CMPE 281 - Cloud Technologies

Smart Street Infrastructure Management on Cloud

Term Project Design Document

Professor: Dr.Jerry Gao,
Department of Computer Engineering
(jerry.gao@sjsu.edu)

Team Members:

1. Name: Vishwas Adiga
   Email: vishwas.adiga@sjsu.edu
   ID: 011822473

2. Name: Kaustubh Hassan Narasimhan
   Email: kaustubh.hassannarasimhan@sjsu.edu
   ID: 013727129

3. Name: Madhu Prasanna Kalluraya
   Email: madhuprasanna.kalluraya@sjsu.edu
   ID: 013708071

4. Name: Prashanth Rajasekar
   Email: prashanth.rajasekar@sjsu.edu
   ID: 011822460

Contents

Abstract

Street lighting is one of the most important aspects of a city's infrastructure, which illuminate a city's streets during dark hours of the day. Conventional street lighting systems in areas with a low frequency of passersby are online most of the night without purpose. The consequence is that a large amount of power is wasted meaninglessly. Smart street lights is a project on intelligent illumination control of street lights to optimize the problem of power consumption and illumination of the streets, late in the night.

1) Requirement Analysis

1a) Introduction and Scope

Making cities smarter helps improve city services and increases citizens' quality of life. Information and communication technologies (ICT) are fundamental for progressing towards smarter city environments. Smart City software platforms potentially support the development and integration of Smart City applications.

Figure (a)

This project is designed to develop, implement, and validate an IOT-based cloud infrastructure system as a Software as a Service(SaaS**)** for Smart Streets in a smart city. As shown in Figure (a) above, a **smart street** consists of several **smart nodes**. Each node is installed on a utility pole on a street. Further, each node is equipped with a few sensors. A **smart cluster** controller among smart nodes will be used to control few such connected smart nodes and supports communication with the back-end server in order to send the collected sensor data for all the nodes. In addition, each smart node has wireless communication capability, which supports node-to-cluster communication. Each cluster node has internet connectivity.

1b) Implementation

The System shall include the following 4 parts -

1) **Simulated sensor station –** This component simulates a smart node. Each node includes 4 Sensors - Motion Sensor, Light Sensor, Temperature Sensor and Rain

Gauge. 5 Nodes together form a Cluster Node. The nodes themselves neither communicate with each other nor with the Server(Cloud). They only send the data of their sensors to the Cluster Nodes. Every Cluster Node collects data from 5 nodes and sends it to the Server(Cloud). The Cluster Nodes in turn receive communication from the Server, based on which they relay information to the nodes to act accordingly.

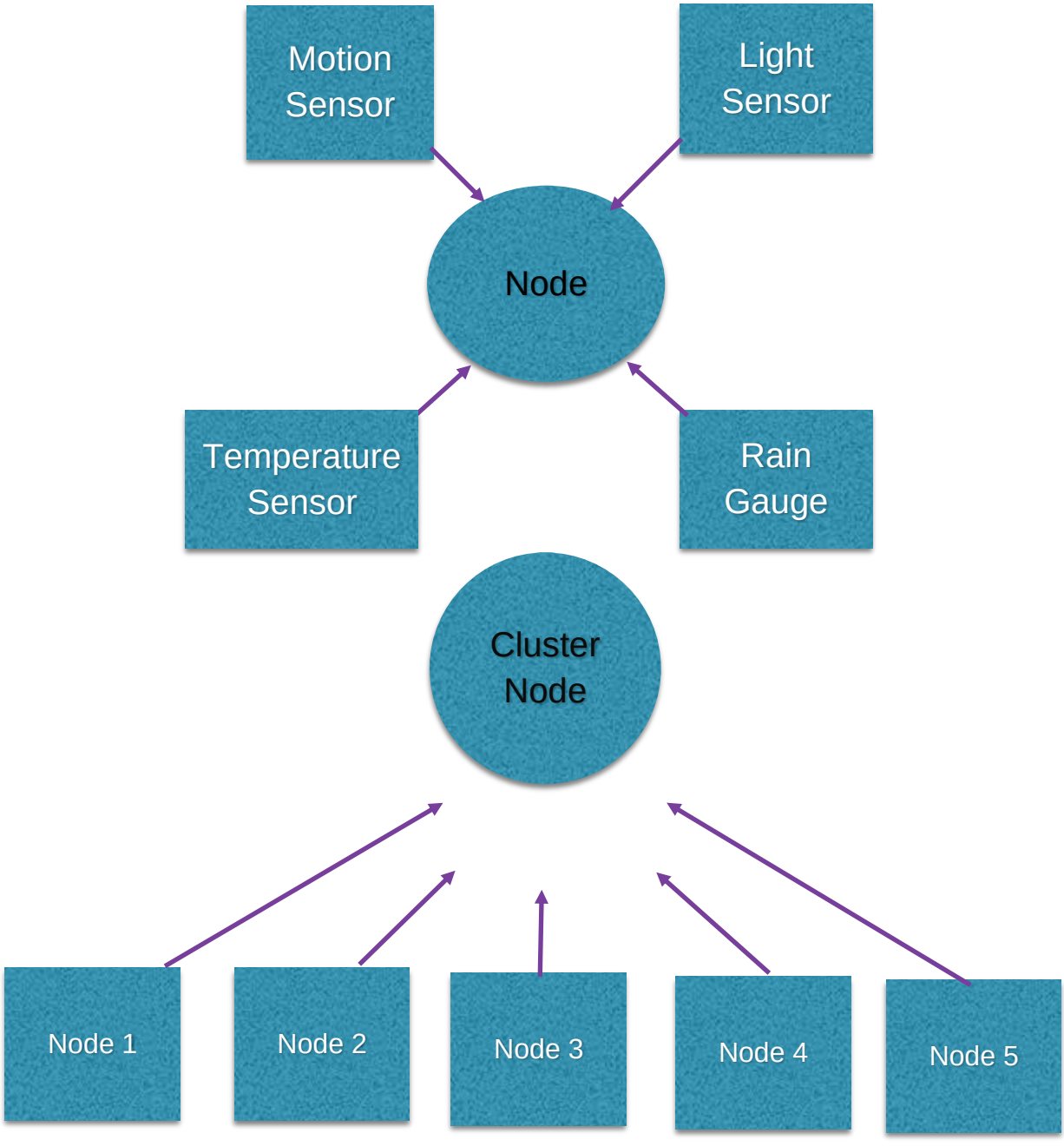


Figure (b)

Every Sensor has one of the following states - **Turn-On, Active, Inactive, Turn-Off** or **Maintenance**.

2. **IOT infrastructure manager** for smart streets – This is a management program, which supports IOT-based smart street management functions below:
    - **Cluster node management:** function for the addition/deletion/viewing etc. of all the clusters involved
    - **Smart node management:** function for adding/deleting/viewing of smart nodes controlled by a cluster, and tracking node status.
    - **Sensor management:** adding/deleting/viewing sensors from a selected smart node sensor and tracking the sensor status.

3. **IOT smart street data manager**
   This component does the following functions -
    - Retrieves sensor data for a smart node based on a given time(entered by the user) range and displays them in a table view.
    - Retrieves sensor data for a cluster node based on a given time range(entered by the user) and displays them in a table view.
    - Displays a cloud connectivity chart which shows the connectivity between all the stations and the back-end sever.

4. **Dashboard**
   This is a User Interface component (including a graphical user interface(GUI)) for our system. It will include the following common functions:
    - It provides statistical information of our smart street infrastructure.
    - A map-based user interface, which allows users to select a region, such as a city, zip code, street name and a given time range to access the following features:
        (a) discover and display smart nodes and cluster nodes for a smart street in a city using a map, and
        (b) discover and display the statistics information about the underlying sensor data in a selected time range.

1c) Functions Involved (in summary)

- Sensor data collection – function that collects sensor data from different sensors based on a-predefined schedule.
- Node-Cluster Communication: function that sends data from 5 nodes to a Cluster
- Cluster-Cloud Communication – function that transfers sensor data from the Cluster to the back-end server on a cloud
- Application Programming Interface(s) - function that responds to sensor data
- Database for Back End - function for storing database of sensor data for a minimum of last 100 days
- Graphical User Interface - function for displaying sensor status for a given time range selected by the user

1d) Technical Specifications

- Dashboard and Front End(User Interface): **ReactJS Version 16.0 and JavaScript(JS) ECMAScript 2018**
- Communication with Back End**: PHP 7.1.3**
- Back End: **AWS EC2 Instance + APIs(Java) + MySQL**

2) System Design and Architecture

2a) Sensor Cloud System Infrastructure

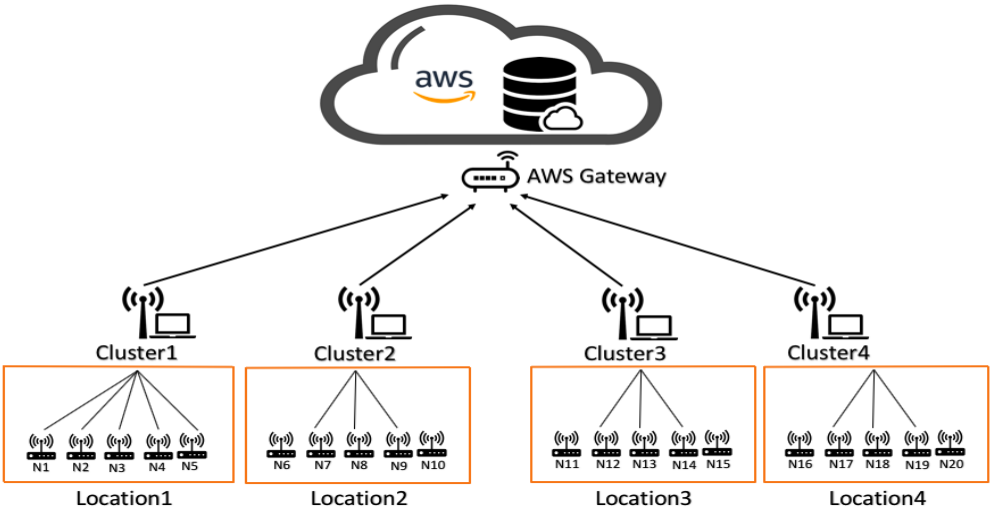


Figure (c)

The above architecture shows the outline of our infrastructure.

Several sensors constitute towards the forming of one Smart Node. The above shown nodes do not have any communication mechanism with each other. All the Smart Nodes send the sensor data to the cluster. Therefore 4 clusters take care of the underlying 20 smart nodes.

Cluster acts as an interface to transfer/transmit its underlying sensor's data to the cloud database. The data from the cluster is sent to the cloud service provider as IP Packets through the ISP. These IP Packets are then received by the Gateway router of the cloud provider and the sensor data is taken out from the packets and then stored in the Database.

2b) Deployment-Oriented System Infrastructure

The data from the sensors will be sent to the cloud provider's database. Then the data is organized in a framework (web page). The statistics, performance and availability of the sensors and its data can be viewed in a structured manner through a webpage. This information will be helpful to the Clients, maintenance people and the infrastructure managers. The smart node can be configured by the maintenance people after viewing and fetching the visual report of the sensors. Configuration of the sensors by the authoritative staffs is possible as the status of each cluster, smart node and each sensor is clearly visible.

The data can be easily accessed by throwing the API request to the Cloud Provider Database. According to the response received, the data is manipulated and plotted/marked. The UI design of the IOT framework allows us to communicate with the cloud service using the ISP. Request packets are sent, and the response packets are received like an HTTP request and response (API request/response).

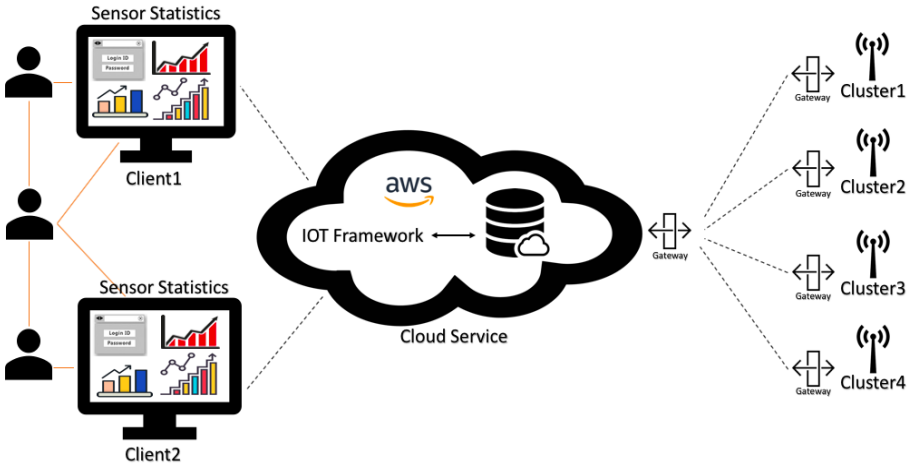


Figure (d)

2c) System-Oriented Component Functional View

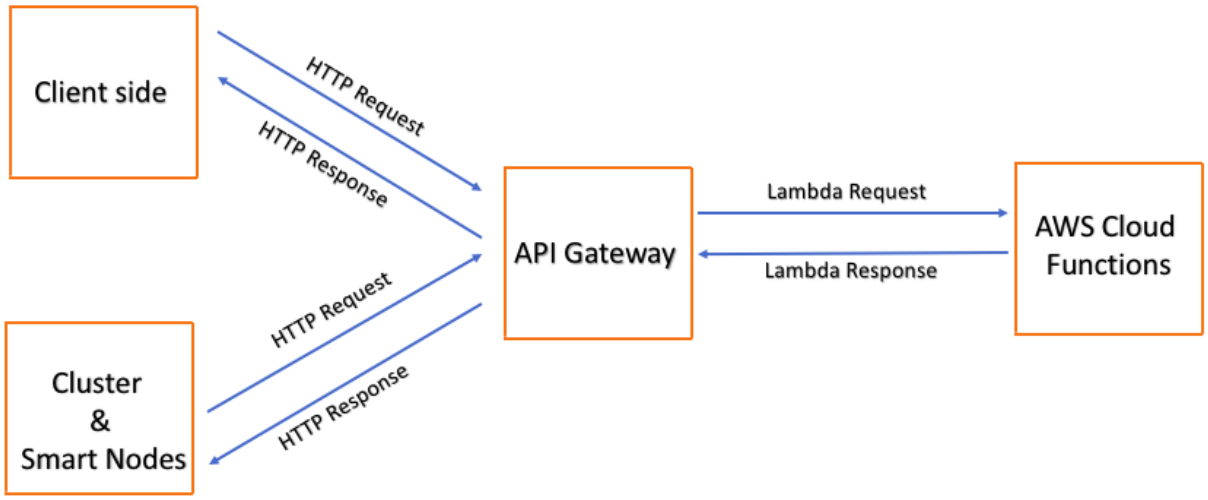


Figure (e)

The above diagram shows the functional overview of our project. The client side consists of Web Users and Administrators. Administrators are the ones who have complete control over the infrastructure. All the events generated by the client side and the data from the cluster side are sent as an HTTP PUT/GET request to the API Gateway. The API gateway then forwards the request for processing. The below diagram shows the insight of AWS cloud functions.

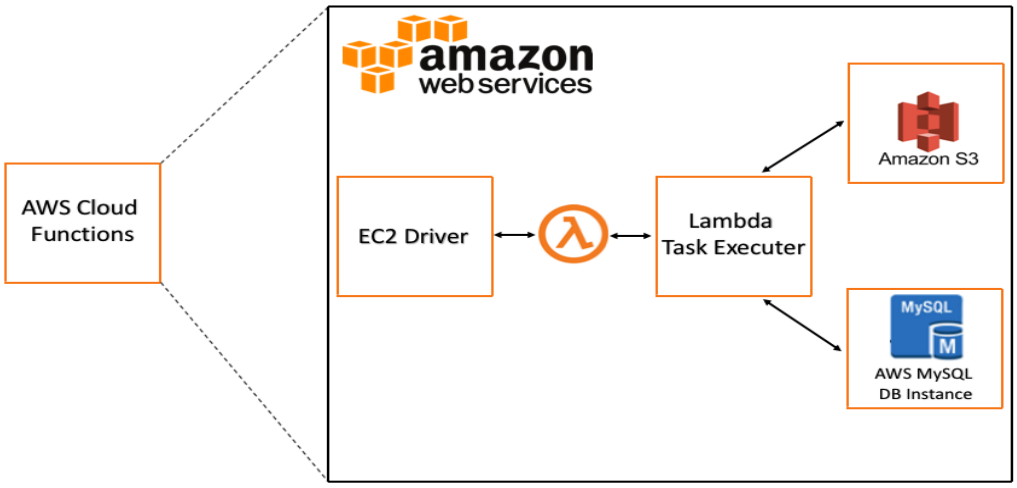


Figure (f)

AWS Cloud Functions consist of an EC2 driver (Compute driver), the request is then forwarded to the Lambda Task executer which processes the requested query. The response is then collected and sent back to the same channel as the HTTP response to the client side. The insight of the client side is shown below in a detailed view.

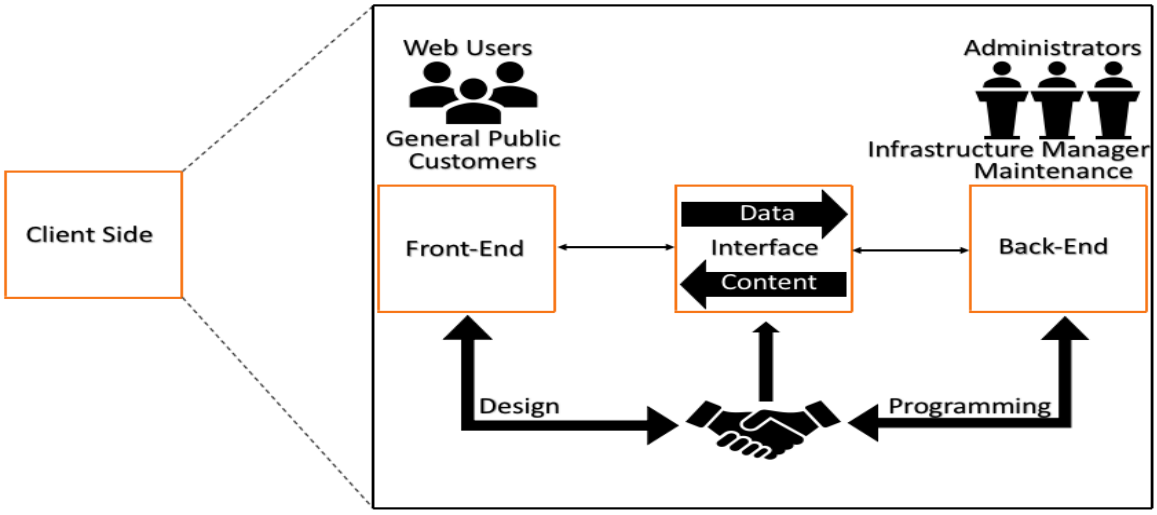


Figure (g)

The client side consists of a front end which deals with the GUI part. The GUI part is event driven which is the front-end. All the events are processed by the back-end. Both the parts collaborate to form the client-side GUI and event handling architecture. The front-end is entirely for the Web users looking for information. The information deliverable process is the backend which will be controlled by Administrators. Updating of node information and maintenance work will be handled by the backend.

3) Database Design

3a) Sensor Database Scheme

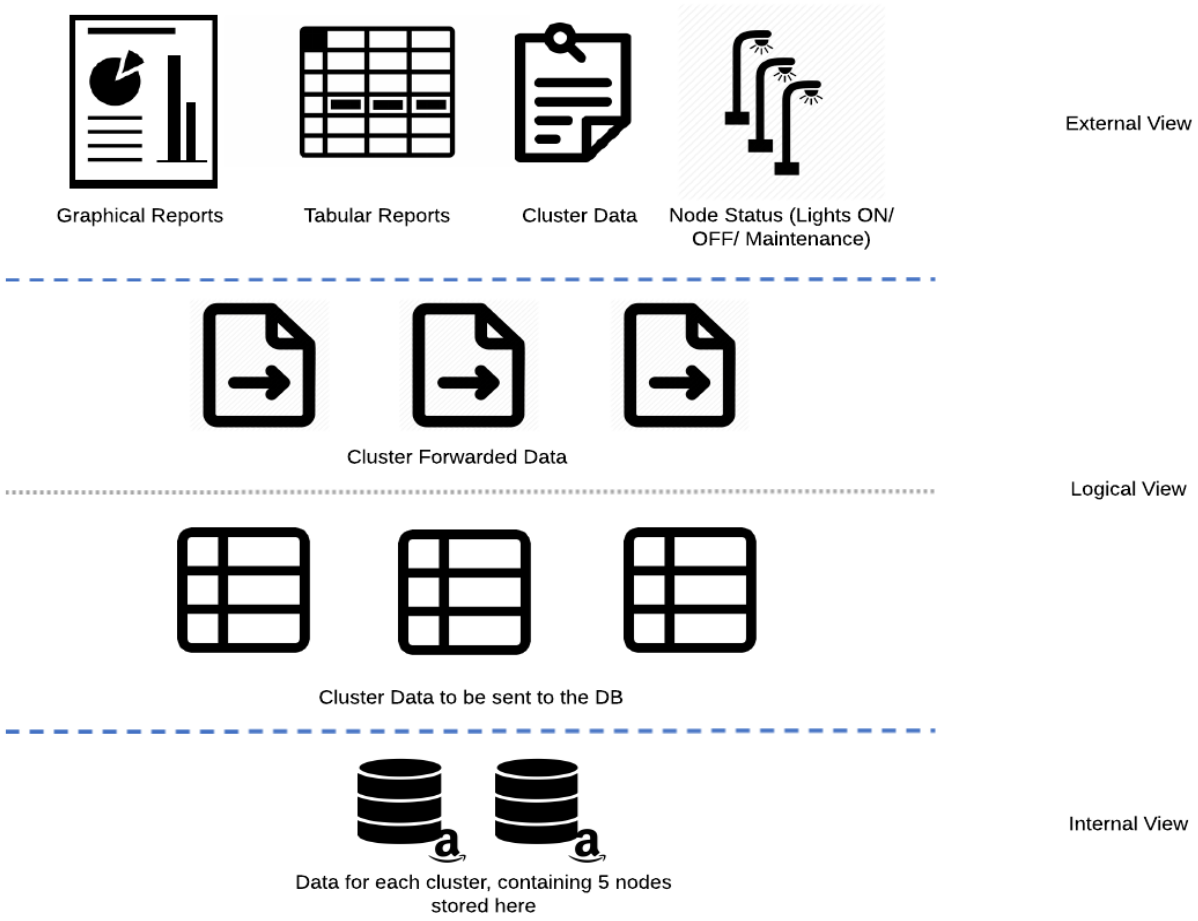


Figure (h)

The above figure is a 3-schema architecture of the Sensor Database model.

- External View - The User Interface comprises of Graphical and Tabular reports, Node status and a summary of Cluster data. Based on the User authentication and validation of his/her access rights, the details get displayed.
- Logical View – In this segment, the data received from the sensors (temperature, rainfall, motion etc.) are forwarded to the clusters. Here the data is tabulated and forward to the Cloud server, where different analytical decisions are taken based on the requirements.
- Internal View – This is the physical storage of all the sensor details that have been sent to the cluster. This will now be forwarded to the backend database hosted on Cloud.

3b) IOT Database Scheme

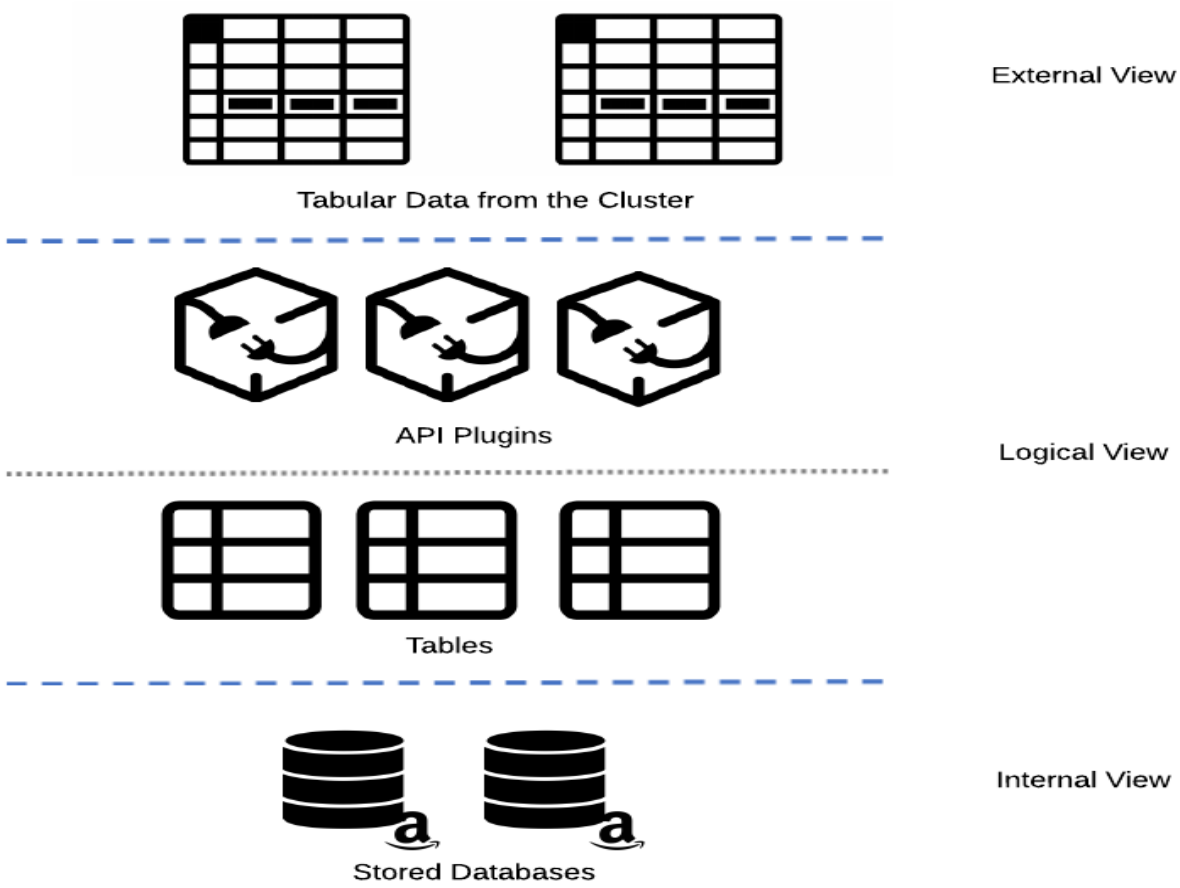


Figure (i)

The above figure is a 3-schema architecture of the IOT Database model.

- External View – With respect to IOT database, the external view is the tabular representation of the values stored in the database. The details provided in these tables are generated based on the user requirements, which is generated through the user interface provided.
- Logical View – This segment involves API utilization to corelate all the details gathered from all the sensors in the street lights. The data collected needs to be tabulated and analyzed to take decisions, as well as store in the database.
- Internal View – This is the physical storage of all the sensor details that have been sent to the cluster. It contains the historical data as well. Thus, anytime the user requests an archived dataset, it needs to be fetched from the backend database. The data stored in the cluster side database needs to be backed up to the backend database on a daily basis.

3c) Data Lifecycle Model

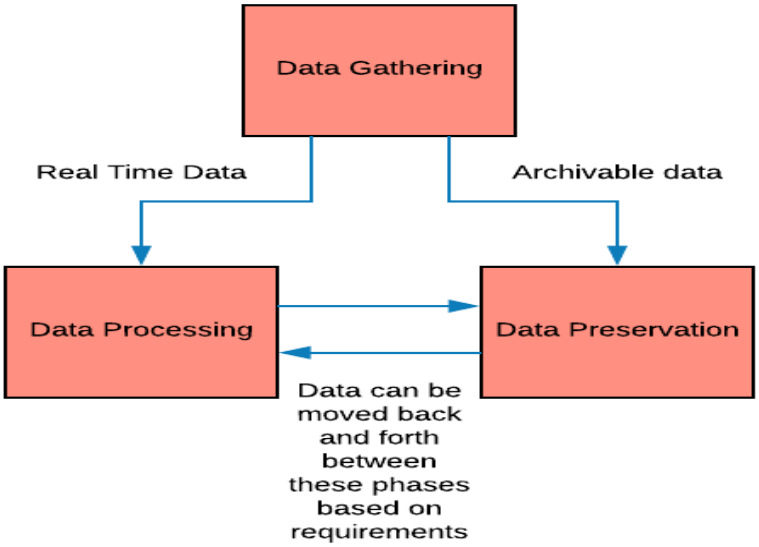


Figure (j)

The above figure is a Data Lifecycle model for the Sensor street lights. The data gathered from the sensors can be used in real–time for data processing or can be stored (archived) for later use. The archived data can be used for data processing at any time, to generate the desired results. Similarly, the data gathered in real time can be archived at any time, based on requirements.

3d) Database Design Model - ER Diagram

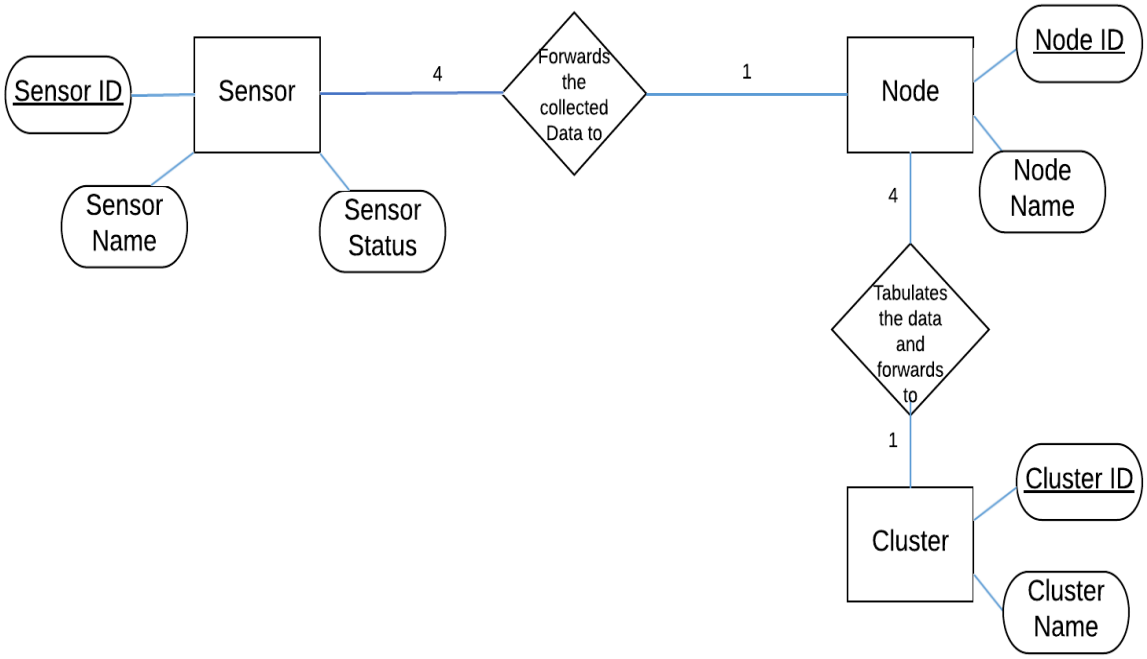


Figure (k)

Figure(k) is an Entity – Relationship Diagram for the Smart City Street Lighting Project. The entities considered here are sensors, nodes and clusters. Sensors collect real – time data and forwards it to their respective nodes. Nodes are responsible for tabulating the data obtained from the 4 sensors attached to it and sending it to the clusters. Each entity has unique ID's which serve as primary keys for identification. 1 sensor is attached to only 1 node. 1 node has 4 sensors attached to it. Similarly, 1 node is attached to only 1 cluster. Each cluster has multiple nodes attached to it.

3e) Data Table Representation

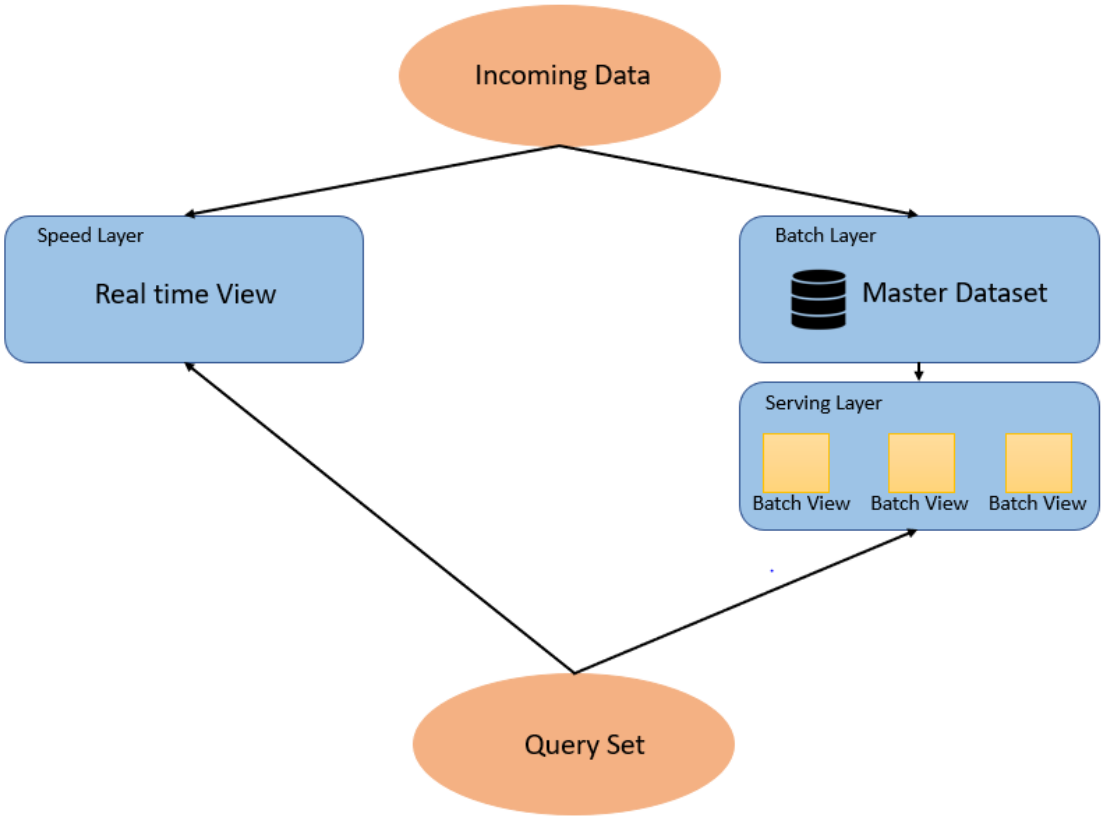
Given below is a Data table representation of a Node which has 4 sensors – Motion, Rainfall, Temperature and Light. Each cluster has 5 nodes attached to it, which in turn has 4 sensors which collect the Motion, Temperature, Light and Rainfall data. Data is collected and stored in 15 minutes interval, throughout the day. This data set is then forwarded to the Cluster, which in turn gets stored on the backend database hosted on cloud.

Cluster	Node	Sensor ID	Sensor Type	Sensor Values
	N1	S1	Motion	
		S2	Temperature	
		S3	Rainfall	
		S4	Light	
	N2	S1	Motion	
		S2	Temperature	
		S3	Rainfall	
		S4	Light	
	N3	S1	Motion	
		S2	Temperature	
		S3	Rainfall	
		S4	Light	
	N4	S1	Motion	
		S2	Temperature	
		S3	Rainfall	
		S4	Light	
	N5	S1	Motion	
		S2	Temperature	
		S3	Rainfall	
		S4	Light	

3f) Big DB Design Model

The below figure (j) is a big data model representation for the Smart City project. The incoming data is used for real time viewing in the speed layer. Since the real time views have high turnover rates, the errors can be quickly expelled. Basically, it provides a real – time access to the incoming data. Batch view on the other hand has the dataset stored in the Master dataset, which is error free. Based on the query set, the results from the master

dataset is provided in batches. In this way, the large amount of data collected can be accessed real time, as well as archived for analysis and resource sharing at any time.



Figure(m)

4) Component Design

The black-box component diagram for the designed architecture is as follows:

The above diagram consists of three major components, i.e., **the sensor cloud system**, the **AWS Cloud functions** and the **deployment-oriented system**. The sensor cloud system contains the various clusters, each containing five smart nodes. These clusters form an interface to forward the generated sensor data back to the cloud database. All the events generated by the client/user and the data from the sensor cloud system are sent as HTTP PUT/GET request to the API gateway. This API gateway in turn forwards these requests to the AWS cloud functions for further processing via the Lambda request/response queries to the cloud functions.

The AWS cloud functions consist of an EC2 driver, i.e., the computer driver, Amazon S3 and the AWS MySQL database. When the requests are received by this component, they are forwarded to the Lambda task executer, which helps in processing the query and responds to the same channel that sent the HTTP response to the client. The response of displaying the collected data to the client is handled by the deployment-oriented system.

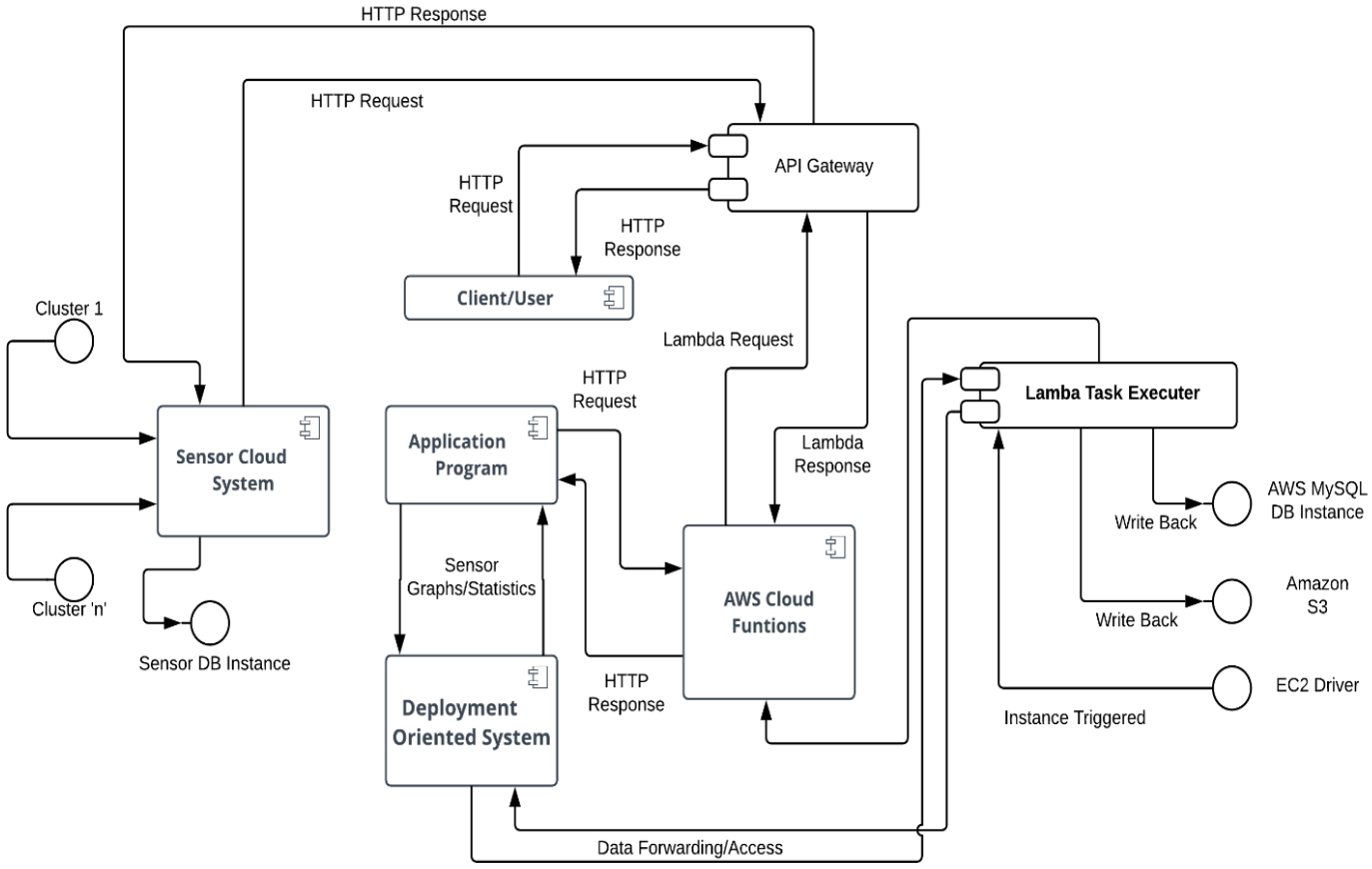


Figure (n)

The deployment-oriented system constitutes the client-side GUI consisting of event-driven (front end) and all the events processed (back end). Both are collectively collaborated to form a basic client-side architecture. The data from the various sensors are sent to the AWS cloud database which is organized in the web page. The sensor data which includes various degree of statistics, performance and availability of the latter, can be viewed in the web page. This organized data proves to be very useful to various clients, maintenance staff as well as the infrastructure administrator. Moreover, these smart nodes/clusters can be easily configured as well as scalability is achieved. This is possible by throwing the API request to the cloud provider's database. The response achieved ensures data manipulation. Respective statistics and graphs will thereafter be generated.

5) References

- https://aws.amazon.com/s3/
- https://aws.amazon.com/rds/
- https://medium.com/stanfy-engineering-practices/