Project Design Document

For

SMART STREETS

CMPE 281: Cloud Technologies FALL 2018

Submitted to:

Dr. Jerry Gao

Team 07 : Phoenix

Supriya Jain (supriya.jain@sjsu.edu)
Priyanka Singhal (priyanka.singhal@sjsu.edu)
DeekshithaReddy Kankanala (deekshithareddy.kankanala@sjsu.edu)
LakshmiPrasanna Kona (lakshmiprasanna.kona@sjsu.edu)

Table of Contents

1. Introduction

This project aims to build a cloud-based solution for smart and green streets in the city of San Jose. The proposed solution aims to make the city streets smart by bringing in energy efficient street light and environment monitoring systems. The system also aims to capture real time monitoring data for streets to detect and prevent anti-social elements and crime watch. This would help reduce street light electricity bills, attract tourism and increase real estate value.

This document covers the high-level component design for the cloud-based solution for the smart street project. The data published from various smart sensors positioned at different geographic locations is controlled and monitored remotely by a cloud based smart solution. The proposed solution is the cloud-based system with various cloud benefits such as elastic computation, auto scalability, remote accessibility and cost effectiveness.

2. System Requirement Analysis

2.1 Proposed System Overview

Infrastructure management team will equip the street utility poles with sensors and cameras connected to Wi-Fi that can communicate with each other and act as smart nodes. An IOT infrastructure and network is formed with a smart cluster monitoring and communicating with the back-end server and all other nodes. The above proposed system enables clients to access the system to collect sensor data and generate required reports of energy efficiency, traffic monitoring, crime watch and environment monitoring. Maintenance people can access the system's smart nodes and sensor status.

2.2 Functional Requirements

Req-ID	Requirement Title	Target Users	Priority
Req-Sys-001	User Authentication	All Users	M
Req-Sys-002	Collect/Transfer Sensor Data	Client/Sensor Data Manager	M
Req-Sys-003	Manage Cluster Node	IOT Infrastructure Manager	M
Req-Sys-004	Manage Smart Node	IOT Infrastructure Manager	M
Req-Sys-005	Manage Sensors	Sensor Data Manager/ IOT Infrastructure Manager	M
Req-Sys-006	Manage Sensor Data	Client/Sensor Data Manager	M
Req-Sys-007	Generate Reports	All Users	M

Req-Sys-001 Login:

Requirement ID	Req-Sys-001
Requirement Title	Login to the system
Priority	Must
Description	All the system users should be able to access the system at any time securely.
Frequency of Use	Daily
Pre-Conditions	User should have a registered account User
Post-Conditions	User should be able to view the system dashboard.

Req-Sys-002 Collect/Transfer Sensor Data:

Requirement ID	Req-Sys-002
Requirement Title	Collect/Transfer Sensor Data
Priority	Must
Description	System should be able to collect sensor reading data from different sensors on to the cloud back end servers.
Frequency of Use	Daily
Pre-Conditions	User should be logged in successfully.
Post-Conditions	Data should be recorded on the cloud db.

Req-Sys-003 Manage Cluster Node:

Requirement ID	Req-Sys-003
Requirement Title	Manage Cluster Node
Priority	Must
Description	Users should be able to perform basic control and configuration functions for the cluster node. Basic functions include cluster creation, updation, deletion and smart node addition.
Frequency of Use	Daily
Pre-Conditions	User should be logged in successfully.
Post-Conditions	NA

Req-Sys-004 Manage Smart Node:

Requirement ID	Req-Sys-004
Requirement Title	Manage Smart Node
Priority	Must
Description	Users should be able to perform basic control and configuration functions for the smart node. Basic functions include node creation, updation, deletion and sensor addition.
Frequency of Use	Daily
Pre-Conditions	User should be logged in successfully.
Post-Conditions	NA

Req-Sys-005 Manage Sensors:

Requirement ID	Req-Sys-005
Requirement Title	Manage Sensors
Priority	Must
Description	Users should be able to perform basic management functions for the sensors such as virtual sensor creation, updation, deletion and track status.
Frequency of Use	Daily
Pre-Conditions	User should be logged in successfully.
Post-Conditions	NA

Req-Sys-006 Manage Sensor Data:

Requirement ID	Req-Sys-006
Requirement Title	Manage Sensor Data
Priority	Must
Description	Users should be able to add, update, delete and view sensor data.
Frequency of Use	Daily
Pre-Conditions	User should be logged in successfully.
Post-Conditions	NA

Req-Sys-007 Generate Report:

Requirement ID	Req-Sys-007
Requirement Title	Generate Report
Priority	Must
Description	Users should be able to generate reports based on sensor reading/usage data for a certain time range (year/month/week/day)
Frequency of Use	Daily
Pre-Conditions	User should be logged in successfully.
Post-Conditions	NA

2.3 Non-Functional Requirement

i. Availability, Maintainability and Reliability

Our design uses Amazon cloud services which ensure high availability with 99.9% confidence. There shall be multiple server replicas, each having their own instance of the application, making each user session stateless, and hence decreasing the dependency on any particular instance. The proposed cloud-based solution also addresses another major issue i.e. accessibility. The data shall be available to any user who is connected to the smart street system via Internet.
Additionally, by using Amazon standard cloud services, the design also offloads the maintenance of its infrastructure to third party AWS security maintenance.
There would be regular backups of the DB data to ensure fast recovery in case of failure.

ii. Performance

Broadly, there are two main request initiators of the system: Client users and Sensor device managers. Each of these users can access the system from distinct geographic locations. Once the request is initiated, the load balancer will delegate the request to the nearest available server. This will ensure fast request handling.

iii. Scalability

Scalability is one of the most important features keeping in mind the proliferating use cases of the smart street system. The same cloud-based system shall be extended to more streets or even to more cities leading to greater usability. The solution shall leverage from the following key cloud features:

a. Auto scaling:

Each of the Amazon EC2 instances shall be capable of up scaling/down scaling itself (i.e. more instances shall be created) based on the usage as per the configured auto scaling policy. Auto scaling policy will allow scaling of VM instances upon a provided threshold value.

b. Dynamic provisioning:

Provisioning of infrastructure components based on demand is at the heart of our cloud-based solution. Each of the components such as Databases and EC2 instances are horizontally scalable. This provides higher handling capability to the user and at the same time avoids component wastage when not in use.

iv. Multitenancy:

Due to its distributed and elastic nature the proposed design shall be capable of catering to multiple requests at the same time. Each of the highly computing application servers shall be shared amongst multiple users. Each of these server instances are capable of up scaling/down scaling as the number of user request increases/decreases.
Additionally, each of the instances will contain their own software stack and shall be capable of offering dedicated experience to the user.

v. Security

Apart from the basic authentication and role-based authorization, the proposed design is encapsulated with AWS infrastructure security capabilities. Network security groups and certificate based secure handshake are some of the salient features of the proposed cloud - based solution.

3. Cloud Based System Infrastructure and Components

3.1 Overview

Our IoT system will have multiple independent smart nodes placed on different locations on the streets for monitoring temperature, humidity, pressure, etc. These smart nodes shall be connected to cluster nodes and send the sensor data to the cloud in real time. The data shall be recorded in the cloud database. The analysis of data would be done in the cloud to aggregate the data and make prediction. Cloud based application shall be used for visualizing the data.

3.2 Sensor Cloud System Infrastructure

Sensor-Cloud Infrastructure

Sensor Cloud Infrastructure is an integrated version of Wireless Sensor Network and Cloud Computing. It uses concept of virtualization of sensor node. It is a scalable, high performance, computing and large scale storage infrastructure for storing of sensor data for real time processing as well as analysis of processed information.

How it works

As shown in Fig 3.2, The Sensor gateway collects information from the sensor nodes of WSN and transmits it back to the cloud gateway after compressing it which then stores it in the cloud storage server after decompressing. Sensor cloud uses physical sensors to accumulate its data and transmit all the sensor data into cloud infrastructure. It enables sensors to be utilized on cloud-infrastructure by virtualizing the physical sensors on cloud according to

users' need. Automatic provisioning of sensors is possible as and when required by users as they are dynamic in nature.

Sensor-Cloud architecture

1. Virtual sensor instance will be automatically provisioned to them when user requests for them.
2. User can create virtual sensor group for the physical sensor nodes.
3. Users can re-configure these templates or virtual groups according to the need.

4. User can delete the virtual sensor instance once it become useless and can avoid the utilization charges for these recourses.
5. Sensor cloud infrastructure provides sensors to end users as and when required in such a way that they are part of their IT resources.
6. Via user interface these instances and their sensor data will be used by end users

7. WSN would be integrated with cloud computing via centralized controller. It would collect the data from the WSN's and passes to the cloud application. Cloud application would then provide service to the users depending upon the need.

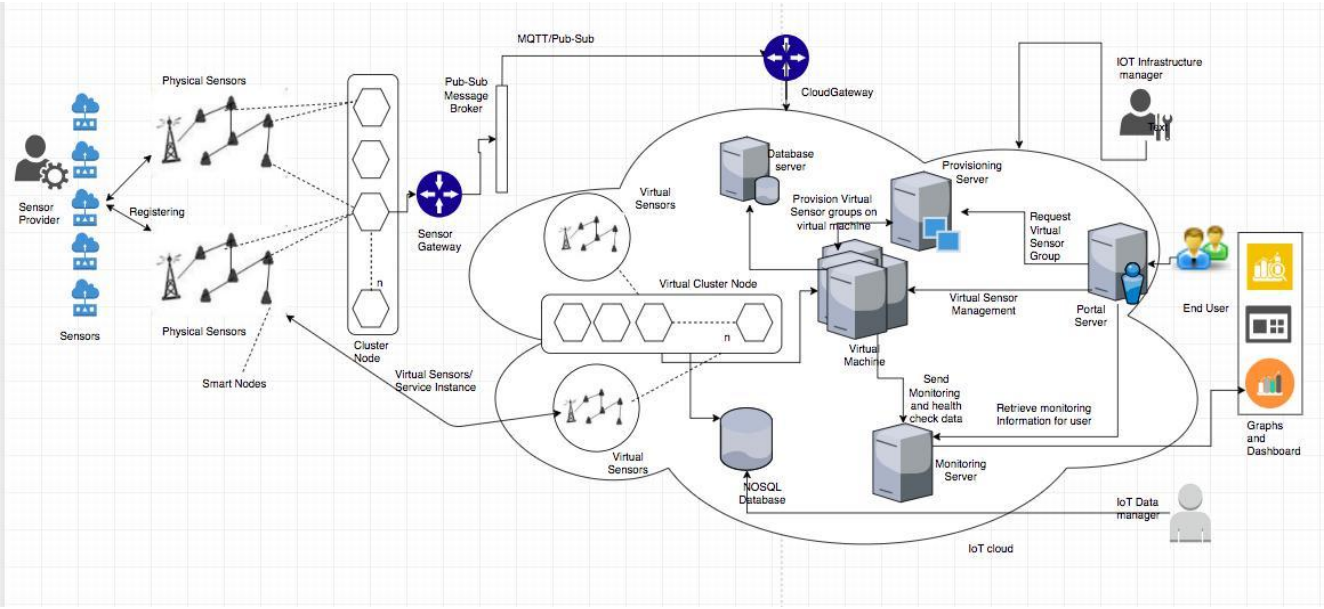


Fig 3.2 Sensor Cloud Architecture

Components in the architecture:

Virtual Sensor Group:

Each virtual sensor group has one or more virtual sensors and it is owned by end user. Virtual sensors can be controlled directly by web browser. End user can be activated or deactivate virtual sensors and also, they can check the status of the their virtual sensors.

Monitoring Server

Monitoring Server will receive the data about the virtual sensors. It will store the data into a database. End Users and Sensor-Cloud admin can monitor the status of the sensors.

Provisioning Server

Upon the requests from the portal server provision server will provision virtual sensor group. It will reserve IT resource pool after receiving the requests. Having retrieve templates of virtual sensor groups, it will provision virtual sensor group including virtual sensors. It will then update definitions of the virtual sensor groups.

Portal server

Depending on the role of the user portal server will determine the available operations when user log into the portal from the web browser.

Database Server

Data storage for smart streets application consists of databases for users, sensors data and virtual sensors group.

Application Server

Our IoT cloud includes the HTTP and MQTT servers which will be developed using Node.js due to its capability in high concurrency. Application servers are capable of handling a large number of concurrent connections as it runs in the form of an asynchronous event loop which performs all I/O operations with a single thread asynchronously.

3.3 Cloud Components used in Project Design

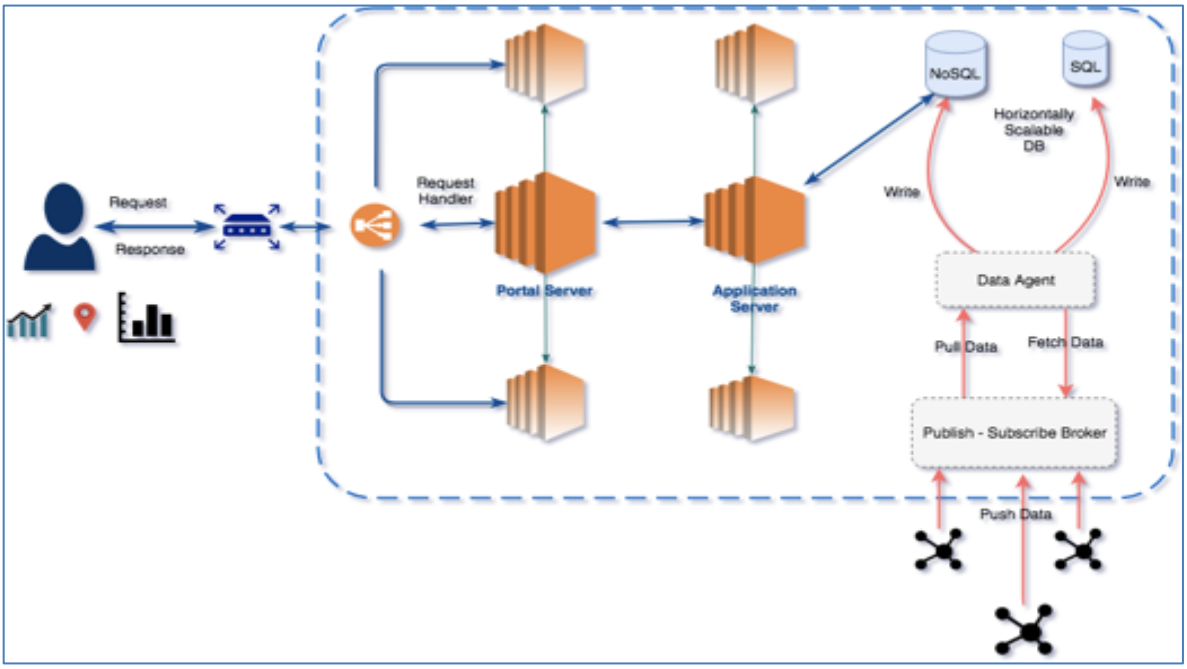


Fig 3.3 AWS cloud architecture

Following are all the cloud component of AWS we will be using for our cloud-based application.

AWS IAM

Identity Access Management will help to manage access to various AWS services like compute, storage, database and application in AWS cloud. Various users' access to aws services can be allowed or denied by creating users, groups and permissions.

AWS Elastic Compute Cloud (EC2)

The smart street application will be deployed on several EC2 instances which can access AWS S3 and AWS RDS. Our EC2 and Databases instances will be separated from each other so that they can be scaled easily.

AWS CloudWatch

To send the notification to administrator when a specific event happens CloudWatch will be used. Statistic information (CPU utilization, memory utilization, network utilization, Memory information, Storage information) will be showed to dashboard. We shall also be using cloudwatch to monitor all EC2 instances and to launch a new EC2 instance automatically.

Nginx Load Balancer

Load balancing is used across multiple server instances. It helps to increase productivity, reduce latency, increase resource utilization, maximize throughout. It also helps to make the application more fault-tolerant. Nginx is very efficient HTTP load balancer. It helps to share traffic among many application servers. It increases performance, make the application more scalable and reliable.

AWS RDS

Smart street web applications will be built to operate at very large scale, so we would be needing database with massive storage scalability, high throughput and high availability. AWS RDS fulfills our needs for highly demanding application.

3.4 Deployment Oriented System Infrastructure

This section describes the deployment-oriented system design for our proposed solution.

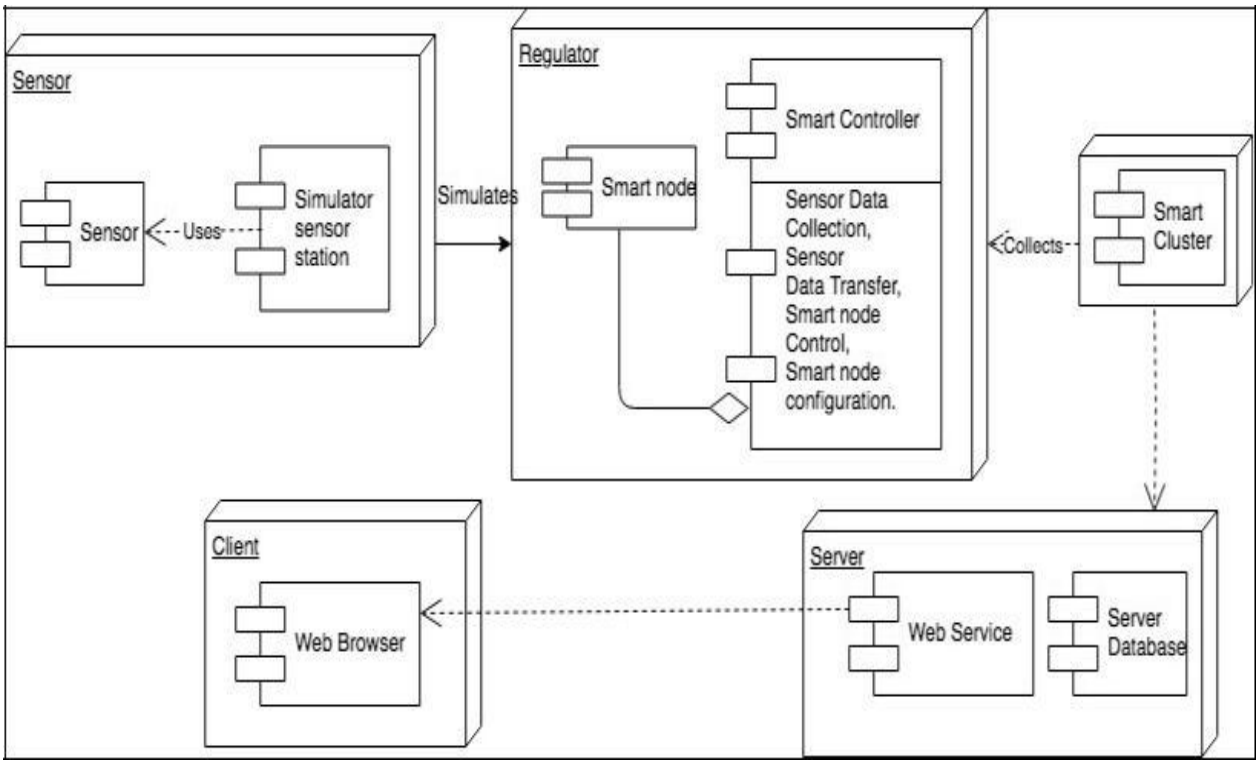


Fig 3.4 System Deployment Diagram

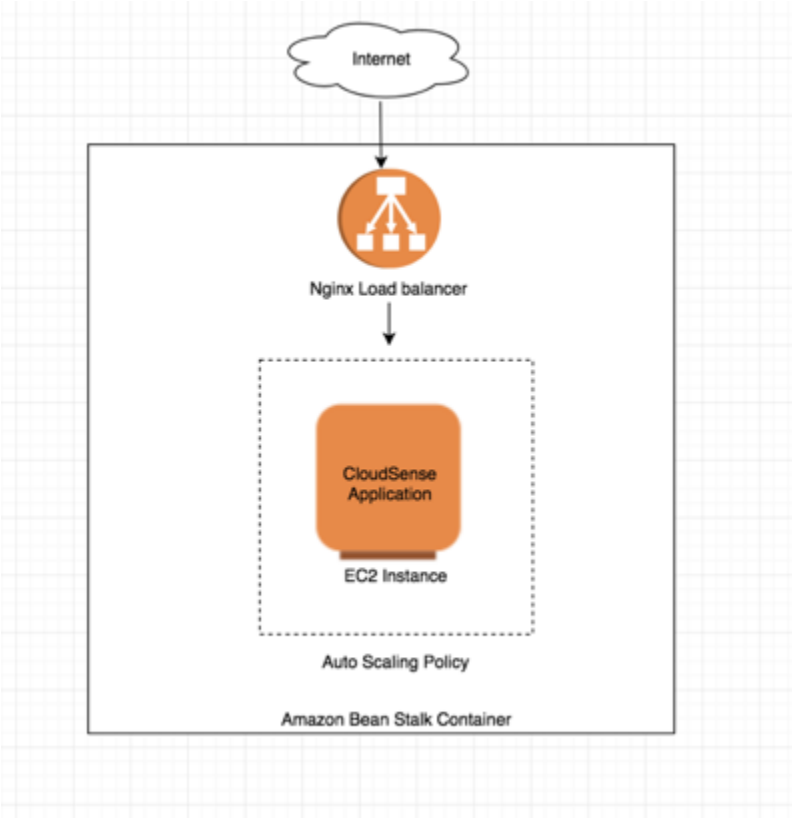


Fig 3.4.1 AWS detailed deployment diagram

Sensors are the main integrated component of the whole system. There is a simulator station which uses the sensors and get the data. Simulator connects to the smart node which is the combination of the sensors. Smart node involves in sensor data collection, sensor data transfer, smart node control, smart node configuration.

There comes a smart controller which is used for controlling all the aspects of the smart node functioning. Smart Cluster is maintained in order to collect all the data from the simulator and then send that to the database with connection to database.

Server provides the web services based on the data collected from the smart cluster and then that service will be available which would be used by the clients through the web browser.

So, for the smart street system, we use these deployment structure so that all the streets are composed with smart nodes with a group of sensors in each node and collects the data in network which is used for the minimizing the wastage of power through the optimization.

3.5 System Oriented Component Functional view

This section describes the functional interaction component design for our proposed solution.

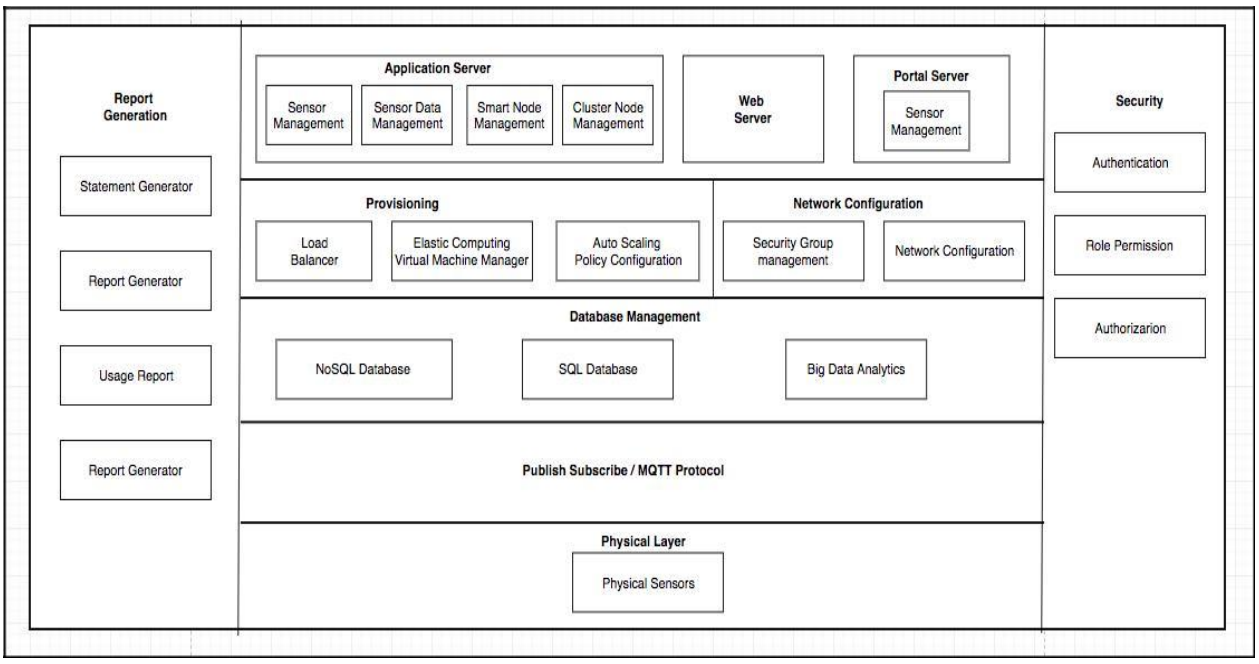


Fig 3.5 System oriented functional design

4. Functional Interaction design

This section describes the functional interaction design for our proposed solution. The proposed system will include the following types of users:

Client:

As represented in Fig 4.2.1, Client users post successful authentication can subscribe to various types of sensors. They can even group these sensors into a more manageable entity called the 'Smart node'. Users can also generate daily/weekly/monthly reports for the sensor data. Additionally, based on the sensor usage the users can generate usage as well as billing statements.

Sensor Device Manager:

As represented in Fig 4.2.2, Device managers configure and maintain the smart nodes as well as the sensors. Post successful login they can register/deregister new sensors. They can group various sensors into smart nodes. They can get a map view of the various sensors and track the status and health of various sensors. They can even control the frequency of the sensor data. Sensor Device Manager can even view reports and statements related to sensor and sensor data.

IOT Infrastructure Manager:

As represented in Fig 4.2.3, Infrastructure Managers are the ones who are responsible to setup, configure, and manage smart nodes, cluster nodes, and sensors, as well as their connections. Similar to other users, they can generate reports, usage statements and bills based on the historical sensor data. They also get to see a rich map-based view of the sensors.

Admin:

As represented in Fig 4.2.4, Admin user performs user management operations such as roles/permission creation, updating and deletion. He can also perform cloud related system functionalities such as network security group maintenance and virtual instance elastic provisioning.

5. UML Diagrams

5.1 Use Case Diagrams

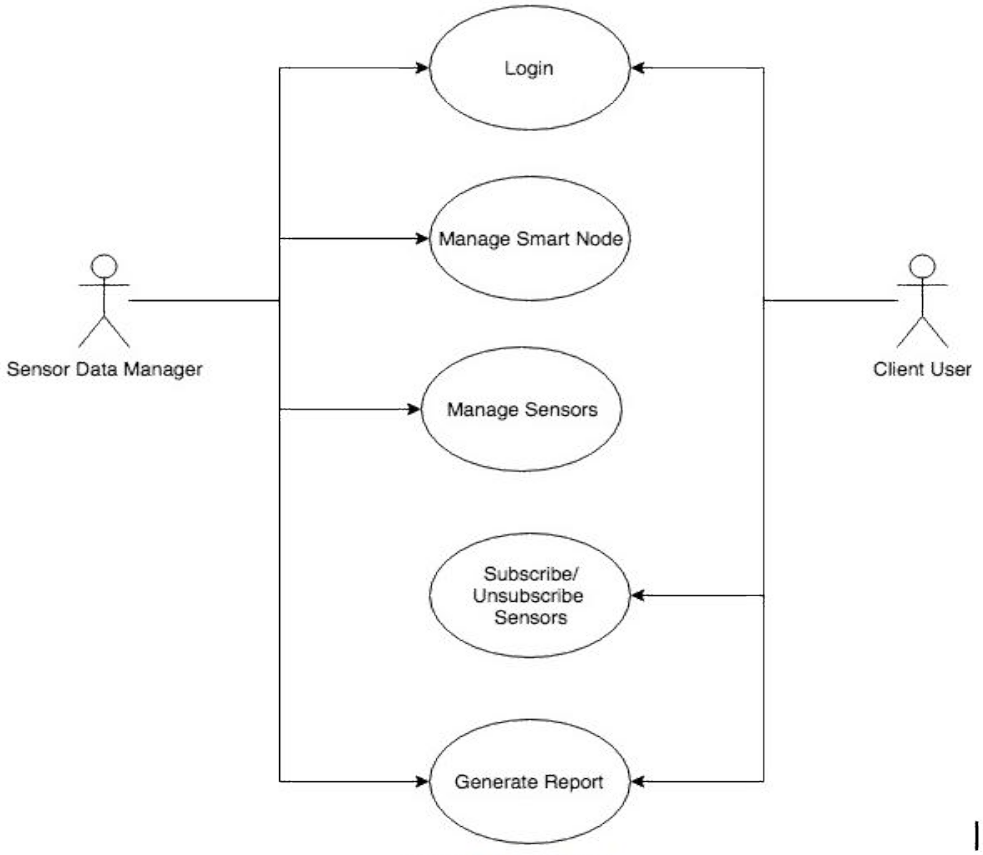


Fig 5.1.1 Use case 1

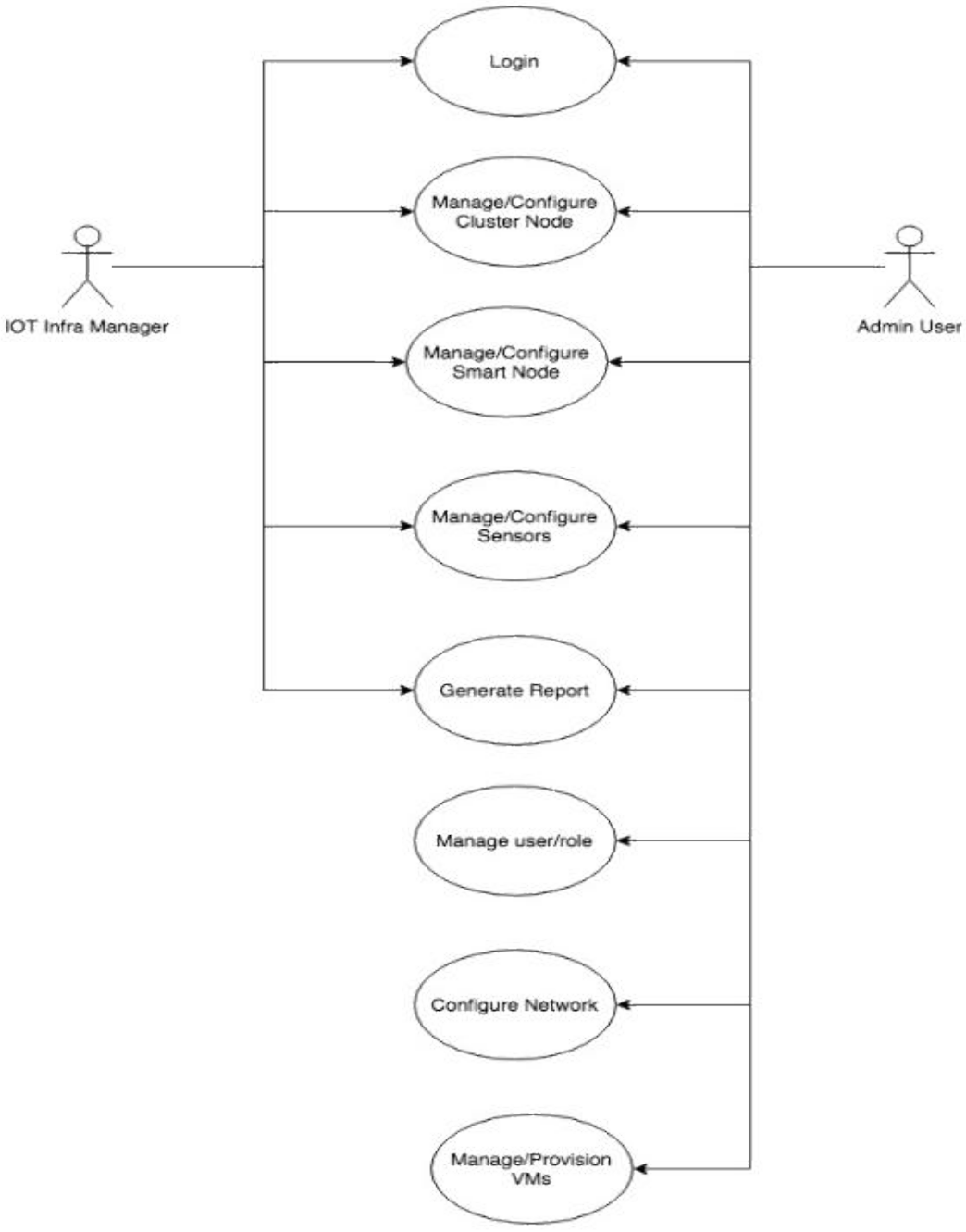


Fig 5.1.2 Use case 2

5.2 Sequence Diagrams

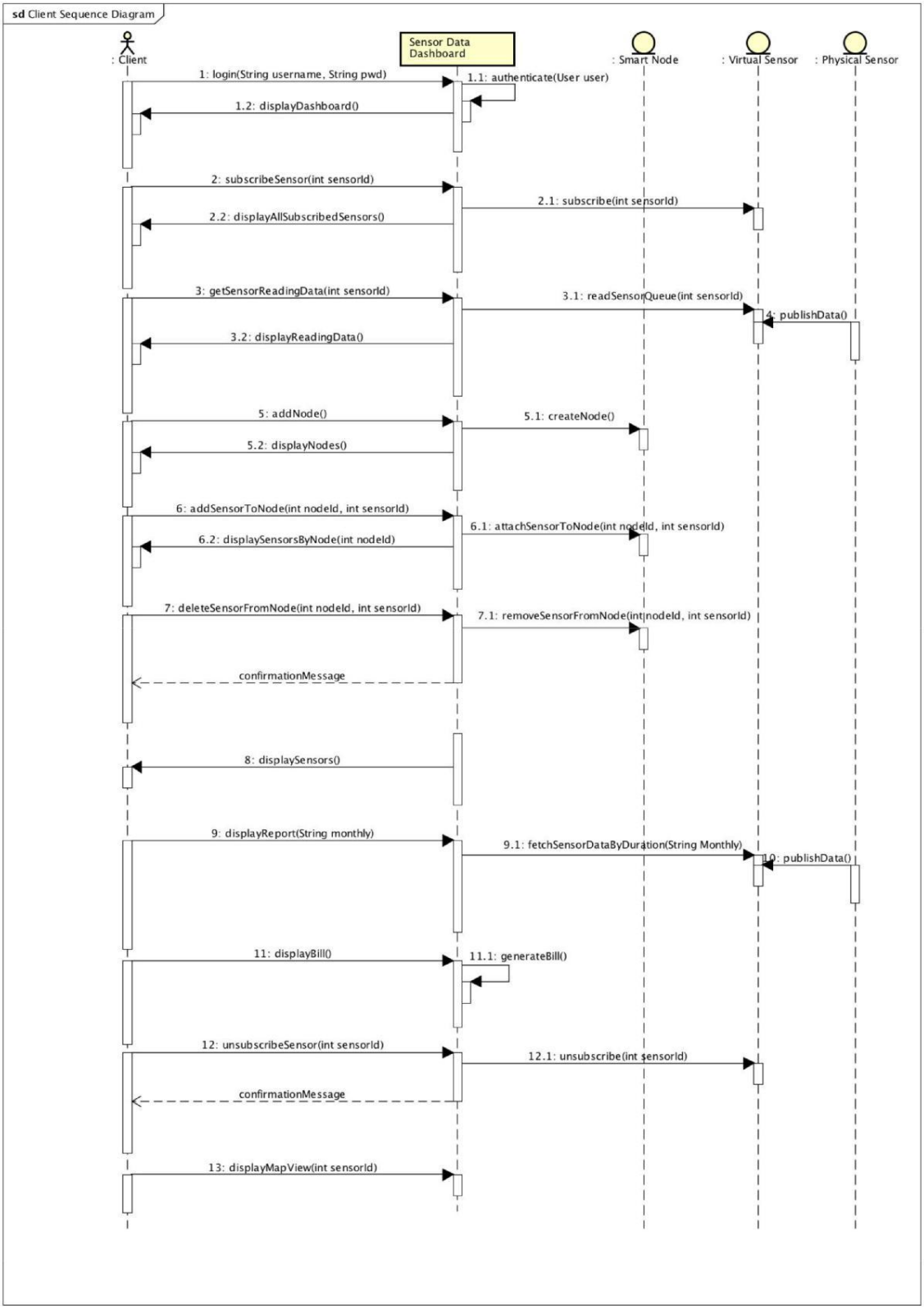


Fig 5.2.1 Client User Sequence Flow

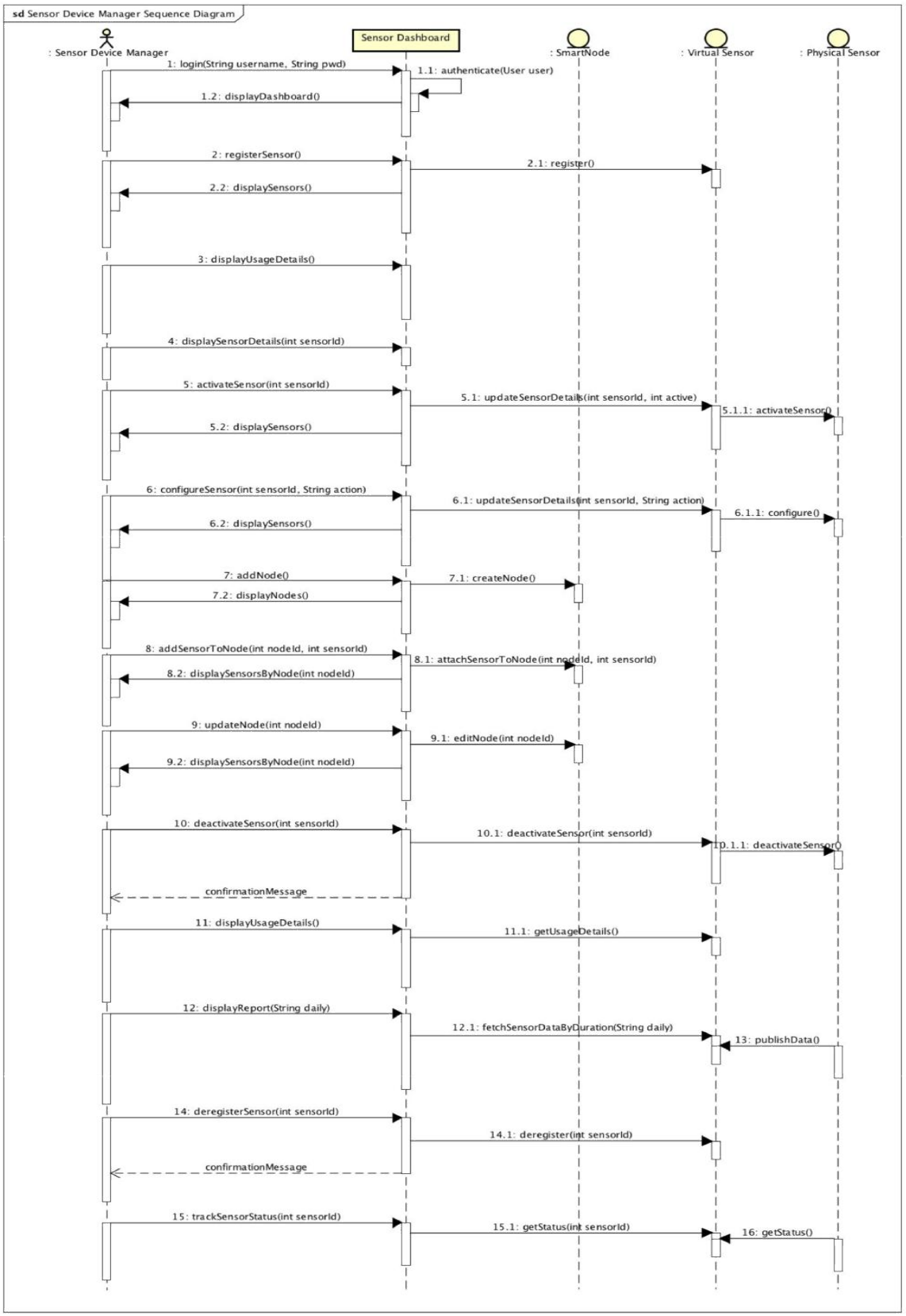


Fig 5.2.2 Sensor Device Manager Sequence Flow

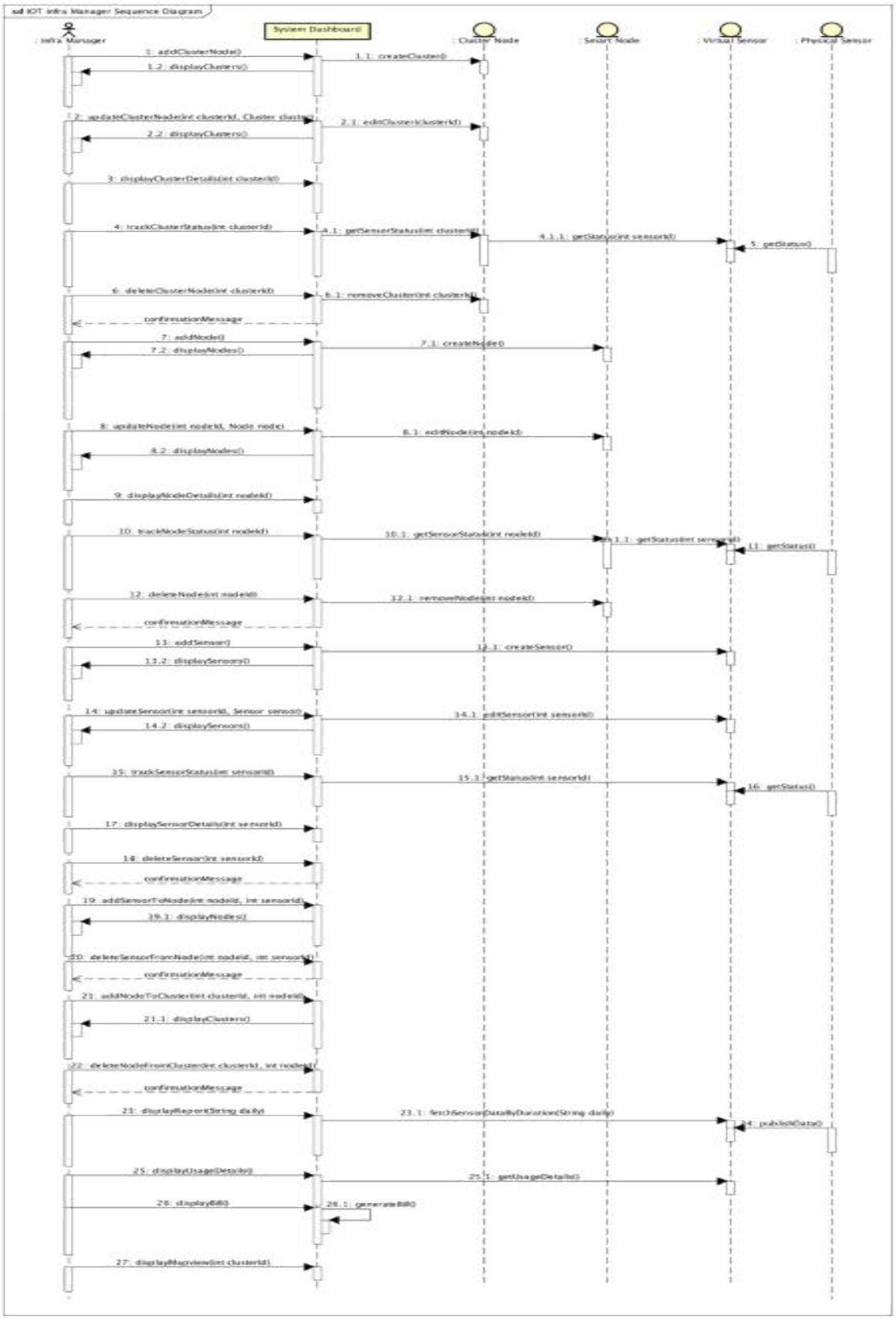


Fig 5.2.3 IoT Infra Manager Sequence Flow

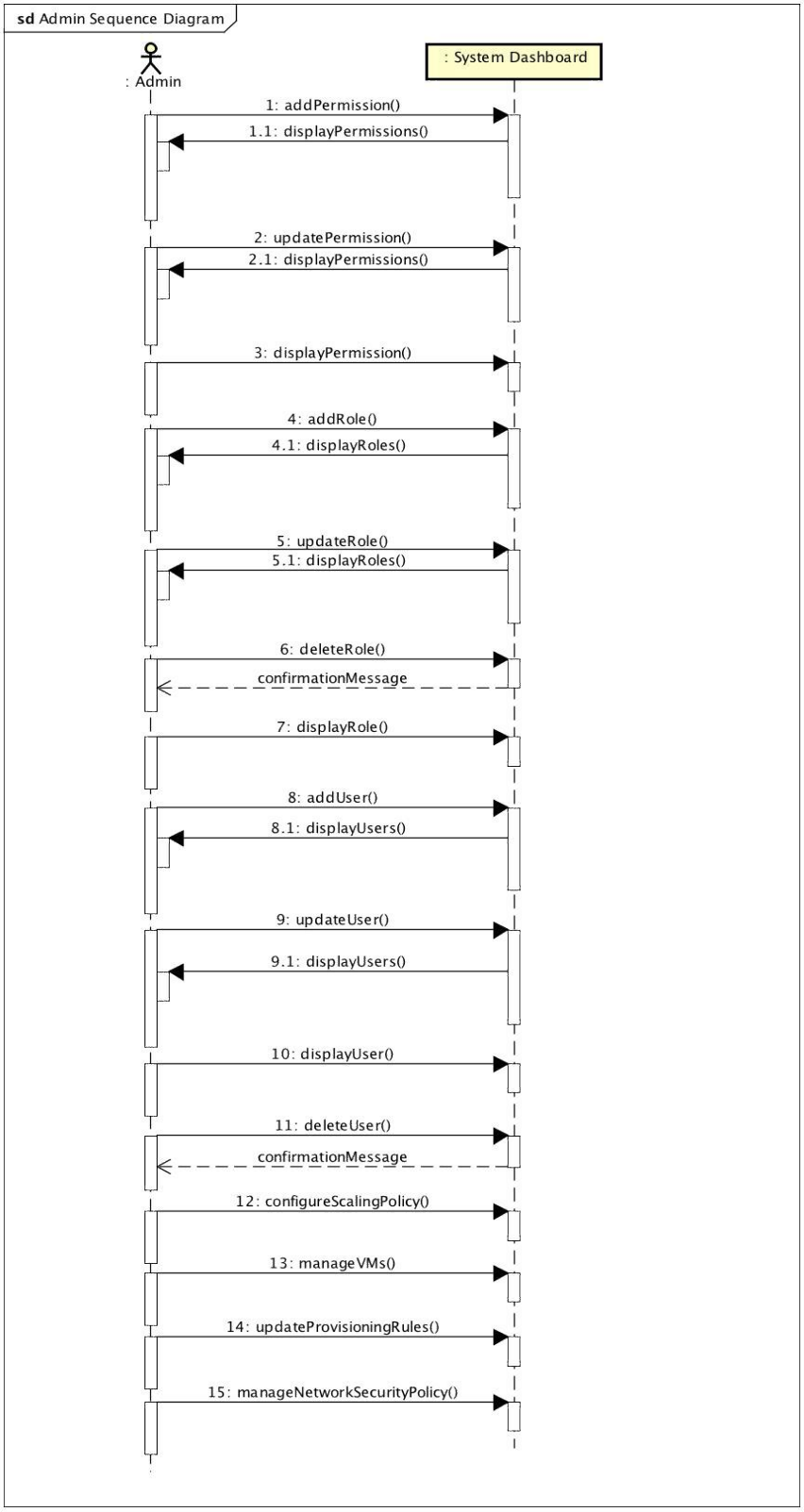


Fig 5.2.4 Admin Sequence Flow

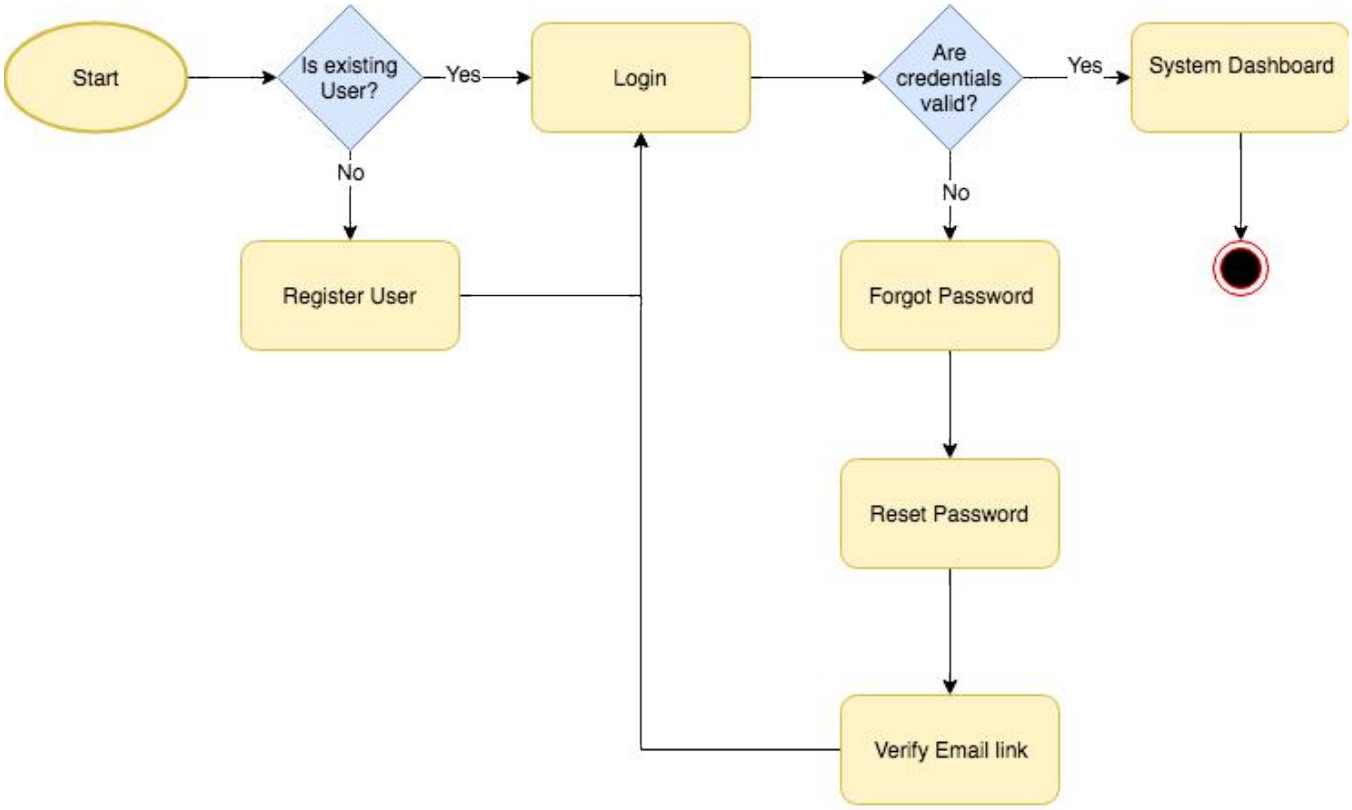
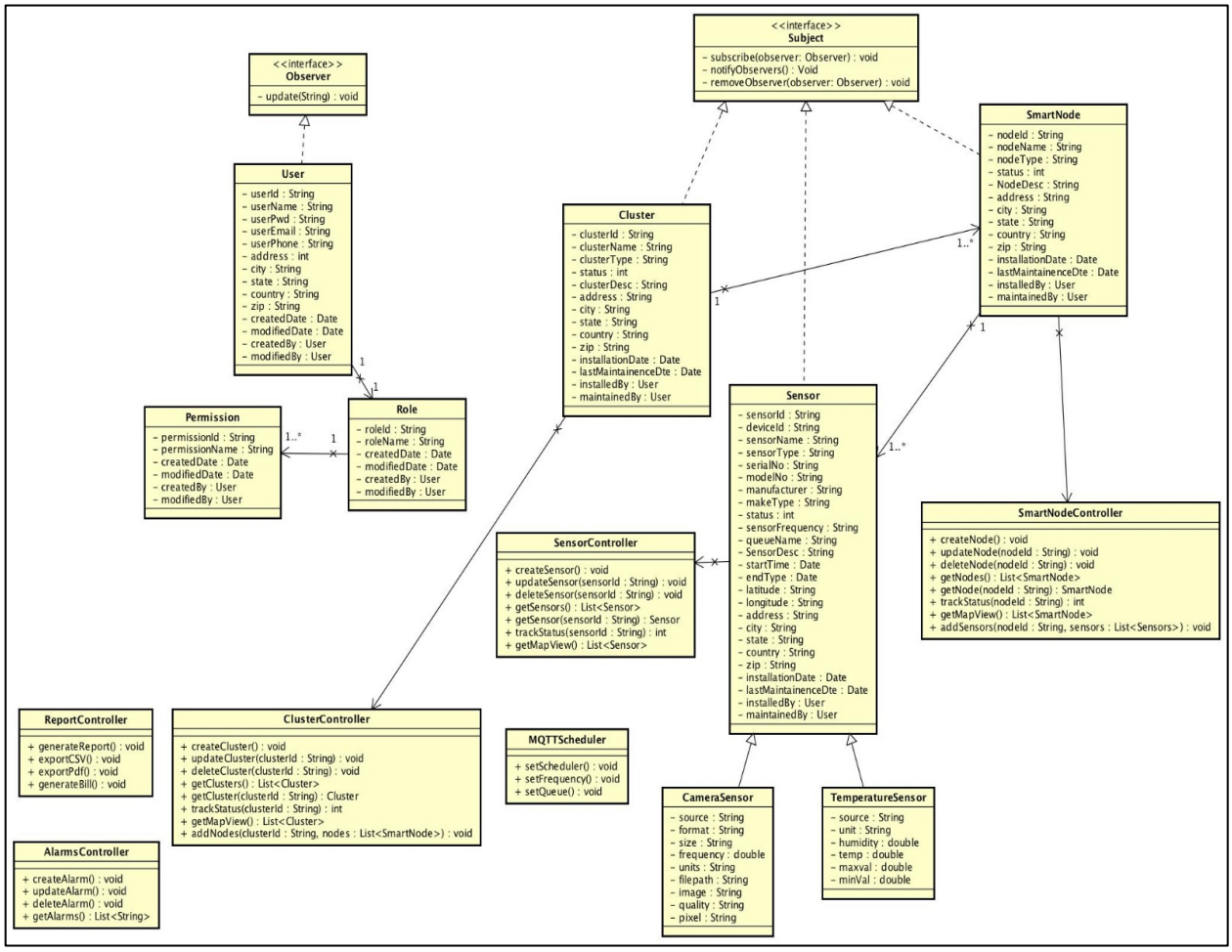


Fig 5.2.5 User Authentication Process

5.3 Class Diagrams



6. Large-Scale IOT Data Design and Implementation

This section describes two kinds of databases schemas to support IoT infrastructure and sensor data.

6.1 SQL DB Schema

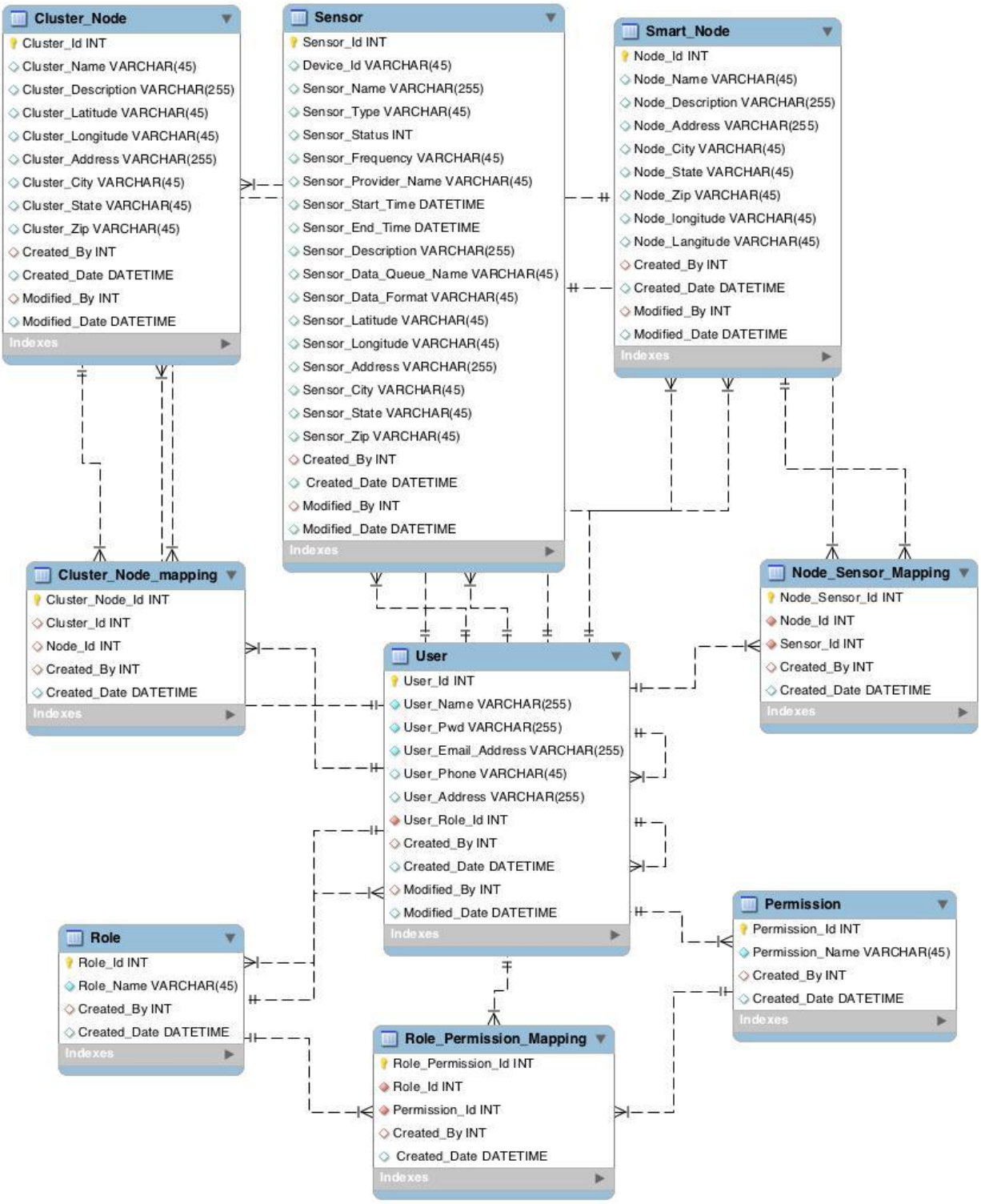


Fig 6.1 SQL DB Schema: system and sensor metadata

6.2 NOSQL DB Schema

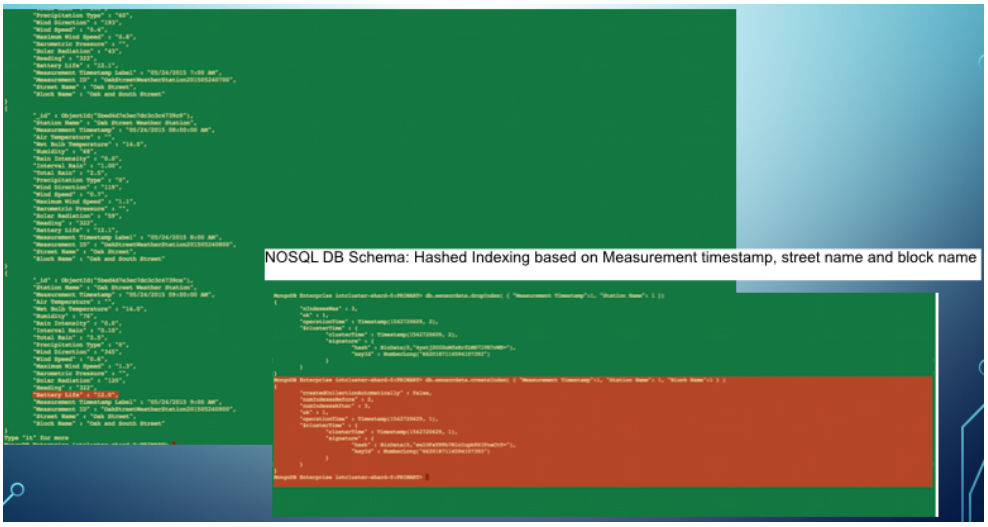
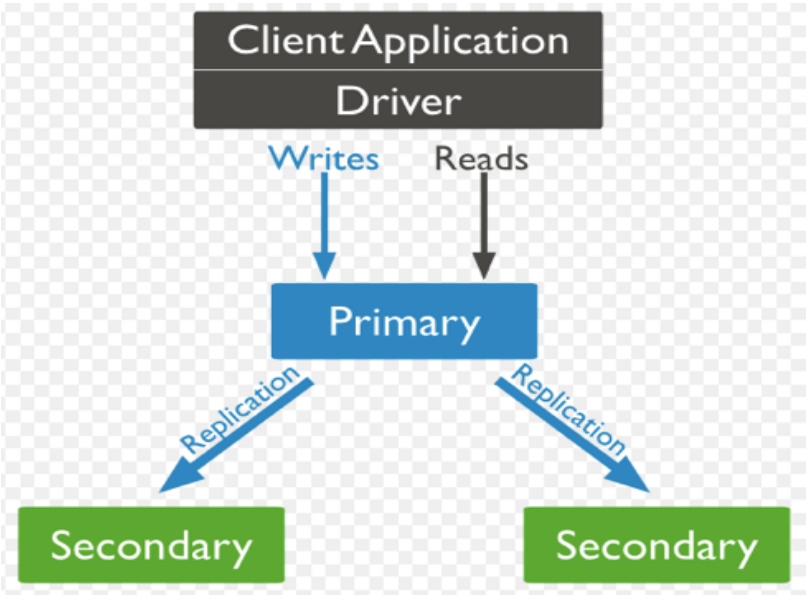


Fig 6.2 NOSQL DB Schema: Sensor IOT Data

Data can be shared across multiple machines using Sharding. MongoDB uses sharding technique. It helps in supporting deployments with very large data sets and high processing operations. A single server cannot support databases systems with large data sets or high processing applications. Sharding helps in capacitating this.

Horizontal Scaling divides the data and the load over multiple servers. It means, additional servers are added to increase the capacity as needed. MongoDB supports horizontal scaling through sharding. Each machine will handle a subset of the overall workload. It helps in providing better efficiency than a single high-speed high-capacity server. Additional servers can be added to increase the capacity of the deployment. This results in lower costs and lower hardware costs for a single machine. However, complexity in infrastructure and maintenance for the deployment is increased.



7. Technology Selection, usage and validation environment

Technologies to be used:

Jquery

We would be using Jquery for DOM Manipulation.

HTML5, CSS and BootStrap

HTML5 and CSS would be used to develop the front-end of our application.

Spring Core & MVC

We shall be using basic Dependency injection concept using Spring core framework. The application shall design REST-ful API using spring MVC.

Amazon cloud Web Services:

Our design suggests the use of Amazon cloud, which offers some of the most popular and mature cloud-based products and services in the cloud marketplace today. Apart from some of the popular cloud features such as elastic computation, auto scaling and dynamic provisioning; AWS offers one of the most transparent ‘pay as you go’ pricing models. Our system would also leverage from various offerings from the AWS IOT platform such as AWS Lambda, kinesis, QuickSight and many more.

MongoDB

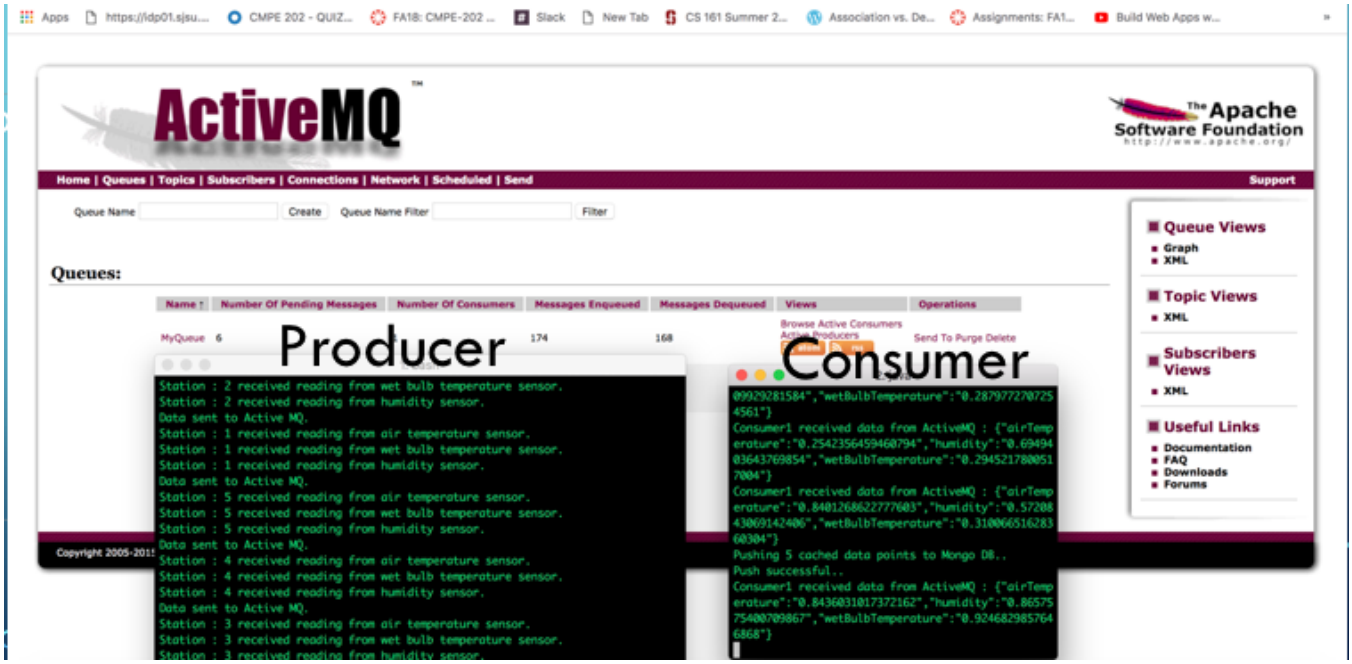
Our design suggests the use of document based NoSQL store for the sensor data persistence. MongoDB Atlas created in cloud fits our requirement as it is one of the popular choices amongst several document store DB engines. It uses optimised querying technique for accessing big data and provides second indexing based on attributes other than primary keys. This technique makes our big data queries fast and efficient. Additionally, BASE (Basically Available, Soft-State and Eventually consistent) property of the NoSQL DBs avoid the unnecessary consistency overhead of SQL DBs and fit our write heavy requirement due to continuous sensor data writes.
MongoDB maintains min of 3 replica sets for each DB node to ensure fault tolerance and also supports horizontal scaling.

Validation Environment

Amazon Cloud platform is to be used for testing of the proposed software against a simulated sensor data.

8. Algorithms used

8.1Pub/Sub message broker



Real-time sensor data simulation - Pub/Sub message broker implementation

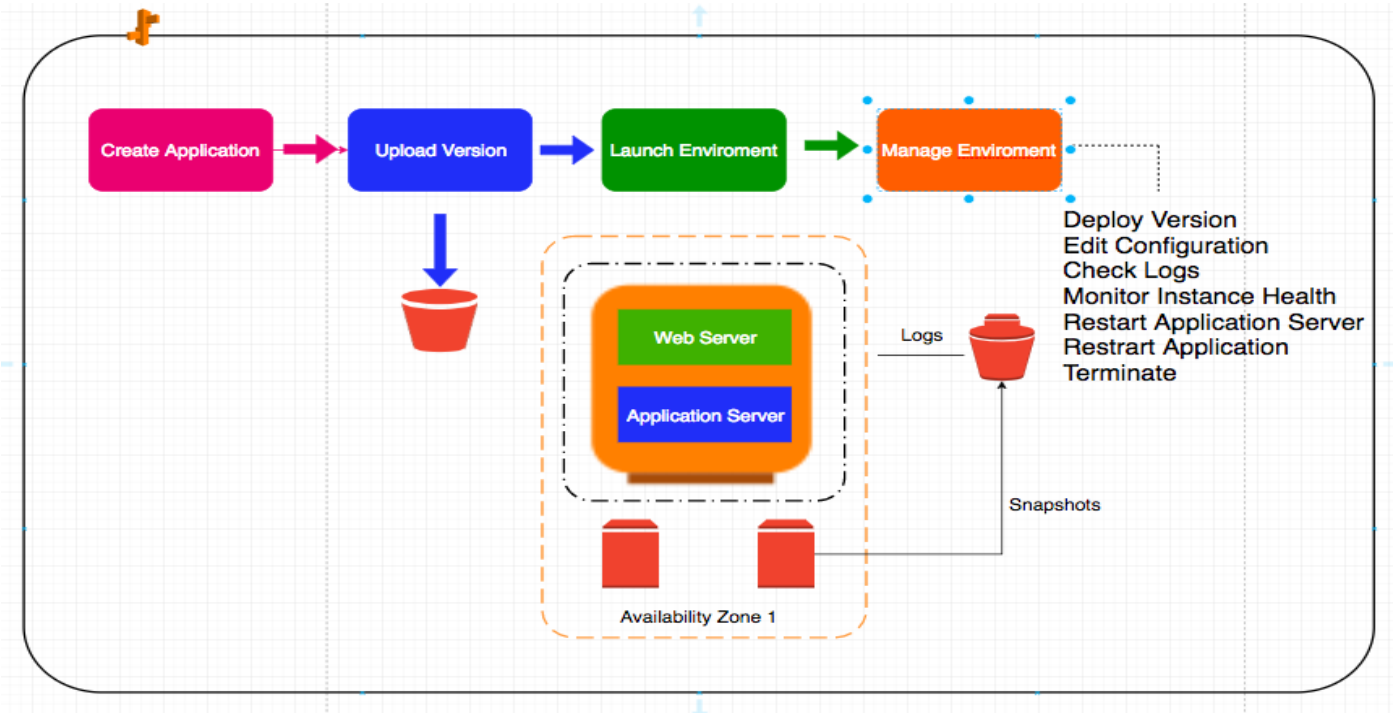
- Created algorithm to produce and consume messages using AWS ActiveMQ
- Producer generates simulated sensor reading data and sends to the queue
- Consumer reads the data in intervals from the queue and pushes into MongoDB

Apache ActiveMQ is an open source message broker. It is written in Java and works with full Java Message Service (JMS) client. It provides a feature to allow communication between more than one client or server. Java via JMS 1.1 and other programming language clients are supported by ActiveMQ. It has the ability to use any database as a JMS persistence provider. Along with this, it can also use virtual memory, cache, and journal persistency for communication. Computer clustering also enables communication to occur.

9. IOT-based cloud system design and services

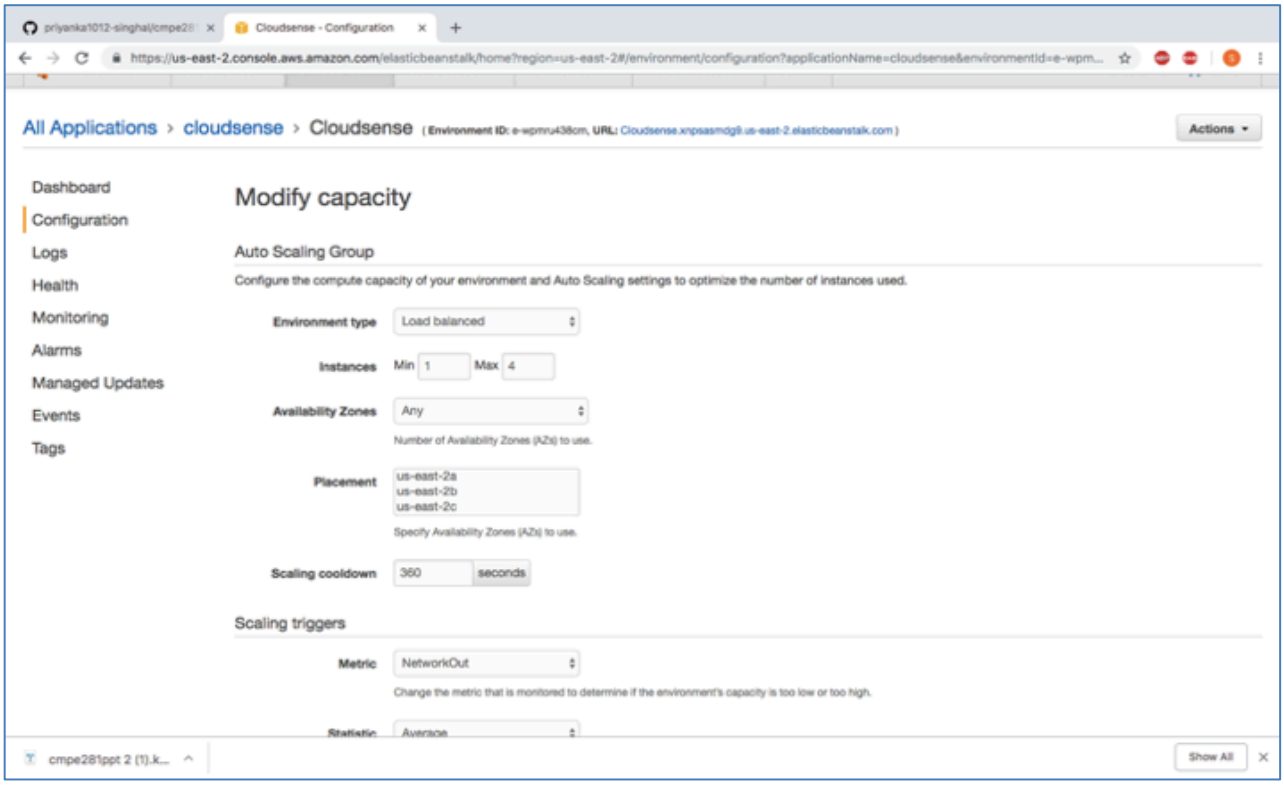
9.1 Elastic Bean Stalk

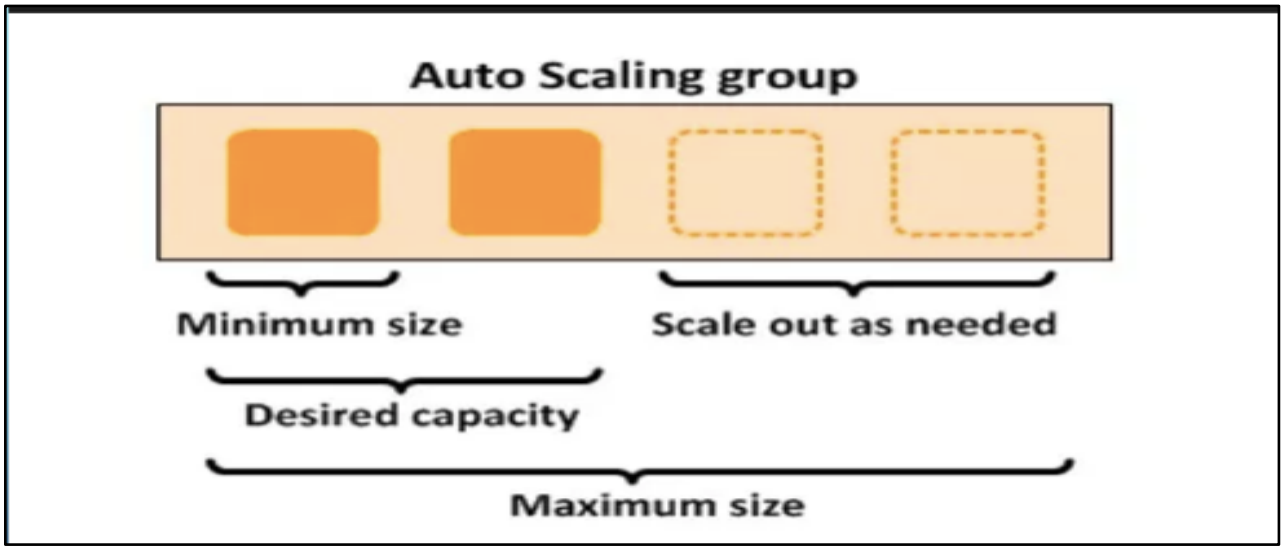
We are using AWS PaaS - Amazon Elastic Beanstalk to deploy our SaaS application. SaaS application is created with REST based micro-services using Spring MVC, Security & JS. With AWS Elastic Beanstalk we could deploy our application with configuration we needed, we could monitor instance health, check logs, restart application and application server.



9.2. Autoscaling

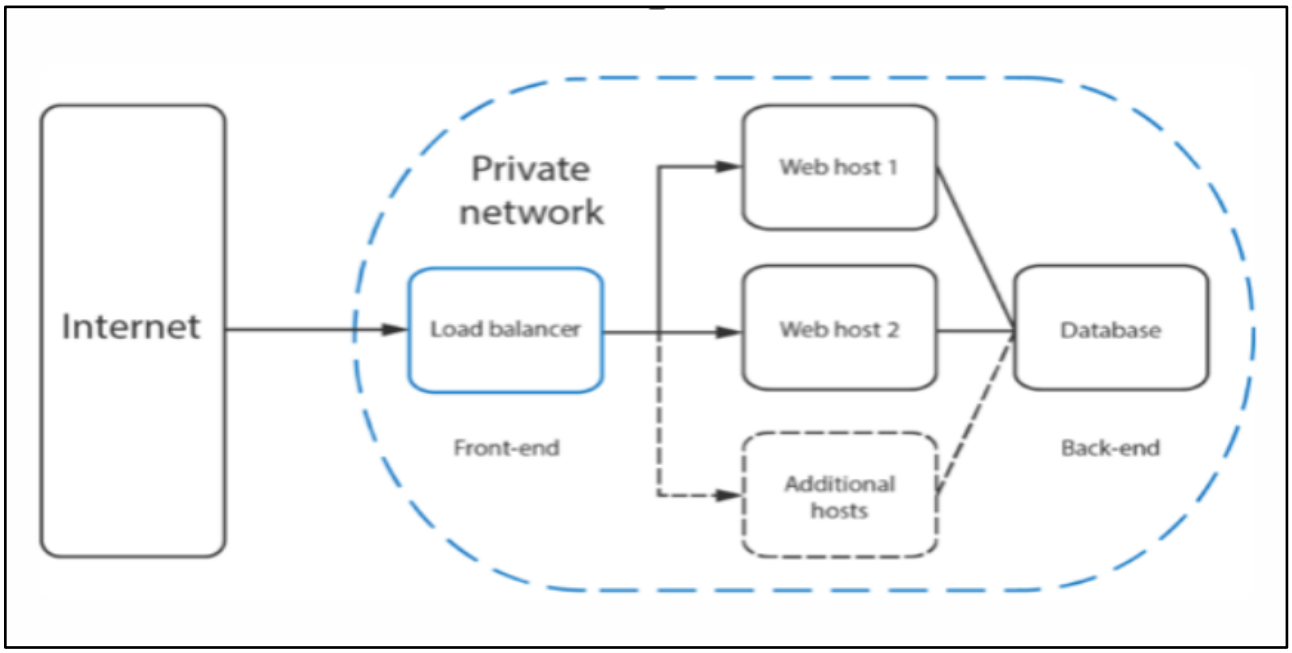
We are using AWS Auto Scaling for better performance. The application is monitored automatically, and the capacity is adjusted simultaneously. It helps to achieve better performance at the lower costs. Multiple resources can be scaled automatically across multiple services easily with the help of AWS Auto Scaling.
Amazon EC2 instances can perform up scaling (create more instances) or down scaling (reduce the running instances) automatically based on the auto scaling policy. Auto scaling policy is configured by the user. It contains the configuration to scale the number of VM instances upto a specified limit value.





9.3 System Load Balance Design and Implementation

Load balancing is used across multiple server instances. It helps to increase productivity, reducing time consumption, increases usage of resources. It also helps to make the application more fault-tolerant. We are using Nginx load balancer. Nginx is very efficient HTTP load balancer. It helps to share traffic among many application servers. It increases performance, make the application more scalable and reliable.



Round-robin technique is used to distribute requests. They are sent to application servers in round-robin manner.

```
# Define which servers to include in the load balancing scheme.
# It's best to use the servers' private IPs for better performance and
security.
# You can find the private IPs at your UpCloud Control Panel Network
section.

upstream backend {
    server 54.243.15.42;
    server 18.212.242.223;
}

# This server accepts all traffic to port 80 and passes it to the
upstream.
# Notice that the upstream name and the proxy_pass need to match.

server {
    listen 80;

    location / {
        proxy_pass http://backend;
    }
}
```

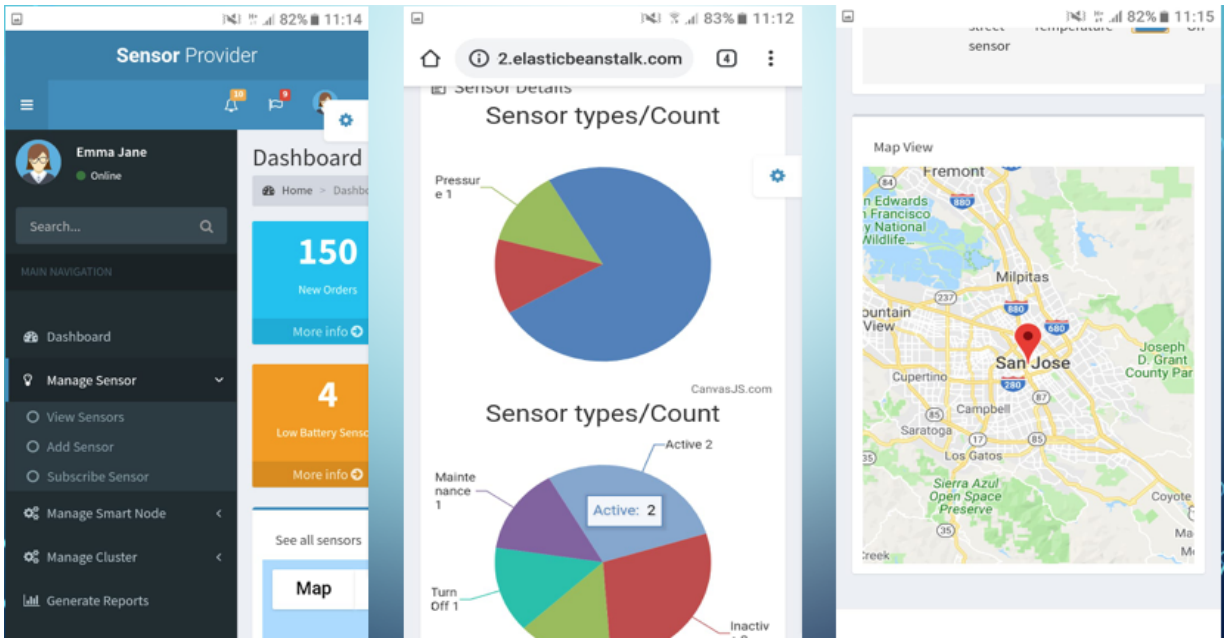
There are 2 more techniques which can be used for load-balancing in nginx:

least-connected: The next request will be served by Server with the minimum number of active connections

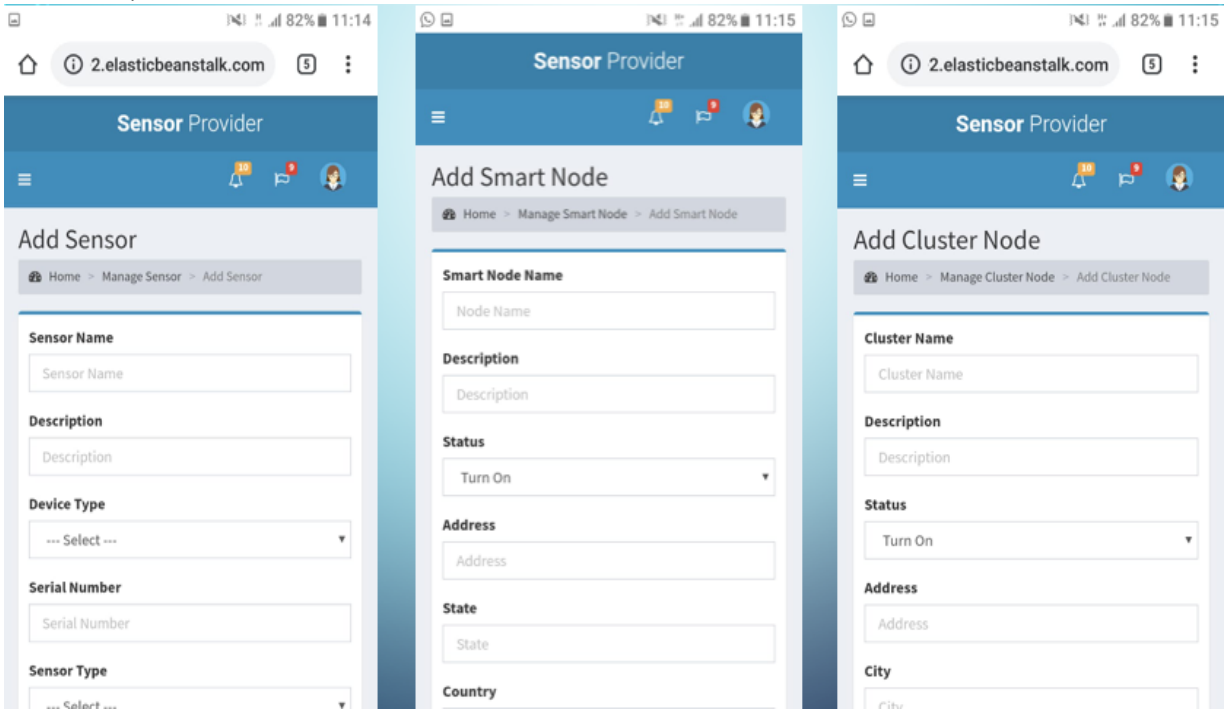
serveip-hash: Which server should be selected for the next request will be determined by hash-function.

10. System GUI Design and Implementation

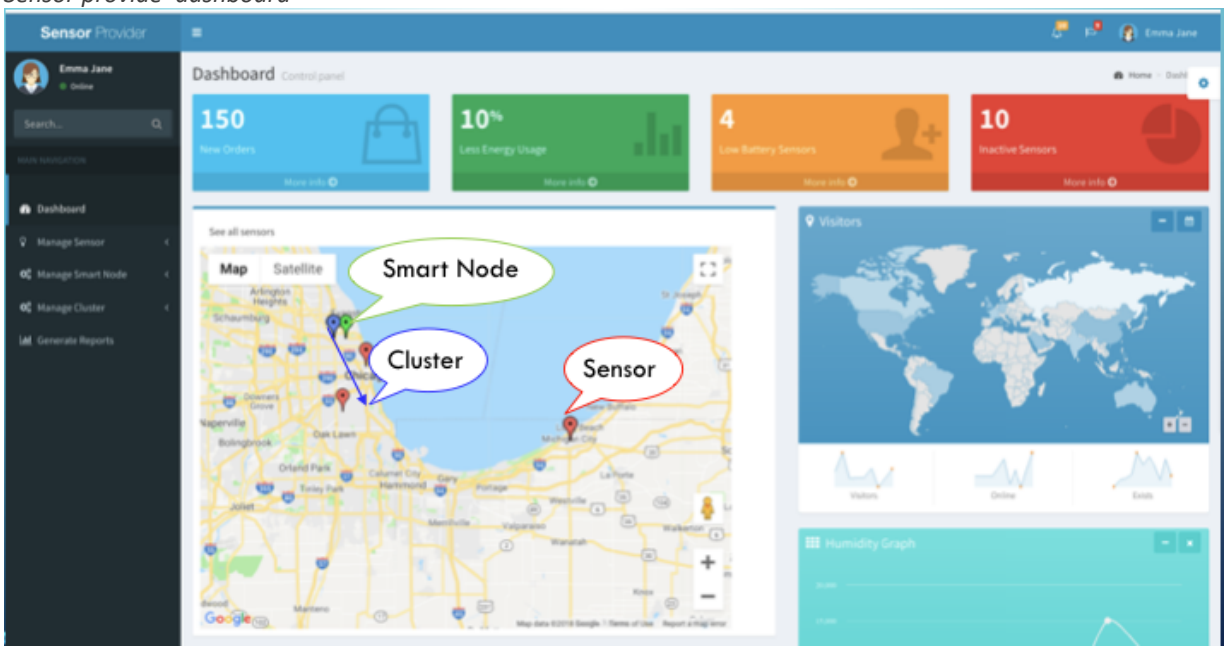
Sensor Provides dashboard



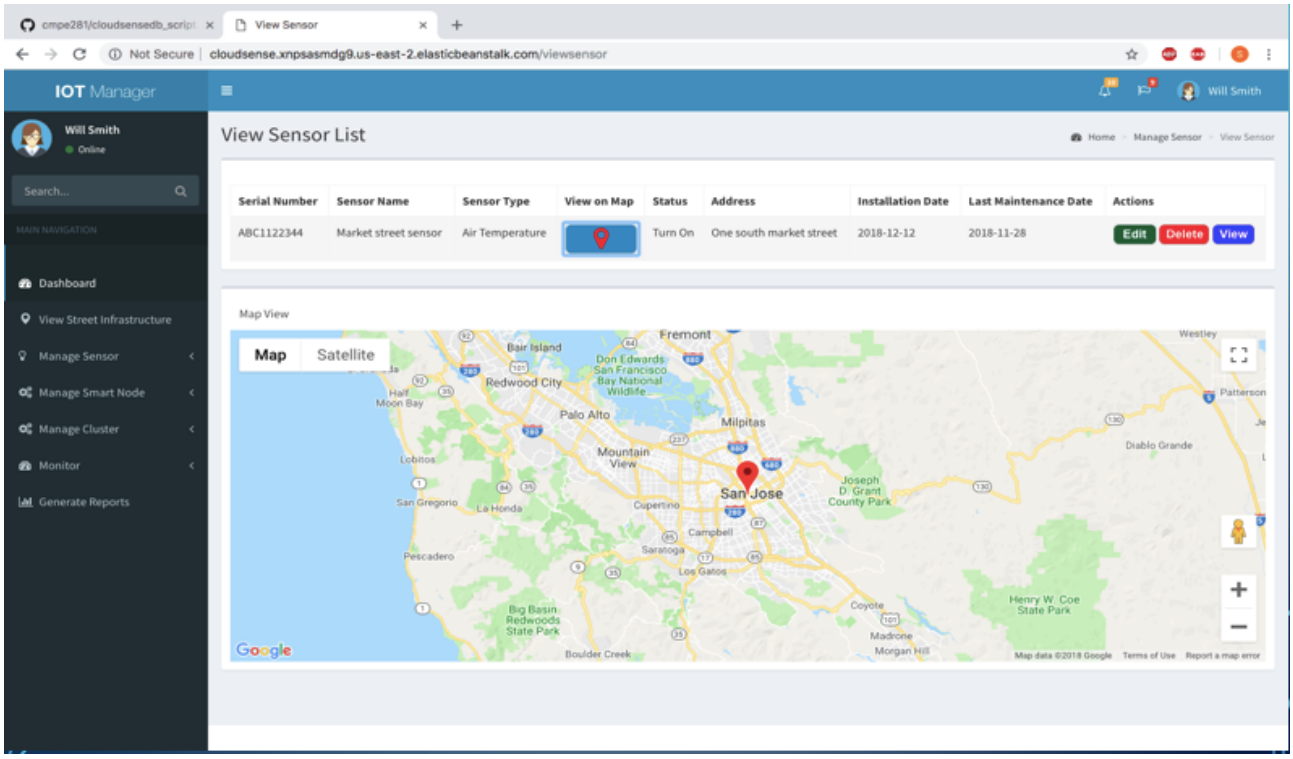
Add sensors, cluster and smart node screen:



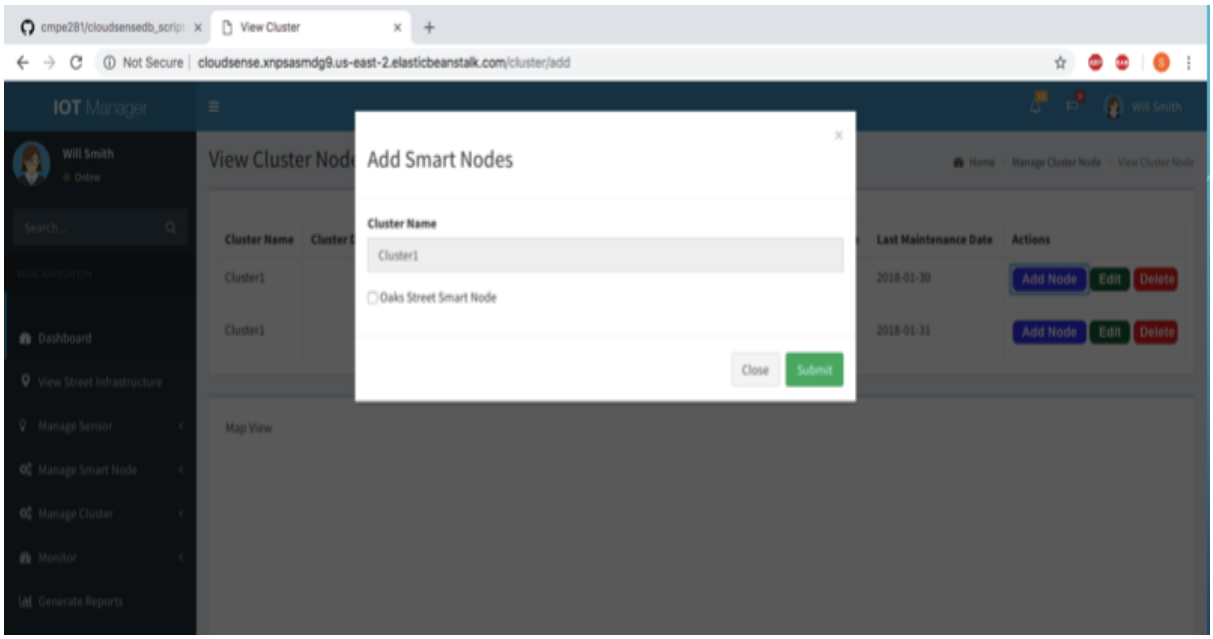
Sensor provide' dashboard



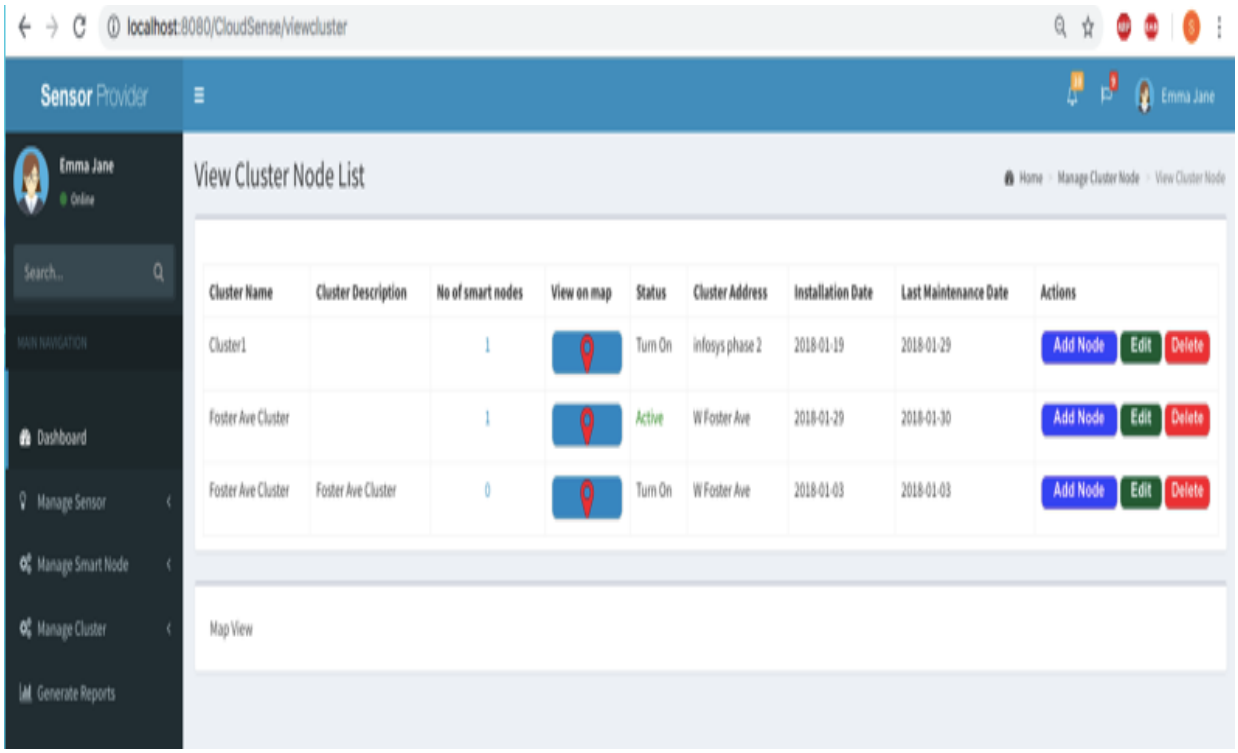
View sensors screen:



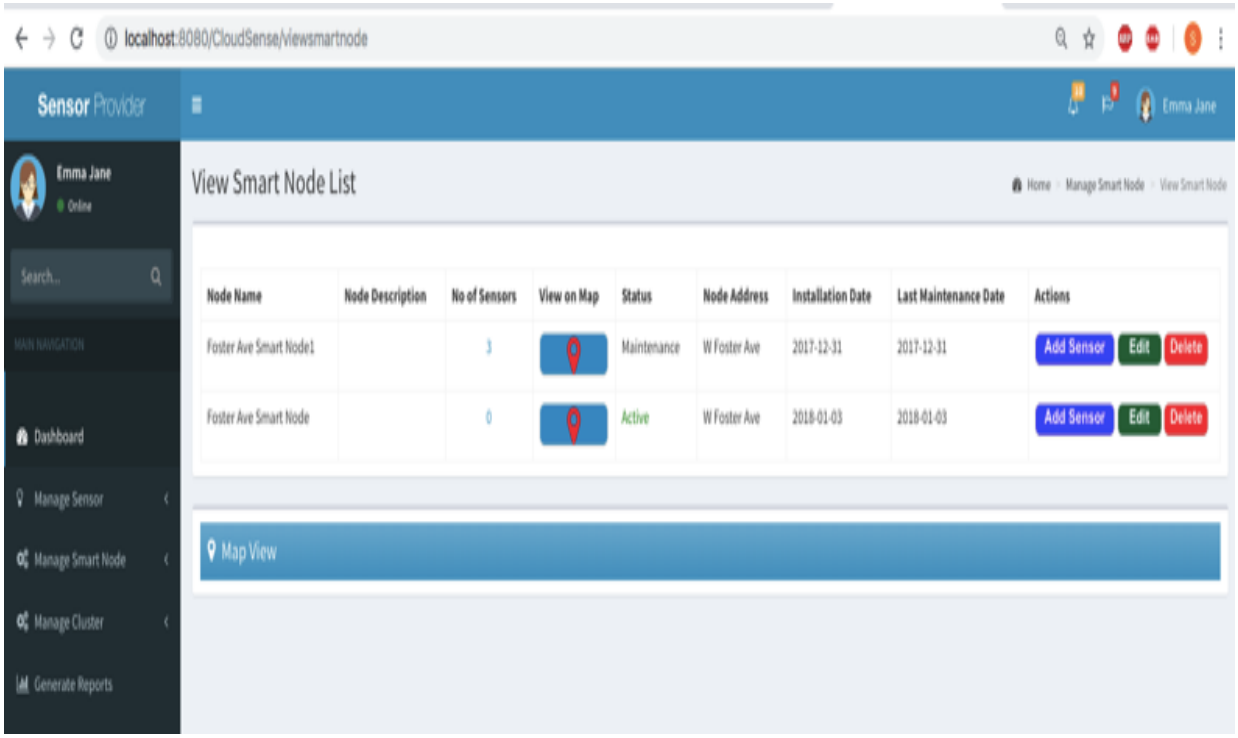
Add sensors to smart node screen:



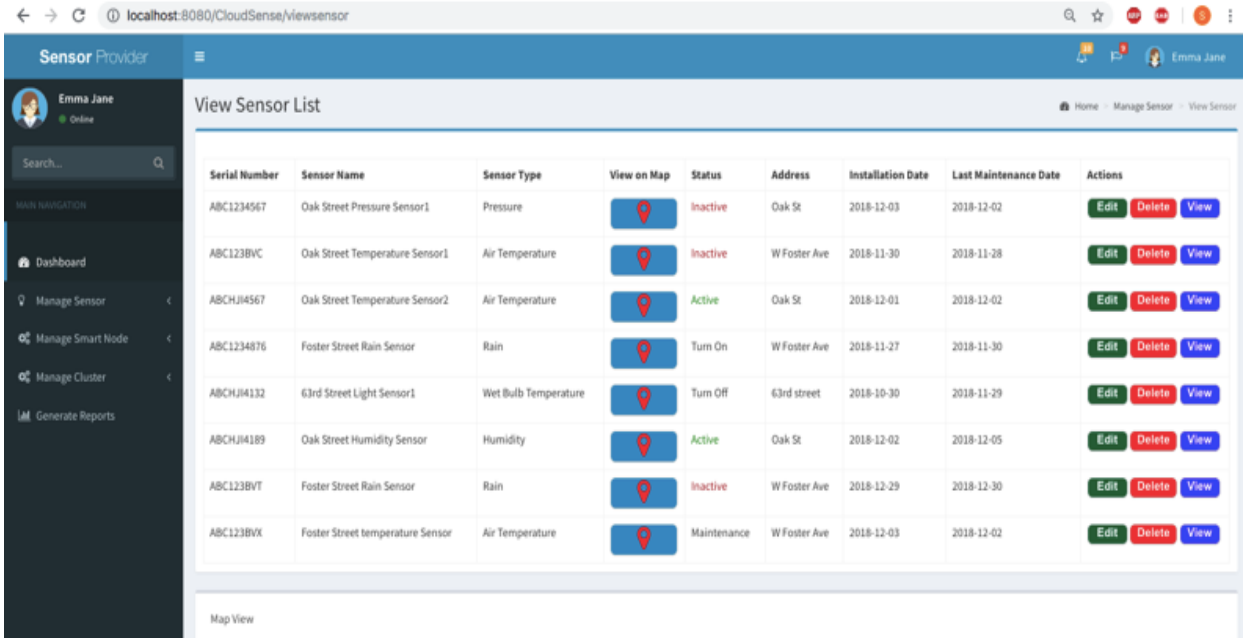
View Cluster node screen:



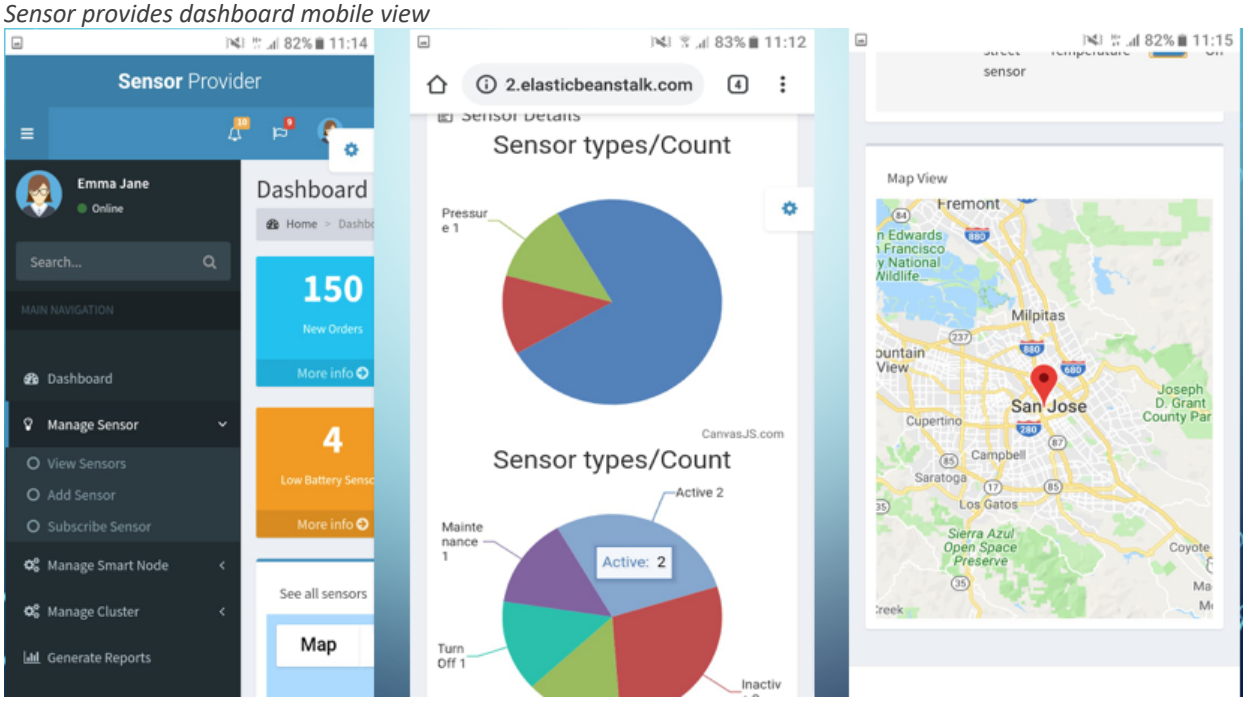
View smart node screen:



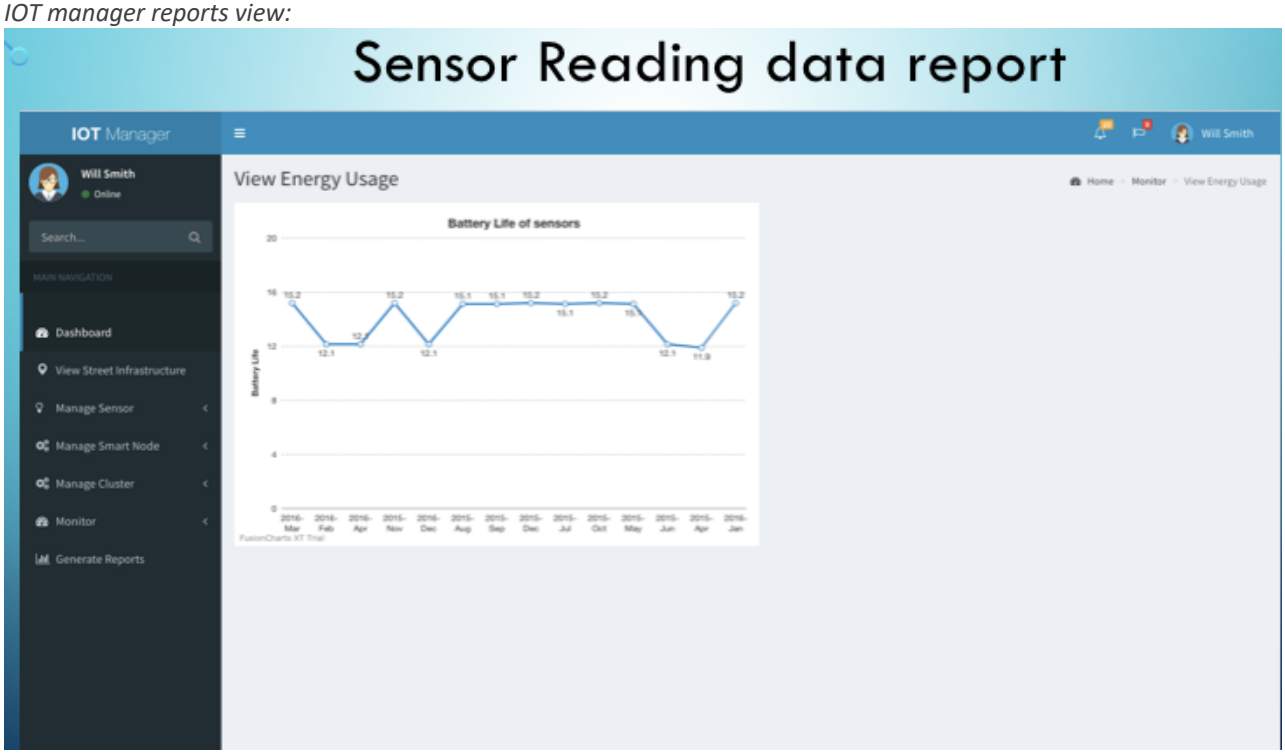
View sensors screen:

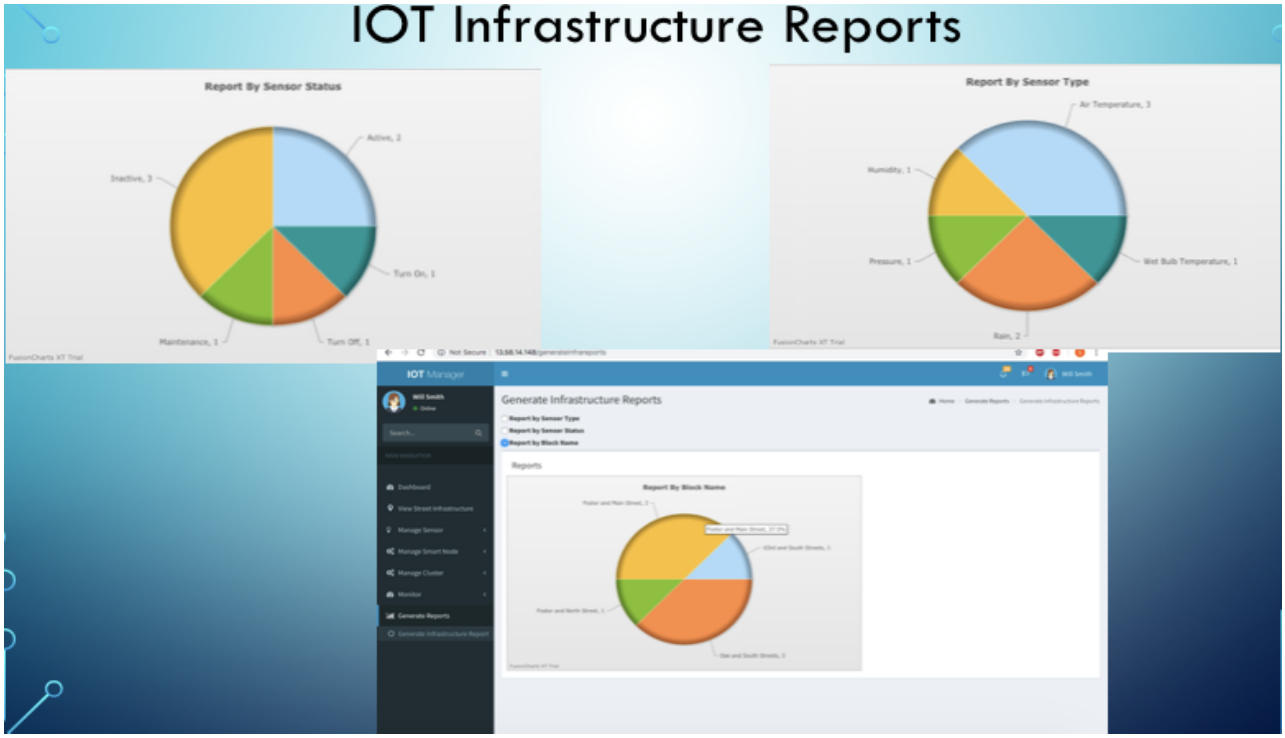


Sensor provides dashboard mobile view

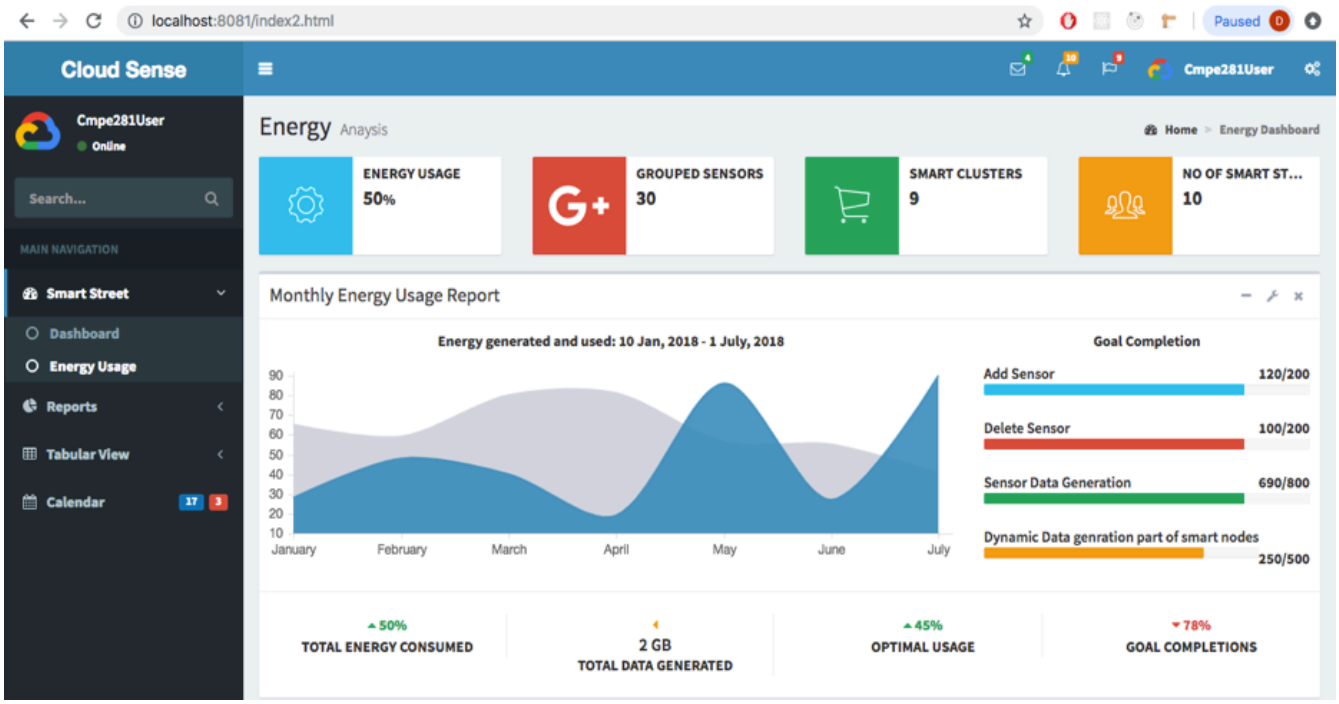


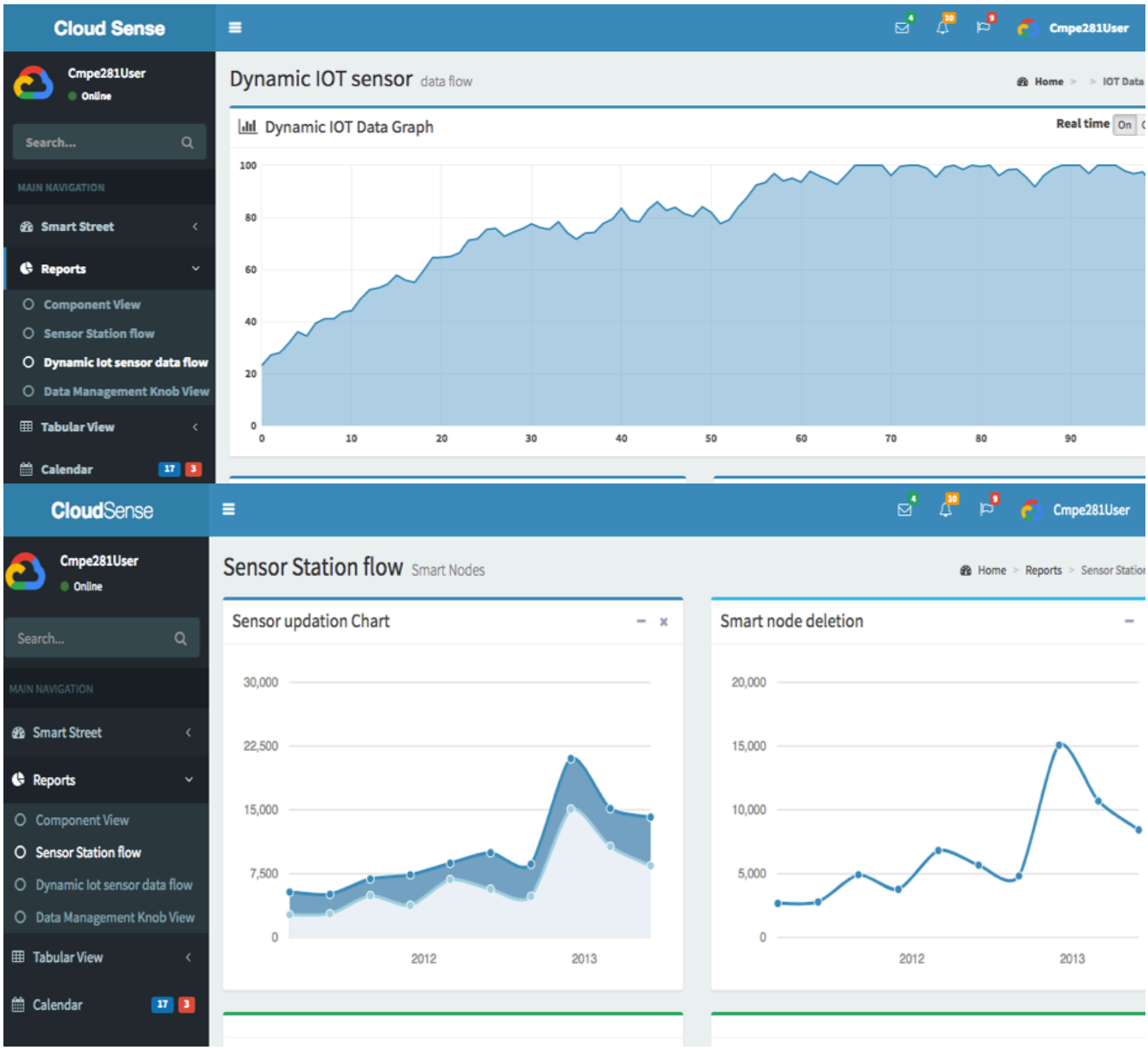
IOT manager reports view:

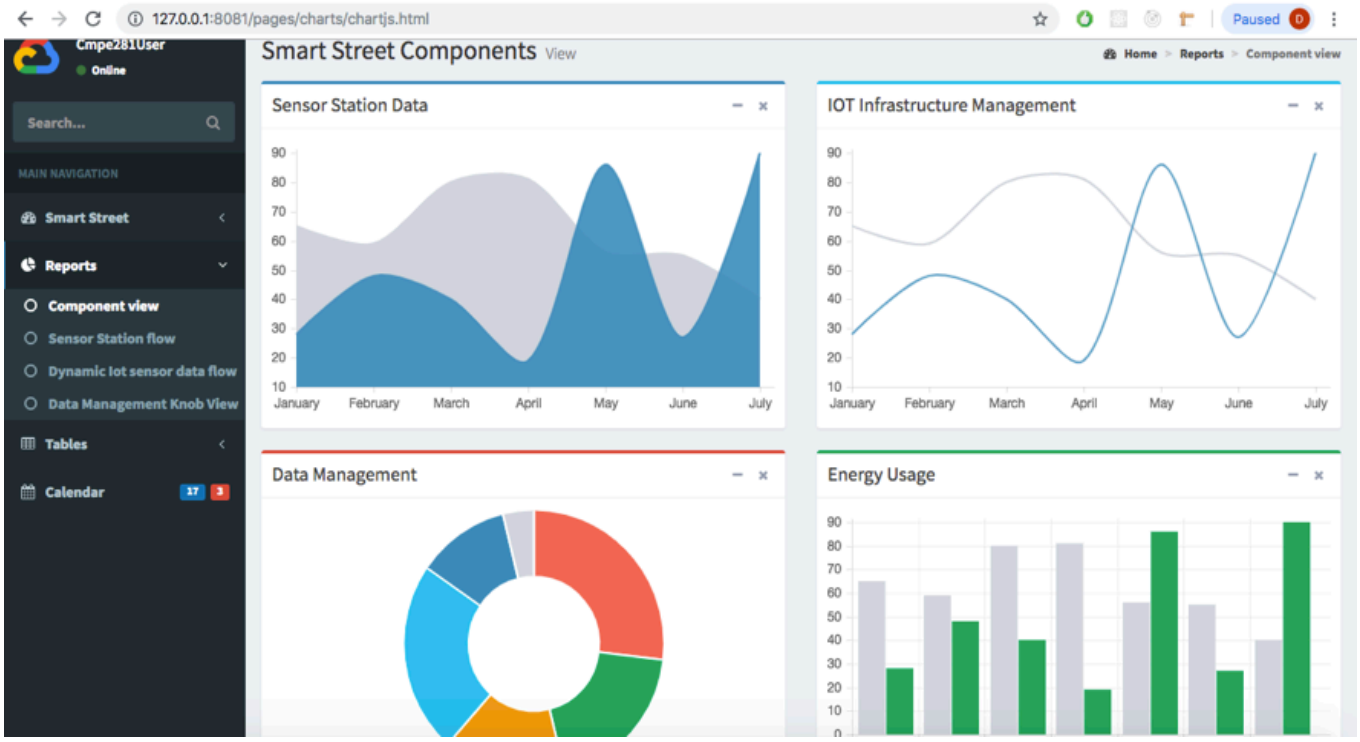


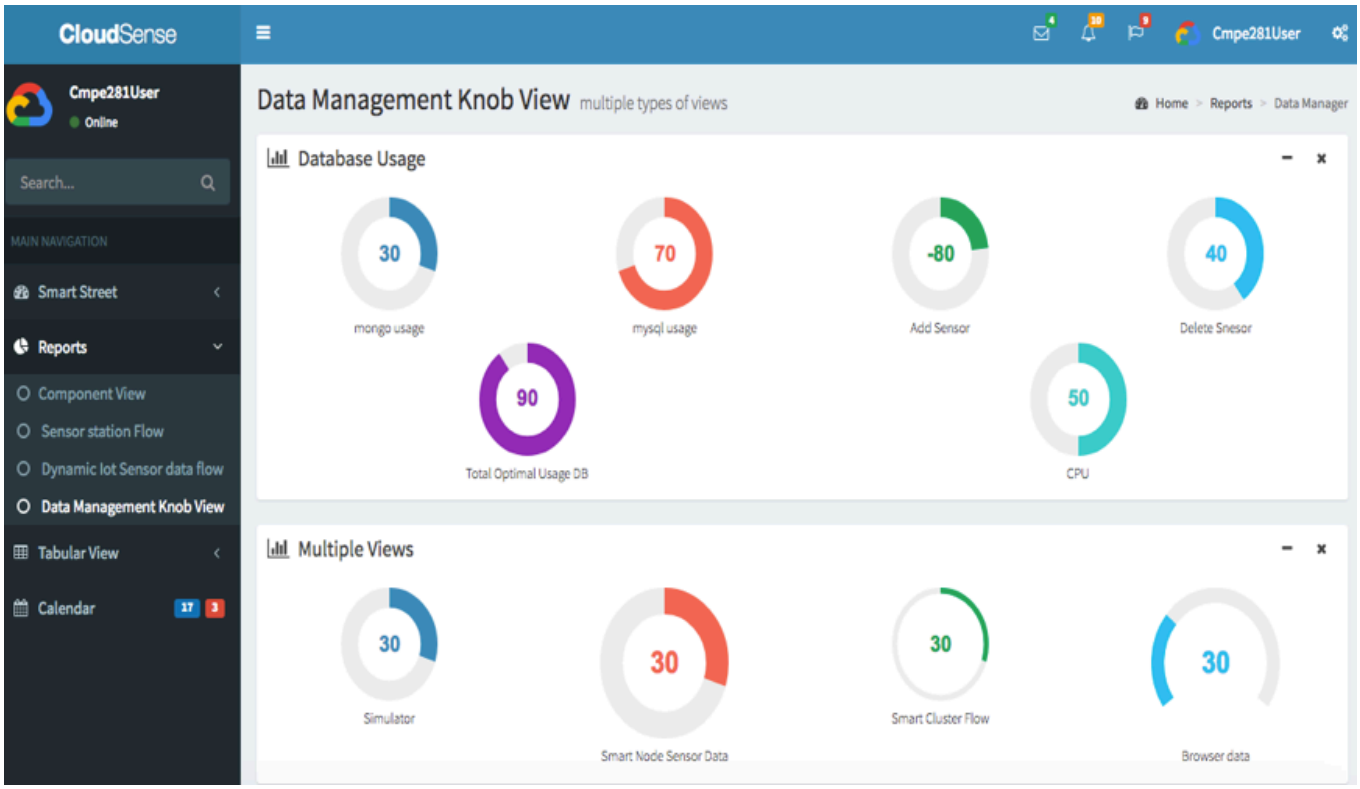


System User dashboard screens:




Cloud Sense
Cmpe281User
Online
Search...
MAIN NAVIGATION
Smart Street
Reports
Component View
Sensor Station flow
Dynamic Iot sensor data flow
Data Management Knob View
Tabular View
Calendar
Dynamic IOT sensor data flow
Dynamic IOT Data Graph
Real time
CloudSense
Sensor Station flow Smart Nodes
Sensor updation Chart
Smart node deletion
30,000
22,500
15,000
7,500
0
20,000
10,000
5,000
2012
2013





10. Conclusion

This document discusses the project to develop, implement, and validate an IOT-based cloud infrastructure system as a SaaS for Smart Streets in a smart city. Huge amounts of energy

wastage and costs associated with lighting applications has driven the need to address both budget issues and environmental concerns.