SAN JOSÉ STATE UNIVERSITY

**CMPE-281**

# Cloud Technologies

## Final Project Report

**Submitted to**

Dr. Jerry Gao

**Date of Submission**

Dec 09, 2016

**Submitted by: Group1**

Sneha Kasetty Sudarshan<br>
Mohamad Shafaat Ali Khan<br>
Khanh Dao<br>
Tran Pham

# ABSTRACT

Transportation is a big issue for any densely populated cities. The traditional solution is to build more lanes to the road. It not only is a costly solution but it's also ineffective over time. We can not pave our way out of traffic congestion but we can increase our efficiency in using existing public transportation system: train, bus, light rail… Efficient smart city transportation is built based on a *connected city transportation*. This can be achieved by combining sensor data in public transportation with new cloud technology & big data to help analyze the data and develop effective solutions.

## 1. Introduction

In this project, we focus on building a cloud infrastructure (Iaas) for smart city transportation to support and manage mobile sensor resources. The IaaS is capable of setting up, control, and management of mobile sensor as an on-demand service. It virtualize sensor networks, abstract the differences between physical sensor by providing virtual sensor template, which support wide range of popular sensor currently mounted on public transportation vehicles. Besides, the web interface provide dashboard for different types of users and tools to monitor and meter mobile sensor status. Furthermore, it can manage load balance and scalability of sensor network. Finally, it provide a billing components for administrator to set price for sensing services and data services.

The mobile sensor cloud infrastructure as a service once successfully deployed will act as enabler, bring sensor data of public transportation closer to the software development community. Sensor Data such as location sensor, Weight sensor , temperature sensor, air pollution sensor, speed sensor, weather sensors etc can be utilised to develop various smart applications

1. One of the smart application could be to determine location of a transportation vehicle as to what time it will be arriving in the bus stop/ train station.
2. Another smart application could be to determine the number of seats available in the public transportation vehicle which can be used by general public to know available seats in the vehicle
3. Sensor Data collected from the vehicles can also be used by transportation companies which can determine the peak hours at certain locations to provide more public vehicles for the peak hours.

We hope such smart software solutions in near future will help us raise the efficiency of our public transportation. Once smart city transportation work effectively, it will help reduce the traffic congestion, air pollution and parking places.

# 2. System Architectures

## 5.1 Mobile Cloud Infrastructure

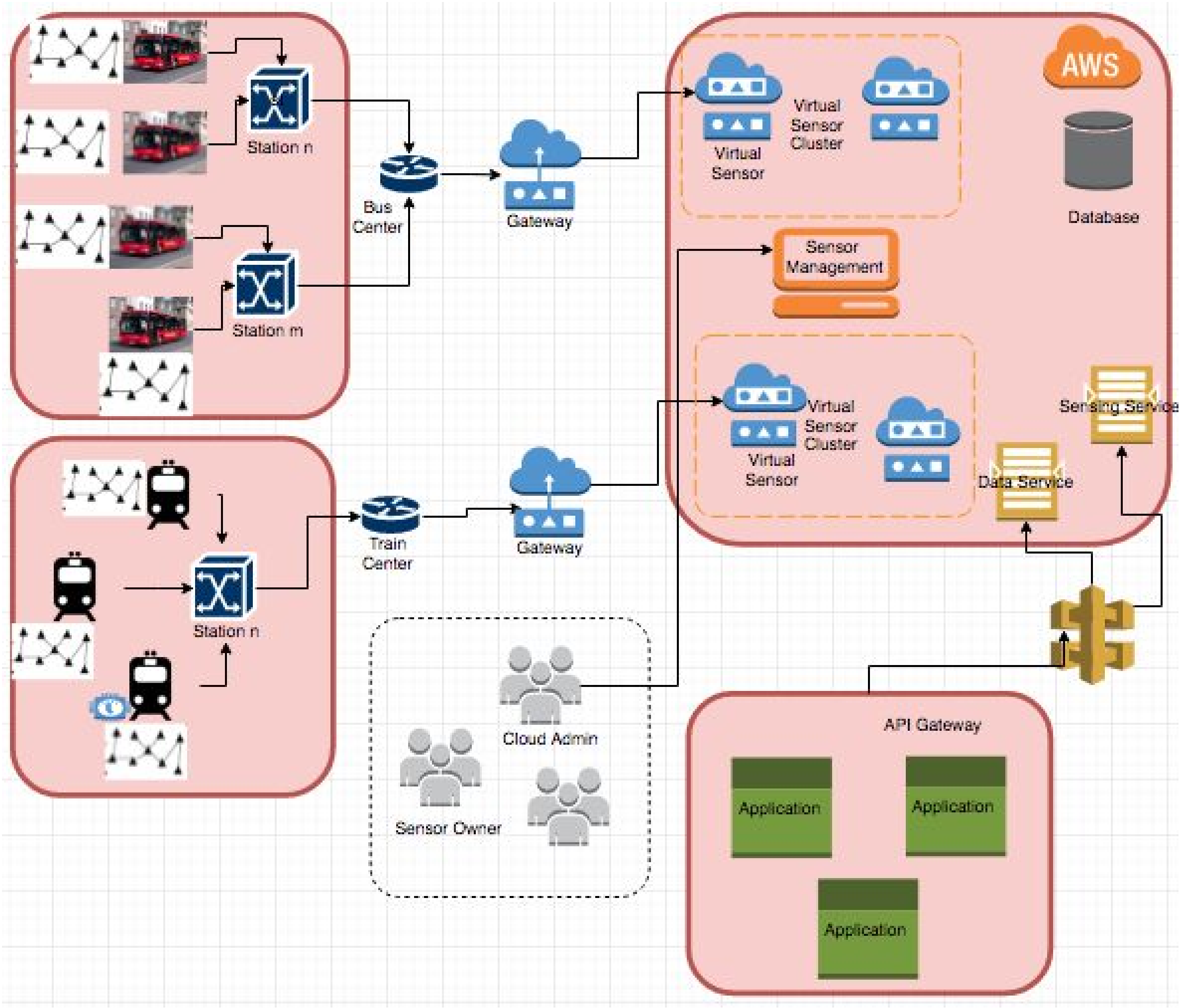


*Figure 1. Mobile sensor infrastructure design*

## 5.2 System Component Design

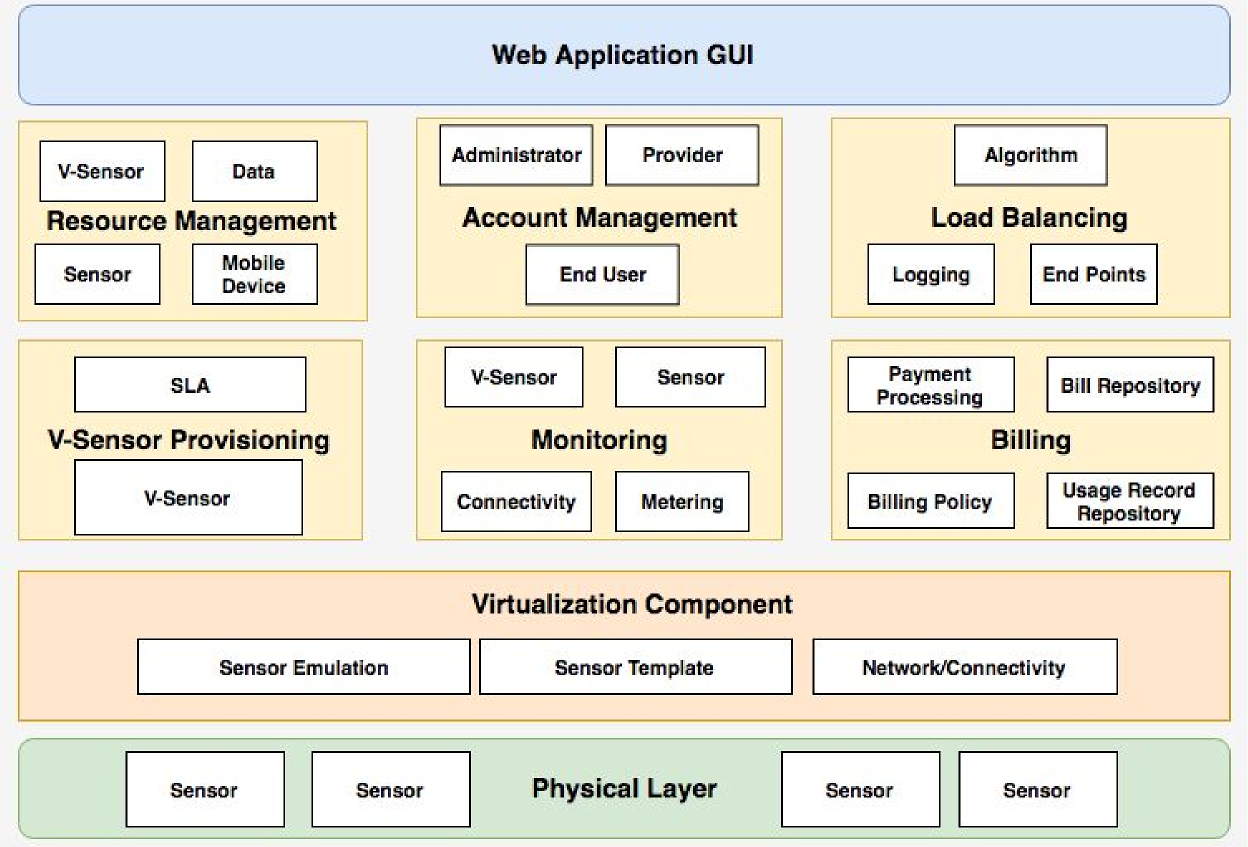


*Figure 2. System Component Design*

### 5.2.1. Dashboard Component Overview

The Dashboard is the first visual screen, which is displayed to the user after they successfully login. The Dashboard displays the registered vehicles, Registered sensors, billing details and the location of the vehicles in google maps. The Dashboard provides links to the functions, which the user can perform, providing easy navigation to the function pages.

The Dashboard Component Has Three Main Users'

1. The Vendor
2. The System Admin
3. The End User

**The Vendor Dashboard** – The Vendor dashboard has different functionality like sensor monitoring, Register Vehicle, Sensor Metering etc. and easy navigation to these pages. The dashboard displays circular pie chart displaying the registered vehicles of the vendor and state (i.e. whether the vehicle is ON or OFF). Another pie chart Displays the Registered sensors and their count. Billing details are in the form of a bar graph for a period of six months. A google map displaying the vehicles, which has the mobile sensors and the route in which the vehicle is travelling.

**The End User Dashboard** – The End User Dashboard Allows the User to Register for a particular sensor and route and displays the Details of the Registered sensors along with the billing details in the form of graphs. Different functionalities of the user are displayed on top and provide easy navigation to those respective pages. A Google map displaying the Registered mobile sensor and route in which it's travelling is displayed.

T**he system admin** - The system Admin Dashboard shows all the vendors and their registered vehicles and sensors in the form of graphs. Different functionalities of the admin are displayed on the dashboard with easy navigation to their respective pages. A Google map displaying all registered vendor vehicle and their route is displayed on the dashboard. The Admin Also can view the load balancing which displays the requests being balanced between 2 servers.

### Dashboard Function Design

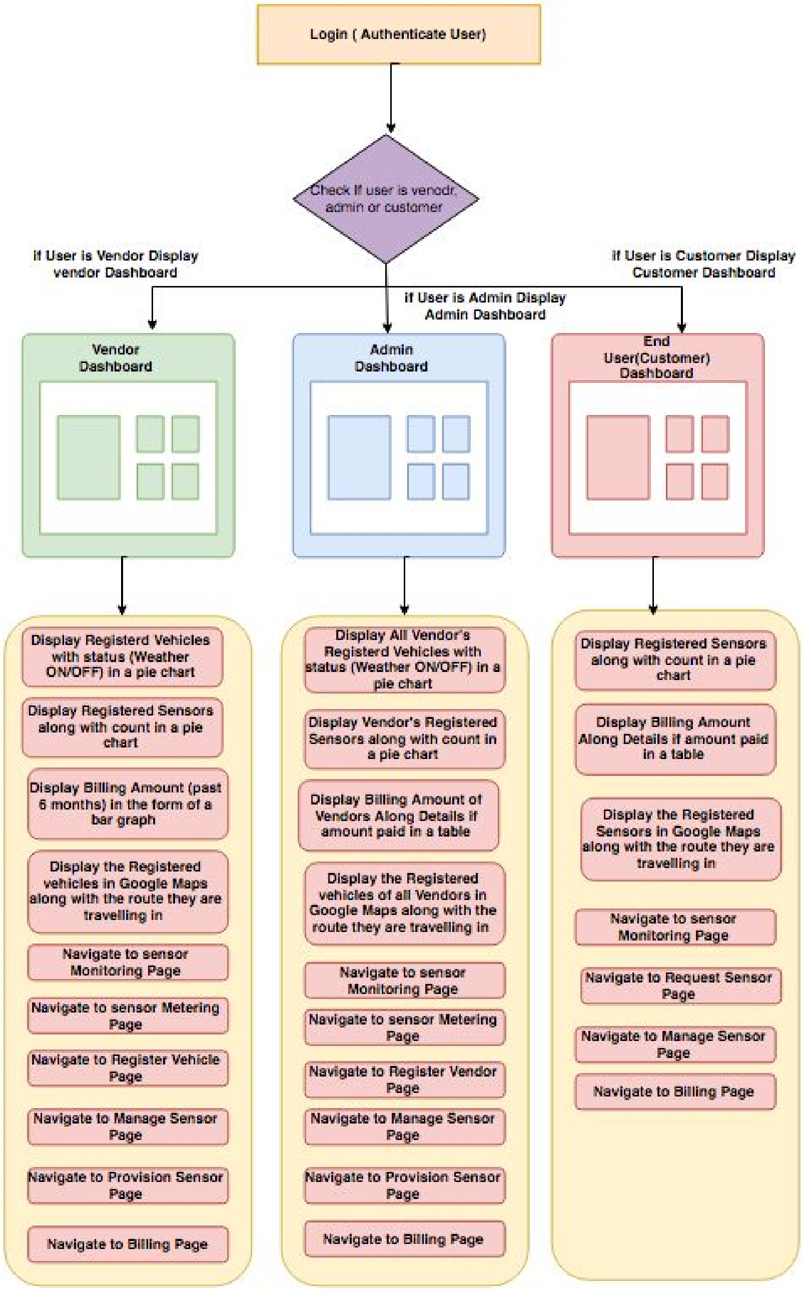

Login ( Authenticate User)
Check If user is venodr, admin or customer
If User is Vendor Display vendor Dashboard
If User is Admin Display Admin Dashboard
If User is Customer Display Customer Dashboard
Vendor Dashboard
Admin Dashboard
End User(Customer) Dashboard
Display Registerd Vehicles with status (Weather ON/OFF) in a pie chart
Display Registered Sensors along with count in a pie chart
Display Billing Amount (past 6 months) in the form of a bar graph
Display the Registered vehicles in Google Maps along with the route they are travelling in
Navigate to sensor Monitoring Page
Navigate to sensor Metering Page
Navigate to Register Vehicle Page
Navigate to Manage Sensor Page
Navigate to Provision Sensor Page
Navigate to Billing Page
Display All Vendor's Registerd Vehicles with status (Weather ON/OFF) in a pie chart
Display Vendor's Registered Sensors along with count in a pie chart
Display Billing Amount of Vendors Along Details if amount paid in a table
Display the Registered vehicles of all Vendors in Google Maps along with the route they are travelling in
Navigate to sensor Monitoring Page
Navigate to sensor Metering Page
Navigate to Register Vendor Page
Navigate to Manage Sensor Page
Navigate to Provision Sensor Page
Navigate to Billing Page
Display Registered Sensors along with count in a pie chart
Display Billing Amount Along Details if amount paid in a table
Display the Registered Sensors in Google Maps along with the route they are travelling in
Navigate to sensor Monitoring Page
Navigate to Request Sensor Page
Navigate to Manage Sensor Page
Navigate to Billing Page

*Figure 3. Dashboard functional design*

### 5.2.2 Load Balancer Component

a. **Overview:**

Load Balancing is performed on the Client's API Requests. Using Round robin algorithm, it alternates requests between servers in a stateless manner. In the current setup the application is running on 2 different servers ports 8081 and 8082 respectively. The load balancer is configured to run on port 8000. The Load Balancer forwards the request to each of the servers. This is how the load on the servers are balanced.

b. **Load Balancer Functional Design**

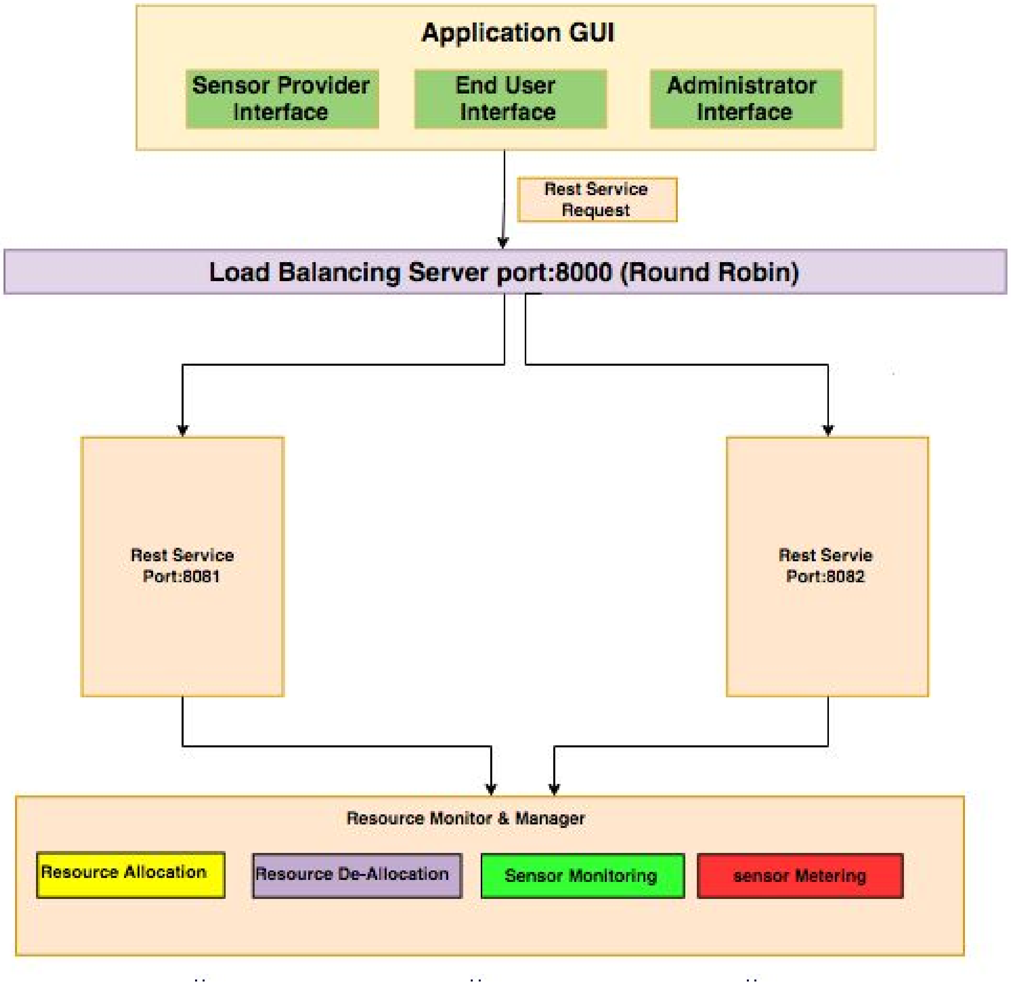
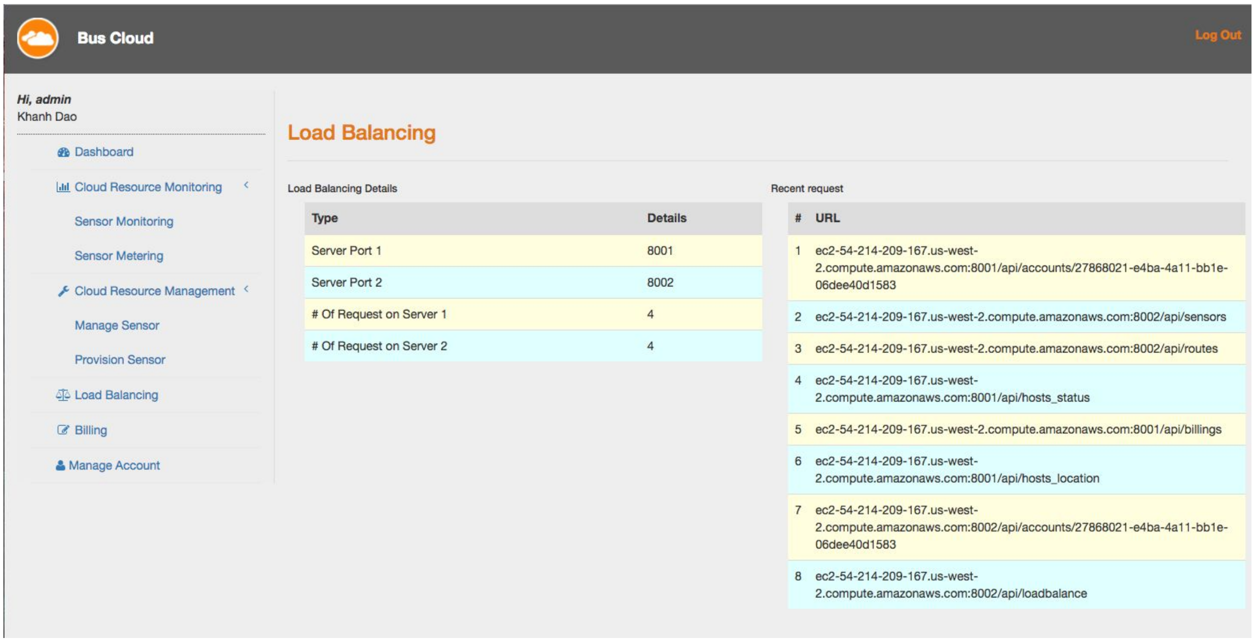


*Figure 4. Load Balancer Functional Design*

### c. Load Balancer UI Design



Load Balancing

Load Balancing Details

| Type | Details |
|---|---|
| Server Port 1 | 8001 |
| Server Port 2 | 8002 |
| # Of Request on Server 1 | 4 |
| # Of Request on Server 2 | 4 |

Recent request

| # | URL |
|---|---|
| 1 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/accounts/27868021-e4ba-4a11-bb1e-06dee40d1583 |
| 2 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/sensors |
| 3 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/routes |
| 4 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/hosts_status |
| 5 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/billings |
| 6 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/hosts_location |
| 7 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/accounts/27868021-e4ba-4a11-bb1e-06dee40d1583 |
| 8 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/loadbalance |

## 5.2.3 Monitoring Component

### a. Overview:

The monitor component provides monitoring capability to the sensor providers, end users and the system admin. It uses different metrics like data frequency, time and sensor state to gauge bandwidth, performance, health and billing. Depend on the user role the monitoring interface for each group of user have different functionalities:

- **The system admin:** monitor the resource usage for billing and sensor state .
- **The end user:** should be able to monitor his personal data usage and service consumption.
- **The sensor network provider** monitor the actual physical state of sensor to perform maintenance, provision and deprovision sensors.

### b. Design:

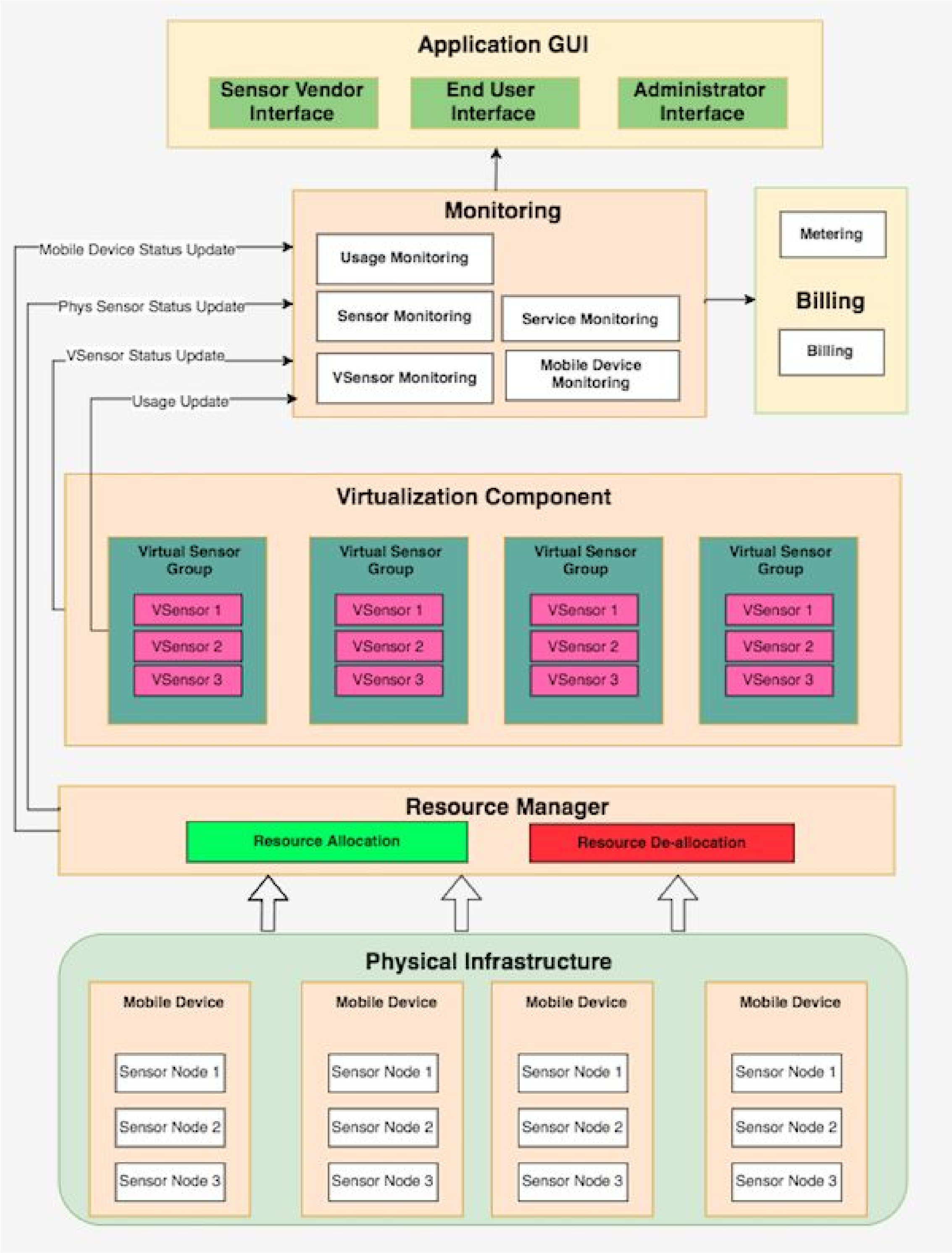


*Figure 5. Sensor Monitor design*

### 5.2.4 Virtualization of Sensor Networks Component

a. **Overview:**

In this component the user request for virtual sensor. The subscriber layer takes in the request of the user and finds an available virtual sensor and allocates it to the user. Creation of virtual sensors is done by the virtualization layer. This layer is present on top of the Physical sensors layer. Physical sensors are present on the mobile device. These sensors are either Location sensor, Clipper sensor, Speed Sensor or Temperature sensor

b. **Design:**

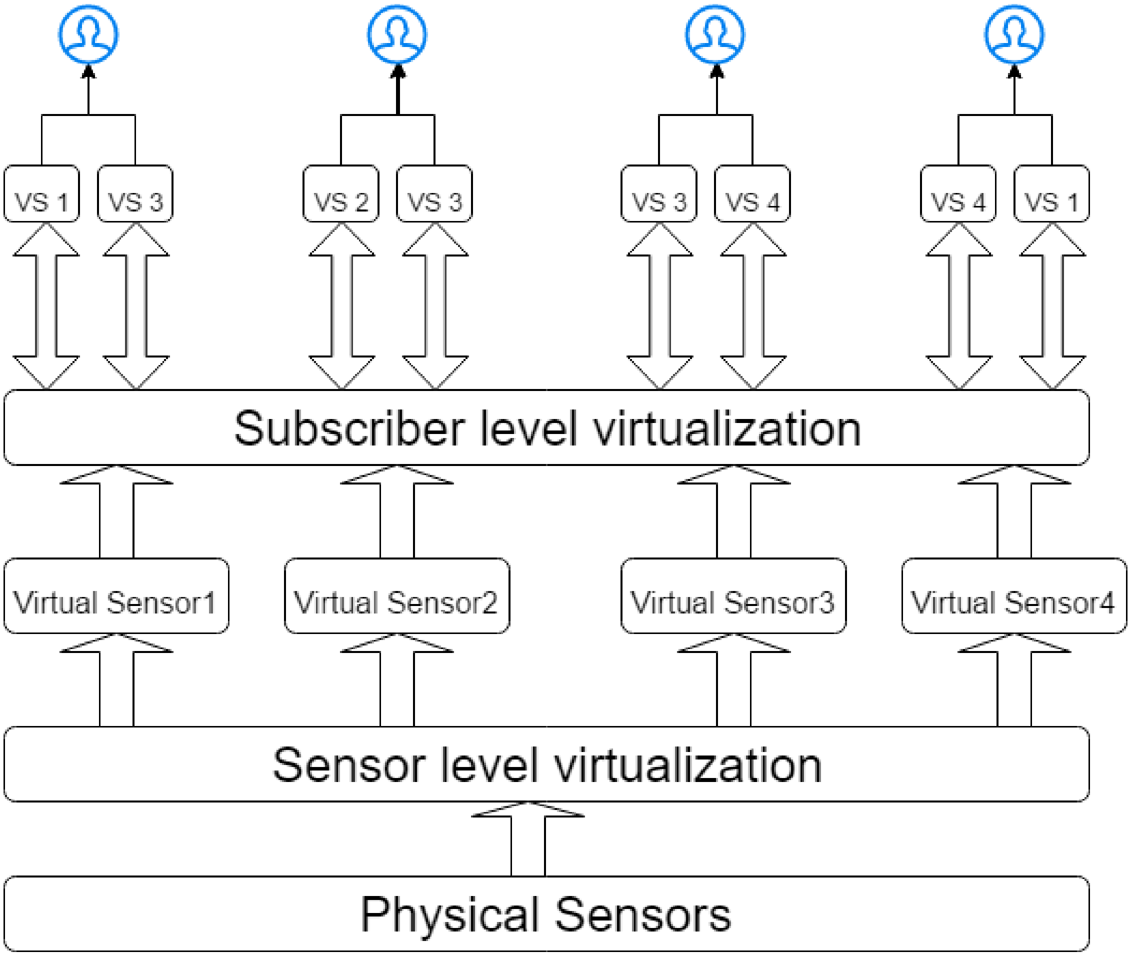


*Figure 6. Virtualization of sensors component design*

### 5.2.5 Sensor Provisioning Component

**a.Overview:**

In this component we show the mapping relationship between the sensors and the user. Relationship between the virtual sensor and the user is one-to-one. That means one user can use one virtual sensor at a time. Relationship between a physical sensor and the user is one-to-many relationship. Multiple virtual sensors can be created on top one physical sensor and each user can be assigned to a user.

**b. Design:**

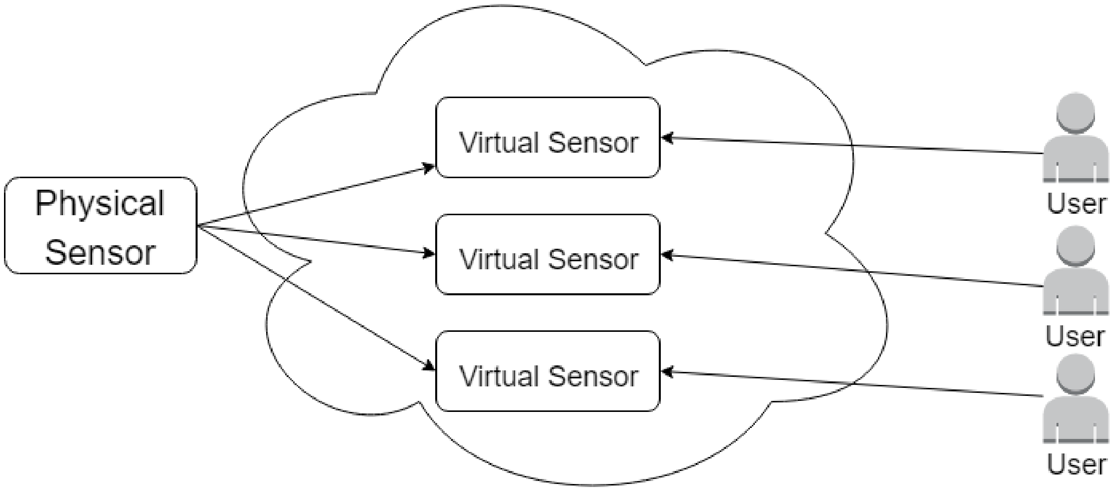
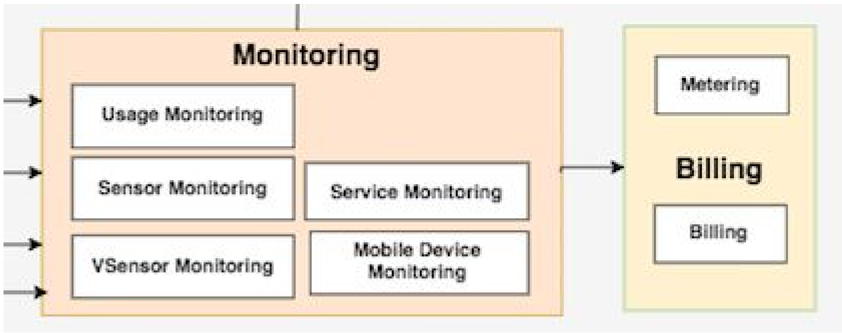


*Figure 7.Sensor Provisioning design*

### 5.2.5 Billing Component

a. Overview
Base on user's data usage compute the bill based on cost model (SLA)

b. Design



c. Cost Model:
- Cost metric: data row count

Bill = Data Row Count * SLA Cost rate * coefficient type

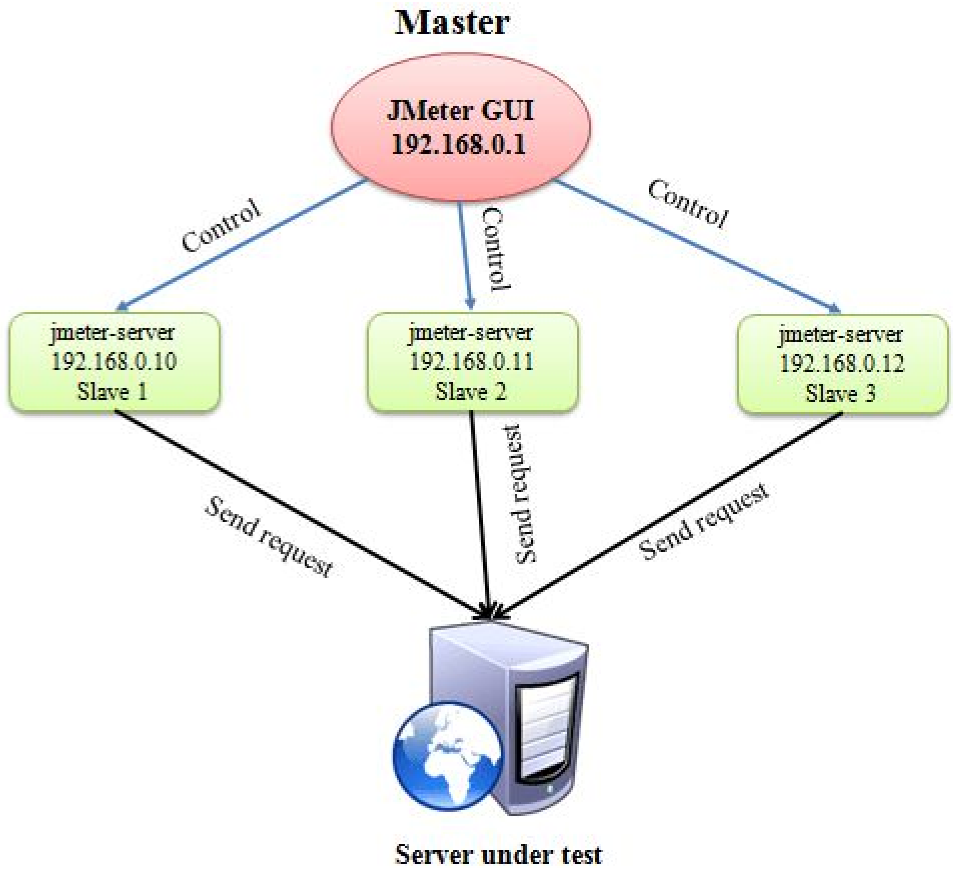
- SLA Cost rate = 0.1 cent
- Coefficient type :
  - 1 - temperature sensor data
  - 1.2 - speed sensor data
  - 1.3 - clipper sensor data

### 5.2.5 Load Traffic Generator:

a. **Overview**

To test the load balancing feature and scalability of application, we use JMeter to simulate user traffic

b. **Design**



c. **Test case:**

**(1) <u>- Provision virtual sensor:</u>**

1. Log in
2. Go to resource management page
3. Request temperature sensor

**(2) <u>- Generate monthly bill:</u>**

1. Log in
2. Go to billing page

## 5.3 System Deployment Diagram

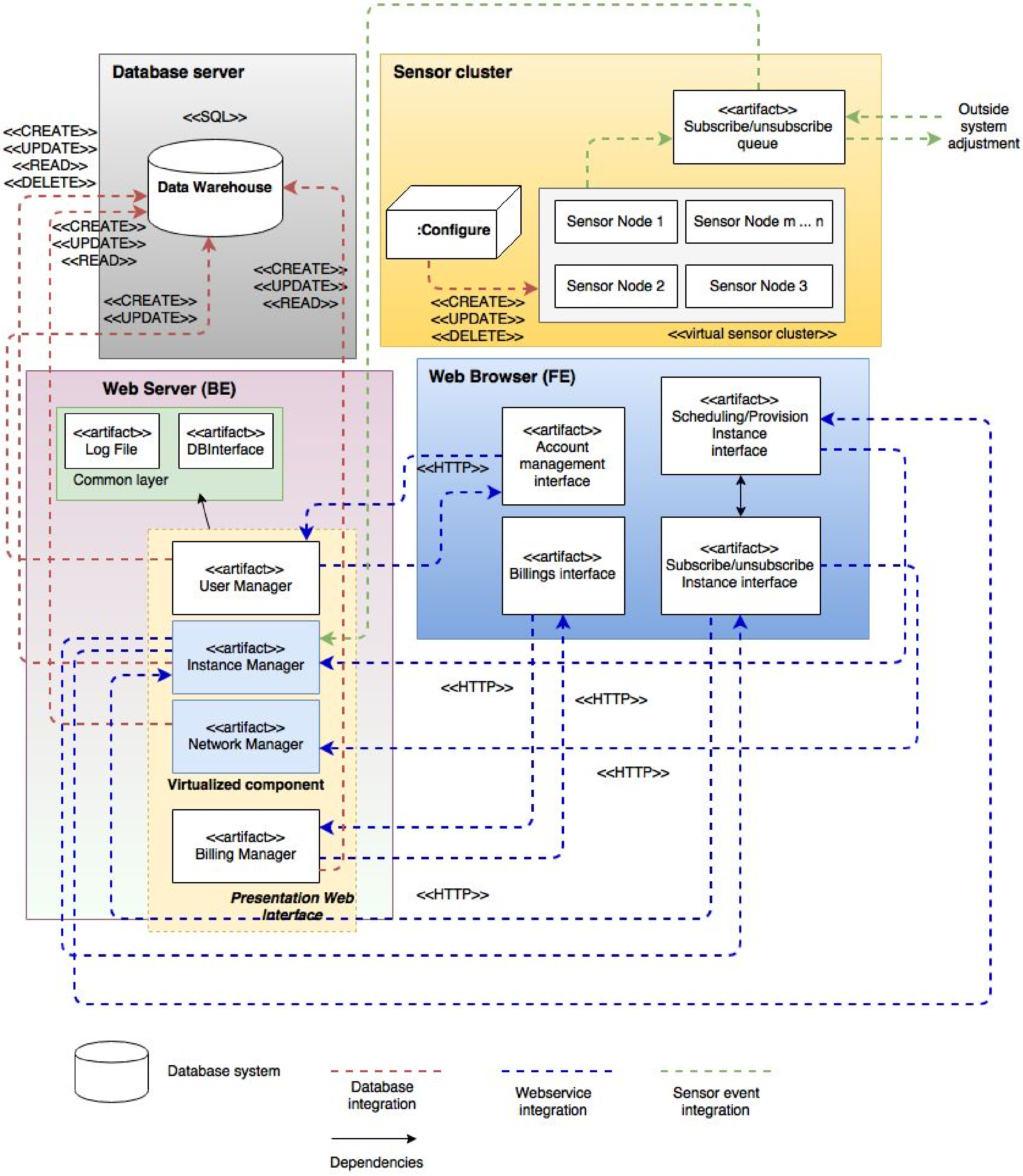


*Figure 8. Systems deployment diagram*

- **Sensor cluster**: System environment in which virtual and/or physical sensor devices are directly managed. User who has admin access to the sensor cluster environment will have the capability to configure/edit and interact with sensor directly. Any changes caused by this action will trigger sensor event, which will notify sensor management in the web server application to trigger the DB update. Sensor nodes are can be varied in types and configurations, such as light sensor, mobile sensor...
- **Database server**: The cloud-based system environment, which is set up for managing database resources. In this project, we will use SQL database server as data storage and configuration for the cloud ministrations.
- **Web Server (BE API)**: System environment where the presentation web interface application is deployed  and monitored. Components to deployed and monitors include billings manager, account manager, instances manager and network manager. These components will interact directly with Database systems to queries and configured all data and resources required, through common DB interface. Log file are instrumented and maintained at this level for system monitoring, health-check and debug purposes. The expected outcome from the Presentation Web Interface Layer will be set of CRUD RESTful APIs (in JSON or XML format).
- **Web Browser (FE)**: This layer will be responsible for the deployment and maintenance of the customer-facing web interface (dashboard UI). Outside users  and admins will interact and monitor their own set of sensors through the comprehensive UI interface. Web FE interact with DB and sensor cluster through APIs services provided by the web server application as mentioned above.

# 3. Database Design

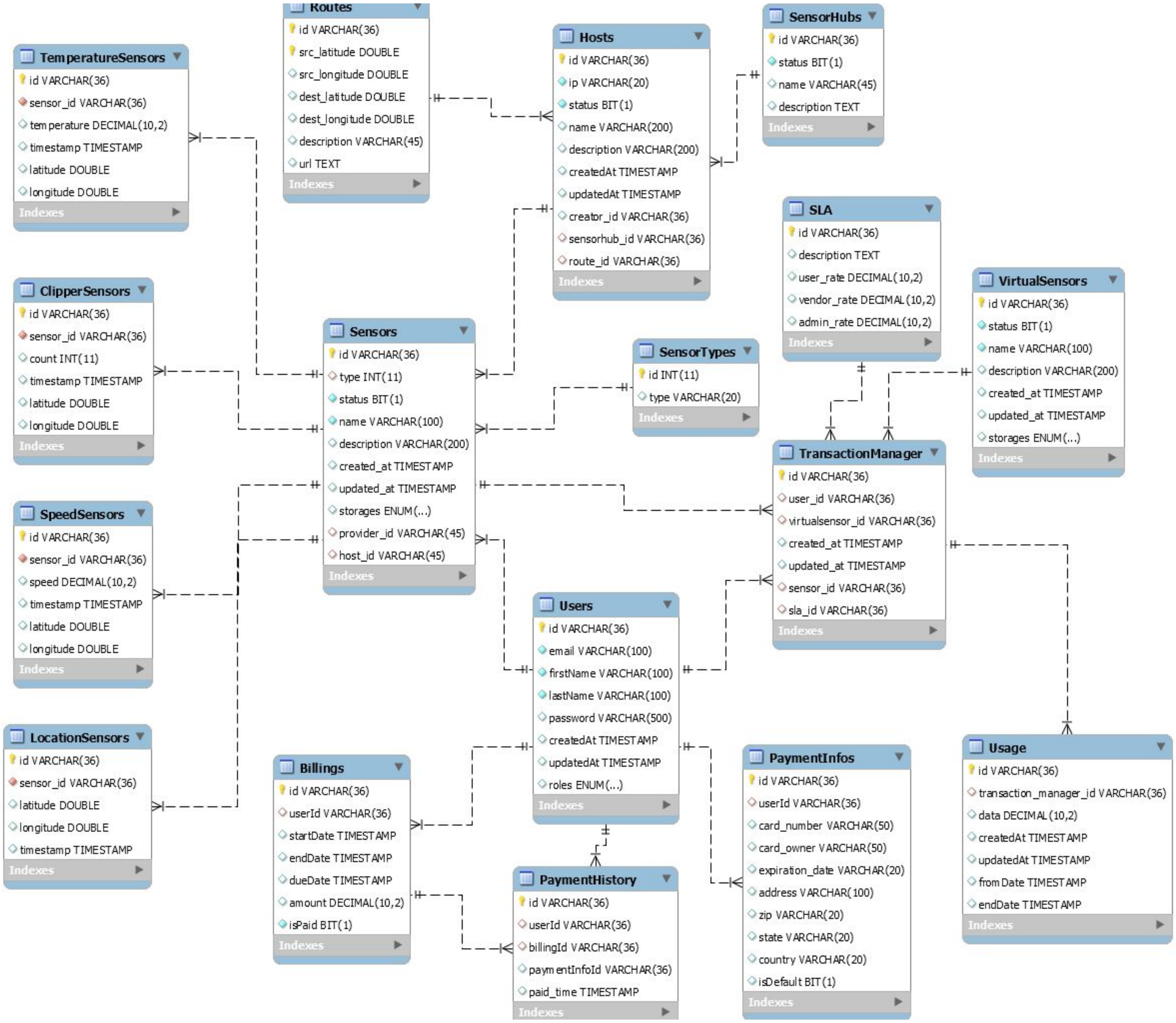


*Figure 9. Database ER diagram*

# 4. Service Component Analysis and Design

## 4.1 Use Cases

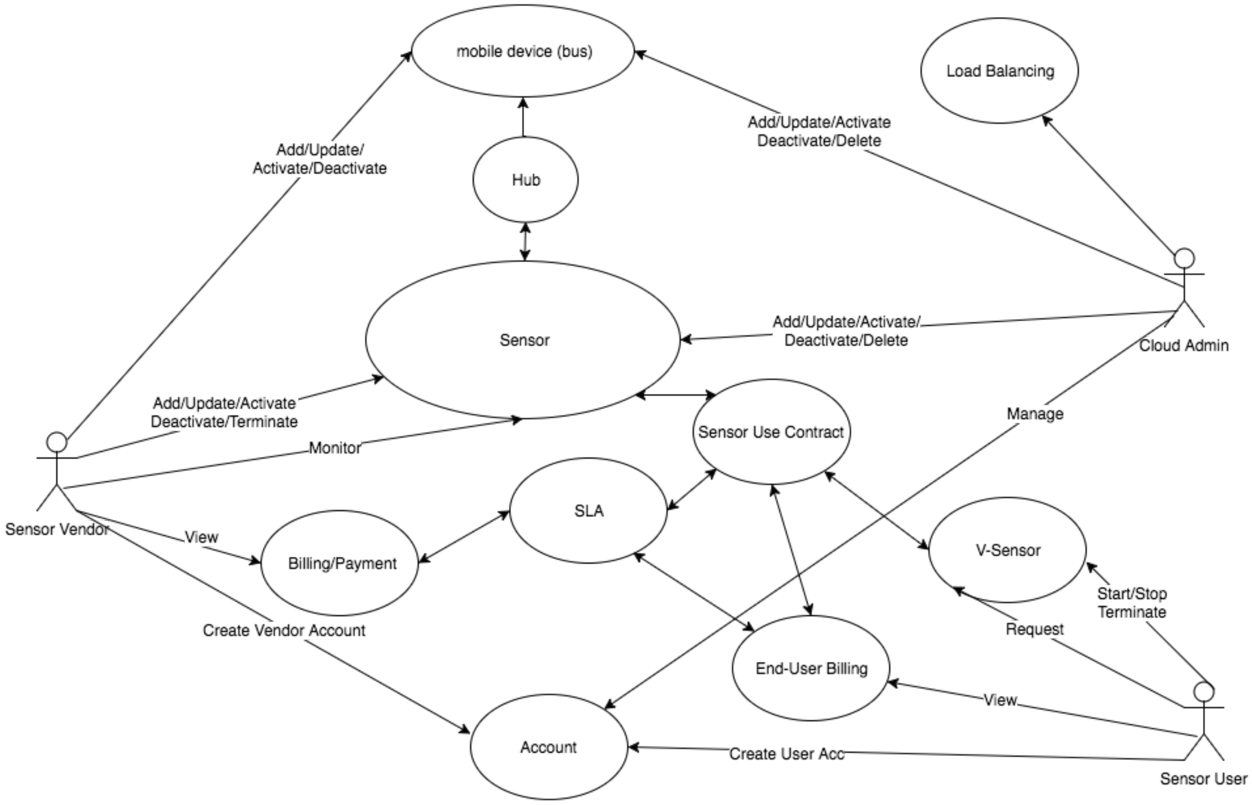


*Figure 10. Use case diagram*

## 4.2 UML Diagram

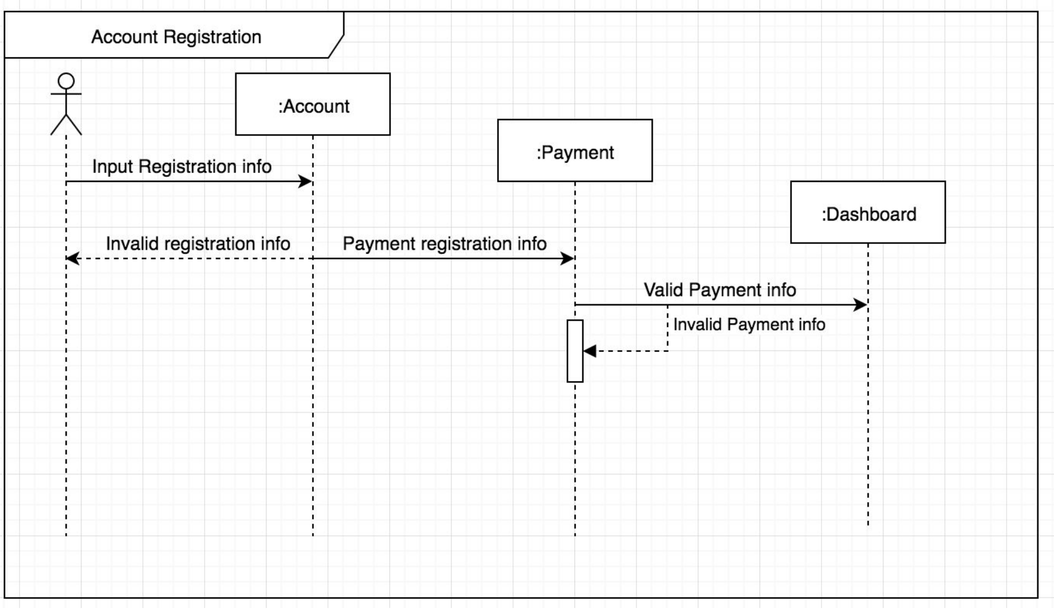


*Figure 11.1: Account Registration flow*

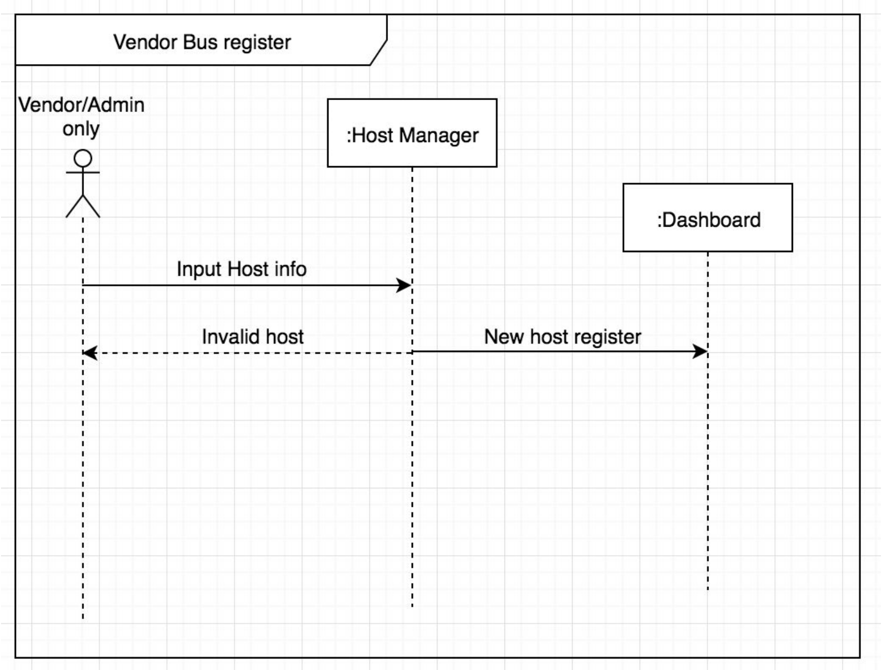


*Figure 11.2: New bus registration flow*

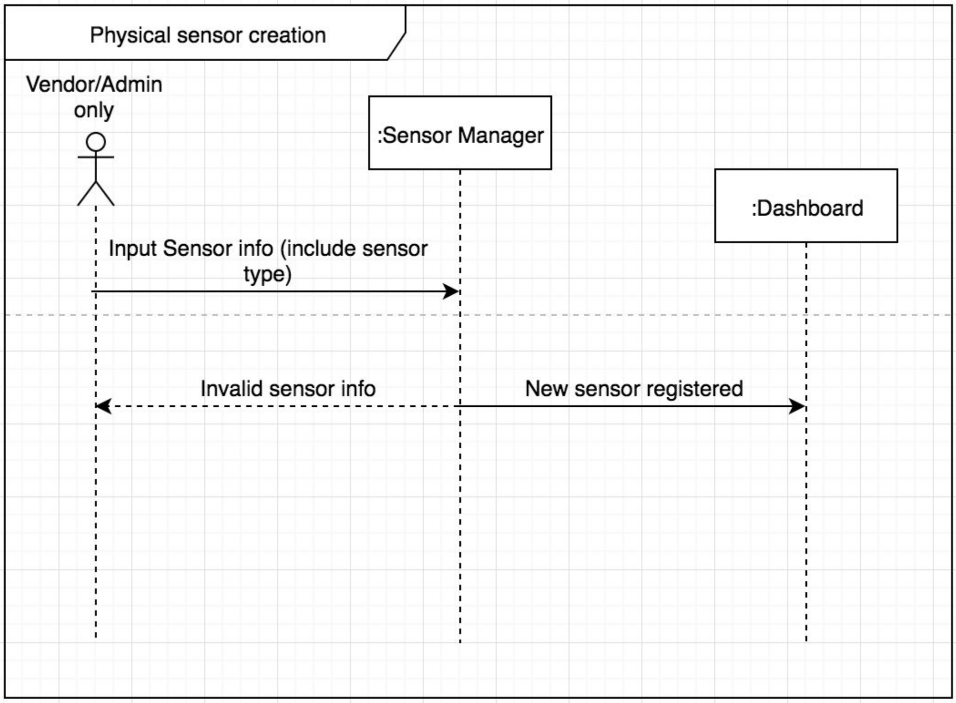


*Figure 10.1: Physical sensor creation flow*

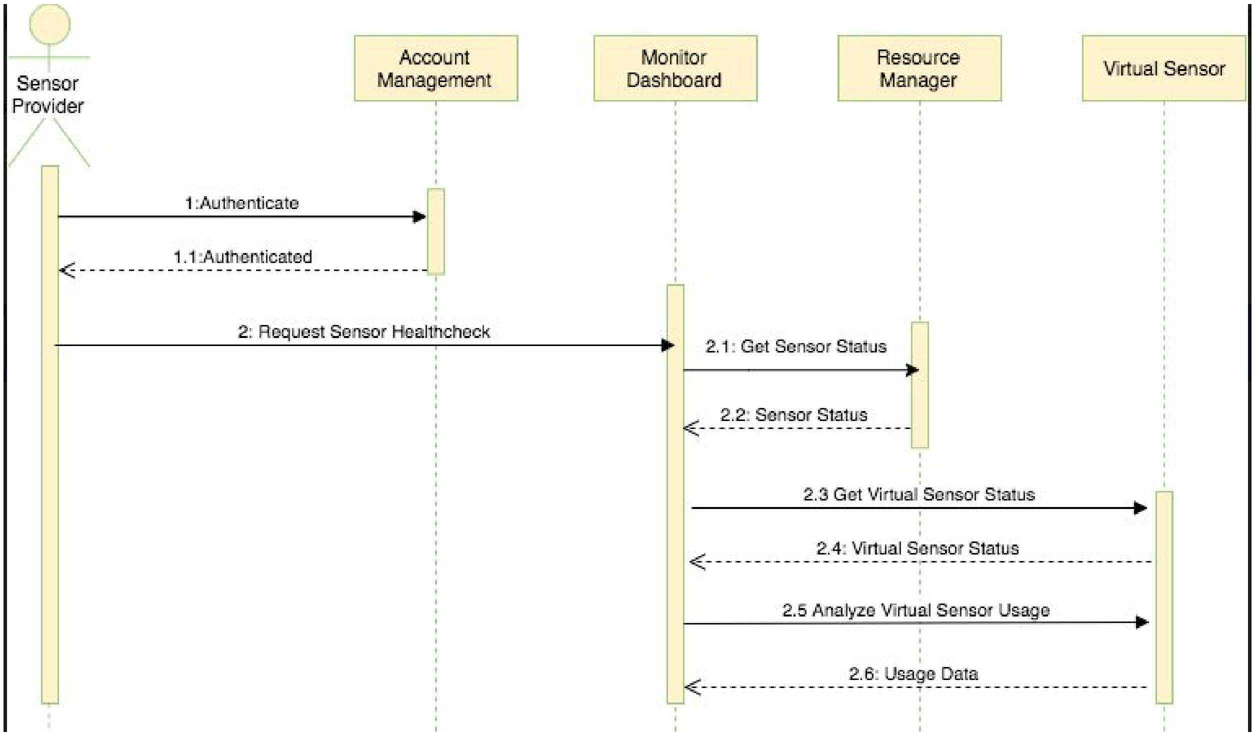
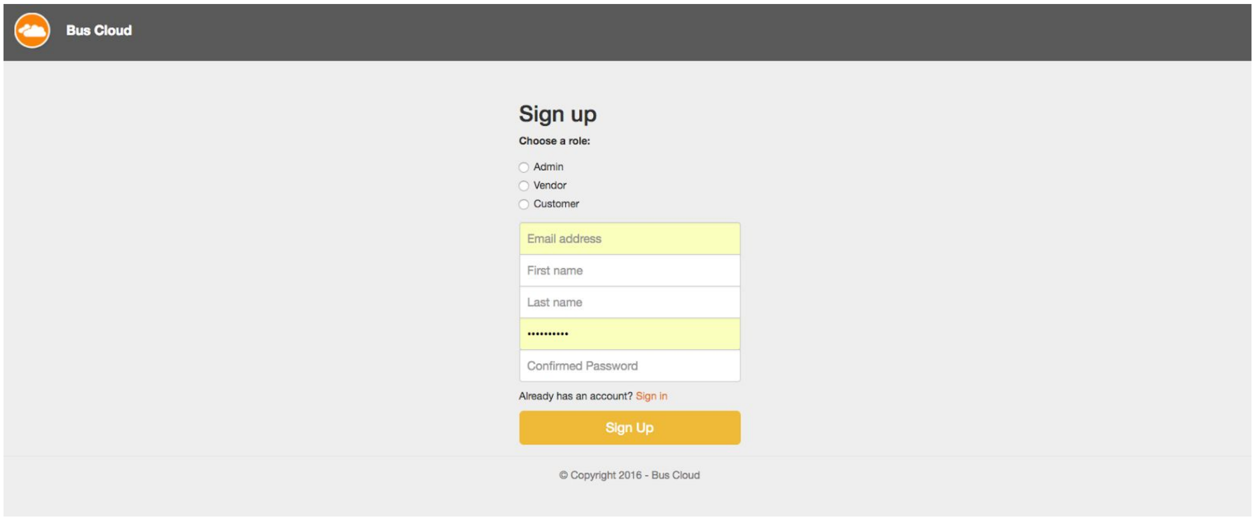


*Figure 11.4: sensor monitoring flow*

# 5. GUI Design

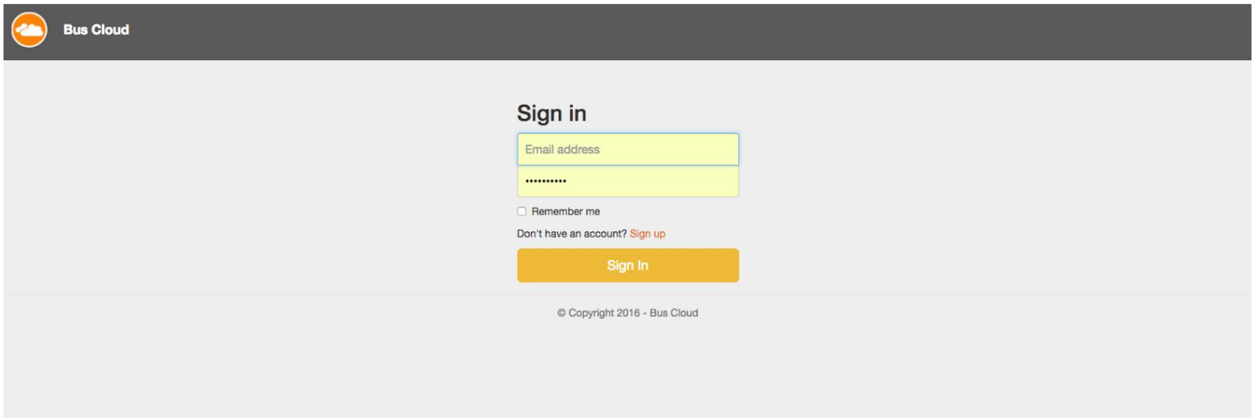
## 5.1 Creating new Account Page



Already has an account? Sign in

© Copyright 2016 - Bus Cloud

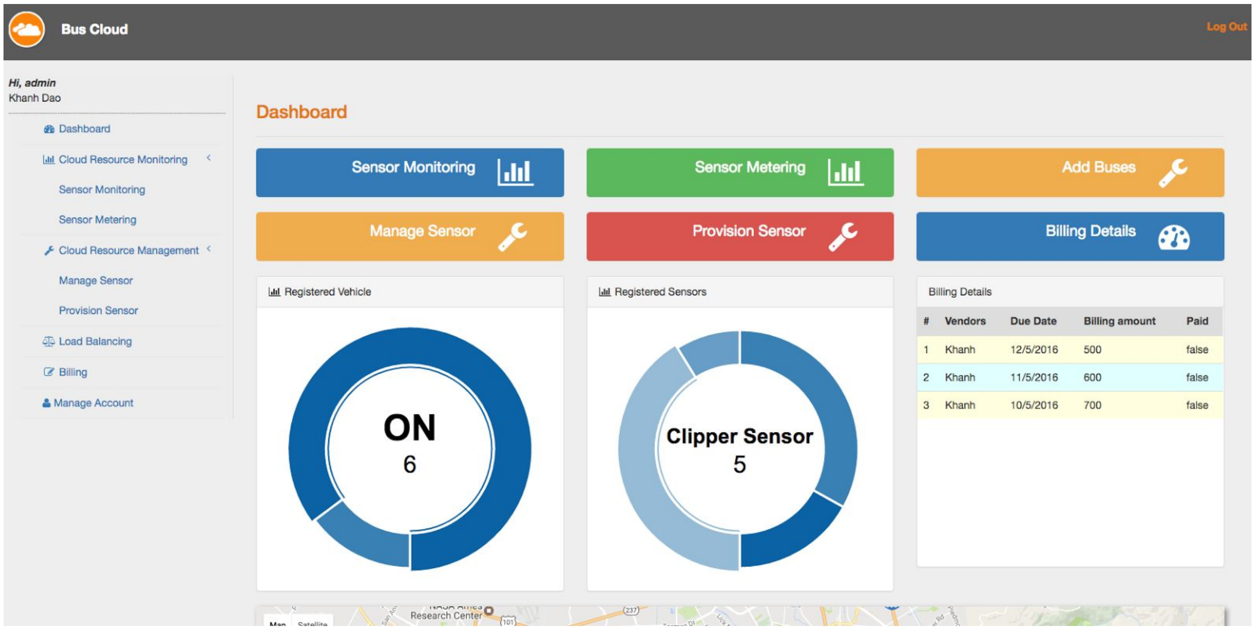
## 5.2 Login Page



Don't have an account? Sign up

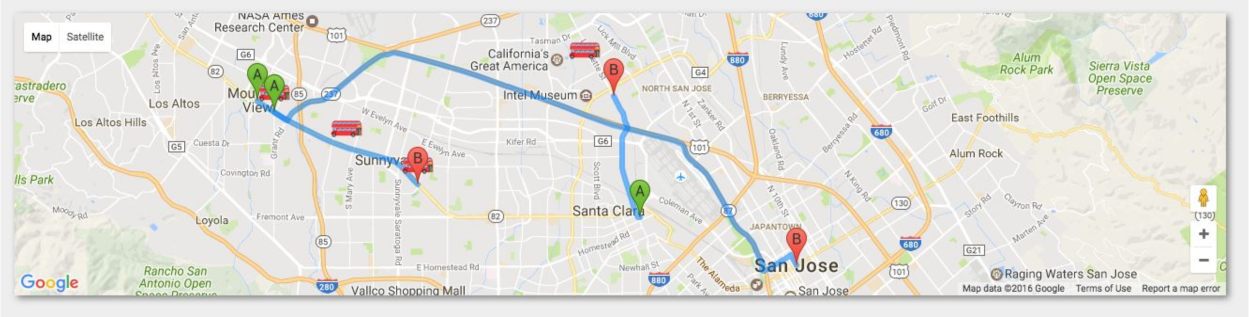
© Copyright 2016 - Bus Cloud

## 5.3 Dashboard page

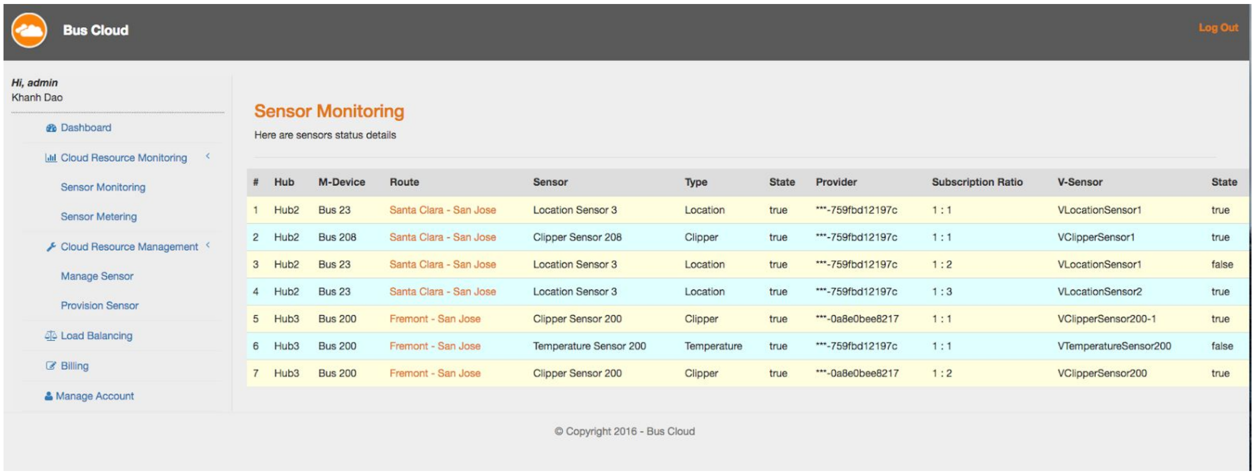


Billing Details

| # | Vendors | Due Date | Billing amount | Paid |
|---|---|---|---|---|
| 1 | Khanh | 12/5/2016 | 500 | false |
| 2 | Khanh | 11/5/2016 | 600 | false |
| 3 | Khanh | 10/5/2016 | 700 | false |



## 5.4 Sensor Monitoring Page



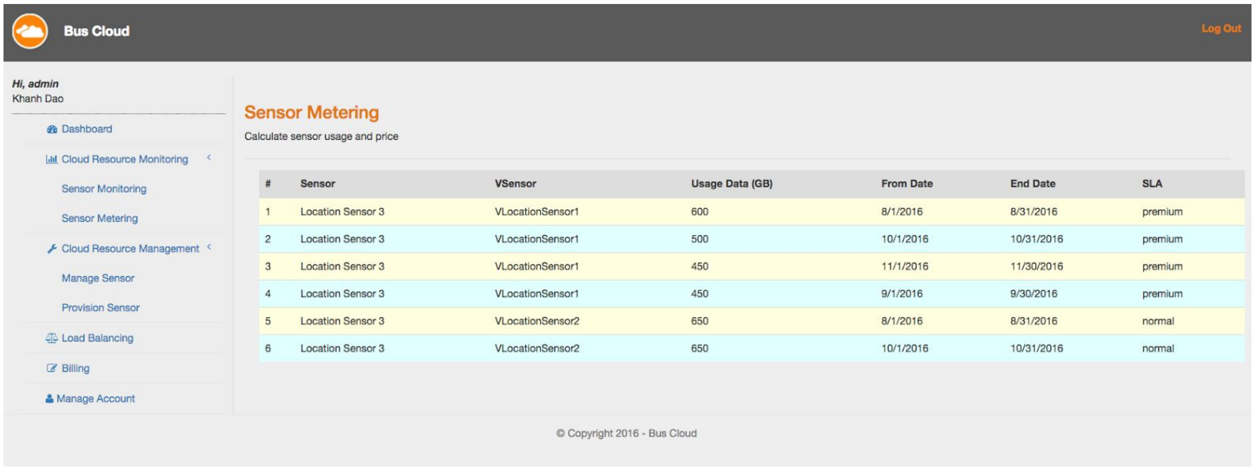
Sensor Monitoring

Here are sensors status details

| # | Hub | M-Device | Route | Sensor | Type | State | Provider | Subscription Ratio | V-Sensor | State |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Hub2 | Bus 23 | Santa Clara - San Jose | Location Sensor 3 | Location | true | ***-759fbd12197c | 1 : 1 | VLocationSensor1 | true |
| 2 | Hub2 | Bus 208 | Santa Clara - San Jose | Clipper Sensor 208 | Clipper | true | ***-759fbd12197c | 1 : 1 | VClipperSensor1 | true |
| 3 | Hub2 | Bus 23 | Santa Clara - San Jose | Location Sensor 3 | Location | true | ***-759fbd12197c | 1 : 2 | VLocationSensor1 | false |
| 4 | Hub2 | Bus 23 | Santa Clara - San Jose | Location Sensor 3 | Location | true | ***-759fbd12197c | 1 : 3 | VLocationSensor2 | true |
| 5 | Hub3 | Bus 200 | Fremont - San Jose | Clipper Sensor 200 | Clipper | true | ***-0a8e0bee8217 | 1 : 1 | VClipperSensor200-1 | true |
| 6 | Hub3 | Bus 200 | Fremont - San Jose | Temperature Sensor 200 | Temperature | true | ***-759fbd12197c | 1 : 1 | VTemperatureSensor200 | false |
| 7 | Hub3 | Bus 200 | Fremont - San Jose | Clipper Sensor 200 | Clipper | true | ***-0a8e0bee8217 | 1 : 2 | VClipperSensor200 | true |

© Copyright 2016 - Bus Cloud

## 5.5 Sensor Metering Page



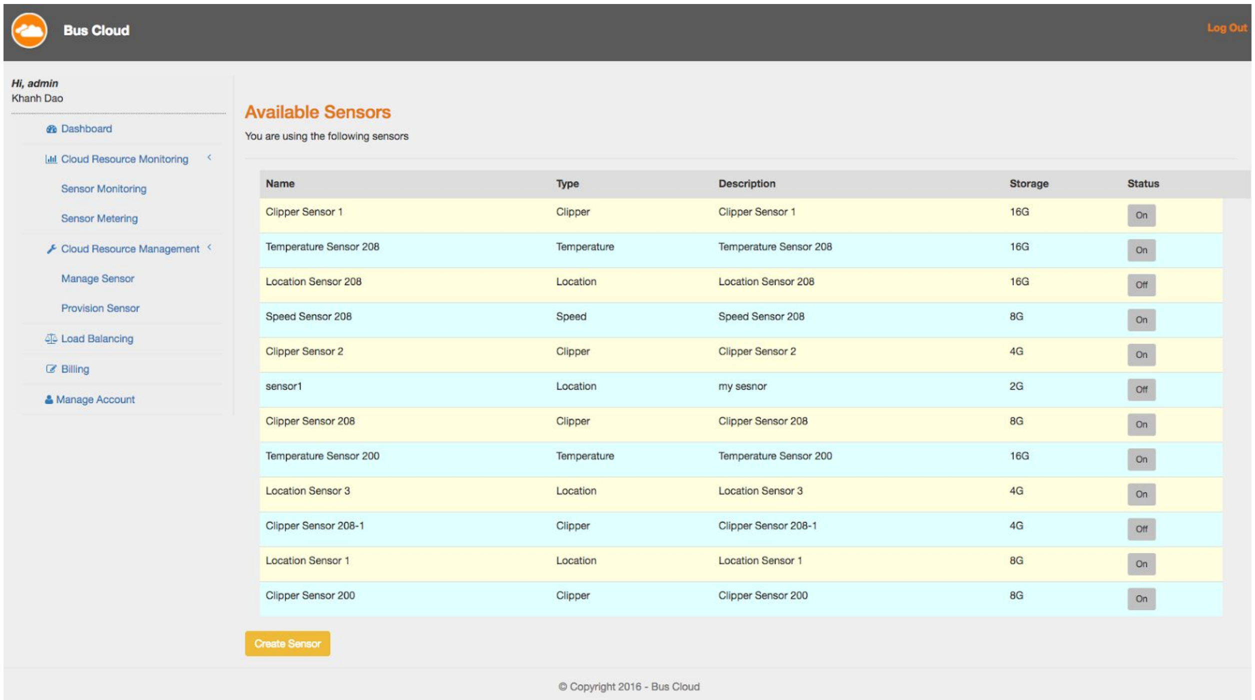
Sensor Metering

Calculate sensor usage and price

| # | Sensor | VSensor | Usage Data (GB) | From Date | End Date | SLA |
|---|---|---|---|---|---|---|
| 1 | Location Sensor 3 | VLocationSensor1 | 600 | 8/1/2016 | 8/31/2016 | premium |
| 2 | Location Sensor 3 | VLocationSensor1 | 500 | 10/1/2016 | 10/31/2016 | premium |
| 3 | Location Sensor 3 | VLocationSensor1 | 450 | 11/1/2016 | 11/30/2016 | premium |
| 4 | Location Sensor 3 | VLocationSensor1 | 450 | 9/1/2016 | 9/30/2016 | premium |
| 5 | Location Sensor 3 | VLocationSensor2 | 650 | 8/1/2016 | 8/31/2016 | normal |
| 6 | Location Sensor 3 | VLocationSensor2 | 650 | 10/1/2016 | 10/31/2016 | normal |

© Copyright 2016 - Bus Cloud

## 5.6 Manage Sensor Page

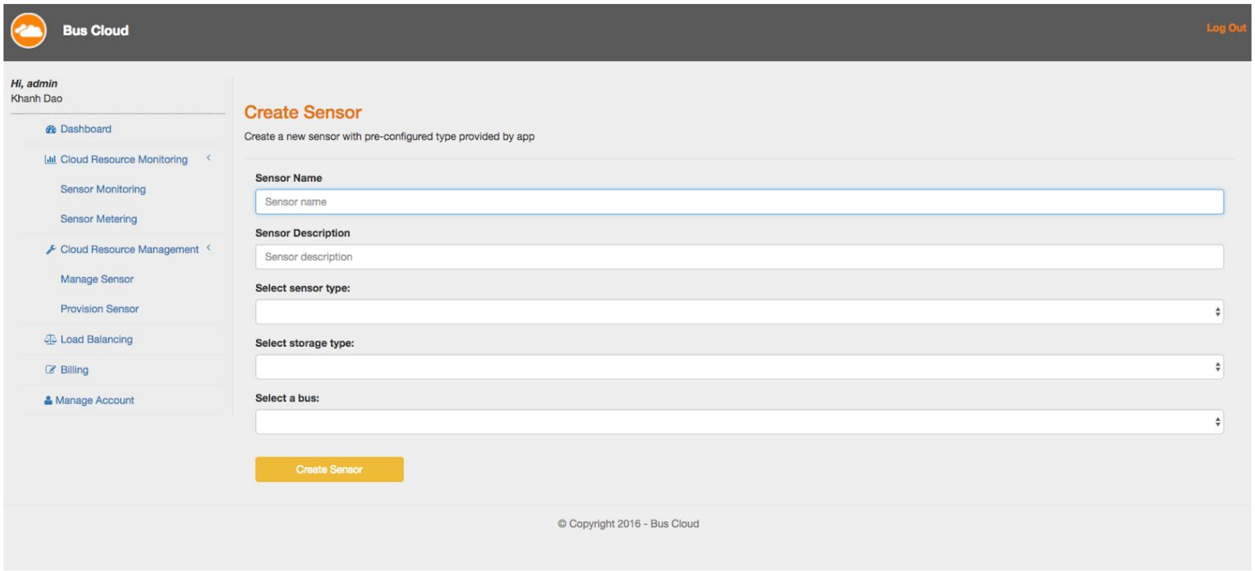


Available Sensors

You are using the following sensors

| Name | Type | Description | Storage | Status |
|---|---|---|---|---|
| Clipper Sensor 1 | Clipper | Clipper Sensor 1 | 16G | On |
| Temperature Sensor 208 | Temperature | Temperature Sensor 208 | 16G | On |
| Location Sensor 208 | Location | Location Sensor 208 | 16G | Off |
| Speed Sensor 208 | Speed | Speed Sensor 208 | 8G | On |
| Clipper Sensor 2 | Clipper | Clipper Sensor 2 | 4G | On |
| sensor1 | Location | my sesnor | 2G | Off |
| Clipper Sensor 208 | Clipper | Clipper Sensor 208 | 8G | On |
| Temperature Sensor 200 | Temperature | Temperature Sensor 200 | 16G | On |
| Location Sensor 3 | Location | Location Sensor 3 | 4G | On |
| Clipper Sensor 208-1 | Clipper | Clipper Sensor 208-1 | 4G | Off |
| Location Sensor 1 | Location | Location Sensor 1 | 8G | On |
| Clipper Sensor 200 | Clipper | Clipper Sensor 200 | 8G | On |

## 5.7 Provision Sensor Page



Create Sensor

Create a new sensor with pre-configured type provided by app

Sensor Name

Sensor name

Sensor Description

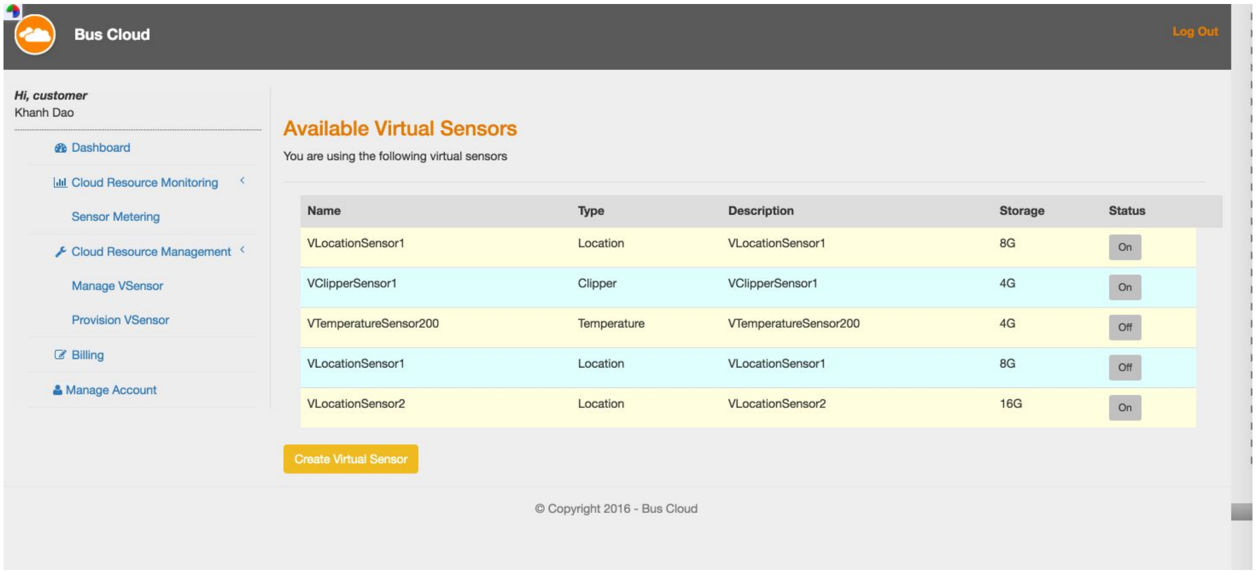
Sensor description

Select sensor type:

Select storage type:

Select a bus:

## 5.8 Manage Virtual Sensor Page



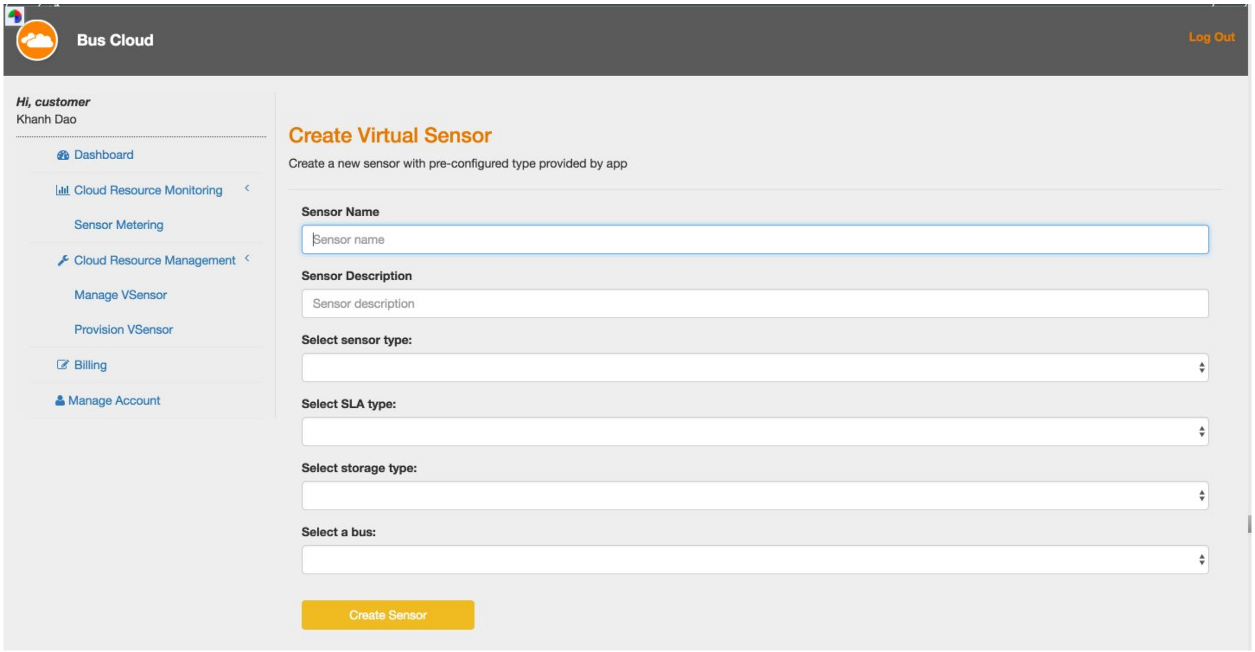
Available Virtual Sensors

You are using the following virtual sensors

| Name | Type | Description | Storage | Status |
|---|---|---|---|---|
| VLocationSensor1 | Location | VLocationSensor1 | 8G | On |
| VClipperSensor1 | Clipper | VClipperSensor1 | 4G | On |
| VTemperatureSensor200 | Temperature | VTemperatureSensor200 | 4G | Off |
| VLocationSensor1 | Location | VLocationSensor1 | 8G | Off |
| VLocationSensor2 | Location | VLocationSensor2 | 16G | On |

© Copyright 2016 - Bus Cloud

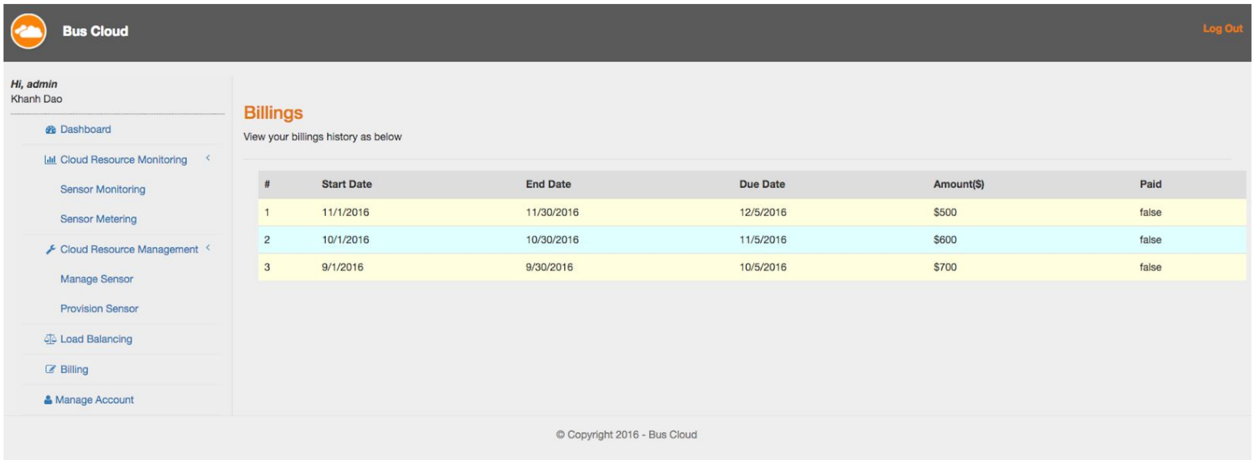
## 5.9 Provision Virtual Sensor Page



Create Virtual Sensor

Create a new sensor with pre-configured type provided by app

## 5.10 Billing Page

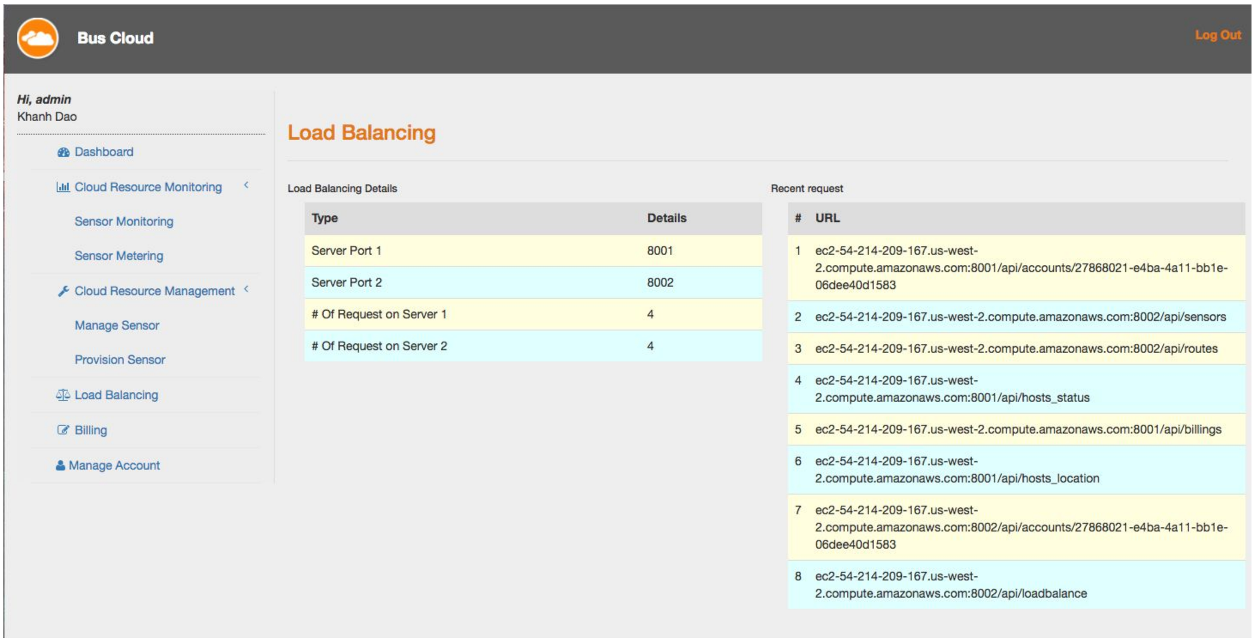


Billings

View your billings history as below

| # | Start Date | End Date | Due Date | Amount($) | Paid |
|---|---|---|---|---|---|
| 1 | 11/1/2016 | 11/30/2016 | 12/5/2016 | $500 | false |
| 2 | 10/1/2016 | 10/30/2016 | 11/5/2016 | $600 | false |
| 3 | 9/1/2016 | 9/30/2016 | 10/5/2016 | $700 | false |

## 5.11 Load Balancing Page



Load Balancing

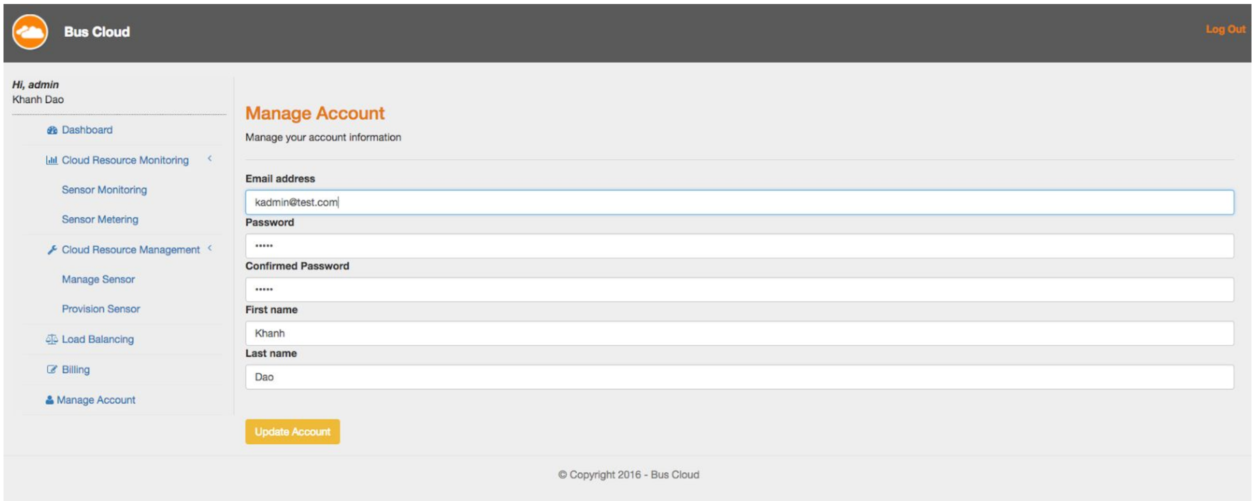
Load Balancing Details

| Type | Details |
|---|---|
| Server Port 1 | 8001 |
| Server Port 2 | 8002 |
| # Of Request on Server 1 | 4 |
| # Of Request on Server 2 | 4 |

Recent request

| # | URL |
|---|---|
| 1 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/accounts/27868021-e4ba-4a11-bb1e-06dee40d1583 |
| 2 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/sensors |
| 3 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/routes |
| 4 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/hosts_status |
| 5 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/billings |
| 6 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/hosts_location |
| 7 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/accounts/27868021-e4ba-4a11-bb1e-06dee40d1583 |
| 8 | ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/loadbalance |

## 5.12 Manage User Account Page



Manage Account

Manage your account information

Email address: kadmin@test.com

Password: •••••

Confirmed Password: •••••

First name: Khanh

Last name: Dao

# 6. API Design

## 6.1 Account Manager

| **Action** | User registration |
|---|---|
| **URI** | {host_name}/api/accounts |
| **HTTP Method** | POST |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",`<br>`}` |
| **Body** | `{`<br>`    firstName: <first_name>,`<br>`    lastName: <last_name>,`<br>`    email: <email>,`<br>`    password: <password>,`<br>`    roles: vendor|admin|customer`<br>`}` |

| | |
|---|---|
| **Return success (data + statusCode)** | HTTP/1.1 201 OK<br>{<br>    id: <user_id>,<br>    firstName: <first_name>,<br>    lastName: <last_name>,<br>    email: <email>,<br>    password: <password>,<br>    roles: vendor\|admin\|customer<br>} |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error -- Fail to fetch data<br>HTTP/1.1 404 User already exist -- Fail to fetch data |

| **Action** | User login |
|---|---|
| **URI** | {host_name}/api/login |
| **HTTP Method** | POST |
| **Headers** | {<br>    Content-Type: "application/json, charset=utf-8",<br>} |
| **Body** | {<br>    email: <email>,<br>    password: <plain_password><br>} |
| **Return success (data + statusCode)** | HTTP/1.1 200 OK<br>{<br>    id: <user_id>,<br>    firstName: <first_name>,<br>    lastName: <last_name>,<br>    email: <email>,<br>    password: <password>, |

| | |
|---|---|
| | `roles: vendor|admin|customer`<br>`}` |
| **Return failure (statusCode)** | HTTP/1.1 404 User already exist -- Fail to fetch data |

| **Action** | User edit profile |
|---|---|
| **URI** | {host_name}/api/accounts/:id |
| **HTTP Method** | PUT |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",`<br>`}` |
| **Body** | `{`<br>`    firstName: <first_name>,`<br>`    lastName: <last_name>,`<br>`    email: <email>,`<br>`    password: <password>`<br>`}` |
| **Return success (data + statusCode)** | HTTP/1.1 200 OK |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

## 6.2 Payment Manager

| **Action** | Register payment |
|---|---|
| **URI** | {host_name}/api/payment |

| | |
|---|---|
| **HTTP Method** | POST |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",`<br>`    u: <user_id>`<br>`}` |
| **Body** | `{`<br>`    card_number: <card_number>,`<br>`    card_owner: <card_onwer>,`<br>`    expiration_date: <expiration_date>,`<br>`    address: <address>,`<br>`    zip: <zip>,`<br>`    state: <state>`<br>`}` |
| **Return success (data + statusCode)** | HTTP/1.1 201 OK |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

## 6.3 Bus (Host) Manager

| | |
|---|---|
| **Action** | Get All Buses (per users) |
| **URI** | {host_name}/api/hosts |
| **HTTP Method** | GET |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",`<br>`    u: <user_id>`<br>`}` |
| **Return success (data** | HTTP/1.1 200 OK |

| | |
|---|---|
| **+ statusCode)** | `[{`<br>`    id: <bus_id>,`<br>`    name: <bus_name>,`<br>`    description: <bus_description>,`<br>`    ip: <ip>`<br>`    status: <status>`<br>`}, ...]` |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

| **Action** | Create Bus |
|---|---|
| **URI** | {host_name}/api/host |
| **HTTP Method** | POST |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",`<br>`    u: <user_id>`<br>`}` |
| **Body** | `{`<br>`    name: <bus_name>,`<br>`    description: <bus_description>`<br>`}` |
| **Return success (data + statusCode)** | HTTP/1.1 201 OK<br>`{`<br>`    id: <bus_id>,`<br>`    name: <bus_name>,`<br>`    description: <bus_description>,`<br>`    ip: <ip>`<br>`    status: <status>`<br>`}` |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

| | |
|---|---|
| **Action** | Delete Bus |
| **URI** | {host_name}/api/host/:id |
| **HTTP Method** | DELETE |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",`<br>`    u: <user_id>`<br>`}` |
| **Return success (data + statusCode)** | HTTP/1.1 200 OK |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

## 6.4 Routes Manager

| | |
|---|---|
| **Action** | Get All Sensors (per users) |
| **URI** | {host_name}/api/routes |
| **HTTP Method** | GET |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",`<br>`    u: <user_id>`<br>`}` |
| **Return success (data + statusCode)** | HTTP/1.1 200 OK<br>`[{`<br>`    id: <route_id>,`<br>`    src_latitude: <src_latitude>,`<br>`    src_longitude: <src_longitude>,` |

| | |
|---|---|
| | dest_latitude: <dest_latitude>,<br>dest_longitude: <dest_longitude>,<br>description: <description>,<br>url: <url><br>}, ...] |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

| **Action** | Get All Sensors (per users) |
|---|---|
| **URI** | {host_name}/api/sensors |
| **HTTP Method** | GET |
| **Headers** | {<br>Content-Type: "application/json, charset=utf-8",<br>u: <user_id><br>} |
| **Return success (data + statusCode)** | HTTP/1.1 200 OK<br>[{<br>id: <sensor_id>,<br>type: <sensor_type>,<br>storages: <sensor_storages>,<br>name: <sensor_name>,<br>description: <sensor_description>,<br>host_id: <host_id><br>status: <status><br>}, ...] |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

| **Action** | Create Sensor |
|---|---|
| **URI** | {host_name}/api/sensors |

| | |
|---|---|
| **HTTP Method** | POST |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",`<br>`    u: <user_id>`<br>`}` |
| **Body** | `{`<br>`    type: <sensor_type>,`<br>`    storages: <sensor_storages>,`<br>`    name: <sensor_name>,`<br>`    description: <sensor_description>,`<br>`    host_id: <host_id>`<br>`    status: <status>`<br>`}` |
| **Return success (data + statusCode)** | HTTP/1.1 201 OK<br>`{`<br>`    id: <sensor_id>,`<br>`    type: <sensor_type>,`<br>`    storages: <sensor_storages>,`<br>`    name: <sensor_name>,`<br>`    description: <sensor_description>,`<br>`    host_id: <host_id>`<br>`    status: <status>`<br>`}` |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

| | |
|---|---|
| **Action** | Delete Sensor |
| **URI** | {host_name}/api/sensors/:id |
| **HTTP Method** | DELETE |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",` |

| | |
|---|---|
| | `    u: <user_id>`<br>`}` |
| **Return success (data + statusCode)** | HTTP/1.1 200 OK |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

## 6.5 Virtual Sensors Manager

| **Action** | Get All Virtual Sensors (per users) |
|---|---|
| **URI** | {host_name}/api/vsensors |
| **HTTP Method** | GET |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",`<br>`    u: <user_id>`<br>`}` |
| **Return success (data + statusCode)** | HTTP/1.1 200 OK<br>`{`<br>`    "data": [{`<br>`        "Sensor": <sensor_obj>,`<br>`        "VirtualSensor": <vsensor_obj>`<br>`    }],`<br>`    "metadata": {`<br>`        "clipper": <vsensor_clipper_count>,`<br>`        "location": <vsensor_location_count>,`<br>`        "speed": <vsensor_speed_count>,`<br>`        "temperature": <vsensor_temperature_count>`<br>`    }`<br>`}` |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

| | |
|---|---|
| **Action** | Create Virtual Sensor |
| **URI** | {host_name}/api/vsensors |
| **HTTP Method** | POST |
| **Headers** | `{`<br>`    Content-Type: "application/json, charset=utf-8",`<br>`    u: <user_id>`<br>`}` |
| **Body** | `{`<br>`    type: <vsensor_type>,`<br>`    storages: <vsensor_storages>,`<br>`    name: <vsensor_name>,`<br>`    description: <vsensor_description>,`<br>`    host_id: <host_id>,`<br>`    sla_id: <sla_id>`<br>`}` |
| **Return success (data + statusCode)** | HTTP/1.1 201 OK<br>`{`<br>`    id: <vsensor_id>,`<br>`    type: <vsensor_type>,`<br>`    storages: <vsensor_storages>,`<br>`    name: <vsensor_name>,`<br>`    description: <vsensor_description>,`<br>`    host_id: <host_id>,`<br>`    sla_id: <sla_id>,`<br>`    status: <status>`<br>`}` |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

## 6.6 Monitor

<table>
<tr><td><b>Action</b></td><td>Sensor monitoring (for admin only)</td></tr>
<tr><td><b>URI</b></td><td>{host_name}/api/monitor/statistics</td></tr>
<tr><td><b>HTTP Method</b></td><td>GET</td></tr>
<tr><td><b>Headers</b></td><td>

```
{
    Content-Type: "application/json, charset=utf-8",
    u: <user_id>
}
```

</td></tr>
<tr><td><b>Return success (data + statusCode)</b></td><td>HTTP/1.1 200 OK

```
[{
    Sensor: {
       Host: {
           Route: {},
           SensorHub: {}
       },
       ...
    }
    VirtualSensor: {...}
}, ...]
```

</td></tr>
<tr><td><b>Return failure (statusCode)</b></td><td>HTTP/1.1 500 Server Error</td></tr>
</table>

## 6.7 Metering (Usage)

<table>
<tr><td><b>Action</b></td><td>Sensor metering</td></tr>
<tr><td><b>URI</b></td><td>{host_name}/api/usage</td></tr>
<tr><td><b>HTTP Method</b></td><td>GET</td></tr>
<tr><td><b>Headers</b></td><td>

```
{
    Content-Type: "application/json, charset=utf-8",
    u: <user_id>
}
```

</td></tr>
</table>

<table>
<tr>
<td><b>Return success (data + statusCode)</b></td>
<td>

HTTP/1.1 200 OK

```
[{
    TransactionManager: {
        SLA: <SLA_object>,
        Sensor: <Sensor_object>,
        VirtualSensor: <VirtualSensor_object>
    },
    data: <amount_data_usage>,
    fromDate: <from_date>,
    endDate: <end_date>
}, ...]
```

</td>
</tr>
<tr>
<td><b>Return failure (statusCode)</b></td>
<td>HTTP/1.1 500 Server Error</td>
</tr>
</table>

## Get Status (ON/OFF) Vehicle

<table>
<tr>
<td><b>Action</b></td>
<td>Get All Sensors (per users)</td>
</tr>
<tr>
<td><b>URI</b></td>
<td>{host_name}/api/status</td>
</tr>
<tr>
<td><b>HTTP Method</b></td>
<td>GET</td>
</tr>
<tr>
<td><b>Headers</b></td>
<td>

```
{
    Content-Type: "application/json, charset=utf-8",
    u: <user_id>
}
```

</td>
</tr>
<tr>
<td><b>Return success (data + statusCode)</b></td>
<td>

HTTP/1.1 200 OK

```
{
   "active":"4",
   "inactive":"3"
}
```

</td>
</tr>
<tr>
<td><b>Return failure (statusCode)</b></td>
<td>HTTP/1.1 500 Server Error</td>
</tr>
</table>

### Get Count of all Sensors

| | |
|---|---|
| **Action** | Get All Sensors (per users) |
| **URI** | {host_name}/api/sensors |
| **HTTP Method** | GET |
| **Headers** | {<br>    Content-Type: "application/json, charset=utf-8",<br>    u: <user_id><br>} |
| **Return success (data + statusCode)** | HTTP/1.1 200 OK<br>{<br>"location":"6",<br>"clipper":"3",<br>"speed":"45",<br>"temperature":"27"<br>} |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

### Get Admin Billing information

| | |
|---|---|
| **Action** | Get All Sensors (per users) |
| **URI** | {host_name}/api/billing |
| **HTTP Method** | GET |
| **Headers** | {<br>    Content-Type: "application/json, charset=utf-8",<br>    u: <user_id><br>} |

| | |
|---|---|
| **Return success (data + statusCode)** | HTTP/1.1 200 OK<br>[<br>{<br>"userid":"user1",<br>"amount":"200",<br>"ispaid":"yes",<br>"date":"01/01/2016"<br>},<br>{<br>"userid":"user2",<br>"amount":"600",<br>"ispaid":"no",<br>"date":"01/02/2016"<br>}<br>] |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

## 6.8 Get Billing information per month

| **Action** | Get All Sensors (per users) |
|---|---|
| **URI** | {host_name}/api/billamount |
| **HTTP Method** | GET |
| **Headers** | {<br>Content-Type: "application/json, charset=utf-8",<br>u: <user_id><br>} |
| **Return success (data + statusCode)** | HTTP/1.1 200 OK<br>[{<br>"a":"200.00",<br>"y":"January"<br>},{<br>"a":"300.00", |

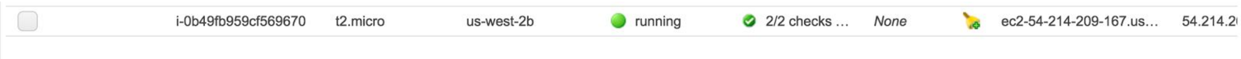
| | |
|---|---|
| | `"y":"February"`<br>`}]` |
| **Return failure (statusCode)** | HTTP/1.1 500 Server Error |

# 8. Mobile sensor cloud system services

| **Services** | **Description** |
|---|---|
| Mobile Sensor Network as a Service | This service allow users to register, configure, and manage different types of sensor networks (sensor clusters) (or mobile sensor clusters).<br>(connectivity between sensor -> sensor cluster) |
| Mobile Sensor as a Service | Perform on-demand automatic provision of different sensor clusters with static or dynamic connectivity |
| Mobile Sensor cloud resources Monitoring | Monitor our provisioned sensors , Display the data values and their availability |
| Mobile Sensor Data as a Service | Provide Database Storage (S3) for storing the Sensor Data in Cloud |
| SEaaS (Sensor Event as a Service) | Change in the Sensor Event is reported. |

## 8.1 Deploy Application to EC2

1. Create new EC2 instance



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| □ | i-0b49fb959cf569670 | t2.micro | us-west-2b | ● running | ✓ 2/2 checks ... | None | ec2-54-214-209-167.us... | 54.214.2 |

2. Install NodeJS and NPM in your ubuntu, also install git

```
sudo apt-get install nodejs

sudo npm install npm -g
```

3. Configure port 80 to route to port 8080 with the following command

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to 8080
```

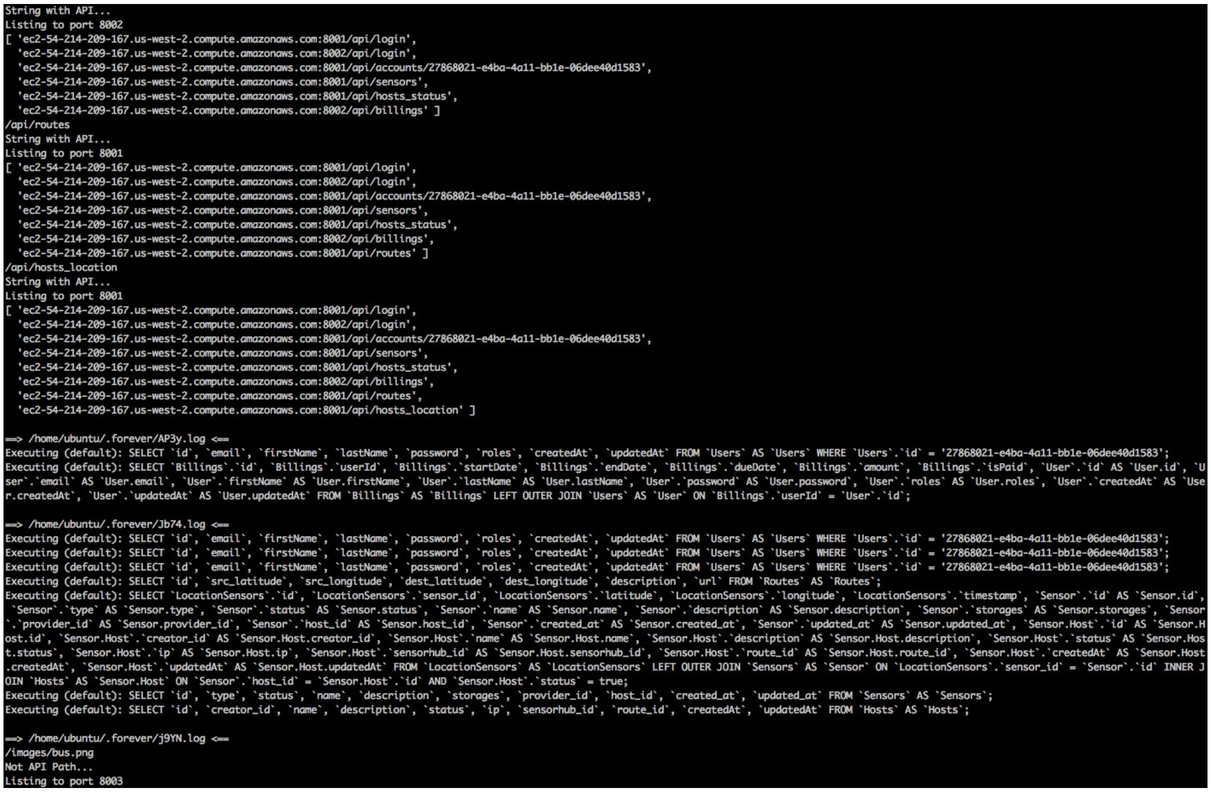
4. Install “npm forever” with the command below

```
npm i forever -g
```

5. Run application

```
forever start index.js 8001 & forever start index.js 8002 & forever start index.js
8003 & forever start load-balancer.js 8080
```

6. Monitor (load balance on 3 ports)



```
String with API...
Listing to port 8002
[ 'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/login',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/login',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/accounts/27868021-e4ba-4a11-bb1e-06dee40d1583',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/sensors',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/hosts_status',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/billings' ]
/api/routes
String with API...
Listing to port 8001
[ 'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/login',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/login',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/accounts/27868021-e4ba-4a11-bb1e-06dee40d1583',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/sensors',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/hosts_status',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/billings',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/routes' ]
/api/hosts_location
String with API...
Listing to port 8001
[ 'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/login',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/login',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/accounts/27868021-e4ba-4a11-bb1e-06dee40d1583',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/sensors',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/hosts_status',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8002/api/billings',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/routes',
  'ec2-54-214-209-167.us-west-2.compute.amazonaws.com:8001/api/hosts_location' ]

==> /home/ubuntu/.forever/AP3y.log <==
Executing (default): SELECT `id`, `email`, `firstName`, `lastName`, `password`, `roles`, `createdAt`, `updatedAt` FROM `Users` AS `Users` WHERE `Users`.`id` = '27868021-e4ba-4a11-bb1e-06dee40d1583';
Executing (default): SELECT `Billings`.`id`, `Billings`.`userId`, `Billings`.`startDate`, `Billings`.`endDate`, `Billings`.`dueDate`, `Billings`.`amount`, `Billings`.`isPaid`, `User`.`id` AS `User.id`, `U
ser`.`email` AS `User.email`, `User`.`firstName` AS `User.firstName`, `User`.`lastName` AS `User.lastName`, `User`.`password` AS `User.password`, `User`.`roles` AS `User.roles`, `User`.`createdAt` AS `Use
r.createdAt`, `User`.`updatedAt` AS `User.updatedAt` FROM `Billings` AS `Billings` LEFT OUTER JOIN `Users` AS `User` ON `Billings`.`userId` = `User`.`id`;

==> /home/ubuntu/.forever/Jb74.log <==
Executing (default): SELECT `id`, `email`, `firstName`, `lastName`, `password`, `roles`, `createdAt`, `updatedAt` FROM `Users` AS `Users` WHERE `Users`.`id` = '27868021-e4ba-4a11-bb1e-06dee40d1583';
Executing (default): SELECT `id`, `email`, `firstName`, `lastName`, `password`, `roles`, `createdAt`, `updatedAt` FROM `Users` AS `Users` WHERE `Users`.`id` = '27868021-e4ba-4a11-bb1e-06dee40d1583';
Executing (default): SELECT `id`, `email`, `firstName`, `lastName`, `password`, `roles`, `createdAt`, `updatedAt` FROM `Users` AS `Users` WHERE `Users`.`id` = '27868021-e4ba-4a11-bb1e-06dee40d1583';
Executing (default): SELECT `id`, `src_latitude`, `src_longitude`, `dest_latitude`, `dest_longitude`, `description`, `url` FROM `Routes` AS `Routes`;
Executing (default): SELECT `LocationSensors`.`id`, `LocationSensors`.`sensor_id`, `LocationSensors`.`latitude`, `LocationSensors`.`longitude`, `LocationSensors`.`timestamp`, `Sensor`.`id` AS `Sensor.id`,
 `Sensor`.`type` AS `Sensor.type`, `Sensor`.`status` AS `Sensor.status`, `Sensor`.`name` AS `Sensor.name`, `Sensor`.`description` AS `Sensor.description`, `Sensor`.`storages` AS `Sensor.storages`, `Sensor
`.`provider_id` AS `Sensor.provider_id`, `Sensor`.`host_id` AS `Sensor.host_id`, `Sensor`.`created_at` AS `Sensor.created_at`, `Sensor`.`updated_at` AS `Sensor.updated_at`, `Sensor.Host`.`id` AS `Sensor.H
ost.id`, `Sensor.Host`.`creator_id` AS `Sensor.Host.creator_id`, `Sensor.Host`.`name` AS `Sensor.Host.name`, `Sensor.Host`.`description` AS `Sensor.Host.description`, `Sensor.Host`.`status` AS `Sensor.Hos
t.status`, `Sensor.Host`.`ip` AS `Sensor.Host.ip`, `Sensor.Host`.`sensorhub_id` AS `Sensor.Host.sensorhub_id`, `Sensor.Host`.`route_id` AS `Sensor.Host.route_id`, `Sensor.Host`.`createdAt` AS `Sensor.Host
.createdAt`, `Sensor.Host`.`updatedAt` AS `Sensor.Host.updatedAt` FROM `LocationSensors` AS `LocationSensors` LEFT OUTER JOIN `Sensors` AS `Sensor` ON `LocationSensors`.`sensor_id` = `Sensor`.`id` INNER J
OIN `Hosts` AS `Sensor.Host` ON `Sensor`.`host_id` = `Sensor.Host`.`id` AND `Sensor.Host`.`status` = true;
Executing (default): SELECT `id`, `type`, `status`, `name`, `description`, `storages`, `provider_id`, `host_id`, `created_at`, `updated_at` FROM `Sensors` AS `Sensors`;
Executing (default): SELECT `id`, `creator_id`, `name`, `description`, `status`, `ip`, `sensorhub_id`, `route_id`, `createdAt`, `updatedAt` FROM `Hosts` AS `Hosts`;

==> /home/ubuntu/.forever/j9YN.log <==
/images/bus.png
Not API Path...
Listing to port 8003
```

## 9. Components Testing

## 10. Technology Selection and Usage

- MySQL

- NodeJS
    - Sequelize
    - Express
    - bcrypt
    - csrf-crypto
    - http-proxy
- Handlebar
- Angular
- Bootstrap UI
- AWS Cloud Management

# 11. IMPLEMENTATIONS

## 11.1 Development Plan

| # | Milestone | Due | Assignee | Progress | Status |
|---|---|---|---|---|---|
| 1 | Build high level system design | Oct 21st | | | |
| | Define Objective & scope | | | | |
| | Brainstorm system components | | | | |
| | Develop Infrastructure design | | | | |
| | Develop Component design | | | | |
| | Brainstorm & define system services | | | | |
| | Finalize equipment & tool to develop project application | | | | |
| 2 | Prepare for environment | | | | |
| | Development env | | | | |
| | Deployment env | | | | |
| 3 | Component Design and Development | Nov 4th | | | |
| | Design Component functions (function partition/process/algorithm) | | | | |
| | Design Component API | | | | |
| | Design Component Logic (class diagram) | | | | |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  | Design Component GUI<br>(GUI layout and operation flow) |  |  |  |  |
|  | Develop prototype for component |  |  |  |  |
|  | Demo & gather feedbacks |  |  |  |  |
| 4 | Final Development & Presentation | Dec 9th |  |  |  |
|  | Brainstorm for modification ideas based on feedbacks |  |  |  |  |
|  | Upgrade application |  |  |  |  |
|  | Testing |  |  |  |  |
|  | Cosmetic update for UI and refactor code |  |  |  |  |
|  | Compose presentation slides |  |  |  |  |

## 10. Conclusion & Future Works

## 11. References

- http://www.lauradhamilton.com/how-to-set-up-a-nodejs-web-server-on-amazon-ec2
- https://github.com/sequelize/sequelize