Sri Lanka Institute of Information Technology



Assignment 2

Fire Alarm Monitoring System

IT 18115444 - Ratnasooriya K.A.L.L - [lahirulakruwan5@gmail.com](mailto:lahirulakruwan5@gmail.com) 0771128607

IT 18121834 – Jayasekara A.S - [anujshashimal456@gmail.com](mailto:anujshashimal456@gmail.com) 0703049454

IT 18126020 - Ranjith K.H.V.S - [subasith@gmail.com](mailto:subasith@gmail.com) 077 2181220

IT 18121766 – Kariyapperuma K.A.D.R.L. - [ravindu.kariyapperuma@gmail.com](mailto:ravindu.kariyapperuma@gmail.com) 071 6371010

Github URL - <https://github.com/anujshashimal/Fire-Alarm-Monitoring-System>

Video URL - <https://www.youtube.com/watch?v=WEhnAvFFyss>

**Distributed Systems - SE3020**

B.Sc. (Hons) in Information Technology Specialized in Software- Engineering

**Introduction**

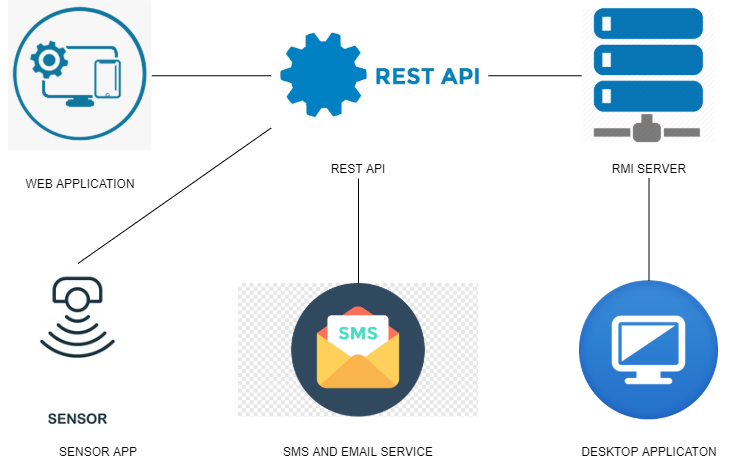
An application that allows recognizing the co2 and smoke level in the system. The system mainly consists of four major components. There are Desktop Application, Web Application, RMI server, and Rest API. On the web, the client application user can view the status of all alarm sensors updated by the sensor app every 15 sec. It will give a warning message when the sensor app details such as co2 and the smoke level go up. The condition consists of smoke level(1-10) and CO2 level (1-10). For the web application, we used to React, MongoDB. For the Rest API we used NodeJS to developed.

A desktop client application where the user can view the same information from the desktop client where the sensor app sent details. This information refresh by every 10 sec. The desktop client application consists of two portals. One is the administrator login. Only an administrator can add a sensor to the system. As well as the administrator can edit sensor details. An alert is displayed on the screen when the CO2 and Smoke level is move to the more than 5. Email and SMS services are combined with the rest APIs when the smoke level and CO2 level goes up. For the Desktop application, we used java to develop the system and the Database is MongoDB. RMI server also built using java.

For the sensor app, we used JSON Data to send the dummy values. It is also run on the localhost. We used JSON-SERVER to it. Through JSON-SERVER it will send the sensor updated details to both web and client applications. According to dummy JSON values, web and desktop application updated their details from time to time. In the side of Desktop application, it will be received sensor details through Rest API and RMI server. On the side of a web application, it will receive sensor details through the REST API.

To the run email service, we used nodemailer service which is the npm module. To run an SMS service, we used the TELESIGN service to di it. Both services are implemented in the rest API. When the co2 and Smoke level goes up it will be sending an email and SMS to a specific phone number and email. There are also consist of the REST API.

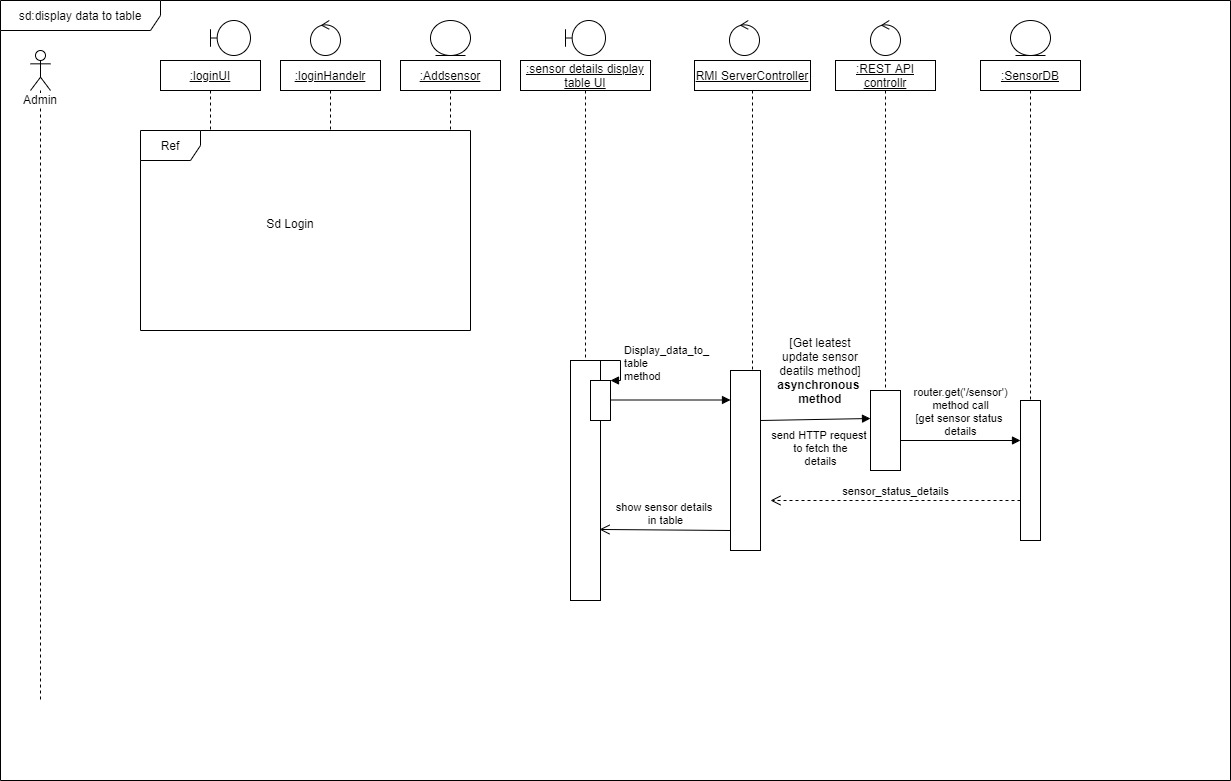
**High Level Architecture Diagram**



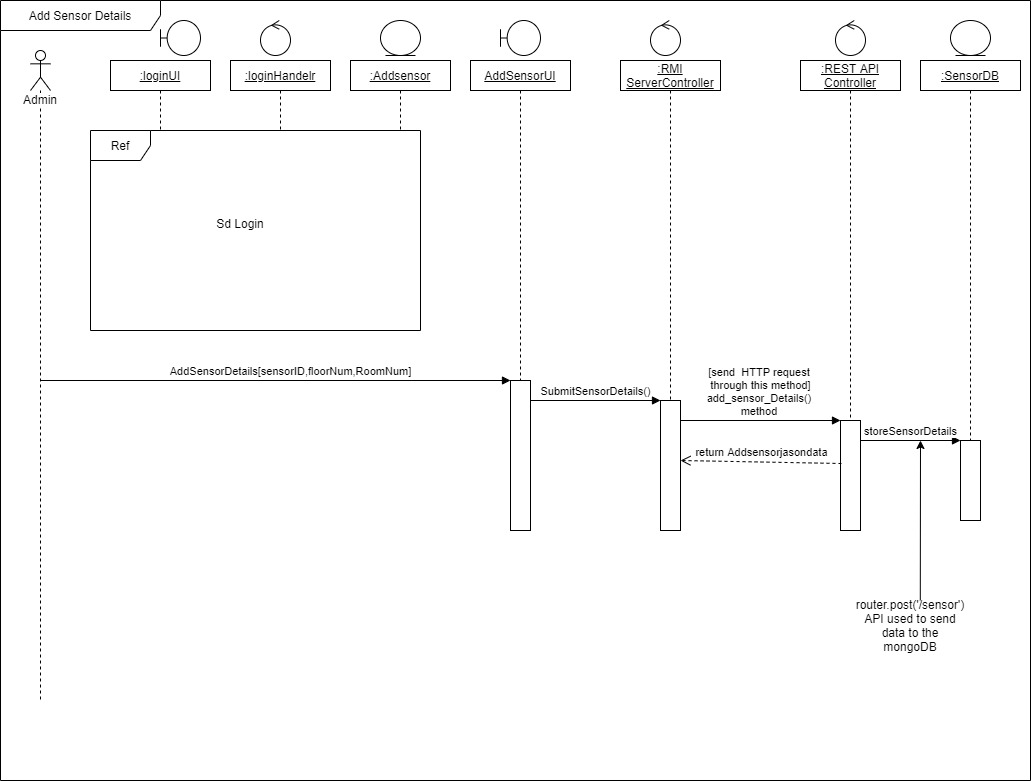
Through the desktop application, the admin can log to the system and register the sensors in the system. Then the sensor will send the update through the rest API to both desktop and web applications. The desktop application will receive that update through the RMI server. When the sensor app’s CO2 and smoke level is higher than 5, Both desktop and web application display an error message. As well as received email and SMS to the relevant emails and phone numbers. RMI interface consists of methods connected to the REST API. RMI Interface’s methods are implemented in the RMI server. That implemented method included URL connect to the REST API. RMI Server and RMI client connected with each other’s (RMI Registry).  When the co2 and smoke level goes up it will call the rest API’s method. RMI server’s asynchronous method called by the Rest APIs. That method will be executed asynchronously.  When the Abnormal situations come, Rest’s method will fire up and sent to the email and SMS to the relevant details. On the other hand, abnormal situations come to the web using a sensor app through the rest API, that method will fire. SMS is configured using TELESIGN service and email service configured using nodemailer.

**Sequence Diagrams**

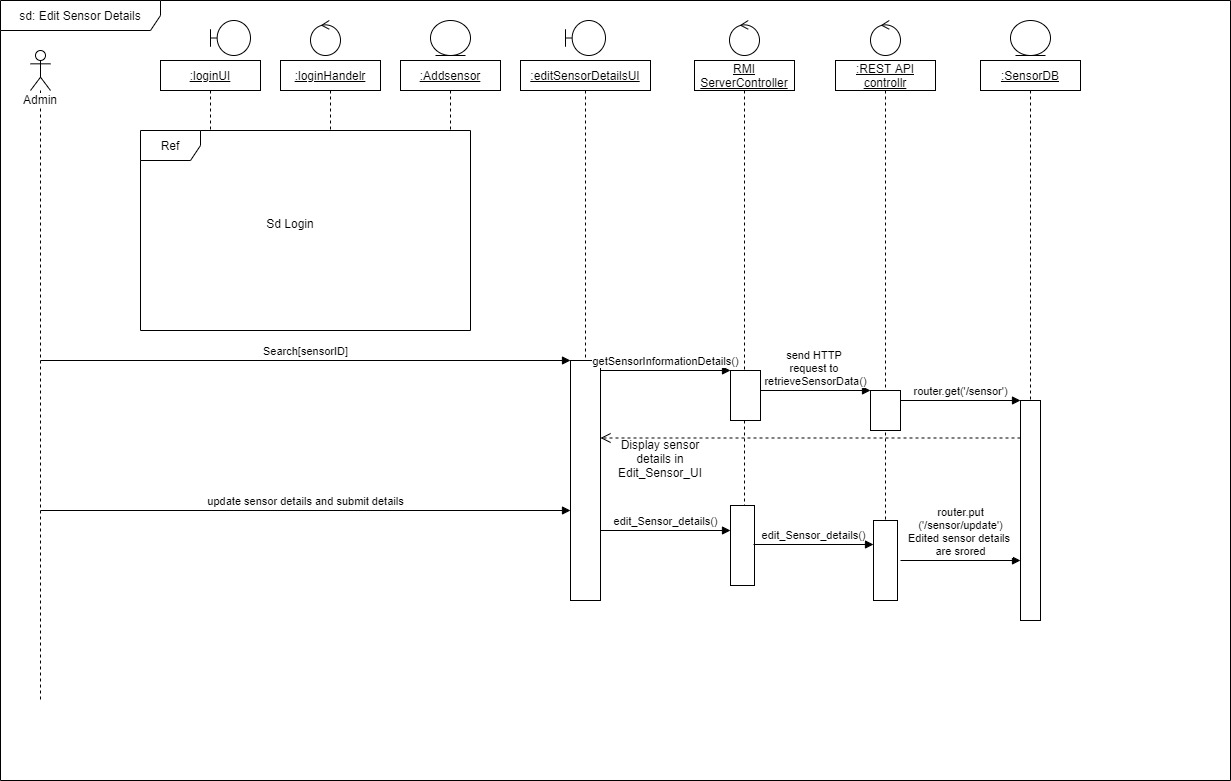
Display data to the desktop application from the received sensor APP. When admin logging into the system and he moves to the sensor details interface all sensor details are displayed in the table.



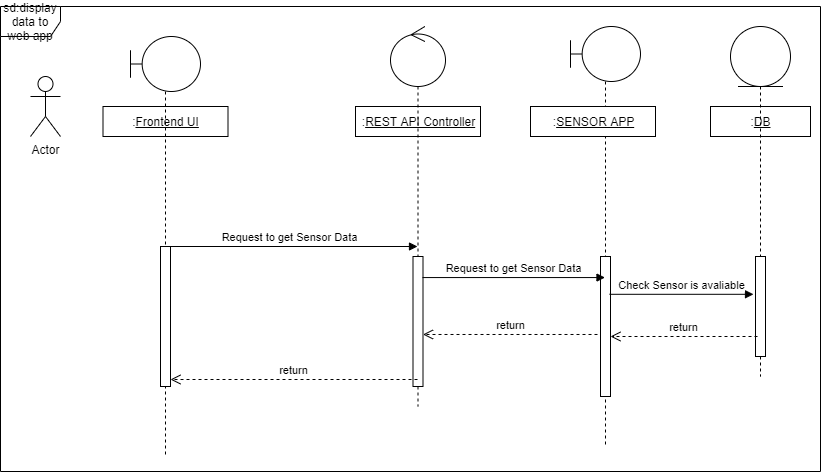
Add a sensor to the Database. Only the admin can add the sensor to the system. When admin logging into the system and he moves to the sensor details interface and there is a search button & sensor id text box. So admin Enter the edit sensor id and press the search button sensor details are displayed in text boxes. So admin can edit the sensor details and then press the update button sensor details are updated.



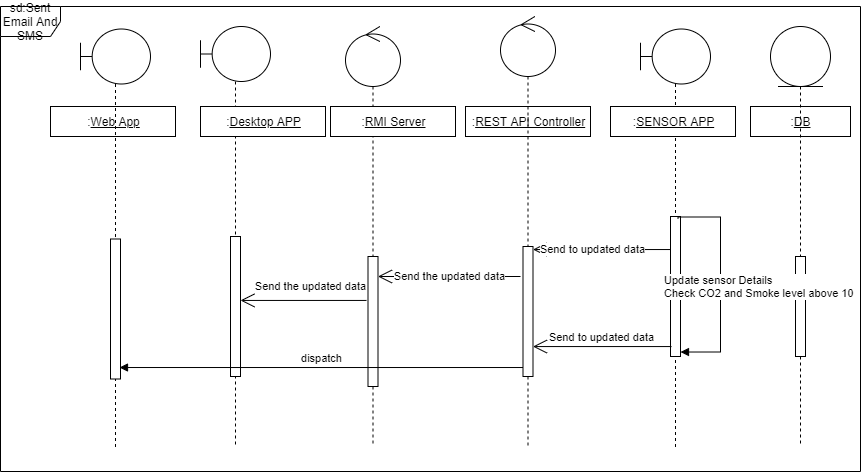
Edit sensor details from the desktop app. Only admin can change the data. When admin logging into the system and he moves to the Add sensor detail interface and he can add the sensor details to the system.[sensor id, Floorno, Roomno]



Web app received the data from the rest API. This will update time to time with the desktop app.



This is how the email and SMS service is working. This will up both web and desktop applications.



APPENDIX

**REST API**

**//GET SENSOR APP DETAILS USIGN THIS API**

**//CREATED THE ASYNC FUNCTION TO GET ANTOTHER HOSTED JSON DUMMY VALUES**

router.get("/sensors", (req, res, next) => {

console.log("response sent");

res.setHeader('Access-Control-Allow-Origin', '\*');

res.setHeader('Access-Control-Allow-Credentials', 'true');

res.setHeader('Access-Control-Allow-Headers', '\*');

const customerq = https.get({hostname: "127.0.0.1",port: 4000, path: '/Sensors',method: 'GET' }, res1 => { res1.on('data', d => {

//console.log(JSON.parse(d)); res.send(JSON.parse(d)); }) });});

**//UPDATED SENSOR DETAILS USING ID**

router.get("/sensor/:id", (req, res, next) => {

res.setHeader('Access-Control-Allow-Origin', '\*');

res.setHeader('Access-Control-Allow-Credentials', 'true');

res.setHeader('Access-Control-Allow-Headers', '\*');

Sensor.find({},{ id: req.params.id }, (err, sensors) => {

var sensorMap = {};

sensors.forEach((sensor) => {

sensorMap[sensor.id] = sensor;

});

res.send(sensorMap);

}).catch(next);

});

//DELETE SENSOR DETAILS THROUGH THIS API

router.delete("/sensor/:id", (req, res, next) => {

Sensor.deleteOne({ id: req.params.id }, (err, result) => {

if (result.deletedCount) { res.json({message: `deleted sensor ${req.params.id}`, });

} else {

res.json({message: `Fail to delete ${req.params.id}`, }); }

}).catch(next);

});

//ADD SENSOR DETAILS THROUGH THIS API

router.post("/sensor", (req, res, next) => {

Sensor.create(req.body)

.then((sensor) => {

res.send(sensor);

})

.catch(next);

});

**//GET TIME TO TIME UPDATED SENSOR APP DETAILS TO DESKTOP APPLICATION**

**//GET TIME TO TIME UPDATED SENSOR APP DETAILS TO DESKTOP APPLICATION**

**//WHEN CO2 AND SMOKE LEVEL GOES UP THIS WILL FIRE**

**//EMAIL AND SMS WILL SEND THROUGH THIS API WHEN THE CO2 AND SMOKE LEVEL GOES UP**

**//CALLED BY BOTH DESKTOP AND WEB APPLICATION**

router.get("/sensorret", (req, res, next) => {

res.setHeader('Access-Control-Allow-Origin', '\*');

res.setHeader('Access-Control-Allow-Credentials', 'true');

res.setHeader('Access-Control-Allow-Headers', '\*');

const customerq = https.get({hostname: "127.0.0.1",port: 4000, path: '/Sensors',method: 'GET' },

res1 => {

res1.on('data', d => {

// console.log(JSON.parse(d));

var json = JSON.parse(d);

res.send(JSON.parse(d))

json.forEach(function (object) {

console.log(object.co2Level )

if(object.co2Level >5 && object.smokeLevel >5){

**//EMAIL CONFIGURATION WHEN ACTION FIRED**

let sentinfo = {

from: 'sanduntharaka258@gmail.com',

to: "anujshashimal456@gmail.com",

subject: "SmokeLevel and CO2level Increased",

text: `SmokeLevel of the sensor ${object.id}

is increased to ${object.smokeLevel}

and CO2 Level of the sensor ${object.id}

increased to ${object.co2Level}`,

};

**//SMS CONFIGURATION WHEN ACTION GET FIRED**

transporter.sendMail(sentinfo, (err, object) => {

if (err) {

console.log(err);

} else {

function messageCallback(error, res) {

if (error === null) {

console.log(`Messaging response for messaging phone number: ${phoneNumber}` +

` => code: ${res['status']['code']}` +

`, description: ${res['status']['description']}`);

} else {

console.error("Unable to send message. " + error);

}

}

client.sms.message(messageCallback, phoneNumber, message, messageType);

}

});

} else if (object.smokeLevel > 5) {

let sentinfo = {

from: 'sanduntharaka258@gmail.com',

to: "anujshashimal456@gmail.com",

subject: "SmokeLevel and CO2 level Increased",

text: `SmokeLevel of the sensor ${object.id} increased to ${object.smokeLevel}`,

};

transporter.sendMail(sentinfo, (err, data) => {

if (err) {

console.log(err);

} else {

console.log("sent");

function messageCallback(error, responseBody) {

if (error === null) {

console.log(`Messaging response for messaging phone number: ${phoneNumber}` +

` => code: ${responseBody['status']['code']}` +

`, description: ${responseBody['status']['description']}`);

} else {

console.error("Unable to send message. " + error); } }

client.sms.message(messageCallback, phoneNumber, message3, messageType); } });

console.log(object.smokeLevel);

}else if (object.co2Level > 5) {

let sentinfo = {

from: 'sanduntharaka258@gmail.com',

to: "anujshashimal456@gmail.com",

subject: "SmokeLevel and CO2 level Increased",

text: `CO2 Level of the sensor ${object.id} increased to ${object.co2Level}`,

};

transporter.sendMail(sentinfo, (err, data) => {

if (err) {

console.log(err);

} else {

console.log("sent");

function messageCallback(error, responseBody) {

if (error === null) {

console.log(`Messaging response for messaging phone number: ${phoneNumber}` +

` => code: ${responseBody['status']['code']}` +

`, description: ${responseBody['status']['description']}`);

} else {

console.error("Unable to send message. " + error); } }

client.sms.message(messageCallback, phoneNumber, message2, messageType);

}

}); } }) console.log('eriush') }) });});

**//REGISTER SENSORS THROUGH THIS API**

router.post("/sensor/Add", (req, res, next) => {

Sensor\_Det.create(req.body)

.then((sensor) => {

res.send(sensor);

})

.catch(next);

});

**//GET ALL REGISTERED DETAILS**

router.get("/sensor", (req, res, next) => {

Sensor\_Det.find({}, (err, sensors) => {

res.setHeader('Access-Control-Allow-Origin', '\*');

res.setHeader('Access-Control-Allow-Credentials', 'true');

res.setHeader('Access-Control-Allow-Headers', '\*');

res.send(sensors);

}).catch(next);

});

**//UPDATE THE REGISTERED DETAILS**

router.post("/sensor/update", (req, res, next) => {

Sensor\_Det.findOneAndUpdate(req.params.Id, req.body, (err, user) => {

if (err) {

return res

.status(500)

.send({error: "unsuccessful"})

};

res.send({success: "success"}); });});

WEB Application

import React, { Component } from 'react';

import { MDBDataTable, Row, Col, Card, CardBody } from 'mdbreact';

import axios from 'axios';

const JSSERVER\_CONFIG = { host: "htpp://127.0.0.1", port: 4000};

class TableSectionInbound extends Component {

constructor(props) {

super(props);

this.state= {

posts: [],

isLoading:true,

tableRows: [],

notify : false

};

}

componentWillUnmount() { clearInterval(this.getAllSensorDet()); }

componentDidMount() { this.getAllSensorDet(); this.interval = setInterval(this.getAllSensorDet, 1000); }

// `${JSSERVER\_CONFIG.host}:${JSSERVER\_CONFIG.port}/Sensors`

// CALL THE REST API TO GET THE SENSOR DETAILS USING AXIOS

getAllSensorDet = async () => {

await axios.get(`http://localhost:5000/api/sensors`)

.then(response => response.data) .then(data => { console.log(data); if(data.co2Level > 4){

alert("Error") console.log('24') }else{ console.log('2') this.setState({ posts: data })

}

})

.then(async() => {

this.setState({ tableRows:this.assemblePosts(), isLoading:true })

console.log(this.state.tableRows);

})

}

assemblePosts= () => {

let posts =this.state.posts.map((post) => {

return (

{

id: post.id,

status: post.status,

co2Level: post.co2Level,

smokeLevel: post.smokeLevel,

floorNo : post.floorNo,

roomNo : post.roomNo } ) }); return posts; }

render() {

**//CONFIGURE THE TABLE DATA**

const data = {

columns: [

{ label:'id', field:'id', },

{ label:'status', field:'status', },

{ label:'co2Level', field:'co2Level', },

{ label:'smokeLevel', field:'smokeLevel', },

{ label:'roomNo', field:'roomNo', },

{ label:'floorNo', field:'floorNo', },

], rows:this.state.tableRows, }

return (

<Col md="12">

<Card>

<CardBody>

<MDBDataTable

data={data}

/>

</CardBody>

</Card>

</Col>

) }}

export default TableSectionInbound;

**Rmi\_Server.java**

**public class Rmi\_Server extends UnicastRemoteObject implements Rmi\_Interface**

**{ private static HttpURLConnection con;**

**public static String getdetails;**

**public static String get\_latest\_update\_details;**

**public static String getsensordetails;**

**public Rmi\_Server() throws RemoteException{**

**super();**

**}**

**//Add sensor details to the database through rest api**

**@Override**

**public void Add\_Sensor\_Details(int id,String floorno1,String roomno1) {**

**try { final String POST\_PARAMS = "{\n"+" \"Id\":"+"\""+id+"\",\r\n" +**

**" \"floorNo\":"+"\""+floorno1+"\",\r\n"+**

**" \"roomNo\":"+"\""+roomno1+"\"" + "\n}";**

**System.out.println(POST\_PARAMS);**

**URL obj = new URL("http://localhost:5000/api/sensor/Add"); //Rest Api Url**

**HttpURLConnection postConnection;**

**postConnection = (HttpURLConnection) obj.openConnection();**

**postConnection.setRequestMethod("POST");**

**postConnection.setRequestProperty("Content-Type", "application/json");**

**postConnection.setDoOutput(true);**

**OutputStream os = postConnection.getOutputStream();**

**os.write(POST\_PARAMS.getBytes());**

**os.flush();**

**os.close();**

**int responseCode = postConnection.getResponseCode();**

**System.out.println("POST Response Code : " + responseCode);**

**System.out.println("POST Response Message : " + postConnection.getResponseMessage());**

**if (responseCode == HttpURLConnection.HTTP\_CREATED) { //success**

**BufferedReader in = new BufferedReader(new InputStreamReader( postConnection.getInputStream()));**

**String inputLine;**

**StringBuffer response = new StringBuffer();**

**while ((inputLine = in .readLine()) != null) {**

**response.append(inputLine);**

**} in .close();**

**// print result**

**System.out.println(response.toString());**

**} else {**

**System.out.println("POST NOT WORKED"); }**

**} catch (MalformedURLException ex) {**

**Logger.getLogger(Rmi\_Server.class.getName()).log(Level.SEVERE, null, ex);**

**} catch (IOException ex) {**

**Logger.getLogger(Rmi\_Server.class.getName()).log(Level.SEVERE, null, ex);}**

**}**

**//Edit sensor details are stored in db**

**@Override**

**public void Edit\_Sensor\_Details(int sensor\_id,String floorno,String roomno) {**

**try {**

**final String POST\_PARAMS = "{\n"+" \"Id\":"+"\""+sensor\_id+"\",\r\n" +**

**" \"floorNo\":"+"\""+floorno+"\",\r\n"+**

**" \"roomNo\":"+"\""+roomno+"\"" + "\n}";**

**System.out.println(POST\_PARAMS);**

**URL obj = new URL("http://localhost:5000/api/sensor/update"); //Rest Api Url**

**HttpURLConnection putConnection;**

**putConnection = (HttpURLConnection) obj.openConnection();**

**putConnection.setRequestMethod("POST");**

**putConnection.setRequestProperty("Content-Type", "application/json");**

**putConnection.setDoOutput(true);**

**OutputStream os = putConnection.getOutputStream();**

**os.write(POST\_PARAMS.getBytes());**

**os.flush();**

**os.close();**

**int responseCode = putConnection.getResponseCode();**

**System.out.println("PUTT Response Code : " + responseCode);**

**System.out.println("PUT Response Message : " + putConnection.getResponseMessage());**

**if (responseCode == HttpURLConnection.HTTP\_CREATED) { //success**

**BufferedReader in = new BufferedReader(new InputStreamReader(**

**putConnection.getInputStream()));**

**String inputLine;**

**StringBuffer response = new StringBuffer();**

**while ((inputLine = in .readLine()) != null) {**

**response.append(inputLine);**

**} in .close();**

**// print result**

**System.out.println(response.toString());**

**} else {**

**System.out.println("Put NOT WORKED"); }**

**} catch (MalformedURLException ex) {**

**Logger.getLogger(Rmi\_Server.class.getName()).log(Level.SEVERE, null, ex); }**

**catch (IOException ex) {**

**Logger.getLogger(Rmi\_Server.class.getName()).log(Level.SEVERE, null, ex); }**

**}**

**//Get sensor information details[sensor id,floor no,room no details]**

**@Override**

**public String Get\_Sensor\_Information\_Details() {**

**try { URL urlForGetRequest = new URL("http://localhost:5000/api/sensor"); //Rest Api Url**

**String readLine = null;**

**HttpURLConnection conection = (HttpURLConnection) urlForGetRequest.openConnection();**

**conection.setRequestMethod("GET");**

**conection.setRequestProperty("userId", "a1bcdef"); // set userId its a sample here**

**int responseCode = conection.getResponseCode();**

**if (responseCode == HttpURLConnection.HTTP\_OK) {**

**BufferedReader in = new BufferedReader(**

**new InputStreamReader(conection.getInputStream()));**

**StringBuffer response = new StringBuffer();**

**while ((readLine = in .readLine()) != null) {**

**response.append(readLine);**

**} in .close();**

**// print result**

**Rmi\_Server.getsensordetails = response.toString();**

**System.out.println("JSON String Result " + response.toString());**

**//GetAndPost.POSTRequest(response.toString());**

**} else {**

**System.out.println("GET NOT WORKED");**

**} } catch (MalformedURLException ex) {**

**Logger.getLogger(Rmi\_Server.class.getName()).log(Level.SEVERE, null, ex);**

**} catch (IOException ex) {**

**Logger.getLogger(Rmi\_Server.class.getName()).log(Level.SEVERE, null, ex);**

**}**

**return Rmi\_Server.getsensordetails;}**

**//Get Sensor Active status Details including sensor id,floor no ,room no,Sensor Active status details .. These Sensor details are displayed in the Table [JTable]**

**public String Get\_Sensor\_Behaviour\_Details() throws RemoteException**

**{ try { URL urlForGetRequest = new URL("http://localhost:5000/api/sensorret"); //Rest Api Url**

**String readLine = null;**

**HttpURLConnection conection = (HttpURLConnection) urlForGetRequest.openConnection();**

**conection.setRequestMethod("GET");**

**conection.setRequestProperty("userId", "a1bcdef");**

**int responseCode = conection.getResponseCode();**

**if (responseCode == HttpURLConnection.HTTP\_OK) {**

**BufferedReader in = new BufferedReader(**

**new InputStreamReader(conection.getInputStream()));**

**StringBuffer response = new StringBuffer();**

**while ((readLine = in .readLine()) != null) {**

**response.append(readLine);**

**} in .close();**

**// print result**

**Rmi\_Server.getdetails = response.toString();**

**System.out.println("JSON String Result " + response.toString());**

**//GetAndPost.POSTRequest(response.toString());**

**} else {**

**System.out.println("GET NOT WORKED");**

**} } catch (MalformedURLException ex) {**

**Logger.getLogger(Rmi\_Server.class.getName()).log(Level.SEVERE, null, ex);**

**} catch (IOException ex) {**

**Logger.getLogger(Rmi\_Server.class.getName()).log(Level.SEVERE, null, ex);}**

**return Rmi\_Server.getdetails;}**

**//Asynchronous method ,, Get latest update sensor update details[sensor id,floor no ,room no,Sensor Active status details] through rest api**

**// Table data update every 15 seconds .**

**//This Asynchronous method call every 15 seconds.**

**public String Get\_Latest\_Update\_Sensor\_Status\_Details() throws RemoteException**

**{ Timer timer = new Timer();**

**timer.schedule(new TimerTask() {**

**@Override**

**public void run() {**

**try { URL urlForGetRequest = new URL("http://localhost:5000/api/sensorret"); //Rest Api Url**

**String readLine = null;**

**HttpURLConnection conection = (HttpURLConnection) urlForGetRequest.openConnection();**

**conection.setRequestMethod("GET");**

**conection.setRequestProperty("userId", "a1bcdef");**

**int responseCode = conection.getResponseCode();**

**if (responseCode == HttpURLConnection.HTTP\_OK) {**

**BufferedReader in = new BufferedReader(**

**new InputStreamReader(conection.getInputStream()));**

**StringBuffer response = new StringBuffer();**

**while ((readLine = in .readLine()) != null) {**

**response.append(readLine);**

**} in .close();**

**// print result**

**Rmi\_Server.getdetails = response.toString();**

**System.out.println("JSON latest update result Result " + response.toString());**

**//GetAndPost.POSTRequest(response.toString());**

**} else {**

**System.out.println("GET NOT WORKED");**

**} } catch (MalformedURLException ex) {**

**Logger.getLogger(Rmi\_Server.class.getName()).log(Level.SEVERE, null, ex);**

**} catch (IOException ex) {**

**Logger.getLogger(Rmi\_Server.class.getName()).log(Level.SEVERE, null, ex);**

**} }**

**},1000,15000);**

**return Rmi\_Server.getdetails;**

**}**

**public static void main(String args[]) throws RemoteException, IOException**

**{ Rmi\_Server Rmiserver = new Rmi\_Server(); //Instantiating & Binding the Service**

**try { Registry reg = LocateRegistry.createRegistry(1022); // Instantiating & Binding the Service**

**reg.rebind("rmiserver",Rmiserver);**

**} catch(Exception e){ } }}**

**Desktop Client**

**Edit\_Sensor\_Details.java**

**public class Edit\_Sensor\_Details extends javax.swing.JFrame {**

**public static int i=0;**

**public static String dataarray ="";**

**public static String getuniquesensordetails ="";**

**public Edit\_Sensor\_Details() {**

**initComponents();**

**//update sensor details table repeatedly**

**Timer timer = new Timer();]**

**timer.schedule(new TimerTask() {**

**@Override**

**public void run() { Display\_data\_to\_table(); //sensor data display in table**

**}**

**},1000,15000); }**

**private void Display\_data\_to\_table(){**

**ArrayList<Integer> smokelevel = new ArrayList<>();**

**ArrayList<Integer> Co2level = new ArrayList<Integer> ();**

**ArrayList<String> floorno = new ArrayList<>();**

**ArrayList<String> roomno = new ArrayList<>();**

**Registry reg=null;**

**try {**

**reg = LocateRegistry.getRegistry("localhost",1022); // Service Lookup**

**} catch (RemoteException ex) {**

**Logger.getLogger(Add\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex); }**

**try {**

**Rmi\_Interface rmiinterface = (Rmi\_Interface) reg.lookup("rmiserver"); // Service Lookup**

**if(i == 0)**

**{ Edit\_Sensor\_Details.dataarray = rmiinterface.Get\_Sensor\_Behaviour\_Details(); // Accessing Service**

**i++;**

**} else{**

**Edit\_Sensor\_Details.dataarray = rmiinterface.Get\_Latest\_Update\_Sensor\_Status\_Details(); // Accessing Service }**

**JSONArray jsonArr = new JSONArray(dataarray);**

**String[] columnNames = {"Sensor\_Id", "Floor\_no","Room\_no","Smoke\_level","Co2\_Level","active"};**

**DefaultTableModel model = new DefaultTableModel(columnNames, 0);**

**for (int i = 0; i < jsonArr.length(); i++)**

**{**

**JSONObject object = jsonArr.getJSONObject(i);**

**int Sensor\_Id = object.getInt("id");**

**String Floor\_no = object.getString("floorNo");**

**String Room\_no = object.getString("roomNo");**

**int Smoke\_level = object.getInt("smokeLevel");**

**int Co2\_Level = object.getInt("co2Level");**

**String active = object.getString("status");**

**smokelevel.add(Smoke\_level);**

**Co2level.add(Co2\_Level);**

**floorno.add(Floor\_no);**

**roomno.add(Room\_no);**

**model.addRow(new Object[] { Sensor\_Id ,Floor\_no,Room\_no,Smoke\_level, Co2\_Level,active });}**

**jTable1.setModel(model)**

**for(int i=0;i<smokelevel.size();i++)**

**{ if(smokelevel.get(i) > 5 )**

**{**

**String floornos = floorno.get(i);**

**String roomnos = roomno.get(i)**

**String settitle = "WARNING";**

**String infoMessage = "FloorNo:"+" "+floornos+" ,,"+"RoomNo:"+" "+roomnos+",, "+"Smoke level is Greater than 5";**

**JOptionPane.showConfirmDialog(this,infoMessage,"WARNING",JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.ERROR\_MESSAGE); }**

**if(Co2level.get(i) > 5 ) {**

**String floornos = floorno.get(i);**

**String roomnos = roomno.get(i);**

**String settitle = "WARNING"**

**String infoMessage = "FloorNo:"+" "+floornos+" ,,"+"RoomNo:"+" "+roomnos+",, "+"Co2 level is Greater than 5";**

**JOptionPane.showConfirmDialog(this,infoMessage,"WARNING",JOptionPane.OK\_CANCEL\_OPTION, JOptionPane.ERROR\_MESSAGE); } }**

**} catch (RemoteException ex) {Logger.getLogger(Add\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex);**

**}catch (NotBoundException ex) { Logger.getLogger(Add\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex); } }**

**private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {**

**// TODO add your handling code here:**

**int id = Integer.parseInt(sensorid.getText());**

**String floorno1 = floorno.getText().toString();**

**String roomno1 = roomno.getText().toString();**

**Registry reg=null;**

**try {**

**reg = LocateRegistry.getRegistry("localhost",1022); // Service Lookup**

**} catch (RemoteException ex) {**

**Logger.getLogger(Add\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex);**

**}**

**Rmi\_Interface rmiinterface = null;**

**try {**

**rmiinterface = (Rmi\_Interface) reg.lookup("rmiserver"); // Service Lookup**

**} catch (RemoteException ex) {**

**Logger.getLogger(Edit\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex);**

**} catch (NotBoundException ex) {**

**Logger.getLogger(Edit\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex); }**

**try {**

**rmiinterface.Edit\_Sensor\_Details(id,floorno1,roomno1); // Accessing Service**

**sensorid.setText("");**

**floorno.setText("");**

**roomno.setText("");**

**} catch (RemoteException ex) {**

**Logger.getLogger(Edit\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex);**

**} }**

**private void SearchActionPerformed(java.awt.event.ActionEvent evt) {**

**// TODO add your handling code here**

**Registry reg=null;**

**try {**

**reg = LocateRegistry.getRegistry("localhost",1022); // Service Lookup**

**} catch (RemoteException ex) {**

**Logger.getLogger(Add\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex); }**

**Rmi\_Interface rmiinterface = null;**

**try {**

**rmiinterface = (Rmi\_Interface) reg.lookup("rmiserver"); // Service Lookup**

**} catch (RemoteException ex) {**

**Logger.getLogger(Edit\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex);**

**} catch (NotBoundException ex) {**

**Logger.getLogger(Edit\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex);**

**} try {**

**Edit\_Sensor\_Details.getuniquesensordetails = rmiinterface.Get\_Sensor\_Information\_Details(); // Accessing Service } catch (RemoteException ex) {**

**Logger.getLogger(Edit\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex);}**

**JSONArray jsonArr = new JSONArray(getuniquesensordetails);**

**for (int i = 0; i < jsonArr.length(); i++)**

**{**

**JSONObject object = jsonArr.getJSONObject(i);**

**int Sensor\_Id = object.getInt("Id");**

**if(Sensor\_Id == Integer.parseInt(sensorid.getText()))**

**{**

**floorno.setText(object.getString("floorNo"));**

**roomno.setText(object.getString("roomNo"));}}}**

**Add\_Sensor\_Details.java**

**public class Add\_Sensor\_Details extends javax.swing.JFrame {**

**//Add Sensor details to the Db**

**private void AddSensorDetailsActionPerformed(java.awt.event.ActionEvent evt) {**

**// table.insert(document);**

**int id = Integer.parseInt(sensorid.getText());**

**String floorno1 = String.valueOf(floorno.getText());**

**String roomno1 =String.valueOf(roomno.getText());**

**Registry reg=null;**

**try {**

**reg = LocateRegistry.getRegistry("localhost",1022); // Service Lookup**

**} catch (RemoteException ex) {**

**Logger.getLogger(Add\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex)}**

**try {**

**Rmi\_Interface rmiinterface = (Rmi\_Interface) reg.lookup("rmiserver"); // Service Lookup**

**rmiinterface.Add\_Sensor\_Details(id, floorno1, roomno1); // Accessing Service**

**} catch (RemoteException ex) {**

**Logger.getLogger(Add\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex);**

**} catch (NotBoundException ex) {**

**Logger.getLogger(Add\_Sensor\_Details.class.getName()).log(Level.SEVERE, null, ex);**

**} sensorid.setText("");**

**floorno.setText("");**

**roomno.setText(""); }**