



Sri Lanka Institute of Information Technology

Fire Alarm Monitoring System

Distributed System (SE3020)

2020

Assignment 2

Submitted by:

1. IT18128482 – (Jayasundara U.L)
2. IT18139822 – (Pinidiya S.C)
3. IT18118346 – (Dinusha Dilshan Siyasinghe)
4. IT18140644 – (Trishan Piyumal M.A)

Submitted to:

Dr. Dharshana Kasthurirathna

Fire Alarm Monitoring System

Introduction

Human is not more sensible people because nowadays they were busy in their life. Therefore, they can't detect smoke and co2 level in environment. For that scientist are checking Co2 level and Smoke level in the environment by using fire alarm system. Also, it has worked automate and sense about the environment. Therefore, we can detect smoke level and co2 level in the environment. If co2 level or smoke level is high than recommended level then informs us by fire alarm system. For that show red light. But we use beep sounds by fire alarm system Instead of using actual Fire Sensor to get details, we implemented a FiresensorMain.java class in the Fire Sensor Project which generates sensor data and sends the status to the database via REST API in every 30 seconds. So, we choose the MySQL database to store data.

There is also have a desktop client which created in Java Swing GUI Desktop client get sensor data from database and display it in a data table. Users can view Sensor ID, Room No, Floor No, Sensor status CO2 level, Smoke level, Date, and time. The information is refreshed every 30 seconds. All sensor details will be returned, and it will be displayed on the main page.

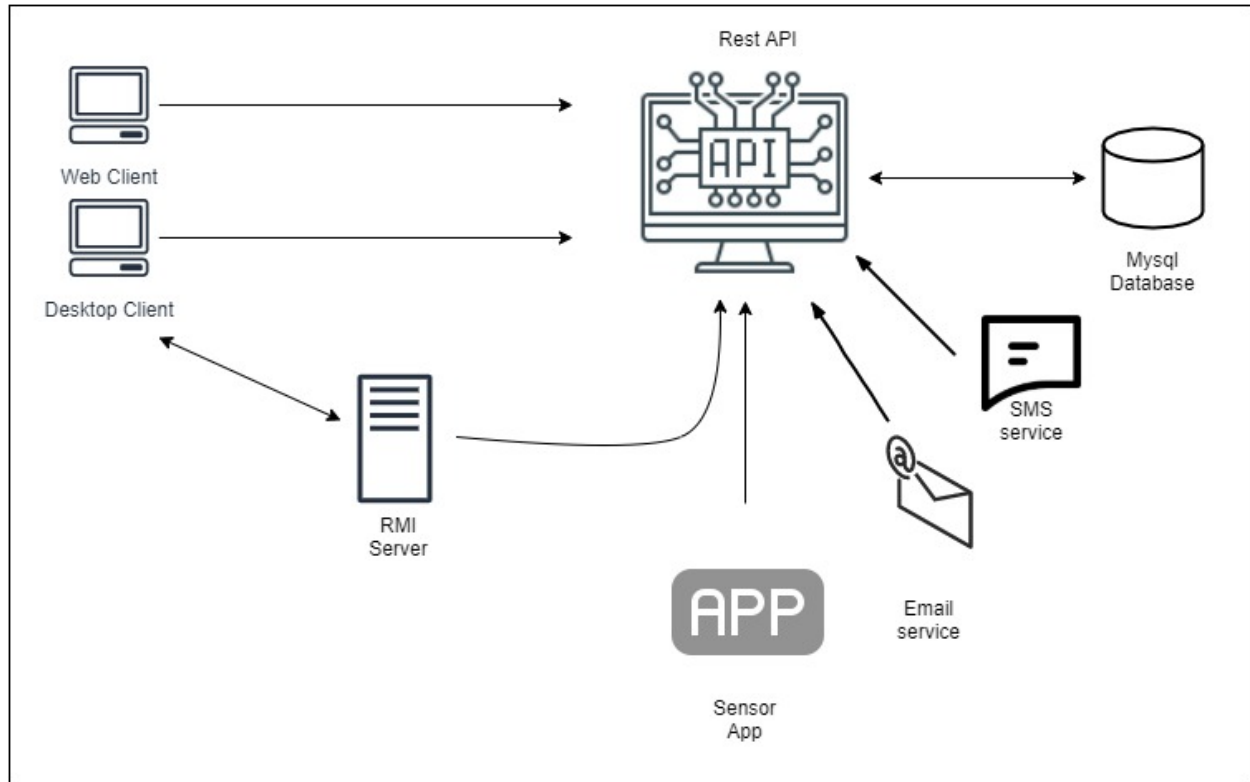
Admin can log in to the system using a given username and password. It is validating in the database and if it is correct admin can go to the admin dashboard. Admin can Add, Update Delete Sensor Details.

A Web client app is created using ReactJS. Any user can view Sensor details such as Sensor ID, Room No, Floor No, Smoke Level, Co2 level, Date, and Time. If the CO2 level or smoke level goes above 5, it will be marked in red, and otherwise, it will be marked in green. Sensor details are refreshed every 40 seconds.

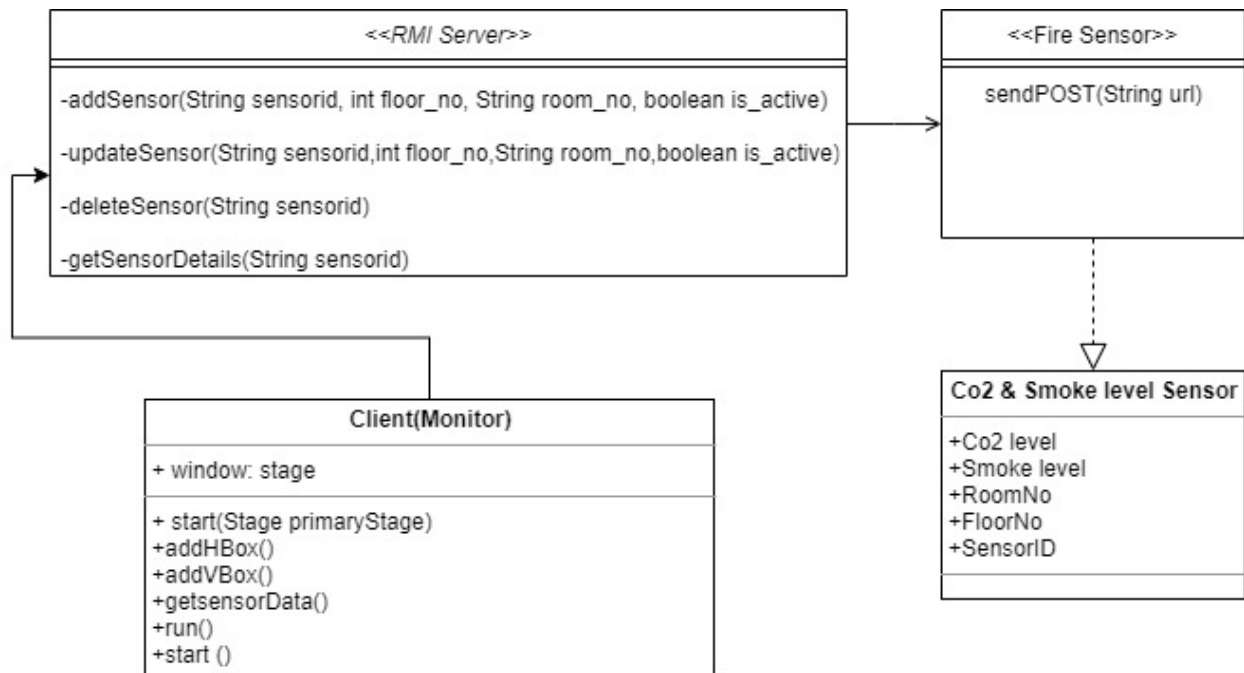
Additionally, Email sending features added. Once the CO2 Level or smoke level goes above 5 Email will be sent to the user.

Architecture of the Fire Alarm Monitoring System

System Architecture Diagram



Class Diagram



Appendix

Frontend

React Web App

```
import React from 'react'

const Sensor = ({ sensordata }) => {
  const danger = {
    color: "red"
  };
  const normal = {
    color: "green"
  };

  return (
    <div className={"container"}>
      <table class="table table-dark">

        <thead>
          <tr>
```

```

        <th>Sensor ID</th>
        <th>Floor Number</th>
        <th>Room Number</th>
        <th>Smoke Level</th>
        <th>CO2 Level</th>
        <th>Date</th>
        <th>Time</th>
    </tr>
</thead>
<tbody>
    {sensordata.map((sensord) => (
        <tr style={sensord.smokeLevel > 5 ? danger : normal}>
            <td>{sensord.sensor.sensorid}</td>
            <td>{sensord.sensor.floorNo}</td>
            <td>{sensord.sensor.roomNo}</td>
            <td style={sensord.smokeLevel > 5 ? danger : normal} >{sensord.smokeLevel}</td>
            <td style={sensord.co2Level > 5 ? danger : normal}>{sensord.co2Level}</td>
            <td>{sensord.date}</td>
            <td>{sensord.time}</td>
        </tr>
    ))}
</tbody>
</table>
</div>
)
};

export default Sensor

```

Backend

Model

Sensor.java

```

package com.techprimers.db.springbootmysql.resource;

import com.techprimers.db.springbootmysql.NotifyUser.SendEmail;
import com.techprimers.db.springbootmysql.model.SensorData;
import com.techprimers.db.springbootmysql.model.Users;
import com.techprimers.db.springbootmysql.repository.SensorDataRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@CrossOrigin
@RequestMapping(value = "/rest/sensordata")

```

```

public class SensorDataResource {

    @Autowired
    SensorDataRepository sensorDataRepository;

    @GetMapping(value = "/all")
    public List<SensorData> getAll(){
        return sensorDataRepository.findAll();
    }

    @PostMapping(value = "/load")
    public List<SensorData> persist(@RequestBody final SensorData sensorData){

        if(sensorData.getCo2Level() >5 ){
            SendEmail.sendEmailtoUser();
        }
        if(sensorData.getSmokeLevel() > 5 ){
            SendEmail.sendEmailtoUser();
        }

        sensorDataRepository.save(sensorData);
        return sensorDataRepository.findAll();
    }
}

```

SensorData.java

```

package com.techprimers.db.springbootmysql.model;

import javax.persistence.*;

@Entity
public class SensorData {

    @Id
    @Column(name = "sensordataid")
    private String sensordataid;

    @ManyToOne
    private Sensor sensor;

    @Column(name = "smokeLevel")
    private int smokeLevel;

    @Column(name = "co2Level")
    private int co2Level;

    @Column(name = "date")
    private String date;

    @Column(name = "time")
    private String time;

    public SensorData() {
    }
}

```

```

public Sensor getSensor() {
    return sensor;
}

public void setSensor(Sensor sensor) {
    this.sensor = sensor;
}

public int getSmokeLevel() {
    return smokeLevel;
}

public void setSmokeLevel(int smokeLevel) {
    this.smokeLevel = smokeLevel;
}

public int getCo2Level() {
    return co2Level;
}

public void setCo2Level(int co2Level) {
    this.co2Level = co2Level;
}

public String getDate() {
    return date;
}

public void setDate(String date) {
    this.date = date;
}

public String getTime() {
    return time;
}

public void setTime(String time) {
    this.time = time;
}

public String getSensordataid() {
    return sensordataid;
}

public void setSensordataid(String sensordataid) {
    this.sensordataid = sensordataid;
}
}

```

User.Java

```

package com.techprimers.db.springbootmysql.model;

import javax.persistence.Column;

```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Users {

    @Id
    @GeneratedValue
    @Column(name = "id")
    private Integer id;
    @Column(name = "name")
    private String name;

    @Column(name = "type")
    private String type;

    @Column(name = "password")
    private String password;

    public Users() {
    }

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```


SendEmail.Java

```
package com.techprimers.db.springbootmysql.NotifyUser;
import java.util.Properties;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class SendEmail {

    public static void sendEmailtoUser(){
        // Recipient's email ID needs to be mentioned.
        String to = "receiver@gmail.com";

        // Sender's email ID needs to be mentioned
        String from = "sender@gmail.com";

        // Assuming you are sending email from through gmails smtp
        String host = "smtp.gmail.com";

        // Get system properties
        Properties properties = System.getProperties();

        // Setup mail server
        properties.put("mail.smtp.host", host);
        properties.put("mail.smtp.port", "465");
        properties.put("mail.smtp.ssl.enable", "true");
        properties.put("mail.smtp.auth", "true");

        // Get the Session object.// and pass username and password
        Session session = Session.getInstance(properties, new javax.mail.Authenticator() {

            protected PasswordAuthentication getPasswordAuthentication() {

                return new PasswordAuthentication("sender@gmail.com", "xxxxxxxxxxx");

            }

        });

        // Used to debug SMTP issues
        session.setDebug(true);

        try {
            // Create a default MimeMessage object.
            MimeMessage message = new MimeMessage(session);

            // Set From: header field of the header.
            message.setFrom(new InternetAddress(from));

            // Set To: header field of the header.
            message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));

            // Set Subject: header field
            message.setSubject("Warning!");
```

```

        // Now set the actual message
        message.setText("Co2 level or Smoke level is above 5");

//        System.out.println("sending...");
//        Send message
        Transport.send(message);
        System.out.println("Sent message successfully....");
    } catch (MessagingException mex) {
        mex.printStackTrace();
    }
}
}

```

SensorDataResource.java

```

package com.techprimers.db.springbootmysql.resource;

import com.techprimers.db.springbootmysql.NotifyUser.SendEmail;
import com.techprimers.db.springbootmysql.model.SensorData;
import com.techprimers.db.springbootmysql.model.Users;
import com.techprimers.db.springbootmysql.repository.SensorDataRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@CrossOrigin
@RequestMapping(value = "/rest/sensordata")
public class SensorDataResource {

    @Autowired
    SensorDataRepository sensorDataRepository;

    @GetMapping(value = "/all")
    public List<SensorData> getAll(){
        return sensorDataRepository.findAll();
    }

    @PostMapping(value = "/load")
    public List<SensorData> persist(@RequestBody final SensorData sensorData){

        if(sensorData.getCo2Level() > 5 ){
            SendEmail.sendEmailtoUser();
        }
        if(sensorData.getSmokeLevel() > 5 ){
            SendEmail.sendEmailtoUser();
        }

        sensorDataRepository.save(sensorData);
        return sensorDataRepository.findAll();
    }
}

```

SensorResource.java

```
package com.techprimers.db.springbootmysql.resource;

import com.techprimers.db.springbootmysql.model.Sensor;
import com.techprimers.db.springbootmysql.model.SensorData;
import com.techprimers.db.springbootmysql.repository.SensorDataRepository;
import com.techprimers.db.springbootmysql.repository.SensorRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@CrossOrigin
@RequestMapping(value = "/rest/sensor")
public class SensorResource {

    @Autowired
    SensorRepository sensorRepository;

    @GetMapping(value = "/all")
    public List<Sensor> getAll(){
        return sensorRepository.findAll();
    }

    @PostMapping(value = "/load")
    public List<Sensor> persist(@RequestBody final Sensor sensor){

        System.out.println(sensor.getFloorNo());
        System.out.println(sensor.isActive());

        sensorRepository.save(sensor);
        return sensorRepository.findAll();
    }

    @PostMapping(value = "/all")
    public List<Sensor> getAll(@RequestBody final Sensor sensor){
        return sensorRepository.findAll();
    }

    @DeleteMapping(value = "/delete/{sensorid}")
    void deleteEmployee(@PathVariable String sensorid) {
        sensorRepository.deleteById(sensorid);
    }

    @GetMapping(value = "/getsensor/{id}")
    Sensor one(@PathVariable String id) throws Exception {
        return sensorRepository.findById(id)
            .orElseThrow(() -> new Exception(String.valueOf(id)));
    }
}
```

UserResource.java

```
package com.techprimers.db.springbootmysql.resource;

import com.techprimers.db.springbootmysql.model.Users;
import com.techprimers.db.springbootmysql.repository.UsersRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping(value = "/rest/users")
public class UsersResource {

    @Autowired
    UsersRepository usersRepository;

    @GetMapping(value = "/all")
    public List<Users> getAll(){
        return usersRepository.findAll();
    }

    @PostMapping(value = "/load")
    public List<Users> persist(@RequestBody final Users user){
        usersRepository.save(user);
        return usersRepository.findAll();
    }

    @PostMapping(value = "/auth")
    public Users authUser(@RequestBody final Users user) throws Exception{
        List<Users> us = usersRepository.findAll();
        for(Users u:us){
            if(u.getName().equals(user.getName()) && u.getPassword().equals(user.getPassword())){
                return u;
            }
        }
        throw new Exception("User not found");
    }
}
```

SpringBootMysqlApplication.java

```
package com.techprimers.db.springbootmysql;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@EnableJpaRepositories(basePackages = "com.techprimers.db.springbootmysql.repository")
@SpringBootApplication
public class SpringBootMysqlApplication {

    public static void main(String[] args) {
```

```

        SpringApplication.run(SpringBootMysqlApplication.class, args);
    }
}

```

application.yml

```

spring:
  datasource:
    driverClassName: com.mysql.jdbc.Driver
    url: jdbc:mysql://localhost:3306/firesensorDB?serverTimezone=UTC
    username: 'root'
    password: ''
  jpa:
    hibernate.ddl-auto: update

```

FireSensorMain.java

```

import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import utils.Utils;

import java.io.IOException;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.concurrent.ThreadLocalRandom;

public class FireSensorMain {
    public static void main(String[] args) {

        try {
            while (true) {

                String result = sendPOST(Utils.SENDPOSTSENSORDATAURL);

                System.out.println(result);

                try {
                    Thread.sleep(30 * 1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static String sendPOST(String url) throws IOException {

```

```

String result = "";
HttpPost post = new HttpPost(url);

post.addHeader("content-type", "application/json");

DateTimeFormatter date = DateTimeFormatter.ofPattern("dd/MM/yyyy");
DateTimeFormatter time = DateTimeFormatter.ofPattern("HH:mm:ss");
LocalDateTime now = LocalDateTime.now();

String sensorID = "sensor4";
int floorNo = 2;
String roomNo = "12";
boolean isActive = true;
int smokeLevel = ThreadLocalRandom.current().nextInt(0, 10 + 1);
int co2Level = ThreadLocalRandom.current().nextInt(0, 10 + 1);

String dt = date.format(now);
String tm = time.format(now);
String sensor = "{\"sensorid\":\"" + sensorID + "\"}";
StringBuilder json = new StringBuilder();
json.append("{");
json.append("\"sensordataid\":\"" + sensorID + ":" + dt + ":" + tm + "\",");
json.append("\"sensor\":\"" + sensor + "\",");
json.append("\"floorNo\":\"" + floorNo + "\",");
json.append("\"roomNo\":\"" + roomNo + "\",");
json.append("\"isActive\":\"" + isActive + "\",");
json.append("\"smokeLevel\":\"" + smokeLevel + "\",");
json.append("\"co2Level\":\"" + co2Level + "\",");
json.append("\"date\":\"" + dt + "\",");
json.append("\"time\":\"" + tm + "\"");
json.append("}");

System.out.println(json);

post.setEntity(new StringEntity(json.toString()));

try (CloseableHttpClient httpClient = HttpClients.createDefault();
    CloseableHttpResponse response = httpClient.execute(post)) {

    result = EntityUtils.toString(response.getEntity());
}

return result;
}

```

RMI

FireAlarmServer.java

```

import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;

```

```

import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import java.net.MalformedURLException;
import java.rmi.AlreadyBoundException;
import java.rmi.Naming;
import java.io.Console;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.lang.SecurityManager;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;

public class FireAlarmServer extends UnicastRemoteObject implements FireAlarmService{

    protected FireAlarmServer() throws RemoteException {
        super();
    }

    @Override
    public String addSensor(String sensorid, int floor_no, String room_no, boolean is_active) throws RemoteException {

        String result = "";
        HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/load");

        post.addHeader("content-type", "application/json");

        StringBuilder json = new StringBuilder();
        json.append("{}");
        json.append("\"sensorid\":\""+sensorid+"\"");
        json.append("\"floor_no\":\""+floor_no+"\"");
        json.append("\"room_no\":\""+room_no+"\"");
        json.append("\"is_active\":\""+is_active+"\"");
        json.append("{}");

        System.out.println(json);

        // send a JSON data
        try {
            post.setEntity(new StringEntity(json.toString()));
        } catch (UnsupportedEncodingException e1) {
            e1.printStackTrace();
        }

        try (CloseableHttpClient httpClient = HttpClients.createDefault();
            CloseableHttpResponse response = httpClient.execute(post)) {

            result = EntityUtils.toString(response.getEntity());
        } catch (Exception e) {

        }

        System.out.println(result);

        return result;
    }
}

```

```
}
```

```
@Override
```

```
public String updateSensor(String sensorid, int floor_no, String room_no, boolean is_active) throws RemoteException {
```

```
    String result = "";
```

```
    HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/load");
```

```
    post.addHeader("content-type", "application/json");
```

```
    StringBuilder json = new StringBuilder();
```

```
    json.append("{}");
```

```
    json.append("\"sensorid\": "+sensorid+",");
```

```
    json.append("\"floor_no\": \""+floor_no+"\"");
```

```
    json.append("\"room_no\": \""+room_no+"\"");
```

```
    json.append("\"is_active\": \""+is_active+"\"");
```

```
    json.append("{}");
```

```
    System.out.println(json);
```

```
    // send a JSON data
```

```
    try {
```

```
        post.setEntity(new StringEntity(json.toString()));
```

```
    } catch (UnsupportedEncodingException e1) {
```

```
        e1.printStackTrace();
```

```
    }
```

```
    try (CloseableHttpClient httpClient = HttpClients.createDefault();
```

```
        CloseableHttpResponse response = httpClient.execute(post)) {
```

```
        result = EntityUtils.toString(response.getEntity());
```

```
    } catch (Exception e) {
```

```
    }
```

```
    System.out.println(result);
```

```
    return result;
```

```
}
```

```
@Override
```

```
public String deleteSensor(String sensorid) throws RemoteException {
```

```
    String result = "";
```

```
    HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/delete/"+sensorid+");
```

```
    try (CloseableHttpClient httpClient = HttpClients.createDefault();
```

```
        CloseableHttpResponse response = httpClient.execute(post)) {
```

```
        result = EntityUtils.toString(response.getEntity());
```

```
    } catch (IOException e) {
```

```
        e.printStackTrace();
```

```
    }
```

```
    System.out.println(result);
```

```
    return result;
```



```

}

@Override
public String getSensorDetails(String sensorid) throws RemoteException {

    String result = "";
    HttpGet get = new HttpGet("http://localhost:8080/rest/sensor/getsensor/"+sensorid+"");

    try (CloseableHttpClient httpClient = HttpClients.createDefault();
        CloseableHttpResponse response = httpClient.execute(get)) {

        result = EntityUtils.toString(response.getEntity());
    } catch (IOException e) {
        e.printStackTrace();
    }

    System.out.println(result);

    return result;
}

public String adminLogin(String username, String password) throws RemoteException {
    String res="";
    try {
        res = sendPOSTAdminLogin(username,password);
    } catch (IOException e) {
        e.printStackTrace();
    }

    return res;
}

public static void main(String[] args) {
    System.setProperty("java.security.policy", "file:allowall.policy");

    try{
        FireAlarmServer svr = new FireAlarmServer();
        // Bind the remote object's stub in the registry
        Registry registry = LocateRegistry.getRegistry();
        registry.bind("FireAlarmService", svr);

        System.out.println ("Service started....");
    }
    catch (RemoteException re){
        System.err.println(re.getMessage());
    }
    catch (AlreadyBoundException abe){
        System.err.println(abe.getMessage());
    }
}

private static String sendPOSTAdminLogin(String username, String password) throws IOException {

    String result = "";

```

```

HttpPost post = new HttpPost("http://localhost:8080/rest/users/auth");

post.addHeader("content-type", "application/json");

StringBuilder json = new StringBuilder();
json.append("{");
json.append("\"username\":\""+username+"");
json.append("\"password\":\""+password+"\"");
json.append("}");

System.out.println(json);

// send a JSON data
post.setEntity(new StringEntity(json.toString()));

try (CloseableHttpClient httpClient = HttpClients.createDefault();
    CloseableHttpResponse response = httpClient.execute(post)) {

    result = EntityUtils.toString(response.getEntity());
}

System.out.println(result);

return result;
}

@Override
public String getAllSensorDetails() throws RemoteException {

    String result = "";
    HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/all");

    post.addHeader("content-type", "application/json");

    StringBuilder json = new StringBuilder();
    json.append("{");
    json.append("}");

    System.out.println(json);

    // send a JSON data
    try {
        post.setEntity(new StringEntity(json.toString()));
    } catch (UnsupportedEncodingException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    try (CloseableHttpClient httpClient = HttpClients.createDefault();
        CloseableHttpResponse response = httpClient.execute(post)) {

        result = EntityUtils.toString(response.getEntity());
    } catch (IOException e) {
        e.printStackTrace();
    }

    System.out.println(result);

    return result;
}

```

```
}  
  
}
```

Desktop App

AdminDashboard.java

```
import org.apache.http.client.methods.CloseableHttpResponse;  
import org.apache.http.client.methods.HttpPost;  
import org.apache.http.entity.StringEntity;  
import org.apache.http.impl.client.CloseableHttpClient;  
import org.apache.http.impl.client.HttpClients;  
import org.apache.http.util.EntityUtils;  
  
import utils.Utils;  
  
import java.io.IOException;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
import java.util.concurrent.ThreadLocalRandom;  
  
public class FireSensorMain {  
    public static void main(String[] args) {  
  
        try {  
            while (true) {  
  
                String result = sendPOST(Utils.SENDPOSTSENSORDATAURL);  
  
                System.out.println(result);  
  
                try {  
                    Thread.sleep(30 * 1000);  
                } catch (InterruptedException e) {  
                    e.printStackTrace();  
                }  
            }  
  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
  
    private static String sendPOST(String url) throws IOException {  
  
        String result = "";  
        HttpPost post = new HttpPost(url);  
  
        post.addHeader("content-type", "application/json");  

```

```

DateTimeFormatter date = DateTimeFormatter.ofPattern("dd/MM/yyyy");
DateTimeFormatter time = DateTimeFormatter.ofPattern("HH:mm:ss");
LocalDateTime now = LocalDateTime.now();

String sensorID = "sensor4";
int floorNo = 2;
String roomNo = "12";
boolean isActive = true;
int smokeLevel = ThreadLocalRandom.current().nextInt(0, 10 + 1);
int co2Level = ThreadLocalRandom.current().nextInt(0, 10 + 1);

String dt = date.format(now);
String tm = time.format(now);
String sensor = "{ \"sensorid\": \"\" + sensorID + \"\" }";
StringBuilder json = new StringBuilder();
json.append("{");
json.append("\"sensordataid\": \"\" + sensorID + \"\" + dt + \"\" + tm + \"\",");
json.append("\"sensor\": \"\" + sensor + \"\",");
json.append("\"floorNo\": \"\" + floorNo + \"\",");
json.append("\"roomNo\": \"\" + roomNo + \"\",");
json.append("\"isActive\": \"\" + isActive + \"\",");
json.append("\"smokeLevel\": \"\" + smokeLevel + \"\",");
json.append("\"co2Level\": \"\" + co2Level + \"\",");
json.append("\"date\": \"\" + dt + \"\",");
json.append("\"time\": \"\" + tm + \"\",");
json.append("}");

System.out.println(json);

post.setEntity(new StringEntity(json.toString()));

try (CloseableHttpClient httpClient = HttpClients.createDefault();
    CloseableHttpResponse response = httpClient.execute(post)) {

    result = EntityUtils.toString(response.getEntity());
}

return result;
}
}

```

Login.java

```

import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;

```

```

import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.stage.Modality;
import javafx.stage.Stage;
import utils.Utils;

import java.sql.*;

import org.json.JSONException;
import org.json.JSONObject;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.Console;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.lang.SecurityManager;
import java.rmi.RemoteException;

public class Login {

    public static void adminLoginView() {
        Stage window = new Stage();
        window.initModality(Modality.APPLICATION_MODAL);
        window.setTitle("Admin Login");
        window.setWidth(400);
        window.setHeight(600);

        GridPane grid = new GridPane();
        grid.setAlignment(Pos.CENTER);
        grid.setHgap(10);
        grid.setVgap(10);
        grid.setPadding(new Insets(25, 25, 25, 25));

        Text scenetitle = new Text("Admin Login");
        scenetitle.setFont(Font.font("Tahoma", FontWeight.BOLD, 20));
        grid.add(scenetitle, 0, 0, 2, 1);

        Label userName = new Label("User Name:");
        grid.add(userName, 0, 1);

        TextField userTextField = new TextField();
        grid.add(userTextField, 1, 1);

        Label pw = new Label("Password:");
        grid.add(pw, 0, 2);

        PasswordField pwBox = new PasswordField();
        grid.add(pwBox, 1, 2);

        Button btn = new Button("Sign in");
        HBox hbBtn = new HBox(10);
        hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
        hbBtn.getChildren().add(btn);
        grid.add(hbBtn, 1, 4);

        btn.setOnAction(e->{

```

```

if(userTextField.getText().equals("") || pwBox.getText().equals("")) {
    Alert a = new Alert(AlertType.NONE);
    a.setAlertType(AlertType.ERROR);
    a.setContentText("Please Enter Username and Password");
    a.show();
}else {
    String username = userTextField.getText();
    String password = pwBox.getText();

    System.setProperty("java.security.policy", "file:allowall.policy");

    FireAlarmService service = null;
    try {
        service = (FireAlarmService) Naming.lookup("//localhost/FireAlarmService");
        String result = service.adminLogin(username,password);

        try {
            JSONObject jsonObject = new JSONObject(result);

            if(!jsonObject.isEmpty()) {
                //Successfully Login
                String loggedInName = ""; // jsonObject.name;
                window.close();
                AdminDashboard.displayAdminDashboard(loggedinName);
            }else {
                //Invalid Username or password
                Alert a = new Alert(AlertType.NONE);
                a.setAlertType(AlertType.ERROR);
                a.setContentText("Incorrect username or password");
                a.show();
            }

        }catch (JSONException err){
            System.out.println(err);
        }

        } catch (NotBoundException ex) {
            ex.printStackTrace();
        } catch (MalformedURLException ex) {
            System.err.println(ex.getMessage());
        } catch (RemoteException ex) {
            System.err.println(ex.getMessage());
        }

    }
});

Scene scene = new Scene(grid, 300, 275);
window.setScene(scene);

window.showAndWait();
}
}

```

SensorData.java

```
import javax.xml.bind.annotation.XmlRootElement;

import java.beans.property.SimpleStringProperty;

public class SensorData {
    private final SimpleStringProperty sensorid;
    private final SimpleStringProperty room_no;
    private final SimpleStringProperty floor_no;
    private final SimpleStringProperty is_active;
    private final SimpleStringProperty co2level;
    private final SimpleStringProperty smoke_level;
    private final SimpleStringProperty date;
    private final SimpleStringProperty time;

    SensorData(String sensorid, String room_no, String floor_no, String is_active, String co2level, String smoke_level,
               String date, String time) {

        this.sensorid = new SimpleStringProperty(sensorid);
        this.room_no = new SimpleStringProperty(room_no);
        this.floor_no = new SimpleStringProperty(floor_no);
        this.is_active = new SimpleStringProperty(is_active);
        this.co2level = new SimpleStringProperty(co2level);
        this.smoke_level = new SimpleStringProperty(smoke_level);
        this.date = new SimpleStringProperty(date);
        this.time = new SimpleStringProperty(time);
    }

    public String getSensorid() {
        return sensorid.get();
    }
    public String getRoom_no() {
        return room_no.get();
    }
    public String getFloor_no() {
        return floor_no.get();
    }
    public String getIs_active() {
        return is_active.get();
    }
    public String getCo2level() {
        return co2level.get();
    }
    public String getSmoke_level() {
        return smoke_level.get();
    }
    public String getDate() {
        return date.get();
    }
    public String getTime() {
        return time.get();
    }

    public void setSensorid(String sensorid) {
        this.sensorid.set(sensorid);
    }
    public void setRoom_no(String room_no) {
        this.room_no.set(room_no);
    }
}
```

```

    public void setFloor_no(String floor_no) {
        this.floor_no.set(floor_no);
    }
    public void setIs_active(String is_active) {
        this.is_active.set(is_active);
    }
    public void setco2level(String co2level) {
        this.co2level.set(co2level);
    }
    public void setSmoke_level(String smoke_level) {
        this.smoke_level.set(smoke_level);
    }
    public void setDate(String date) {
        this.date.set(date);
    }
    public void setTime(String time) {
        this.time.set(time);
    }
}

```

FireAlarmServer.java

```

import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import java.net.MalformedURLException;
import java.rmi.AlreadyBoundException;
import java.rmi.Naming;
import java.io.Console;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.lang.SecurityManager;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;

public class FireAlarmServer extends UnicastRemoteObject implements FireAlarmService{

    protected FireAlarmServer() throws RemoteException {
        super();
    }

    @Override
    public String addSensor(String sensorid, int floor_no, String room_no, boolean is_active) throws RemoteException {

        String result = "";
        HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/load");

        post.addHeader("content-type", "application/json");
    }
}

```



```

Stringbuilder json = new StringBuilder();
json.append("{}");
json.append("\"sensorid\":\""+sensorid+"");
json.append("\"floor_no\":\""+floor_no+"\"");
json.append("\"room_no\":\""+room_no+"\"");
json.append("\"is_active\":\""+is_active+"\"");
json.append("{}");

System.out.println(json);

// send a JSON data
try {
    post.setEntity(new StringEntity(json.toString()));
} catch (UnsupportedEncodingException e1) {
    e1.printStackTrace();
}

try (CloseableHttpClient httpClient = HttpClients.createDefault();
    CloseableHttpResponse response = httpClient.execute(post)) {

    result = EntityUtils.toString(response.getEntity());
} catch (Exception e) {

}

System.out.println(result);

return result;

}

@Override
public String updateSensor(String sensorid, int floor_no, String room_no, boolean is_active) throws RemoteException {

    String result = "";
    HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/load");

    post.addHeader("content-type", "application/json");

    StringBuilder json = new StringBuilder();
    json.append("{}");
    json.append("\"sensorid\":\""+sensorid+"");
    json.append("\"floor_no\":\""+floor_no+"\"");
    json.append("\"room_no\":\""+room_no+"\"");
    json.append("\"is_active\":\""+is_active+"\"");
    json.append("{}");

    System.out.println(json);

    // send a JSON data
    try {
        post.setEntity(new StringEntity(json.toString()));
    } catch (UnsupportedEncodingException e1) {
        e1.printStackTrace();
    }

    try (CloseableHttpClient httpClient = HttpClients.createDefault();
        CloseableHttpResponse response = httpClient.execute(post)) {

```

```

        result = EntityUtils.toString(response.getEntity());
    } catch (Exception e) {

    }

    System.out.println(result);

    return result;
}

@Override
public String deleteSensor(String sensorid) throws RemoteException {
    String result = "";
    HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/delete/"+sensorid+"");

    try (CloseableHttpClient httpClient = HttpClients.createDefault();
        CloseableHttpResponse response = httpClient.execute(post)) {

        result = EntityUtils.toString(response.getEntity());
    } catch (IOException e) {
        e.printStackTrace();
    }

    System.out.println(result);

    return result;
}

@Override
public String getSensorDetails(String sensorid) throws RemoteException {

    String result = "";
    HttpGet get = new HttpGet("http://localhost:8080/rest/sensor/getsensor/"+sensorid+"");

    try (CloseableHttpClient httpClient = HttpClients.createDefault();
        CloseableHttpResponse response = httpClient.execute(get)) {

        result = EntityUtils.toString(response.getEntity());
    } catch (IOException e) {
        e.printStackTrace();
    }

    System.out.println(result);

    return result;
}

public String adminLogin(String username, String password) throws RemoteException {
    String res="";
    try {
        res=sendPOSTAdminLogin(username,password);
    } catch (IOException e) {

```

```

        e.printStackTrace();
    }

    return res;
}

public static void main(String[] args) {
    System.setProperty("java.security.policy", "file:allowall.policy");

    try{
        FireAlarmServer svr = new FireAlarmServer();
        // Bind the remote object's stub in the registry
        Registry registry = LocateRegistry.getRegistry();
        registry.bind("FireAlarmService", svr);

        System.out.println ("Service started....");
    }
    catch (RemoteException re){
        System.err.println(re.getMessage());
    }
    catch (AlreadyBoundException abe){
        System.err.println(abe.getMessage());
    }
}

private static String sendPOSTAdminLogin(String username, String password) throws IOException {

    String result = "";
    HttpPost post = new HttpPost("http://localhost:8080/rest/users/auth");

    post.addHeader("content-type", "application/json");

    StringBuilder json = new StringBuilder();
    json.append("{}");
    json.append("\"username\":\""+username+"");
    json.append("\"password\":\""+password+"\"");
    json.append("}");

    System.out.println(json);

    // send a JSON data
    post.setEntity(new StringEntity(json.toString()));

    try (CloseableHttpClient httpClient = HttpClients.createDefault();
        CloseableHttpResponse response = httpClient.execute(post)) {

        result = EntityUtils.toString(response.getEntity());
    }

    System.out.println(result);

    return result;
}

@Override
public String getAllSensorDetails() throws RemoteException {

    String result = "";

```

```

HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/all");

post.setHeader("content-type", "application/json");

StringBuilder json = new StringBuilder();
json.append("{");
json.append("}");

System.out.println(json);

// send a JSON data
try {
    post.setEntity(new StringEntity(json.toString()));
} catch (UnsupportedEncodingException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

try (CloseableHttpClient httpClient = HttpClients.createDefault();
    CloseableHttpResponse response = httpClient.execute(post)) {

    result = EntityUtils.toString(response.getEntity());
} catch (IOException e) {
    e.printStackTrace();
}

System.out.println(result);

return result;

}
}

```

ClientApp.java

```

import com.google.gson.*;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Stage;
import org.apache.http.HttpEntity;
import org.apache.http.ParseException;

```

```

import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import java.io.IOException;

public class ClientApp extends Application {

    Stage window;

    private TableView<SensorData> table = new TableView<SensorData>();
    private ObservableList<SensorData> data =
        FXCollections.observableArrayList();

    public static void main(String[] args) {
        // start();
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        window = primaryStage;
        window.setTitle("Fire-Sensor Desktop Client");
        getsensorData();
        BorderPane borderpane = new BorderPane();
        borderpane.setTop(addHBox());
        borderpane.setCenter(addVBox());

        Scene scene = new Scene(borderpane, 800, 400);
        window.setScene(scene);
        window.show();
    }

    public HBox addHBox() {
        HBox hbox = new HBox();
        hbox.setPadding(new Insets(15, 12, 15, 12));
        hbox.setSpacing(10);
        hbox.setStyle("-fx-background-color: #336699;");

        Button buttonCurrent = new Button("Admin Login");
        buttonCurrent.setPrefSize(150, 20);

        buttonCurrent.setOnAction(e->{
            Login.adminLoginView();
        });

        hbox.getChildren().addAll(buttonCurrent);

        return hbox;
    }

    public VBox addVBox() {

        final Label label = new Label("Fire-Sensor Monitor");
        label.setFont(new Font("Arial", 20));
    }

```

```

// TableView table = new TableView();

table.setEditable(false);

TableColumn sensoridCol = new TableColumn("Sensor ID");
sensoridCol.setMinWidth(100);
sensoridCol.setCellValueFactory( new PropertyValueFactory<SensorData, String>("sensorid"));

TableColumn room_noCol = new TableColumn("Room No");
room_noCol.setMinWidth(100);
room_noCol.setCellValueFactory( new PropertyValueFactory<SensorData, String>("room_no"));

TableColumn floor_noCol = new TableColumn("Floor No");
floor_noCol.setMinWidth(100);
floor_noCol.setCellValueFactory( new PropertyValueFactory<SensorData, String>("floor_no"));

TableColumn is_activeCol = new TableColumn("Sensor Status");
is_activeCol.setMinWidth(100);
is_activeCol.setCellValueFactory( new PropertyValueFactory<SensorData, String>("is_active"));

TableColumn co2levelCol = new TableColumn("CO2 Level");
co2levelCol.setMinWidth(100);
co2levelCol.setCellValueFactory( new PropertyValueFactory<SensorData, String>("co2level"));

TableColumn smoke_levelCol = new TableColumn("Smoke Level");
smoke_levelCol.setMinWidth(100);
smoke_levelCol.setCellValueFactory( new PropertyValueFactory<SensorData, String>("smoke_level"));

TableColumn dateCol = new TableColumn("Date");
dateCol.setMinWidth(100);
dateCol.setCellValueFactory( new PropertyValueFactory<SensorData, String>("date"));

TableColumn timeCol = new TableColumn("Time");
timeCol.setMinWidth(100);
timeCol.setCellValueFactory( new PropertyValueFactory<SensorData, String>("time"));

table.setItems(data);

table.getColumns().addAll(sensoridCol, room_noCol, floor_noCol, is_activeCol, co2levelCol, smoke_levelCol, dateCol,
timeCol);

final VBox vbox1 = new VBox();
vbox1.setSpacing(5);
vbox1.setPadding(new Insets(10, 0, 0, 10));
vbox1.getChildren().addAll(label, table);

return vbox1;
}

public void getsensorData() {

String result = "";
HttpGet request = new HttpGet("http://localhost:8080/rest/sensordata/all");

//
//
//
//
// try (CloseableHttpClient httpClient = HttpClients.createDefault();
//     CloseableHttpResponse response = httpClient.execute(get)) {

```

```

//
// result = EntityUtils.toString(response.getEntity());
// } catch (IOException e) {
//     e.printStackTrace();
// }

try (CloseableHttpClient httpClient = HttpClients.createDefault();
    CloseableHttpResponse response = httpClient.execute(request)) {

    // Get HttpResponse Status
    System.out.println(response.getProtocolVersion()); // HTTP/1.1
    System.out.println(response.getStatusLine().getStatusCode()); // 200
    System.out.println(response.getStatusLine().getReasonPhrase()); // OK
    System.out.println(response.getStatusLine().toString()); // HTTP/1.1 200 OK

    HttpEntity entity = response.getEntity();
    if (entity != null) {
        // return it as a String
        result = EntityUtils.toString(entity);
    }
    System.out.println(result);

} catch (ParseException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

data.removeAll(data);
// System.out.println(result);
JSONArray arr = new Gson().fromJson(result, JSONArray.class);
for (JsonElement e : arr) {
    System.out.println(e.getAsJsonObject());
    JsonObject o = e.getAsJsonObject();

    data.add(new SensorData(
        o.get("sensor").getAsJsonObject().get("sensorid").toString(),
        o.get("sensor").getAsJsonObject().get("roomNo").toString(),
        o.get("sensor").getAsJsonObject().get("floorNo").toString(),
        o.get("sensor").getAsJsonObject().get("active").toString(),
        o.get("co2Level").toString(),
        o.get("smokeLevel").toString(),
        o.get("date").toString(),
        o.get("time").toString()));
}

// System.setProperty("java.security.policy", "file:allowall.policy");
//
// FireAlarmService service = null;
// try {
//     service = (FireAlarmService) Naming.lookup("//localhost/FireAlarmService");
//     String result = service.getAllSensorDetails();
//     try {
//         JSONObject jsonObject = new JSONObject(result);
//         // jsonObject to ObservableList<SensorData> data
//     } catch (JSONException err) {
//         System.out.println(err);
//     }
// }

```

```
//     }  
// } catch (NotBoundException ex) {  
//     ex.printStackTrace();  
// } catch (MalformedURLException ex) {  
//     System.err.println(ex.getMessage());  
// } catch (RemoteException ex) {  
//     System.err.println(ex.getMessage());  
// }  
  
}  
  
// @Override  
// public void run() {  
//     getsensorData();  
//     try {  
//         Thread.sleep(30);  
//     } catch (InterruptedException e) {  
//         e.printStackTrace();  
//     }  
// }  
// }  
// }  
// public static void start () {  
//     Thread t = new Thread ();  
//     t.start ();  
// }  
}
```