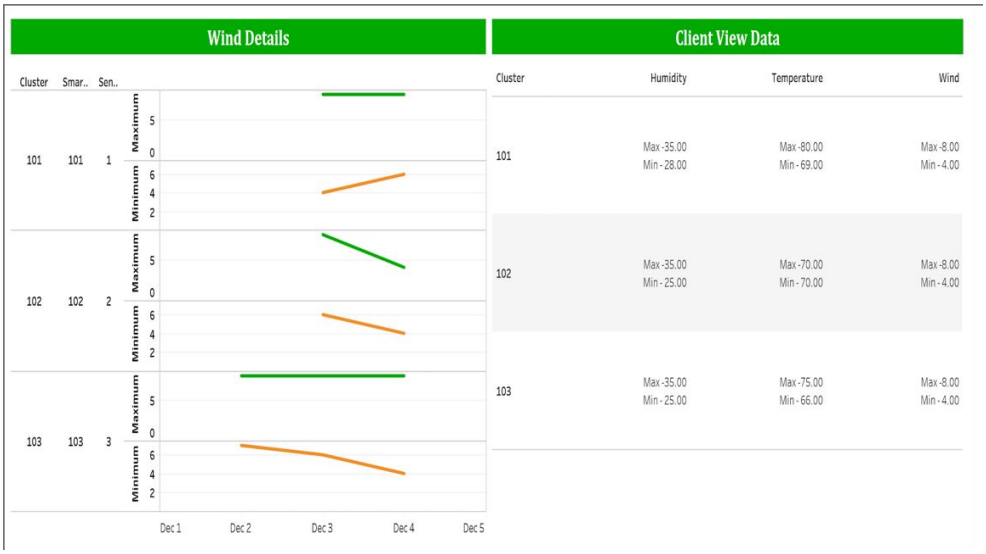


Fig: Interactive view for client

3.4.2. Dashboard view for Infrastructure Manager

Key Points:

- Functionality - Interactive Dashboard

Data Points:

- Count of Sensors Created
- Count of Sensors Updated
- Count of Sensors Deleted
- Cluster Connectivity

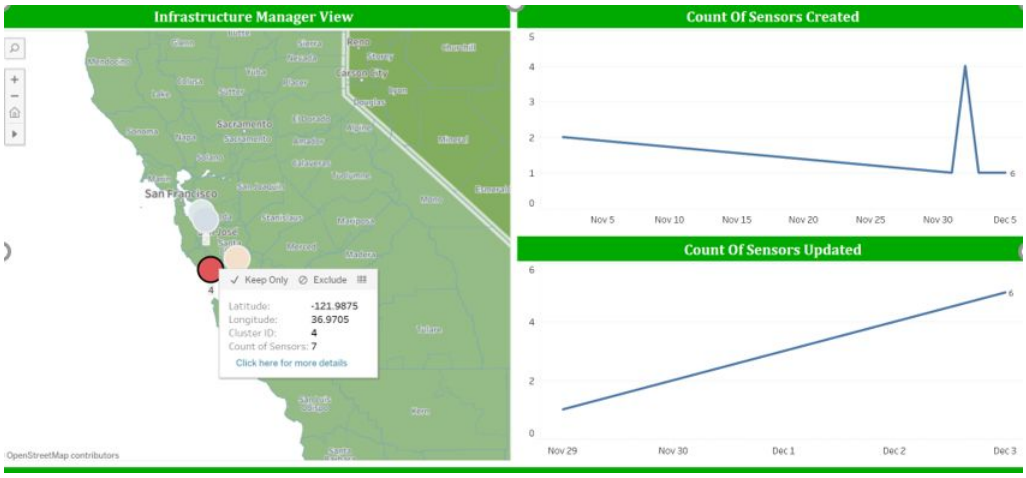


Fig: Interactive view for Infrastructure Manager

Cluster Details		
Cluster ID	SmartNode ID	Sensors
2	4	5,
3	5	4, 6,
	7	2, 14, 15, 16, 22,
4	2	3,
	5	8, 9, 10, 11, 12, 13,
6	2	7, 17, 18, 19, 20, 21, 23, 24, 25,

Fig: Cluster Details for Infrastructure Manager

3.4.3 Dashboard view for Maintenance Crew

Key Points:

- **Functionality - Interactive Dashboard**

Data Points:

- Sensor Connectivity Stats
- Active
- Inactive
- Paused
- Broken

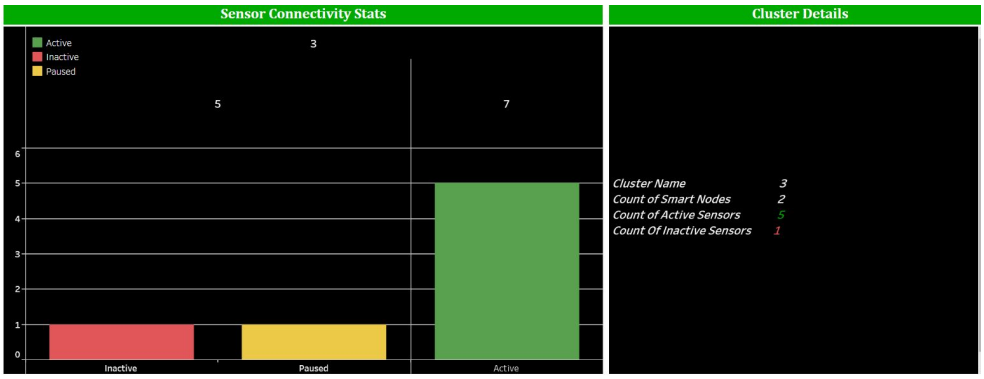


Fig: Sensor Connectivity Statistics for Maintenance Crew

Fig: Cluster Statistics for Maintenance Crew

7. Edge Station Design and Simulation

Edge stations are smart nodes or devices which are used in a distributed computing paradigm. In this project, each node is considered to be installed on a utility pole on some street.
Each node consists of a collection of sensors like temperature, wind, humidity, proximity, etc. Multiple such nodes make up a smart cluster which is equipped with internet and has means to communicate with the backend server.
Apart from this, each smart node can also communicate with the cluster since they have wireless communication capability.

Each node includes the following items:

1. **Controller** -
   - Collection of sensor data

This function refers to the collection of data from various sensors of the smart node at a polling frequency decided by the administrator while creating the node.

   - Transfer sensor data

Sensor data should be transferred from the back-end server frequently. This function is triggered when user requests for sensor data. The back end server fetched the requested data from the database.

   - Smart node control

This module performs basic operations such as node creation, node updating, node deletion and viewing the status change in the list of nodes

   - Smart node configuration

You can configure a node as per your requirements by adding/updating, deleting or viewing the existing sensors.

   - Every sensor can be monitored by the following status – Active/Inactive/Maintenance etc

2. **Cluster**
   - This controls and supports a number of smart nodes, and collects sensor data from them, and send sensor data to the back-end server.

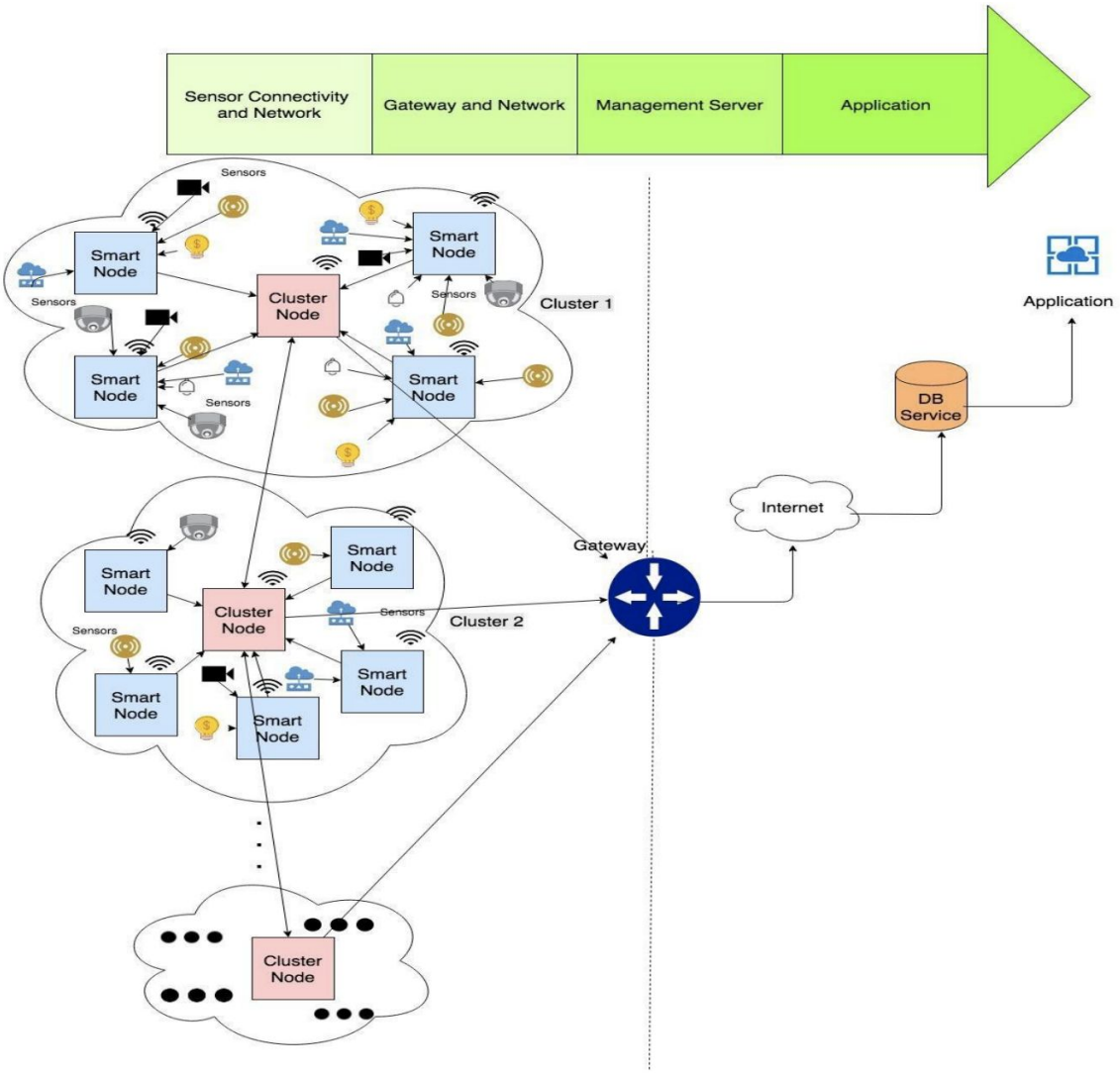


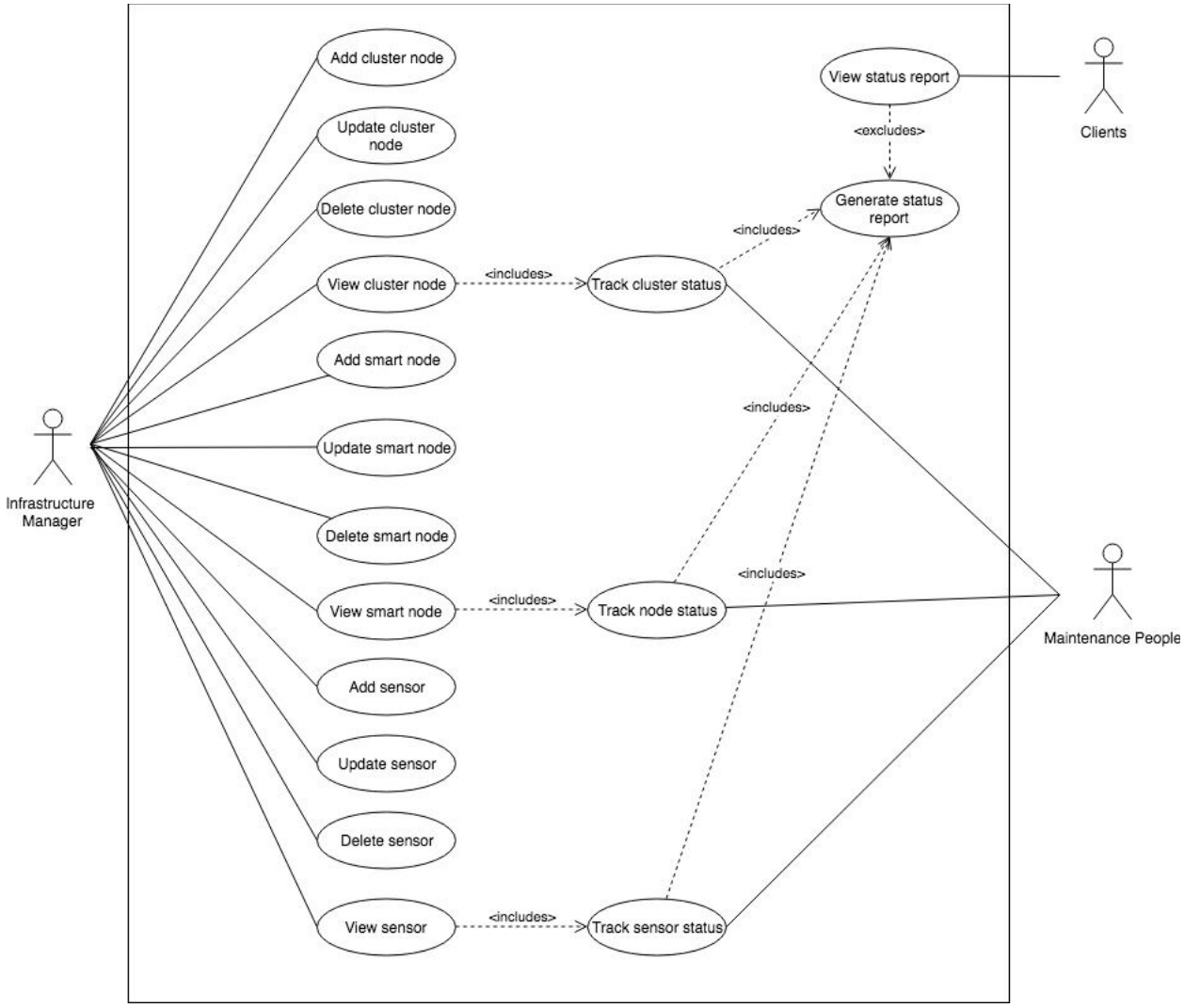
Fig : Sensor Station Design and interaction

8. System Application Examples

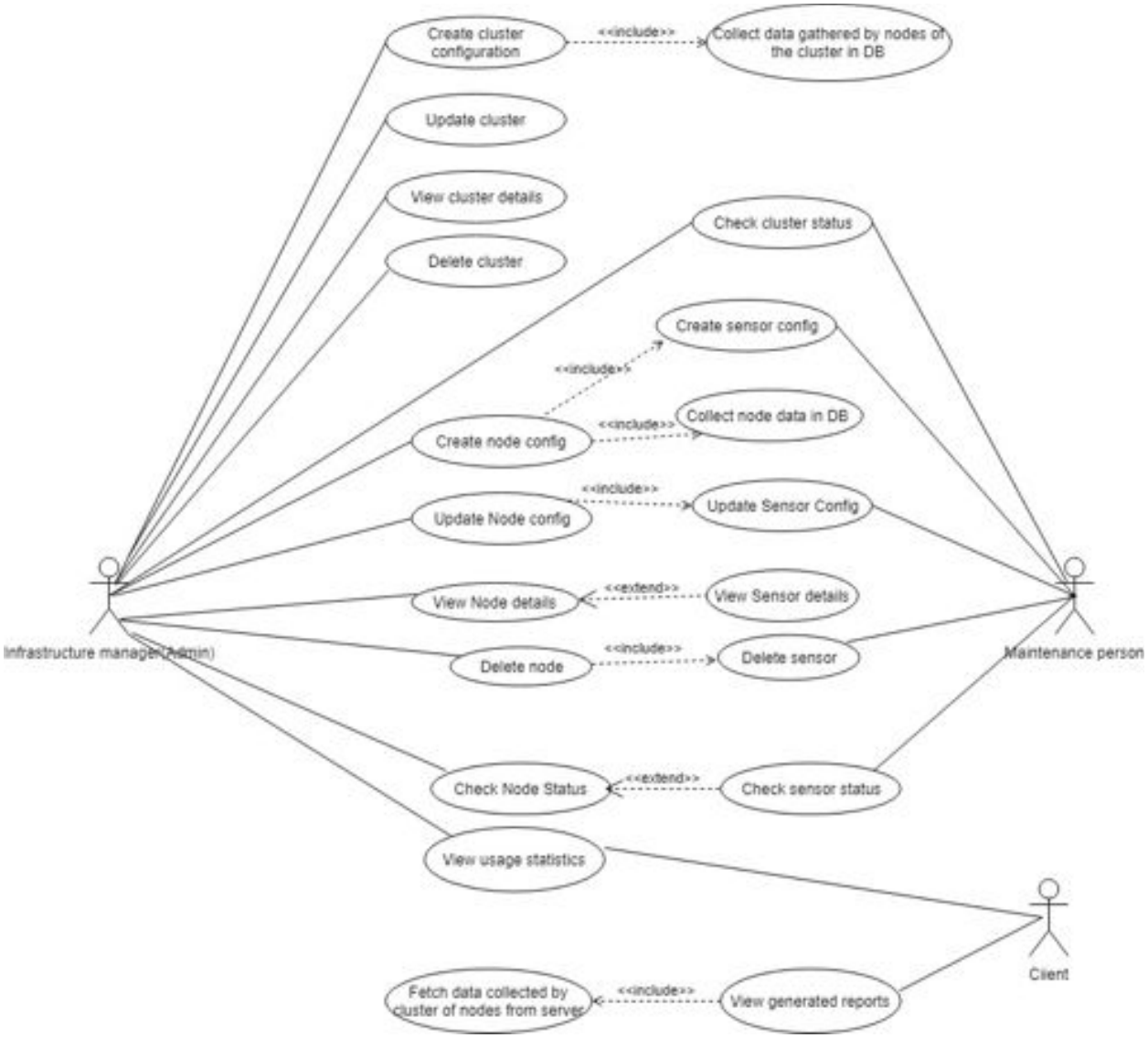
8.1 Scenario-based system application example

Example 1: For the infrastructure manager, it mainly focuses on this users' functions. The infrastructure manager performs the set-up, configurations and management of smart nodes, sensors and cluster controllers.

In a scenario where a infrastructure manager Mike wants can add/update/delete various installed nodes from the map(cluster node/ smart node/ sensors), can track their statuses, and also view their details on the map on click of the nodes.
These actions performed by Mike are shown in the use-case diagram below.



Example 2: Consider another scenario where a manager, Harvey wants to not only perform the functions specified above but also configure sensors, frequency of polling the data and sending the data generated by smart nodes within some cluster to the backend. Following are the use cases that the simulator module can be used for:

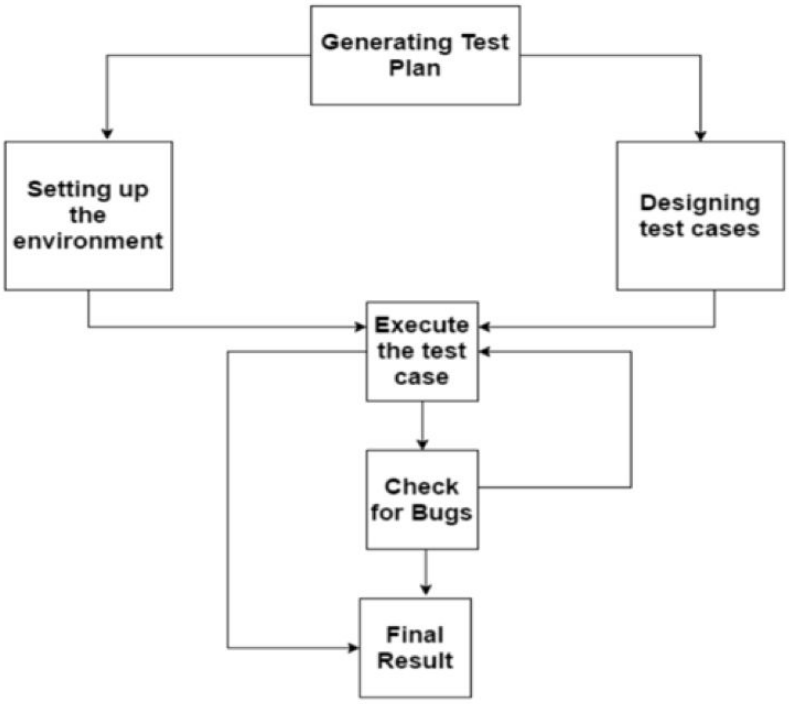

Create cluster configuration
<<include>>
Collect data gathered by nodes of the cluster in DB
Update cluster
View cluster details
Delete cluster
Check cluster status
Create sensor config
<<include>>
Collect node data in DB
Create node config
<<include>>
<<include>>
Update Sensor Config
Update Node config
<<extend>>
View Sensor details
View Node details
<<include>>
Delete sensor
Delete node
<<extend>>
Check sensor status
Check Node Status
View usage statistics
<<include>>
View generated reports
Fetch data collected by cluster of nodes from server
Infrastructure manager (Admin)
Maintenance person
Client

9. System Performance Evaluation and Experiments

All the possible use- cases and modules that were tested. The modules might be extended later as per the requirement.

Testing and Experimentation Steps

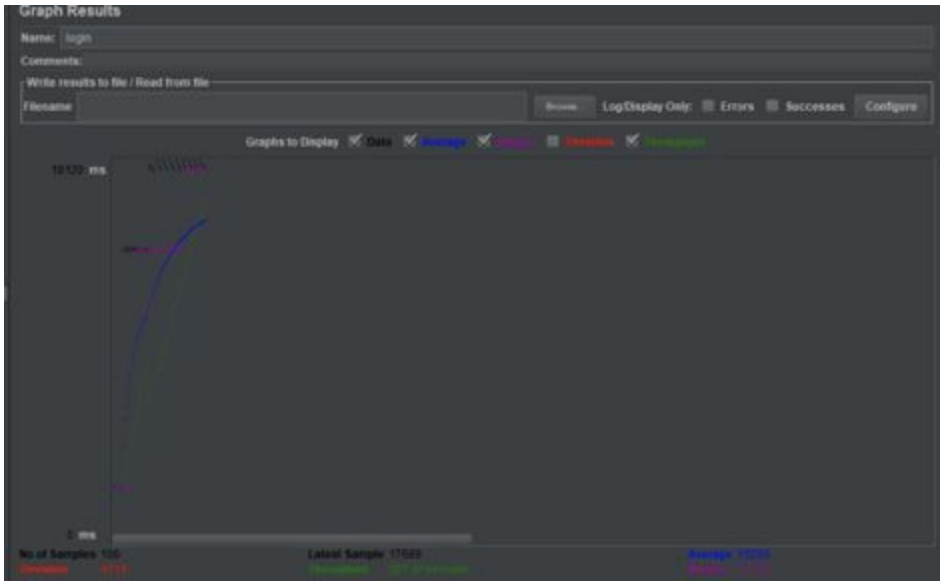
- Testing tools installation- JMeter, Selenium
- Testing Environment Setup -Install Java Runtime
- Creation of test cases- Selenium automation test scripts in java
- Running the test script
- Debugging and bug Fixing cases
- Manual Testing again for verification



Testing Approach

Results of JMeter testing:

1. Login route for 100 users:



UNIT TEST CASES USING MOCHA-

The green marks indicate that the expected results matched the expected results.

Deployment

Application is deployed the application as a Software as a Service on Cloud using Amazon EC2 (Elastic Cloud Compute 2) instance. It is used to start servers virtually. Advantages of application deployment on cloud are the features like flexibility, elasticity, scalability, high availability, no additional hardware installation, pay as we use model also we do not have to acre about the sea-up, maintenance and configurations- it is handled by the cloud services. In AWS EC2, there are a large variety of instance types which we can select according to our requirement (for e.g. various combinations of CPU, networking capabilities, memory, storage, etc.) Scaling up or down is easily

possible according to the server load. AWS EC2 instances are virtual servers that run applications.

Application deployment steps:

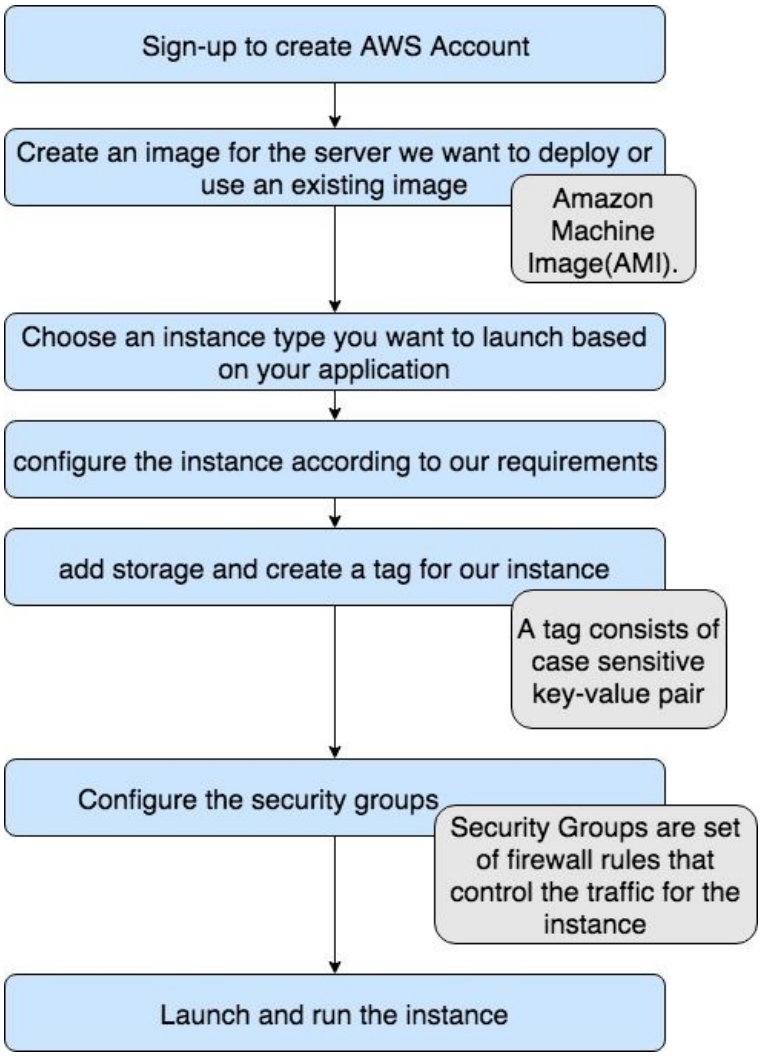
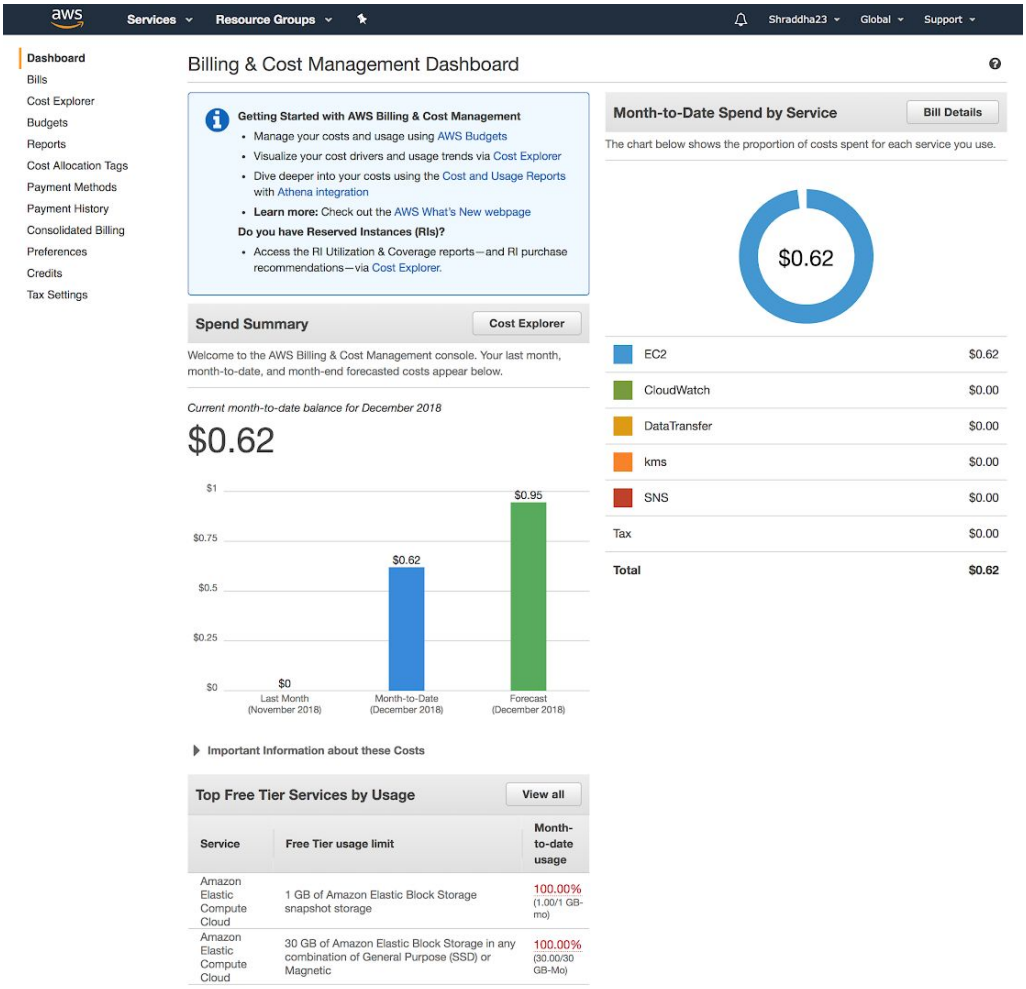


Figure: System Deployment



10. Conclusion

We have successfully developed an IOT Smart Streets Application, which can be used by infrastructure managers, maintenance crew as well as clients like City Council for better management of smart nodes and sensors. The data gathered from these smart stations can further be analysed using machine learning algorithms and used for development of the city.

11. References

[1] Amazon CloudWatch, [online] Available: https://aws.amazon.com/cloudwatch/

[2] Jerry Gao, “CMPE281 – Cloud Technology – Project Assignment ,Option #1 - An IOT-based smart street infrastructure management system on a cloud”, Fall 2018, 9/04/2018

[3] Flowchart Maker and Online Diagram Software, [online] Available: https://www.draw.io/

[4] Elastic Load Balancing, [online] Available: https://aws.amazon.com/elasticloadbalancing/

[5] Elastic Load Balancing, [online] Available: https://aws.amazon.com/elasticbeanstalk/

[6] What Is Amazon Relational Database Service (Amazon RDS)?, [online] Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html

[7] IoT Device Simulator, [online] Available: https://aws.amazon.com/answers/iot/iot-device-simulator/

[8]The 4 stages of an IoT architecture, [online] Available: https://techbeacon.com/4-stages-iot-architecture

[9] “Powering the smart cities with intelligent lighting”, [online] Available: http://www.eprmagazine.com/features/powering-the-smart-cities-with-intelligent-lighting/