**Report**

**Distributed Systems**

|  |  |
| --- | --- |
| **Registration No** | **Name** |
| IT18011494 | Nasome P.V.R.D.V. |
| IT18044096 | Malavige M.O.D. |

**High Level Diagram**

**A screenshot of a cell phone

Description automatically generated**

**Interface TemperatureSensor**

* public String getConnection()

Checking whether the connection is successful or not.

* public void addTemperatureListener(TemperatureListener listener )

Adding the desktop object as a listener

* public void removeTemperatureListener(TemperatureListener listener )

Removing the desktop object

* public String displaySensors()

Sending all the sensors details by connecting with REST API.

If levels are exceeding sending notifyListeners.

* public String login(String un,String pw)

Receiving user entity from REST API by providing user name and send whether the login is success or not.

**REST API(services)**

**sensor**

public List<sensor> getAll()

🡪(<http://localhost:8080/sensor>)

Receiving all sensors details

public ResponseEntity<?> addSensorDetails(@RequestBody sensor s) 🡪(<http://localhost:8080//sensor/add>)

Adding/Updating a new sensor details.

Simulating email and msg as S.O.P.

public ResponseEntity<?> getLocation(@PathVariable String location) 🡪([http://localhost:8080/sensor/{location}](http://localhost:8080/sensor/%7blocation%7d))

Receive a particular sensor entity (sensor details) by proving a sensor’s location

**user**

public List<user> getAll()

🡪(<http://localhost:8080/user>)

Get all users details

public ResponseEntity<?> getName(@PathVariable String name)

🡪([http://localhost:8080/user/{name}](http://localhost:8080/user/%7bname%7d))

Get a particular user entity by proving user name

Diagram 1(Methods that are in red are services.)A screenshot of a cell phone

Description automatically generated

Diagram 2

A close up of a piece of paper

Description automatically generated

Appendix 1

**React Application-WEB CLIENT**

**App.js**

**import *React***, { Component } **from 'react'**;  
**class** App **extends** Component {  
 constructor(props) {  
 **super**(props);  
 **this**.**state** = {  
 **sensors**: []  
 }  
 }

*//Calling this method every 40 s to retrive the senosr data from rest api*

**async** componentDidMount() {  
 **try** {  
 *setInterval*(**async** () => {  
 **const** url = **"http://localhost:8080/sensor"**;  
 *fetch*(url)  
 .then(response => response.json())  
 .then(json => **this**.setState({ **sensors**: json }))  
 }, 40000);  
 } **catch**(e) {  
 ***console***.log(e);  
 }  
 }

*//checking whether the smoke level or/and is above or equal to 5 and produce alert boxes*

highSml(smokelevel,co2){  
 *//alert('smoke level is high');* **if** (smokelevel>=5 || co2>=5){  
 *alert*(**'Sensor is exceeding limits'**);  
 }  
 }  
 render() {  
 **const** { sensors } = **this**.**state**;  
 **return** (  
  
 <**div className="container"**>  
 <**div className="jumbotron"**>  
 <**h1 className="display-4"**>Sensor Details</**h1**>  
 </**div**>  
 {sensors.map((post) => (  
 <**div className="card" key=**{post.**location**}>  
 <**div className="card-header"**>  
 <**table class="table table-striped table-bordered"**>  
  
 <**th**>Location</**th**>  
 <**th**>Activity</**th**>  
 <**th**>Smoke Level</**th**>  
 <**th**>Co2 Level</**th**>  
  
 <**tr**>  
 <**td**>{post.**location**}</**td**>  
 <**td**>{post.activity}</**td**>  
 <**td**>{post.smokelevel}</**td**>  
 <**td**>{post.co2}</**td**>  
 </**tr**>  
  
 {**this**.highSml(post.smokelevel,post.co2)}  
 </**table**>  
 </**div**>  
  
 </**div**>  
 ))}  
 </**div**>  
 );  
 }  
}  
**export default** App;

**Springboot Application-REST API**

**MainRestClass**

package com.example.REST;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

@EnableJpaRepositories(basePackages="com.example.REST.repository")

@SpringBootApplication

public class MainRestClass {

public static void main(String[] args) {

// TODO Auto-generated method stub

SpringApplication.run(MainRestClass.class, args);

}

}

**sensor**

package com.example.REST.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

@Entity

public class sensor {

@Id

@Column(name="location")

private String location;

@Column(name="activity")

private String activity;

@Column(name="smokelevel")

private int smokelevel;

@Column(name="co2")

private int co2;

public sensor() {

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

public String getActivity() {

return activity;

}

public void setActivity(String activity) {

this.activity = activity;

}

public int getSmokelevel() {

return smokelevel;

}

public void setSmokelevel(int smokelevel) {

this.smokelevel = smokelevel;

}

public int getCo2() {

return co2;

}

public void setCo2(int co2) {

this.co2 = co2;

}

}

**user**

package com.example.REST.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

@Entity

public class user {

@Id

@GeneratedValue

@Column(name="id")

private int id;

@Column(name="login")

private String login;

@Column(name="password")

private String password;

public user() {

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getLogin() {

return login;

}

public void setLogin(String login) {

this.login= login;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**Sensor\_repositary**

package com.example.REST.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.REST.model.sensor;

public interface sensor\_repositary extends JpaRepository<sensor, Integer> {

sensor findBylocation(String location);

}

**User\_repositary**

package com.example.REST.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.example.REST.model.user;

public interface user\_repositary extends JpaRepository<user, Integer>{

user findBylogin(String login);

}

**Sensor\_resource**

package com.example.REST.resource;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.example.REST.model.sensor;

import com.example.REST.model.user;

import com.example.REST.repository.sensor\_repositary;

@CrossOrigin

@RestController

public class sensor\_resource {

@Autowired

sensor\_repositary rep;

//These methods can used by any applications.In order to retrieve details

@RequestMapping(value="/sensor")

public List<sensor> getAll(){

return rep.findAll();

}

@PostMapping(value="/sensor/add")

public ResponseEntity<?> addSensorDetails(@RequestBody sensor s){

rep.save(s);

//When updating checking whether the exceeding of level and simulating sending emails and massages

if(s.getCo2()>=5 || s.getSmokelevel()>=5 ) {

System.out.println("Smoke levels and or co2 levels are exceeding. Sending emails and massages.................");

}

return new ResponseEntity<String>("success",HttpStatus.OK);

}

@GetMapping(value="/sensor/{location}")

public ResponseEntity<?> getLocation(@PathVariable String location){

sensor sensor=rep.findBylocation(location);

return new ResponseEntity<sensor>(sensor,HttpStatus.OK);

}

}

**User\_resource**

package com.example.REST.resource;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.example.REST.model.sensor;

import com.example.REST.model.user;

import com.example.REST.repository.user\_repositary;

@RestController

public class user\_resource {

@Autowired

user\_repositary u\_m\_r;

//These methods can used by any applications.In order to retrieve details

@RequestMapping(value="/user")

public List<user> getAll(){

System.out.println("details");

System.out.println(u\_m\_r.findAll());

return u\_m\_r.findAll();

}

@GetMapping(value="/user/{name}")

public ResponseEntity<?> getName(@PathVariable String name){

user user=u\_m\_r.findBylogin(name);

return new ResponseEntity<user>(user,HttpStatus.OK);

}}

**Application.yml**

**spring:**

**datasource:**

**driverClassName:** com.mysql.jdbc.Driver

**url:** jdbc:mysql://localhost:3306/sensor

**username:** root

**password:** 1234

**jpa:**

**hibernate.ddl-auto:** update

**generate-ddl:** true

**show-sql:** true

**Temp-FOLDER RMI CLIENT(DESKTOP) AND SERVER**

**Desktop**

public class Desktop extends javax.swing.JFrame implements TemperatureListener,Runnable,Serializable{

/\*\*

\* Creates new form Desktop

\*/

private static final long serialVersionUID = 1L;

ArrayList<sensor> arr;

/\*private String y="";

private String x="";\*/

public Desktop() {

initComponents();

//After Login as Admin this will be visible

adminpanel.setVisible(false);

}

//This method will be called after click add New Sensor button.It will create a new Sensor App Class

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

SensorApp sensor=new SensorApp();

Thread t =new Thread(sensor);

t.start();

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

try {

// Creating a get Registry from TemperatureSensorServer registry.And check with the login credentials that is send by the TemperatureSensorServer.

Registry reg= LocateRegistry.getRegistry("localhost", 4444);

TemperatureSensor sensor=(TemperatureSensor) reg.lookup("welcome");

String name=un.getText().toString();

String password=pw.getText().toString();

String x=sensor.login(name,password);

System.out.println("displaying ....login"+x);

if(x.equals("logged")){

if(name.equals("admin")){

adminpanel.setVisible(true);

}

}

} catch (RemoteException ex) {

Logger.getLogger(Desktop.class.getName()).log(Level.SEVERE, null, ex);

} catch (NotBoundException ex) {

Logger.getLogger(Desktop.class.getName()).log(Level.SEVERE, null, ex);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Desktop.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Desktop.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Desktop.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Desktop.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

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

try {

//Get the Connection Registry from TemperatureSensorServer

Registry reg= LocateRegistry.getRegistry("localhost", 4444);

TemperatureSensor sensor=(TemperatureSensor) reg.lookup("welcome");

String reading = sensor.getConnection();

System.out.println("Progress:"+reading);

Desktop monitor = new Desktop();

//Add a new remote monitor Object.

UnicastRemoteObject.exportObject(monitor, 0);

sensor.addTemperatureListener(monitor);

monitor.run();

} catch (RemoteException re) {

System.out.println(re);

} catch (NotBoundException nbe) {

System.out.println(nbe);

}

}

@Override

public void run() {

System.out.println("Running Desktop Appplication......");

this.setVisible(true);

//Reading data from API in every 30s and display the data.if exceeded the fire limits call notify listeneser method.

for(;;){

String xx="";

try {

Registry reg= LocateRegistry.getRegistry("localhost", 4444);

TemperatureSensor sensor=(TemperatureSensor) reg.lookup("welcome");

String x=sensor.displaySensors();

System.out.println("desktop display");

String str[] = x.split(",");

List<String> al = new ArrayList<String>();

al = Arrays.asList(str);

//System.out.println("dispay123"+al);

for(String s: al){

String str2[] = s.split(",");

List<String> a2 = new ArrayList<String>();

a2 = Arrays.asList(str2);

for(String y:a2){

System.out.println(y);

xx=xx.concat(y);

tt.setText("displaying \n\n"+xx);

}

xx=xx.concat("\n");

}

Thread.sleep(30000);

} catch (RemoteException ex) {

Logger.getLogger(Desktop.class.getName()).log(Level.SEVERE, null, ex);

} catch (NotBoundException ex) {

Logger.getLogger(Desktop.class.getName()).log(Level.SEVERE, null, ex);

} catch (InterruptedException ex) {

Logger.getLogger(Desktop.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

//This method is called inside notifyListeners in TemperatureSENSORSERVER.

@Override

public void temperatureChanged(String loc) throws RemoteException {

System.out.println("\nTemperature change event : Sending massages and Emails for exceeding sensors in :"+loc+"\n");

//Starting a new email class

Emal\_msg email=new Emal\_msg(loc);

Thread t1 =new Thread(email);

t1.start();

}

}

**Email\_msg**

package TemperatureMonitor\_Code;

import java.util.logging.Level;

import java.util.logging.Logger;

/\*\*

\*

\* @author vnaso

\* This is email class.This doesnt have a main method

\* this is call inside desktop to simulate the email and msg behaviour

\*/

public class Emal\_msg extends javax.swing.JFrame implements Runnable{

/\*\*

\* Creates new form Emal\_msg

\*/

private String loc;

public Emal\_msg() {

initComponents();

}

public Emal\_msg(String loc){

initComponents();

this.loc=loc;

this.setVisible(true);

String x="Attention!!!\n You have recieved 1 msg and 1 email \n regarding fire alarm."+"\n"+loc+" :has exceeded the fire limits\n\n";

location.setText(x);

}

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;

private javax.swing.JTextField location;

// End of variables declaration

@Override

public void run() {

try {

Thread.sleep(10000);

System.out.println("dispose jframe");

this.dispose();

} catch (InterruptedException ex) {

Logger.getLogger(Emal\_msg.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

**SensorApp**

public class SensorApp extends javax.swing.JFrame implements Runnable{

// Use this con variable to start the connection in update method.

private static HttpURLConnection con;

Boolean checkStart=false;

/\*\*

\* Creates new form frame1

\*/

public SensorApp() {

initComponents();

}

/\*\*

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

//Calling the Add Sensor Method

addSensorMethod("active",3,3,location123.getText());

panel.setVisible(false);

}

private void location123ActionPerformed(java.awt.event.ActionEvent evt) {

}

@Override

public void run() {

String activity="active";

String location=location123.getText();

//Initially adding values to smokelevel and CO2.

//This values will be changing in every 40s with using random number genarator and a simple logic.

int smokeLevel=4;

int co2=1;

Random r=new Random();

this.setVisible(true);

for(;;){

int num = r.nextInt();

if(smokeLevel>0 &&smokeLevel<11){

if(num%2==0)

{

if(smokeLevel!=10){

smokeLevel=smokeLevel+1;

}

}else{

if(smokeLevel!=1){

smokeLevel=smokeLevel-1;

}

}

}else{

smokeLevel=4;

}

if(co2>0 &&co2<11){

if(num%2==0)

{

if(co2!=10){

co2=co2+1;

}

}else{

if(co2!=1){

co2=co2-1;

}

}

}else{

co2=2;

}

//CheckStart variable is used to stop calling UpdateSensorMethod and Displaysensor method before adding sensor details to the api and database.

if(checkStart==true){

System.out.println("#######"+location+" "+activity+" "+" Smokelevel:"+smokeLevel+" Co2level:"+co2+"####"+"\n");

UpdateSensorMethod(activity, smokeLevel, co2, location);

DisplaySensorMethod();

}

try {

Thread.sleep(10000);

} catch (InterruptedException ex) {

Logger.getLogger(SensorApp.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

//Using API add sensor details.

private void addSensorMethod(String ac,int sm,int c,String loc){

try {

URL myurl = new URL("http://localhost:8080/sensor/add");

HttpURLConnection con12 = (HttpURLConnection)myurl.openConnection();

con12.setRequestMethod("POST");

con12.setRequestProperty("Content-Type", "application/json; utf-8");

con12.setRequestProperty("Accept", "application/json");

con12.setDoOutput(true);

String passingData = "{\"location\": \""+loc+"\", \"activity\": \""+ac+"\",\"smokelevel\": \""+sm+"\",\"co2\": \""+c+"\"}";

try(OutputStream os = con12.getOutputStream()) {

byte[] input = passingData.getBytes("utf-8");

os.write(input, 0, input.length);

}

int code = con12.getResponseCode();

System.out.println(code);

try(BufferedReader br = new BufferedReader(new InputStreamReader(con12.getInputStream(), "utf-8"))){

StringBuilder response = new StringBuilder();

String responseLine = null;

while ((responseLine = br.readLine()) != null) {

response.append(responseLine.trim());

}

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

}

checkStart=true;

} catch (IOException ex) {

Logger.getLogger(SensorApp.class.getName()).log(Level.SEVERE, null, ex);

}

}

//Using API updating sensor datails.

private void UpdateSensorMethod(String ac,int sm,int c,String loc){

try {

URL myurl = new URL("http://localhost:8080/sensor/add");

HttpURLConnection con12 = (HttpURLConnection)myurl.openConnection();

con12.setRequestMethod("POST");

con12.setRequestProperty("Content-Type", "application/json; utf-8");

con12.setRequestProperty("Accept", "application/json");

con12.setDoOutput(true);

String locNew=location123.getText().toString();

String passingData = "{\"location\": \""+locNew+"\", \"activity\": \""+ac+"\",\"smokelevel\": \""+sm+"\",\"co2\": \""+c+"\"}";

try(OutputStream os = con12.getOutputStream()) {

byte[] input = passingData.getBytes("utf-8");

os.write(input, 0, input.length);

}

int code = con12.getResponseCode();

System.out.println(code);

try(BufferedReader br = new BufferedReader(new InputStreamReader(con12.getInputStream(), "utf-8"))){

StringBuilder response = new StringBuilder();

String responseLine = null;

while ((responseLine = br.readLine()) != null) {

response.append(responseLine.trim());

}

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

}

} catch (IOException ex) {

Logger.getLogger(SensorApp.class.getName()).log(Level.SEVERE, null, ex);

}

}

//This method is used to call to display sensor details in sensorApp UI.Sensor data is retrived from API.

private void DisplaySensorMethod(){

try {

URL myurl = new URL("http://localhost:8080/sensor/"+location123.getText());

con = (HttpURLConnection) myurl.openConnection();

con.setRequestMethod("GET");

System.out.println("showing updating.....1");

if(con.getResponseCode()==200){

System.out.println("showing updating.....2");

InputStream im=con.getInputStream();

StringBuffer sb=new StringBuffer();

BufferedReader br=new BufferedReader(new InputStreamReader(im));

String line=br.readLine();

while(line != null){

System.out.println("showing updating....."+line);

JSONObject jsonObj = new JSONObject(line);

String location=jsonObj.getString("location");

String activity=jsonObj.getString("activity");

int smokelevel=jsonObj.getInt("smokelevel");

int co2=jsonObj.getInt("co2");

det.setText("#######"+location+"\*\* \*\*"+activity+"\*\* \*\*"+" Smokelevel:"+smokelevel+" Co2level:"+co2+"####"+"\n");

line=br.readLine();

}

}

} catch (MalformedURLException ex) {

Logger.getLogger(TemperatureSensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (ProtocolException ex) {

Logger.getLogger(TemperatureSensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(TemperatureSensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (JSONException ex) {

Logger.getLogger(TemperatureSensorServer.class.getName()).log(Level.SEVERE, null, ex);

} finally {

con.disconnect();

}

}

}

**TemperatureListener**

package TemperatureMonitor\_Code;

interface TemperatureListener extends java.rmi.Remote

{

public void temperatureChanged(String loc) throws java.rmi.RemoteException;

}

**TemperatureSensor**

package TemperatureMonitor\_Code;

import java.util.ArrayList;

interface TemperatureSensor extends java.rmi.Remote

{

public String getConnection() throws

java.rmi.RemoteException;

public void addTemperatureListener

(TemperatureListener listener )

throws java.rmi.RemoteException;

public void removeTemperatureListener

(TemperatureListener listener )

throws java.rmi.RemoteException;

public String displaySensors() throws java.rmi.RemoteException;

public String login(String un,String pw) throws java.rmi.RemoteException;

}

**TemperatureSensorServer**

public class TemperatureSensorServer extends UnicastRemoteObject implements

TemperatureSensor, Runnable {

private static HttpURLConnection con;

//Make ArrayList Synchronized inorder to add or get arraylist objects.

List<TemperatureListener> list = Collections.synchronizedList(new ArrayList<TemperatureListener>());

private ArrayList<String> sensorslist = new ArrayList<>();

public TemperatureSensorServer() throws java.rmi.RemoteException {

}

//Notify about the connection with rmi server

public String getConnection() throws java.rmi.RemoteException {

return "Connnected to rmi server";

}

//Add a listener to arralist

public void addTemperatureListener(TemperatureListener listener)

throws java.rmi.RemoteException {

System.out.println("adding listener -" + listener);

list.add(listener);

}

//Remove an listener

public void removeTemperatureListener(TemperatureListener listener)

throws java.rmi.RemoteException {

System.out.println("removing listener -" + listener);

list.remove(listener);

}

//To check login with provided username and password and return "logged" if credentials are true

@Override

public String login(String un,String pw) throws RemoteException {

System.out.println("Inside login method"+un+"\n\n\n");

try {

URL myurl = new URL("http://localhost:8080/user/"+un);

con = (HttpURLConnection) myurl.openConnection();

con.setRequestMethod("GET");

if(con.getResponseCode()==200){

InputStream im=con.getInputStream();

StringBuffer sb=new StringBuffer();

BufferedReader br=new BufferedReader(new InputStreamReader(im));

String line=br.readLine();

while(line != null){

System.out.println("displayig line....."+line);

JSONObject jsonObj = new JSONObject(line);

String getpw=jsonObj.getString("password");

if(getpw.equals(pw)){

return "logged";

}

line=br.readLine();

}

}

} catch (MalformedURLException ex) {

Logger.getLogger(TemperatureSensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (ProtocolException ex) {

Logger.getLogger(TemperatureSensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(TemperatureSensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (JSONException ex) {

Logger.getLogger(TemperatureSensorServer.class.getName()).log(Level.SEVERE, null, ex);

} finally {

con.disconnect();

}

return "incorrect";

}

//This is the main thread

public void run() {

try {

System.out.println("starting.........");

//this is run after every 5 s

//retrieve the sensor detalis and notify if exceeded

for(;;){

sensorslist.clear();

System.out.println("Refresh Sensor Data.........");

URL u=new URL("http://localhost:8080/sensor");

HttpURLConnection hr=(HttpURLConnection) u.openConnection();

if(hr.getResponseCode()==200){

InputStream im=hr.getInputStream();

StringBuffer sb=new StringBuffer();

BufferedReader br=new BufferedReader(new InputStreamReader(im));

String line=br.readLine();

while(line != null){

JSONArray jsonArr = new JSONArray(line);

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

{

JSONObject jsonObj = jsonArr.getJSONObject(i);

System.out.println("json object"+jsonObj);

sensor ob=new sensor();

ob.setLocation(jsonObj.getString("location"));

ob.setActivity(jsonObj.getString("activity"));

ob.setCo2(jsonObj.getInt("co2"));

ob.setSmokelevel(jsonObj.getInt("smokelevel"));

if(jsonObj.getInt("co2")>=5 || jsonObj.getInt("smokelevel")>=5 ){

if(jsonObj.getInt("co2")>=5 && jsonObj.getInt("smokelevel")>=5 ){

ob.setHigh("Your co2 and smoke level is above or equals to 5.");

this.notifyListeners( ob.getLocation());

}else if(jsonObj.getInt("co2")>=5){

ob.setHigh("Your co2 is above or equals to 5.");

this.notifyListeners(ob.getLocation());

}else{

ob.setHigh("Your smoke level is above or equals to 5.");

this.notifyListeners(ob.getLocation());

}

}

System.out.println("data"+ob);

String toStringVal=ob.toString();

sensorslist.add(toStringVal);

}

line=br.readLine();

}

}

Thread.sleep(15000);

}

} catch (Exception ex) {

Logger.getLogger(TemperatureSensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

}

//notify if exceeded

private void notifyListeners(String loc) {

try{

for (TemperatureListener t:list){

t.temperatureChanged(loc);

}

}catch(RemoteException e){}

}

@Override

public String displaySensors() throws java.rmi.RemoteException{

//Displaying all sensors details.

String listString = String.join(", ", sensorslist);

return listString;

}

public static void main(String[] args) {

//To start the rmi server

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

System.out.println("Loading temperature service");

try {

TemperatureSensorServer sensor = new TemperatureSensorServer();

Registry reg=LocateRegistry.createRegistry(4444);

reg.rebind("welcome", sensor);

System.out.println("server is ready.........");

Thread thread = new Thread(sensor);

thread.start();

} catch (RemoteException re) {

System.err.println("Remote Error - " + re);

} catch (Exception e) {

System.err.println("Error - " + e);

}

}

}

**Allowall.policy**

grant{

permission java.security.AllPermission;

};

**Sensor**

package TemperatureMonitor\_Code;

public class sensor {

private String location;

private String activity;

private int smokelevel;

private int co2;

private String high;

public String getHigh() {

return high;

}

public void setHigh(String high) {

this.high = high;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

public String getActivity() {

return activity;

}

public void setActivity(String activity) {

this.activity = activity;

}

public int getSmokelevel() {

return smokelevel;

}

public void setSmokelevel(int smokelevel) {

this.smokelevel = smokelevel;

}

public int getCo2() {

return co2;

}

public void setCo2(int co2) {

this.co2 = co2;

}

@Override

public String toString() {

return "sensor{" + "location=" + location + ", activity=" + activity + ", smokelevel=" + smokelevel + ", co2=" + co2 + ", high=" + high + '}';

}

}