SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY

Distributed Systems

Group Members

IT18150544 – Jayawardana O.R.

IT18163728 – Ransika N.G.Y.

IT18154368 – Wickramanayake T.R.D.

IT18146516 – Wijerathna W.G.D.D.

Table of Content

1. **Introduction…….. 3**

2. **High Level Architectural Diagram …… 4**

3. **Sequence Diagrams and Application Explanation ……. 5**
    a. **Web Application…… 5**
    b. **Desktop Application…… 7**

4. **Appendix**

1. Introduction

This Project is about developing a fire alarm Monitoring System. In this Project we have developed four main components to succeed this project. Those are,

1. REST API
2. Web Application
3. RMI Server
4. Desktop Application

REST API is used to establish the connection with Database. And Web Application communicate with REST API to receive data from the Mongo Database. Web application will repeat this process every 20s so that, the fire alarm Sensor data can be displayed up to date.

In Desktop Application it does not directly communicate with the REST API. So that, there is a middleware which helps the Desktop Application to communicate with the REST API. This middleware is the RMI Server. When Desktop application wants to send or retrieve data from the database, it will invoke the methods which is inside the RMI Server. Then RMI Server will invoke the REST API methods. This will help to establish the connection between REST API and Desktop Application.

Another requirement of this project was to add email Service to notify a person about the level of Smoke and CO_2. To achieve this, we have used NodeMailer.

The High-level architectural diagram of this project is shown below.

2. High Level Architectural Diagram

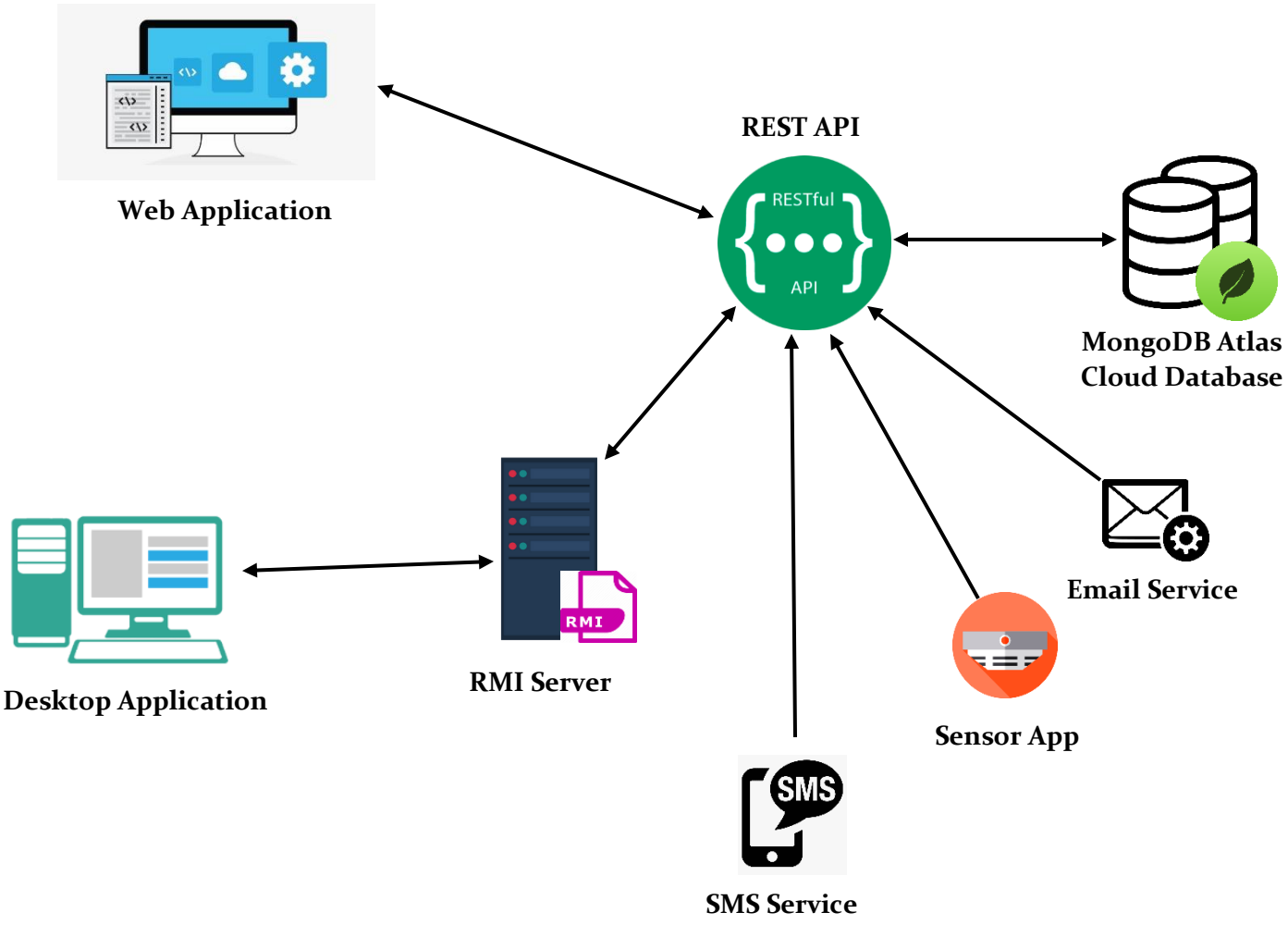

Figure 1 : High Level Architectural Diagram

3. Sequence Diagrams and Application Explanation

1. Web Application

The technology which is used to develop the web application is ReactJS. This web application will communicate with the REST API, to retrieve the fire sensor details from the Database. To retrieve it, Web application uses the GET method which is reside in the Express REST API. For each 20s React web client will call the GET methods in REST API. It allows to keep the Web client up to date.

a. Web Application Interface

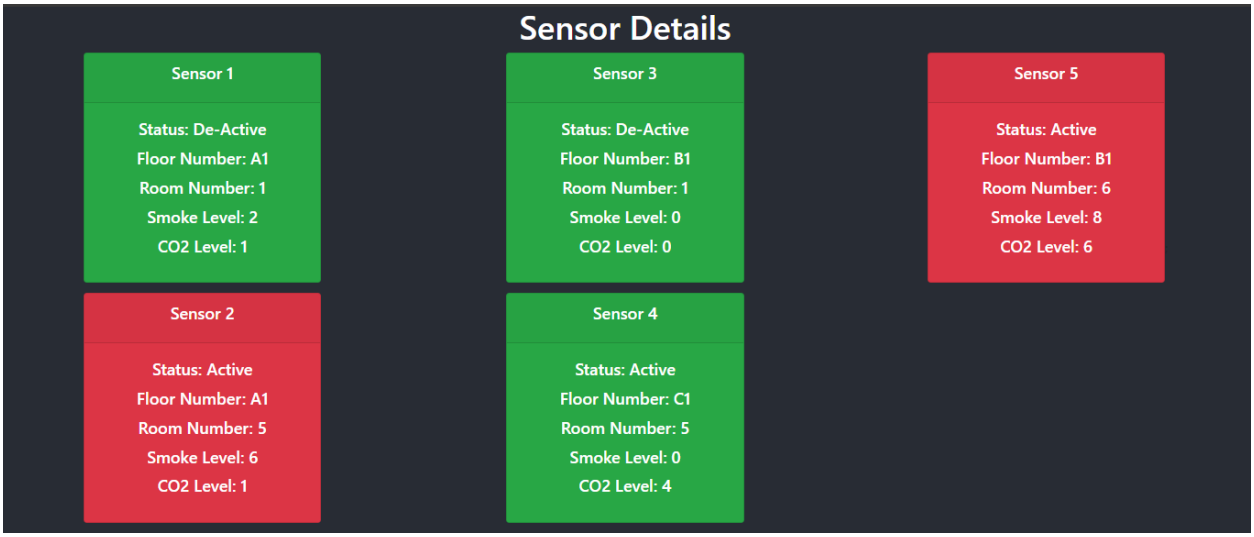

Figure 2 : Web Application Interface

b. Sequence Diagram

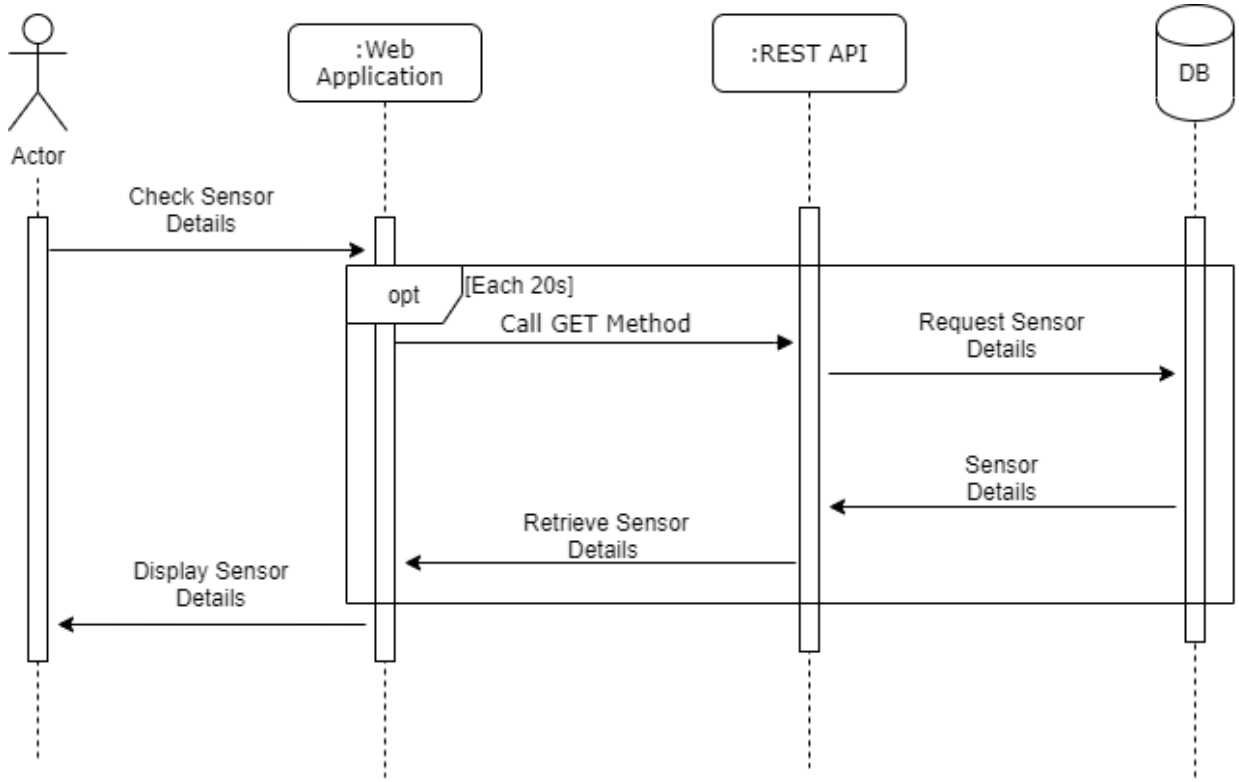

Figure 3 : Web Application User Sequence Diagram

2. Desktop Application

The technology which is used to develop the Desktop application is Java. To establish the communication between the REST API and Desktop Application there is a middleware called RMI Server. All the REST API method are invoked through RMI Server. There are two types of users who can use this desktop application.

The Two Users are.

1. Admin

   Admin Can Check the Sensor Details, add new Sensor Details, Edit Sensor Details. To achieve these tasks admin first has to login to the application by providing username and password.

2. Client

   Client only can view the Sensor Details. Clients do not have to login to check the Sensor Details.

a. Desktop Application Interface

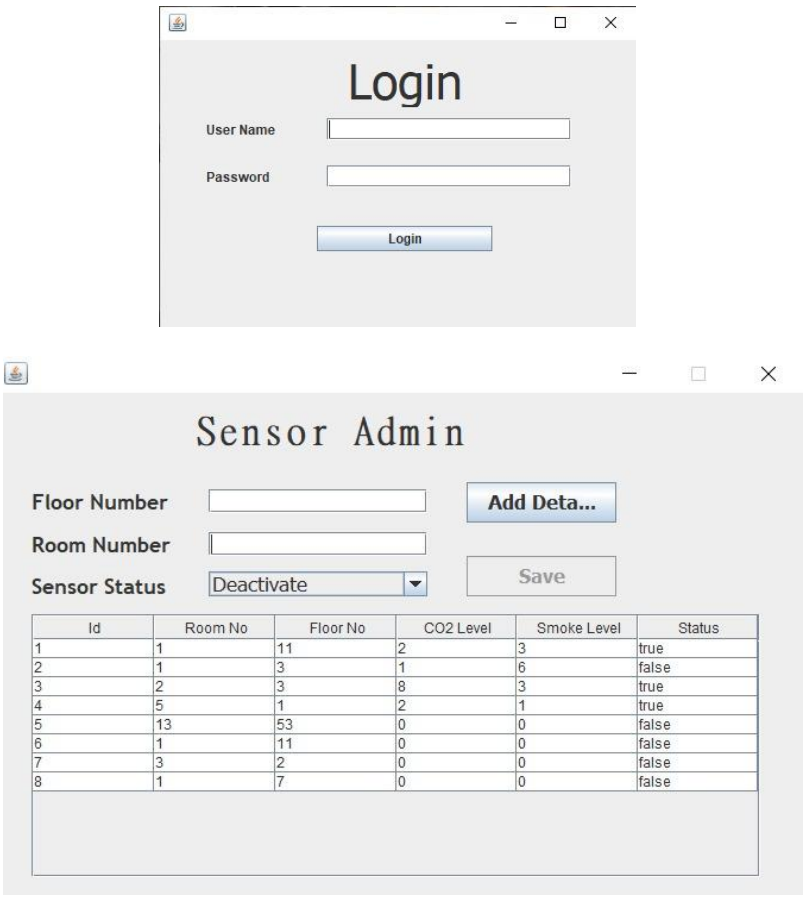

Figure 4 : Desktop Application Interfaces.

b. Sequence Diagram

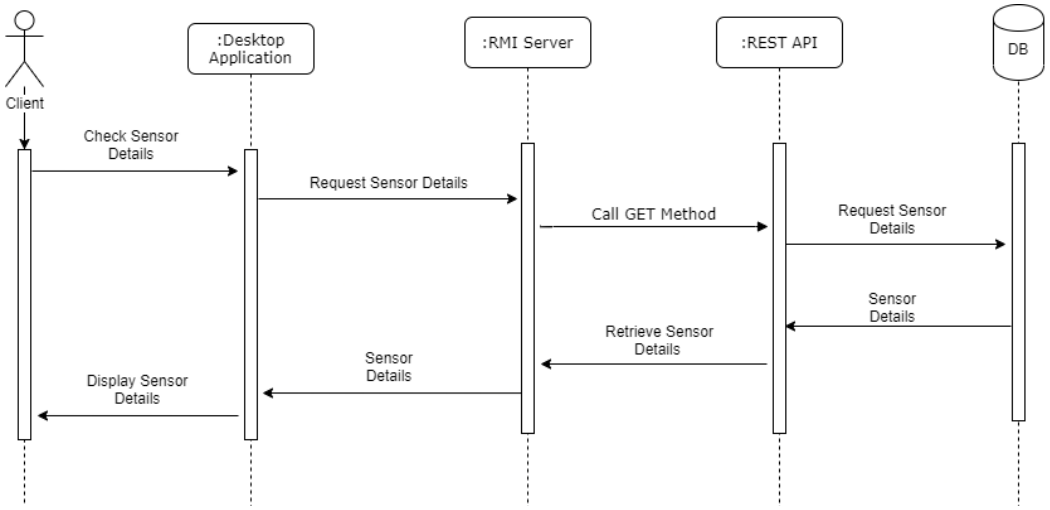

Figure 5 : Desktop Application User Client Sequence Diagram

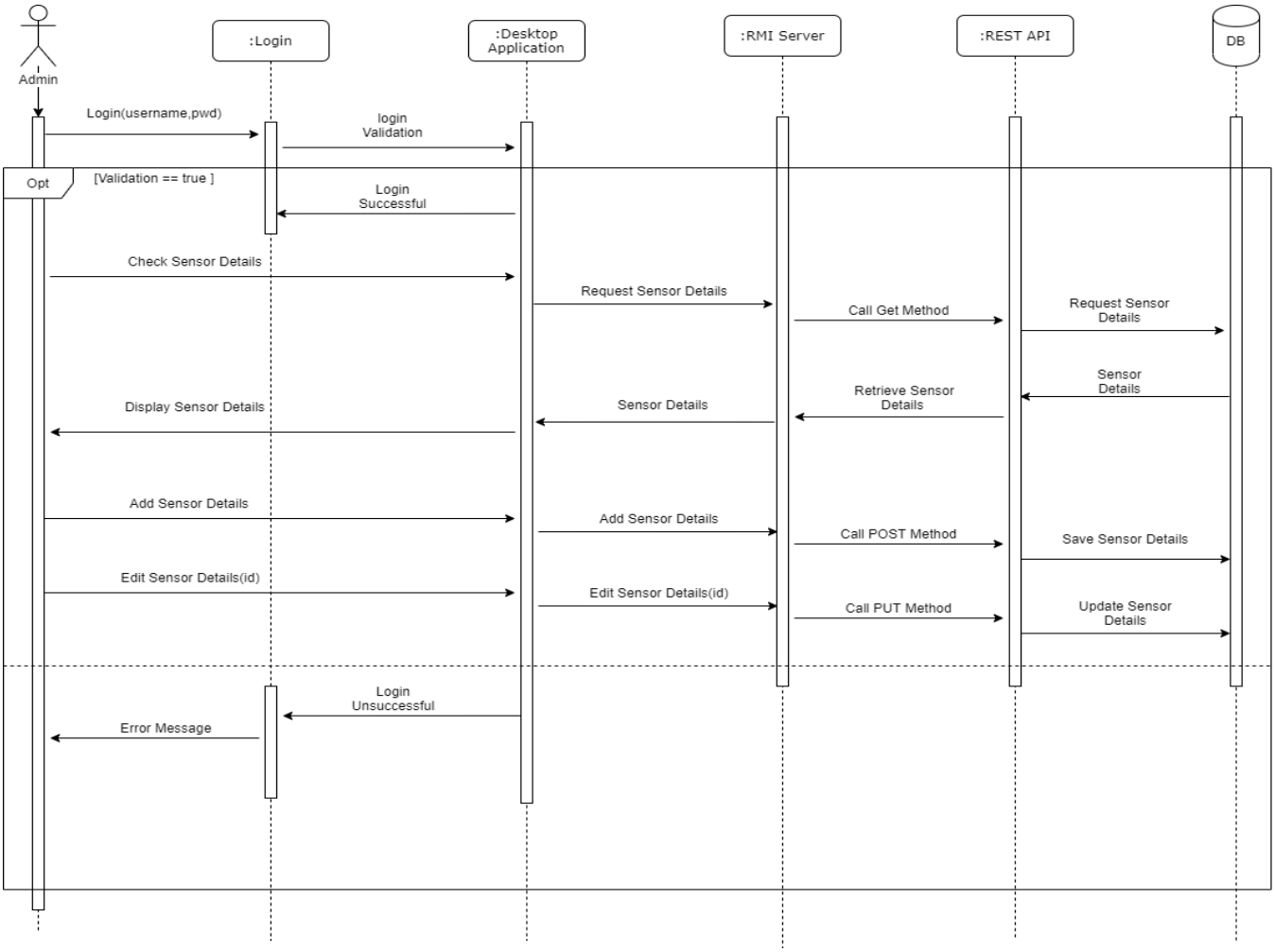

Figure 6: Desktop Application User Admin Sequence Diagram

4. Appendix

REST-API

App.js

```
//Importing Express
const express = require('express');
//Importing Mongoose
const mongo =  require('mongoose');
//Importing bodyParser
const bParser = require('body-parser');
//Importing Cors
const cors = require('cors');

const app = express();

require('dotenv').config();

app.use(bParser.urlencoded({extended: true}));
app.use(bParser.json());
app.use(cors());

app.use("/SensorAPI", require("./Route/FireSensorRoute"));

app.use(function (err, req, res, next) {
    console.error(err.stack);
    res.status(422).send({ error: err.message });
});


//Creating the connection with MongoDB.
 mongo.connect(
     "mongodb+srv://abcd:abcd@cluster0-
fsn7z.mongodb.net/ABCD?retryWrites=true&w=majority"
).then(() => {
    app.listen(5000);
}).catch((err) => {
    console.log(err);
});
```

FireSensorDetModel.js

```
//Importing Mongoose Modules.
const mongo = require('mongoose');
//const autoIncrement = require('mongoose-auto-increment');

//Defining Schema Object
const Schema = mongo.Schema;

//Adding Auto Incrementing Plugin.
const AutoIncrementFactory = require('mongoose-sequence');
const connection = mongo.createConnection('mongodb+srv://abcd:abcd@cluster0-
fsn7z.mongodb.net/ABCD?retryWrites=true&w=majority');
const AutoIncrement = AutoIncrementFactory(connection);

//Defining Fields for the Schema.
const fireSensorSchema = new Schema({
    activeStatus: {type: Boolean, required: false, default:false},
    floorNumber: {type: String, required: true},
    roomNumber: {type:Number, required:true},
    smokeLvl :{type:Number, required:false, default: 0},
    co2Lvl :{type:Number, required:false, default:0}
});

//Assigning the plugin to Schema.
fireSensorSchema.plugin(AutoIncrement, {inc_field: 'id'})

//Creating FireSensor Model and exporting it.
module.exports = mongo.model('FireSensors', fireSensorSchema)
```

FireSensorRoute.js

```
//Importing Express
const express = require('express');
//Importing the Fire Sensor Det Model
const AlarmSensorModel = require('../Model/FireSensorDetModel');

const route = express.Router();

//Email Service
const emailService = require('nodemailer');
require('dotenv').config();

let transporter = emailService.createTransport({
    service: 'gmail',
    auth:{
        user:'it18146516@my.sliit.lk',
        pwd: process.env.PASSWORD
    }
});

route.post('/alarmSensor', (req,res,next) => {
```

```
    AlarmSensorModel.create(req.body).then((alarmSensor) =>{
        res.send(alarmSensor);
    }).catch(next)
});


route.put('/alarmSensor/:id', (req,res,next) => {
   AlarmSensorModel.findOneAndUpdate(
       {id: req.params.id},
       {$set: req.body},
       {new : true},
       (error, document) => {
           if(document.co2Lvl > 5 || document.smokeLvl > 5){
               let email = {
                   from:'it18146516@my.sliit.lk',
                   to:'it18163728@my.sliit.lk',
                   subject:'Alert of CO2 and Smoke Level',
                   text:`Smoke Level or CO2 Level has passed its basic Limits.`
               };
               transporter.sendMail(email, function(error, info){
                   if (error) {
                       console.log(error);
                   } else {
                       console.log('Email sent: ' + info.response);
                   }
               });

               console.log("Email Has been Send to it18163728@my.sliit.lk");
               console.log("From :- it18146516@my.sliit.lk");
               console.log("Title :- Alert of CO2 and Smoke Level");
               console.log("Text:- Smoke Level or CO2 Level has passed its basic
Limits.");

               console.log("SMS has send to 0772134768");
           }

           res.send(document);
       }
   )
});

route.get('/alarmSensor', (req,res,next) =>{
    AlarmSensorModel.find({}, (error,alarmSen) =>{
        let map = {};
        alarmSen.forEach((element) =>{
            map[element.id] = element;
        });

        res.send(map);
    }).catch(next);

});

route.get('/alarmSen', (req,res,next)=>{
    AlarmSensorModel.find({}, (error, alarmSen)=>{
```

```
        res.send(alarmSen);
        //console.log(process.env.PASSWORD);
    }).catch(next);
});

route.get('/alarmSensor/:id', (req,res,next) =>{
    AlarmSensorModel.find({id:req.params.id}, (error,alarmSen) =>{
        let map = {};
        alarmSen.forEach((element) =>{
            map[element.id] = element;
        });

        res.send(map);
    }).catch(next);
});

route.delete('/alarmSensor/:id' ,(req,res,next) =>{
    AlarmSensorModel.deleteOne({id: req.params.id}, (error, output) =>{
        if(output.deletedCount){
            console.log(`Delete Complete. ${req.params.id}`);
        }else{
            console.log(`Error While Deleting ${req.params.id}`);
        }
        res.send(200);
    }).catch(next);

});

module.exports = route;
```

Web Application

Index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';
import 'bootstrap/dist/css/bootstrap.css';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);

// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
```

app.js

```
import React from 'react';
import 'bootstrap/dist/css/bootstrap.css';
import { BrowserRouter } from "react-router-dom";
import './App.css';

import Sensor from './Component/Sensor';

function App() {
  return (
      <BrowserRouter>
        <div className="App">
            <h1>Sensor Details</h1>

            <Sensor/>
        </div>
      </BrowserRouter>
  );
}

export default App;
```

Sensor.js

```
import React, {Component} from 'react';
import 'bootstrap/dist/css/bootstrap.css';
import Card from 'react-bootstrap/Card'
import './Sensor.css';
import CardColumns from 'react-bootstrap/CardColumns'
import CardDeck from 'react-bootstrap/CardDeck'
import axios from "axios";

export default class Sensor extends Component{
    constructor(props) {
        super(props);
        this.state = {alarmSen: [] };
    }

    getData = () => {
        axios.get("http://localhost:5000/SensorAPI/alarmSen")
            .then((response)=>{
                const data = response.data;
                this.setState({alarmSen : data});
                console.log(this.state.alarmSen);
                console.log("Data Has Received");

            }).catch(() =>{
                alert("Error");
            });
    }
```

```
    componentDidMount() {
        this.getData();
        setInterval(this.getData, 20000);
    }

    render() {
        console.log(this.state.alarmSen);
        return (
            <div className ="cardList">

                <CardColumns >
                    {this.state.alarmSen.map((alarmSen) => {
                        return(alarmSen.smokeLvl > 5 || alarmSen.co2Lvl > 5) ? (
                            <Card text="white" bg = "danger" style={{ width: '18rem'
}} key={alarmSen.id}>
                                <Card.Header><Card.Title>Sensor
{alarmSen.id}</Card.Title></Card.Header>
                                <Card.Body>
                                    <Card.Title>Status: {alarmSen.activeStatus ?
"Active" : "De-Active"}</Card.Title>
                                    <Card.Title>Floor Number:
{alarmSen.floorNumber}</Card.Title>
                                    <Card.Title>Room Number:
{alarmSen.roomNumber}</Card.Title>
                                    <Card.Title>Smoke Level:
{alarmSen.smokeLvl}</Card.Title>
                                    <Card.Title>CO2 Level:
{alarmSen.co2Lvl}</Card.Title>
                                </Card.Body>
                            </Card>
                            ) : (
                            <Card  text="white"  bg = "success" style={{ width:
'18rem' }} key={alarmSen.id}>
                                <Card.Header><Card.Title>Sensor
{alarmSen.id}</Card.Title></Card.Header>
                                <Card.Body>
                                    <Card.Title>Status: {alarmSen.activeStatus ?
"Active" : "De-Active"}</Card.Title>
                                    <Card.Title>Floor Number:
{alarmSen.floorNumber}</Card.Title>
                                    <Card.Title>Room Number:
{alarmSen.roomNumber}</Card.Title>
                                    <Card.Title>Smoke Level:
{alarmSen.smokeLvl}</Card.Title>
                                    <Card.Title>CO2 Level:
{alarmSen.co2Lvl}</Card.Title>
                                </Card.Body>
                            </Card>
                        )})};
                </CardColumns>
            </div>
        );
    }
}
```

RMI Server And Desktop Application

ISensor.java

```
import java.awt.List;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.ArrayList;

//Interface extesnd remote
public interface ISensor extends Remote {
	public int addsensor(SensorModel sensormodel) throws RemoteException;
	//addSensor method
	public ArrayList<SensorModel> getSensors() throws RemoteException;
	//getSensor method
	public void updatesensor(SensorModel sensormodel) throws
	RemoteException;    //updateSensor Method
}
```

Login.java

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.Font;

public class Login extends JFrame {

	private JPanel contentPane;
	private JTextField uName;
	private JTextField pWord;
	//Login frameLogin = new Login();
	SensorAdmin sensorAdmin = new SensorAdmin();

	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Login frame = new Login();
					frame.setVisible(true);
```

```
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        });
}

/**
 * Create the frame.
 */

public Login() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 450, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("User Name");
        lblNewLabel.setBounds(42, 74, 98, 14);
        contentPane.add(lblNewLabel);

        JLabel lblNewLabel_1 = new JLabel("Password");
        lblNewLabel_1.setBounds(42, 117, 98, 14);
        contentPane.add(lblNewLabel_1);

        uName = new JTextField();
        uName.setBounds(152, 71, 223, 20);
        contentPane.add(uName);
        uName.setColumns(10);

        pWord = new JTextField();
        pWord.setBounds(152, 114, 223, 20);
        contentPane.add(pWord);
        pWord.setColumns(10);

        //Checking is the userName = admin and password = admin
        JButton btnNewButton = new JButton("Login");
        btnNewButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        if (uName.getText().equals("admin") &&
pWord.getText().equals("admin") ) {
                                Login login = new Login();
                                login.setVisible(false);
                                //frameLogin.dispose();
                                sensorAdmin.setVisible(true);
                        }else {
                                JOptionPane.showMessageDialog(null, "Wrong
Password");
                        }
                }
        });
        btnNewButton.setBounds(143, 169, 160, 23);
        contentPane.add(btnNewButton);
```

```
                JLabel lblLogin = new JLabel("Login");
                lblLogin.setFont(new Font("Tahoma", Font.PLAIN, 44));
                lblLogin.setBounds(171, 13, 105, 48);
                contentPane.add(lblLogin);
        }
}
```

SensorAdmin.java

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;
import javax.xml.ws.soap.Addressing;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.DefaultComboBoxModel;
import java.awt.Font;
import javax.swing.JTable;
import javax.swing.JScrollPane;
import java.awt.event.ActionListener;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.util.ArrayList;
import java.awt.event.ActionEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;



public class SensorAdmin extends JFrame {

        private JPanel contentPane;
        private JTextField txtFloorNo;
        private JTextField txtRoomNo;
        private JTable table;
        private int roomNumber;
        private int floorNumber;
        private int co2Level;
        private int smokeLevel;
        private boolean sensorStaus;
```

```
	SensorModel sensorModel = new SensorModel();
	private JTextField txtCo2Level;
	private JTextField txtSmokeLevel;
	JComboBox txtSensorStatus = new JComboBox();


	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					SensorAdmin frame = new SensorAdmin();
					frame.setVisible(true);
					JOptionPane.showMessageDialog(null, "Select
Records To Edit..\n If You Want To Add New Record Just Skip This Message !");
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public SensorAdmin() {
		setResizable(false);

		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 682, 457);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);

		JLabel lblNewLabel = new JLabel("Sensor Admin");
		lblNewLabel.setFont(new Font("MingLiU_HKSCS-ExtB", Font.BOLD,
33));
		lblNewLabel.setBounds(164, 11, 328, 42);
		contentPane.add(lblNewLabel);

		JLabel lblNewLabel_1 = new JLabel("Sensor Status");
		lblNewLabel_1.setFont(new Font("Trebuchet MS", Font.BOLD, 18));
		lblNewLabel_1.setBounds(26, 152, 126, 21);
		contentPane.add(lblNewLabel_1);

		JLabel lblNewLabel_2 = new JLabel("Floor Number");
		lblNewLabel_2.setFont(new Font("Trebuchet MS", Font.BOLD, 18));
		lblNewLabel_2.setBounds(26, 77, 126, 28);
		contentPane.add(lblNewLabel_2);

		JLabel lblNewLabel_3 = new JLabel("Room Number");
		lblNewLabel_3.setFont(new Font("Trebuchet MS", Font.BOLD, 18));
```

```
		lblNewLabel_3.setBounds(26, 116, 126, 23);
		contentPane.add(lblNewLabel_3);

		txtFloorNo = new JTextField();
		txtFloorNo.setFont(new Font("Tahoma", Font.PLAIN, 17));
		txtFloorNo.setBounds(175, 81, 184, 20);
		contentPane.add(txtFloorNo);
		txtFloorNo.setColumns(10);

		txtRoomNo = new JTextField();
		txtRoomNo.setFont(new Font("Tahoma", Font.PLAIN, 17));
		txtRoomNo.setBounds(175, 117, 184, 20);
		contentPane.add(txtRoomNo);
		txtRoomNo.setColumns(10);

		JButton btnAddDetails = new JButton("Add Details");

		btnAddDetails.setFont(new Font("Tahoma", Font.BOLD, 16));
		btnAddDetails.setBounds(392, 75, 166, 34);
		contentPane.add(btnAddDetails);

		JButton btnSave = new JButton("Save");
		btnSave.setEnabled(false);
		btnSave.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {


				//Get inputs from the fields and convert in to int

				roomNumber =Integer.parseInt(txtRoomNo.getText());
//Int
				floorNumber
=Integer.parseInt(txtFloorNo.getText());//Int
				co2Level =Integer.parseInt(txtCo2Level.getText());
//Int
				smokeLevel =
Integer.parseInt(txtSmokeLevel.getText()); //Int

				//Sensor staus boolean set acording to the dropdiown
sellections
				if
(txtSensorStatus.getSelectedItem().toString().equals("Activate")) {
					sensorStaus = true; //If Sensor staus
'Activate' this boolean true
				}else {
					sensorStaus = false; //Else False
				}

				//asignin filtered inputs to the model class via
getters and setters

				sensorModel.setRoomNumber(roomNumber);
				sensorModel.setFloorNumber(floorNumber);
				sensorModel.setSmokeLevel(smokeLevel);
				sensorModel.setCO2Level(co2Level);
```

```
                        sensorModel.setStatus(sensorStaus);

                        ArrayList<SensorModel> xArrayList =new
ArrayList<>();
                        try {
                              ISensor tt = (ISensor)
Naming.lookup("rmi://localhost/ABC");
                              tt.updatesensor(sensorModel);
                              xArrayList = tt.getSensors();

                        } catch (Exception e1) {

                              // TODO Auto-generated catch block
                              e1.printStackTrace();
                        }
                        btnSave.setEnabled(false);
                        tableLoad();
                        JOptionPane.showMessageDialog(null,"Details Updated
Sucessfully!");


                  }
            });
            btnSave.setFont(new Font("Tahoma", Font.BOLD, 16));
            btnSave.setBounds(392, 137, 126, 34);
            contentPane.add(btnSave);

            txtSensorStatus = new JComboBox();
            txtSensorStatus.setFont(new Font("Tahoma", Font.PLAIN, 17));
            txtSensorStatus.setModel(new DefaultComboBoxModel(new String[]
{"Activate", "Deactivate"}));
            txtSensorStatus.setBounds(175, 150, 184, 21);
            contentPane.add(txtSensorStatus);

            JScrollPane scrollPane = new JScrollPane();
            scrollPane.setBounds(26, 186, 613, 221);
            contentPane.add(scrollPane);

            table = new JTable();
            table.addMouseListener(new MouseAdapter() {
                  @Override
                  public void mouseClicked(MouseEvent e) {

                        int rowNumber = table.getSelectedRow();

      sensorModel.setId(Integer.parseInt(table.getValueAt(rowNumber,
0).toString()));
                        txtRoomNo.setText(table.getValueAt(rowNumber,
1).toString());
                        txtFloorNo.setText(table.getValueAt(rowNumber,
2).toString());
                        txtCo2Level.setText(table.getValueAt(rowNumber,
3).toString());
                        txtSmokeLevel.setText(table.getValueAt(rowNumber,
4).toString());
```

```
                        if (table.getValueAt(rowNumber,
5).toString()=="true") {
                                txtSensorStatus.setSelectedItem("Activate");

      //System.out.println(table.getValueAt(rowNumber, 5));
                        }else if(table.getValueAt(rowNumber,
5).toString()=="false") {

      txtSensorStatus.setSelectedItem("Deactivate");
//
      System.out.println(table.getValueAt(rowNumber, 5));
                        }
                        btnSave.setEnabled(true);
                  }
            });
            scrollPane.setViewportView(table);

            txtCo2Level = new JTextField();
            txtCo2Level.setEnabled(false);
            txtCo2Level.setText("9");
            txtCo2Level.setFont(new Font("Tahoma", Font.PLAIN, 17));
            txtCo2Level.setColumns(10);
            txtCo2Level.setBounds(175, 151, -34, 20);
            contentPane.add(txtCo2Level);

            txtSmokeLevel = new JTextField();
            txtSmokeLevel.setEnabled(false);
            txtSmokeLevel.setText("9");
            txtSmokeLevel.setFont(new Font("Tahoma", Font.PLAIN, 17));
            txtSmokeLevel.setColumns(10);
            txtSmokeLevel.setBounds(175, 183, -22, 20);
            contentPane.add(txtSmokeLevel);



            //Add Details button action perfome
            btnAddDetails.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        //Get inputs from the fields and convert in to int

                        roomNumber =Integer.parseInt(txtRoomNo.getText());
//Int
                        floorNumber
=Integer.parseInt(txtFloorNo.getText());//Int
                        co2Level =Integer.parseInt(txtCo2Level.getText());
//Int
                        smokeLevel =
Integer.parseInt(txtSmokeLevel.getText()); //Int

                        //Sensor staus boolean set acording to the dropdiown
sellections
                        if
(txtSensorStatus.getSelectedItem().toString().equals("Activate")) {
```

```
                                sensorStaus = true; //If Sensor staus
'Activate' this boolean true
                        }else {
                                sensorStaus = false; //Else False
                        }

                        //asignin filtered inputs to the model class via
getters and setters
                        sensorModel.setRoomNumber(roomNumber);
                        sensorModel.setFloorNumber(floorNumber);
                        sensorModel.setSmokeLevel(smokeLevel);
                        sensorModel.setCO2Level(co2Level);
                        sensorModel.setStatus(sensorStaus);

                        //Add data to the remote
                        ArrayList<SensorModel> xArrayList =new
ArrayList<>();
                        try {
                                ISensor obj = (ISensor)
Naming.lookup("rmi://localhost/ABC");
                                int i =obj.addsensor(sensorModel);
                                xArrayList = obj.getSensors();

                        } catch (Exception e1) {

                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                        }

                        //Setting data to the table
                        Object[] cols = new String[] {
                                        "Id","Room No","Floor No", "CO2 Level",
"Smoke Level", "Status"
                        };

                        Object[][] dataRow = new
Object[xArrayList.size()][6];
                        for(int t = 0; t<xArrayList.size();t++) {
                                dataRow[t][0] = xArrayList.get(t).getId();
                                dataRow[t][1] =
xArrayList.get(t).getRoomNumber();
                                dataRow[t][2] =
xArrayList.get(t).getFloorNumber();
                                dataRow[t][3] =
xArrayList.get(t).getCO2Level();
                                dataRow[t][4] =
xArrayList.get(t).getSmokeLevel();
                                dataRow[t][5] =
xArrayList.get(t).getStatus();
                        }

                        TableModel tableModel = new
DefaultTableModel(dataRow,cols);

                        table.setModel(tableModel);
```

```
                }
            });


            tableLoad();

        }

        //Load data to table

        public void tableLoad() {
            ArrayList<SensorModel> xArrayList =new ArrayList<>();
            try {
                ISensor obj = (ISensor)
Naming.lookup("rmi://localhost/ABC");
//                int i =obj.addsensor(sensorModel);
                xArrayList = obj.getSensors();

            } catch (Exception e1) {

                // TODO Auto-generated catch block
                e1.printStackTrace();
            }


            Object[] cols = new String[] {
                        "Id","Room No","Floor No", "CO2 Level", "Smoke
Level", "Status"
            };

            Object[][] dataRow = new Object[xArrayList.size()][6];
            for(int t = 0; t<xArrayList.size();t++) {
                dataRow[t][0] = xArrayList.get(t).getId();
                dataRow[t][1] = xArrayList.get(t).getRoomNumber();
                dataRow[t][2] = xArrayList.get(t).getFloorNumber();
                dataRow[t][3] = xArrayList.get(t).getCO2Level();
                dataRow[t][4] = xArrayList.get(t).getSmokeLevel();
                dataRow[t][5] = xArrayList.get(t).getStatus();
            }

            TableModel tableModel = new DefaultTableModel(dataRow,cols);

            table.setModel(tableModel);


        }

}
```

SensorDetails.Java

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JTable;
import java.awt.Color;
import javax.swing.border.BevelBorder;
import javax.swing.table.TableModel;

import java.awt.event.ActionListener;
import java.rmi.Naming;
import java.util.ArrayList;
import java.awt.event.ActionEvent;
import javax.swing.table.DefaultTableModel;
import javax.swing.JScrollPane;
import java.awt.Font;

public class SensorDetails {

	private JFrame frame;

	SensorModel sensorModel = new SensorModel();
	Login logingPage = new Login();
	private JTable table;



	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					SensorDetails window = new SensorDetails();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the application.
	 */
	public SensorDetails() {
		initialize();
		tableLoad();
	}
```

```
	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {
		frame = new JFrame();
		frame.setBounds(100, 100, 596, 364);
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		frame.getContentPane().setLayout(null);

		JLabel lblSensoreDetails = new JLabel("Sensore Details");
		lblSensoreDetails.setFont(new Font("Sitka Subheading",
Font.PLAIN, 39));
		lblSensoreDetails.setBounds(59, 25, 302, 36);
		frame.getContentPane().add(lblSensoreDetails);

		JButton btnNewButton = new JButton("Login");
		btnNewButton.setFont(new Font("Tahoma", Font.PLAIN, 14));

		btnNewButton.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				frame.dispose();
				logingPage.setVisible(true);
			}
		});

		btnNewButton.setBounds(461, 41, 109, 36);
		frame.getContentPane().add(btnNewButton);

		JScrollPane scrollPane = new JScrollPane();
		scrollPane.setBounds(10, 93, 560, 206);
		frame.getContentPane().add(scrollPane);

		table = new JTable();
		scrollPane.setViewportView(table);
	}

	//Load data to the table
	public void tableLoad() {
		ArrayList<SensorModel> xArrayList =new ArrayList<>();
		try {
			ISensor obj = (ISensor)
Naming.lookup("rmi://localhost/ABC");
//			int i =obj.addsensor(sensorModel);
			xArrayList = obj.getSensors();

		} catch (Exception e1) {

			// TODO Auto-generated catch block
			e1.printStackTrace();
		}

		//Creating colunm names
		Object[] cols = new String[] {
				"Id","Room No","Floor No", "CO2 Level", "Smoke
Level", "Status"
```

```
            };


            //Setiing data to the table
            Object[][] dataRow = new Object[xArrayList.size()][6];
            for(int t = 0; t<xArrayList.size();t++) {
                  dataRow[t][0] = xArrayList.get(t).getId();
                  dataRow[t][1] = xArrayList.get(t).getRoomNumber();
                  dataRow[t][2] = xArrayList.get(t).getFloorNumber();
                  dataRow[t][3] = xArrayList.get(t).getCO2Level();
                  dataRow[t][4] = xArrayList.get(t).getSmokeLevel();
                  dataRow[t][5] = xArrayList.get(t).getStatus();
            }

            TableModel tableModel = new DefaultTableModel(dataRow,cols);

            table.setModel(tableModel);


      }

}
```

SensorImpl.java

```
import java.rmi.RemoteException;
import java.rmi.server.*;
import java.util.ArrayList;
import java.io.StringWriter;
import java.awt.List;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
import java.io.Serializable;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;

import org.json.JSONObject;
import org.json.JSONObject.*;
import org.json.JSONArray;

public class SensorImpl extends UnicastRemoteObject implements ISensor {

      public SensorImpl() throws RemoteException {
            super();
            // TODO Auto-generated constructor stub
      }

      //Add Sensor Details Method Implementation
```

```
	@Override
	public int addsensor(SensorModel sensormodel) throws RemoteException{
		// TODO Auto-generated method stub



			boolean status = sensormodel.getStatus();
			int floorNumber = sensormodel.getFloorNumber();
			int roomNumber = sensormodel.getRoomNumber();
			int smokeLevel = sensormodel.getSmokeLevel();
			int co2level = sensormodel.getCO2Level();

			final String POST_PARAMS = "{\n" +  "\"floorNumber\":
\""+floorNumber+"\",\r\n"  +
		"    \"roomNumber\": \""+roomNumber+"\" \r\n}";



			System.out.println(POST_PARAMS);
		try {
		URL obj = new URL("http://localhost:5000/SensorAPI/alarmSensor");

		HttpURLConnection 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 (Exception e) {
                                        e.printStackTrace();
                }


                int word = sensormodel.getRoomNumber();

                return word;




        }


        //Get Sensor details Implementation
        public  ArrayList<SensorModel> getSensors() throws RemoteException {
                // TODO Auto-generated method stub
                ArrayList<SensorModel> xArrayList =new ArrayList<>();
                //SensorModel smm = new SensorModel();
                try {
                        //Rest Api url
                URL urlForGetRequest = new
URL("http://localhost:5000/SensorAPI/alarmSen");
            String readLine = null;

            HttpURLConnection conection = (HttpURLConnection)
urlForGetRequest.openConnection();
            conection.setRequestMethod("GET");


            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
                System.out.println("JSON String Result " + response.toString());
                //GetAndPost.POSTRequest(response.toString());

                JSONArray  obj_JSONArray = new JSONArray(response.toString());
                System.out.println("JSON Array Result " + obj_JSONArray);
                System.out.println("JSON Array length " +
obj_JSONArray.length());


               //Setting respond data to object
```

```
                for(int i=0;i<obj_JSONArray.length();i++) {

                      SensorModel smm = new SensorModel();
                      JSONObject jo = obj_JSONArray.getJSONObject(i);
                      smm.setId(jo.getInt("id"));
                      smm.setStatus(jo.getBoolean("activeStatus"));

          smm.setFloorNumber(Integer.parseInt(jo.getString("floorNumber")));
                      smm.setRoomNumber(jo.getInt("roomNumber"));
                      smm.setCO2Level(jo.getInt("co2Lvl"));
                      smm.setSmokeLevel(jo.getInt("smokeLvl"));

                      xArrayList.add(smm);
                }

            } else {
                System.out.println("GET NOT WORKED");
            }
               } catch (Exception e) {

                      e.printStackTrace();
               }

               return xArrayList;
          }

          //Update Sensor details implementation
          public void updatesensor(SensorModel sensormodel) throws RemoteException
{
               // TODO Auto-generated method stub
               // TODO Auto-generated method stub


               int id = sensormodel.getId();
               boolean status = sensormodel.getStatus();
               int floorNumber = sensormodel.getFloorNumber();
               int roomNumber = sensormodel.getRoomNumber();
               int smokeLevel = sensormodel.getSmokeLevel();
               int co2level = sensormodel.getCO2Level();

               //Converting Java Object to jason string
               final String POST_PARAMS = "{\n" +  "\"id\": \""+id+"\",\r\n"  +
                            "      \"activeStatus\": \""+status+"\",\r\n"  +
                            "      \"floorNumber\": \""+floorNumber+"\",\r\n"  +
                            "      \"co2Lvl\": \""+co2level+"\",\r\n"  +
                            "      \"smokeLvl\": \""+smokeLevel+"\",\r\n"  +
                            "   \"roomNumber\": \""+roomNumber+"\" \r\n}";



               System.out.println("chikeeey "+POST_PARAMS);


          try {
          URL obj = new URL("http://localhost:5000/SensorAPI/alarmSensor/"+ id);
```

```
        HttpURLConnection postConnection = (HttpURLConnection)
obj.openConnection();
        postConnection.setRequestMethod("PUT");

        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 (Exception e) {
                            e.printStackTrace();
        }
        }
}
```

SensorModel.java

```
import java.io.Serializable;

public class SensorModel implements Serializable  {

        //Sensor Model
        int id;

        boolean Status;
        int FloorNumber;
        int RoomNumber;
        int SmokeLevel;
        int CO2Level;
```

```
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}

	public boolean getStatus() {
		return Status;
	}
	public void setStatus(boolean status) {
		Status = status;
	}
	public int getFloorNumber() {
		return FloorNumber;
	}
	public void setFloorNumber(int floorNumber) {
		FloorNumber = floorNumber;
	}
	public int getRoomNumber() {
		return RoomNumber;
	}
	public void setRoomNumber(int roomNumber) {
		RoomNumber = roomNumber;
	}
	public int getSmokeLevel() {
		return SmokeLevel;
	}
	public void setSmokeLevel(int smokeLevel) {
		SmokeLevel = smokeLevel;
	}
	public int getCO2Level() {
		return CO2Level;
	}
	public void setCO2Level(int cO2Level) {
		CO2Level = cO2Level;
	}
}
```

Server.java

```
import java.rmi.*;
import java.rmi.registry.*;

public class Server {
	public static void main(String a[]) throws Exception
	{
		SensorImpl obj = new SensorImpl();
		Naming.rebind("rmi://localhost/ABC",obj);
		System.out.println("Server Started");
	}
}
```