

Fire Alarm Monitoring System

Assignment 2

Distributed Systems

(SE3020)

|  |  |
| --- | --- |
| IT18046830 | Samaraweera D.H.M. |
| IT18028874 | Deshan A.S.S.B. |

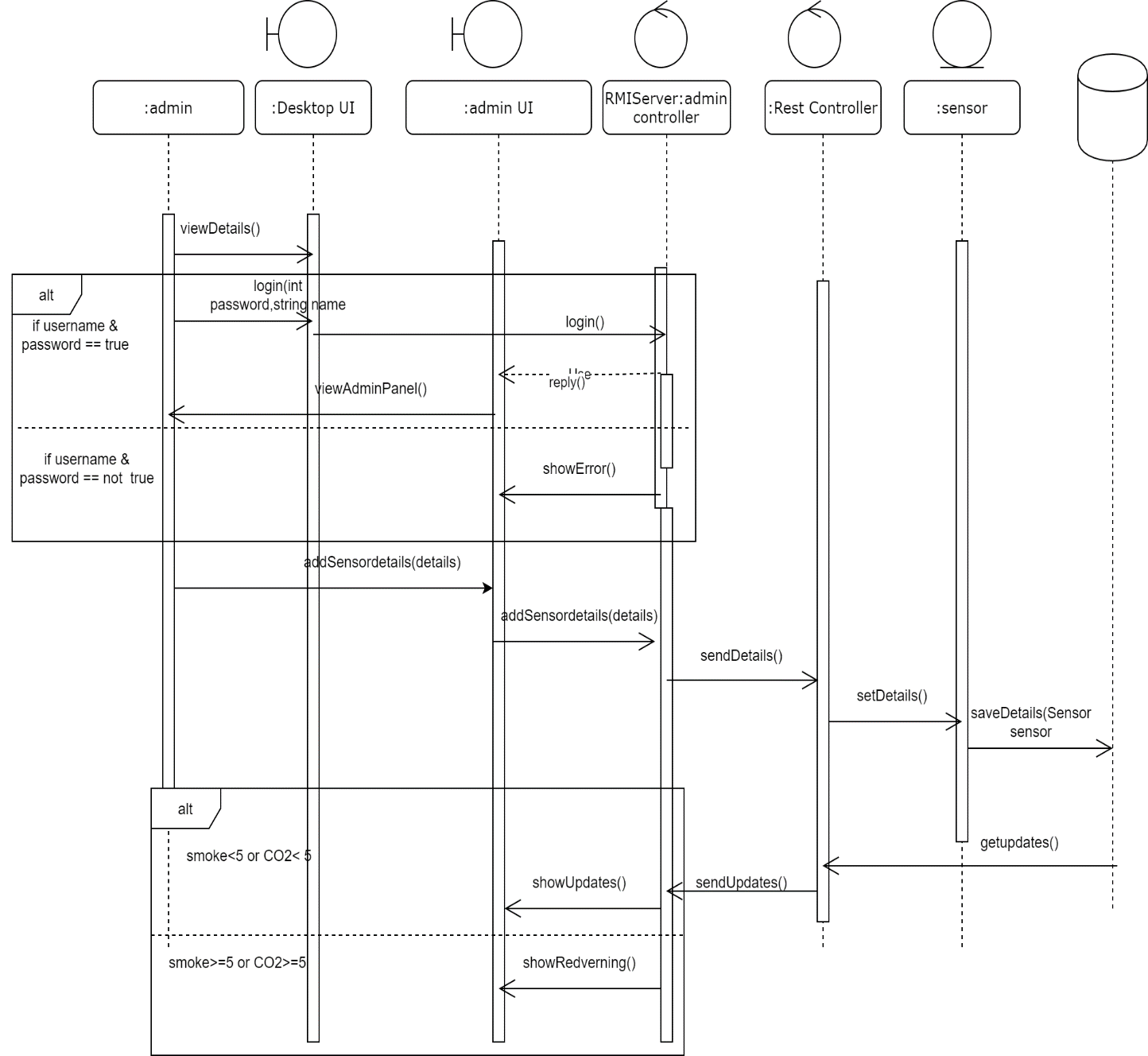
Introduction

This is a group project of developing a fire alarm monitoring system. This system has a web client application that shows the users states of all fire alarm sensors. Users can see the floor number, room number, CO2 level, Smoke level and active or inactive state of all alarms. To develop this web client application, we used ReactJS and WebStorm application. We Developed an RMI Server and an RMI desktop client as the desktop client application. In this application users can see the same information from a desktop client. And moreover, desktop application has an admin login where admin can add new fire alarm sensors. Admin can register a fire alarm sensor by giving room number and floor number and admin can edit sensor details as well. To develop this desktop client, we used netBeans application and Java swing. We used mySLQ database as our datastore to store sensor information and user information. Web client application and the mySQL database are connected by using REST API which developed in InteliJ IDEA application as a Spring boot project. We implemented a simple client application to simulate the behavior of a Fire sensor. These applications send the fire alarm status to the REST API every 10 seconds.

High Level Diagram

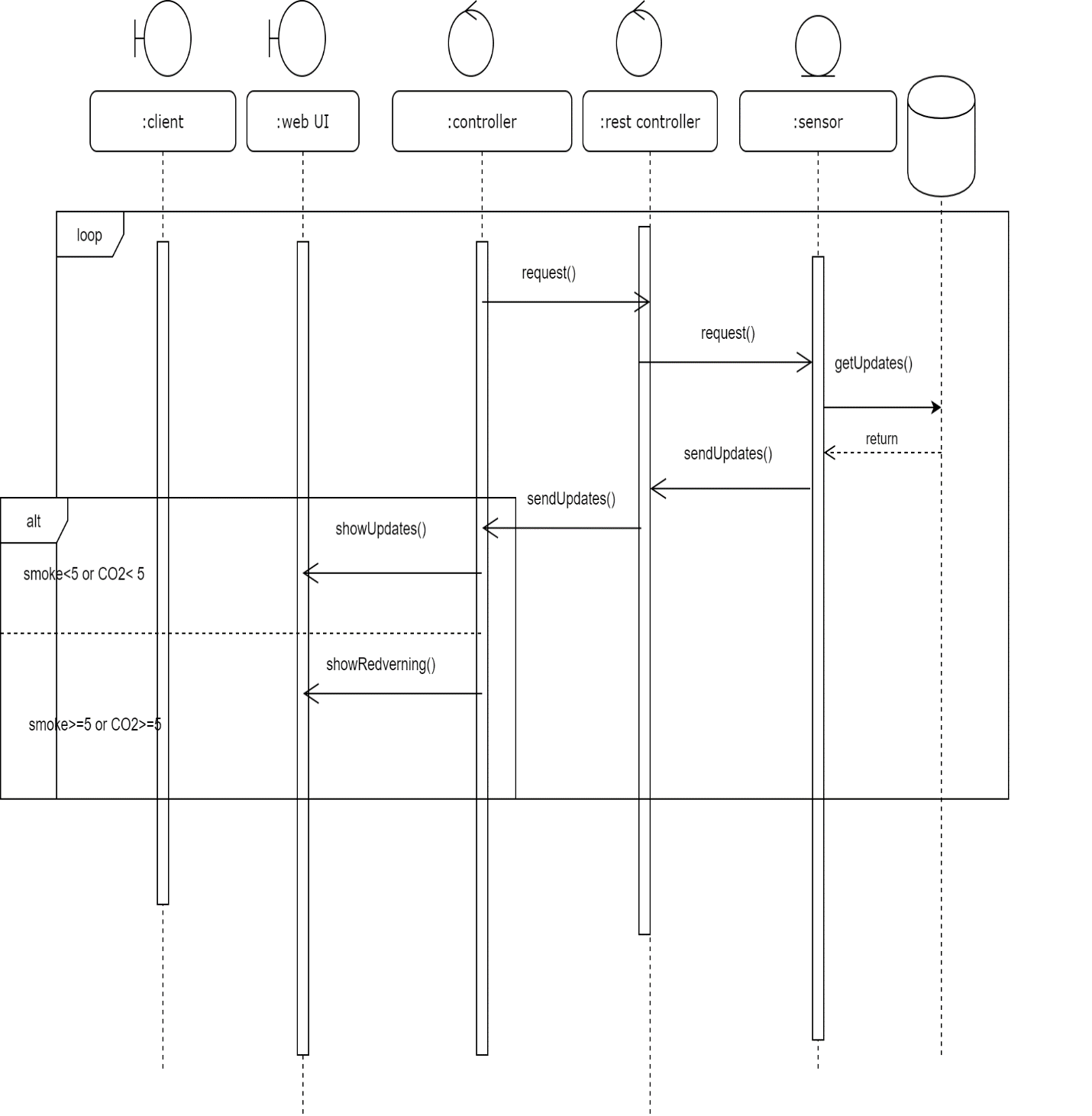
Workflow and Service Interfaces

**Desktop Client**

****Desktop client application is developed as an RMI client. It is connected to the REST API through the RMI server and store data in MySQL database. The desktop client can view details about every sensor that are connected to the system. It shows the sensor id, smoke level, CO2 level and activeness of the sensor. Admin of the system can login to the system using correct username and password and they are validated by RMI server. It will show the admin panel. The admin can add sensor details like sensor id, floor number and room number using admin panel. The added details will be sent to REST API by the RMI server. REST API is responsible to save data in the MySQL database. The registered sensor updates are taking from the database and REST API send them to RMI server. It will update desktop client frequently. If the smoke level or CO 2 level is increasing dangerously desktop client will be notified by showing red alert.

**Web Client**

The web client is implemented in React Js and it shows the sensor updates which are currently connected to the system. The web controller uses the callback functions to REST API receive sensor updates frequently and quickly. REST API will send updates of sensors to web client by retrieving from database. The web client will show red alert by web controller when smoke level or CO2 level is high.

****

**Sensor Application**

The sensor application is implemented as an RMI client and the RMI server handles the updates. Sensor is implemented to change rapidly and detect the CO2 level and smoke level. Those details will send to RMI server. The sensor server sends details to REST API and they will save in the database. This asynchronous will happened all the time in the state of active sensor.

**A screenshot of a cell phone

Description automatically generated**

**REST API**

**A screenshot of a cell phone

Description automatically generated**The REST API is implemented as a spring boot application. This will handle all the connections of the system. Other than that API is responsible for sending e-mail and SMS for relevant clients when sensor’s smoke and CO2 level is high. It checks the updates that are receive from the sensor and analyze what client should get notified. If the smoke or CO2 goes to dangerous level API will send the e-mail and SMS to clients.

# Appendix

Reactjs - App.js

**import *React***, {Component} **from 'react'**;  
**import './App.css'**;  
**import *axios* from "axios"**;  
**import** Table **from 'react-bootstrap/Table'**;  
  
  
**class** App **extends** Component {  
  
   
  
 constructor(props) {  
 **super**(props);  
 **this**.**state** = {  
 **values**: [],  
 **isLoaded**: **false** }  
 }  
 **interval** = **null**;  
  
 componentDidMount() {  
 **this**.**interval** = *setInterval*(**this**.*getData*, 40000);  
 **this**.*getData*();  
 }  
  
 componentWillUnmount() {  
 *clearInterval*(**this**.**interval**);  
 }  
  
 *getData* = () => {  
 ***axios***.get(**'http://localhost:8080/sensor/getsensor'**).then((response) => {  
 **this**.setState({  
 **values**: response.**data**,  
 })  
 });  
 }  
  
 render() {  
  
 **let** values = **this**.**state**.**values**.map((x) => {  
  
 **if**(x.coLevel>5 || x.smokeLevel>6){  
 **return** (  
  
 <**tr bgcolor=**{**"#c70404"**} **key=**{x.sensorId} **style=**{{ **color**: **'white'** }}>  
  
 <**td**>{x.sensorId}</**td**>  
 <**td**>{x.roomNo}</**td**>  
 <**td**>{x.floorNo}</**td**>  
 <**td**>{x.coLevel}</**td**>  
 <**td**>{x.smokeLevel}</**td**>  
 <**td**>Active</**td**>  
  
 </**tr**>  
 );  
 }  
 **else  
 return**(  
 <**tr key=**{x.sensorId}>  
  
 <**td**>{x.sensorId}</**td**>  
 <**td**>{x.roomNo}</**td**>  
 <**td**>{x.floorNo}</**td**>  
 <**td**>{x.coLevel}</**td**>  
 <**td**>{x.smokeLevel}</**td**>  
 <**td**>Active</**td**>  
 </**tr**>  
 )  
 });  
 **return** (  
 <**div className="App container"**>  
 <**div className='mt-5'**>  
 <**div className='mt-5'**>  
 <**h2**>Fire Alarm Table</**h2**>  
 </**div**>  
 <**Table striped bordered** >  
 <**thead**>  
 <**tr bgcolor=**{**"#373a3d"**} **style=**{{ **color**: **'white'** }}>  
 <**th**>Sensor Id</**th**>  
 <**th**>Room No</**th**>  
 <**th**>Floor No</**th**>  
 <**th**>CO2 Level</**th**>  
 <**th**>Smoke Level</**th**>  
 <**th**>State</**th**>  
 </**tr**>  
  
 </**thead**>  
  
 <**tbody**>  
 {values}  
 </**tbody**>  
  
 </**Table**>  
 </**div**>  
 </**div**>  
 );  
 }  
}  
  
**export default** App;

rest API – Model – Sensor

@Entity  
**public class** Sensor {  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 **int sensorId**;  
 String **roomNo**;  
 String **floorNo**;  
 **double coLevel**;  
 **double smokeLevel**;  
  
 **public int** getSensorId() {  
 **return sensorId**;  
 }  
  
 **public void** setSensorId(**int** sensorId) {  
 **this**.**sensorId** = sensorId;  
 }  
  
 **public** String getRoomNo() {  
 **return roomNo**;  
 }  
  
 **public void** setRoomNo(String roomNo) {  
 **this**.**roomNo** = roomNo;  
 }  
  
 **public** String getFloorNo() {  
 **return floorNo**;  
 }  
  
 **public void** setFloorNo(String floorNo) {  
 **this**.**floorNo** = floorNo;  
 }  
  
 **public** Sensor() {  
 }  
  
 **public double** getCoLevel() {  
 **return coLevel**;  
 }  
  
 **public void** setCoLevel(**double** coLevel) {  
 **this**.**coLevel** = coLevel;  
 }  
  
 **public double** getSmokeLevel() {  
 **return smokeLevel**;  
 }  
  
 **public void** setSmokeLevel(**double** smokeLevel) {  
 **this**.**smokeLevel** = smokeLevel;  
 }  
}

Rest API- Model-User

@Entity  
**public class** User {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 **int Id**;  
 String **name**;  
 String **email**;  
 **int phoneno**;  
  
 **public** User() {  
 }  
  
 **public int** getId() {  
 **return Id**;  
 }  
  
 **public void** setId(**int** id) {  
 **Id** = id;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public** String getEmail() {  
 **return email**;  
 }  
  
 **public void** setEmail(String email) {  
 **this**.**email** = email;  
 }  
  
 **public int** getPhoneno() {  
 **return phoneno**;  
 }  
  
 **public void** setPhoneno(**int** phoneno) {  
 **this**.**phoneno** = phoneno;  
 }  
}

Rest API- Controller – SensorController

@CrossOrigin  
@RestController  
@RequestMapping(value = **"/sensor"**)  
**public class** SensorController {  
  
 @Autowired  
 **private** SensorService **sensorService**;  
  
 @PostMapping(value = **"/addsensor"**)  
 **public** Sensor addSensor(@RequestBody Sensor sensor){  
 **return sensorService**.addSensor(sensor);  
 }  
  
 @GetMapping(value = **"/getsensor"**)  
 **public** List<Sensor> getallSensor(){  
 **return sensorService**.getallSensor();  
 }  
  
 @PostMapping(value = **"/update"**)  
 **public** Sensor update(@RequestBody Sensor sensor){  
 **return sensorService**.update(sensor);  
 }  
 @GetMapping(value =**"/search/{id}"**)  
 **public** Sensor search(@PathVariable **int** id){  
 **return sensorService**.search(id);  
 }  
  
  
}

Rest API- Controller – UserController

@CrossOrigin  
@RestController  
@RequestMapping(value = **"/user"**)  
**public class** UserController {  
  
 @Autowired  
 UserService **userService**;  
  
 @PostMapping(value = **"/send"**)  
 **public void** send(String id){  
 **userService**.send(id);  
 }  
}

Rest API- Service-SensorService

@Service  
**public class** SensorService{  
  
 @Autowired  
 **private** SensorRepository **sensorRepository**;  
  
 **public** Sensor addSensor(Sensor sensor) {  
 **return sensorRepository**.save(sensor);  
 }  
  
 **public** List<Sensor> getallSensor() {  
 **return sensorRepository**.findAll();  
 }  
  
 **public** Sensor update(Sensor sensor) {  
 **return sensorRepository**.save(sensor);  
 }  
  
 **public** Sensor search(**int** id) {  
 **return sensorRepository**.search( id);  
 }  
  
  
}

Rest API -Service-UserService

@Service  
**public class** UserService {  
 **private** JavaMailSender **javaMailSender**;  
  
 @Autowired  
 **public** UserService(JavaMailSender javaMailSender){  
 **this**.**javaMailSender** = javaMailSender;  
 }  
  
 **public void** send(String id)**throws** MailException {  
 SimpleMailMessage mail = **new** SimpleMailMessage();  
 mail.setTo(**"sithijabimsara@gmail.com"**);  
 mail.setFrom(**"hansika.m99@gmail.com"**);  
 mail.setSubject(**"Fire Alarm Alert!!!"**);  
 mail.setText(**"Your room is on fire"**+ id);  
 **javaMailSender**.send(mail);  
  
 }  
}

Rest API – Repository – SensorRepository

**public interface** SensorRepository **extends** JpaRepository<Sensor,String> {  
  
  
 @Query(value = **"from Sensor where sensorId=?1"**)  
 Sensor search(**int** id);  
  
}

rest API – application.properties

**spring.jpa.hibernate.ddl-auto**=**update  
spring.datasource.url**=**jdbc:mysql://localhost:3306/fireAlarm?createDatabaseIfNotExist=true  
spring.datasource.username**=**root  
spring.datasource.password**=**1996  
spring.jpa.show-sql**=**true  
spring.jpa.database-platform**=**org.hibernate.dialect.MySQL57Dialect  
spring.mail.host**= **smtp.gmail.com  
spring.mail.username**=**dsassignsliit@gmail.com  
spring.mail.password**= **DSassign2020  
spring.mail.port**= **587  
spring.mail.properties.mail.smtp.starttls.enable** = **true  
spring.mail.properties.mail.smtp.auth** = **true  
spring.mail.properties.mail.smtp.socketFactory.port** = **465  
spring.mail.properties.mail.smtp.socketFactory.class** = **javax.net.ssl.SSLSocketFactory  
spring.mail.properties.mail.smtp.socketFactory.fallback** = **false**

DesktopClient- Firealaremclient – LoginServer

public class LoginServer {

public static void main(String[] args){

try {

Registry reg = LocateRegistry.createRegistry(1099);

LoginImplement lp = new LoginImplement();

reg.rebind("login", lp);

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

} catch (RemoteException ex) {

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

}

}

}

DesctopClient- Firealaremclient – LoginInterface

public interface LoginInterface extends Remote {

public boolean getlogin(String user,String pass)throws RemoteException;

}

DesctopClient- Firealaremclient – LoginImplement

public class LoginImplement extends UnicastRemoteObject implements LoginInterface{

public LoginImplement()throws RemoteException{

}

@Override

public boolean getlogin(String user, String pass) throws RemoteException {

boolean match = false;

try{

if(user.equals("admin")&& pass.equals("admin")){

return match = true;

}else{

return match = false;

}

}catch(Exception e){

e.printStackTrace();

}

return match;

}

}

DesctopClient- Firealaremclient – Login

public class Login extends javax.swing.JFrame {

/\*\*

\* Creates new form Login

\*/

public Login() {

initComponents();

DisplaySensor();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

txtUsername = new javax.swing.JTextField();

jLabel4 = new javax.swing.JLabel();

txtPassword = new javax.swing.JPasswordField();

jButton1 = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

table = new javax.swing.JTable();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

jLabel1.setText("Fire Alarm List");

jLabel2.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N

jLabel2.setText("Login");

jLabel3.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

jLabel3.setText("UserName");

jLabel4.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

jLabel4.setText("Password");

jButton1.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

jButton1.setText("Login");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

table.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] {

"Title 1", "Title 2", "Title 3", "Title 4", "Title 5"

}

));

jScrollPane1.setViewportView(table);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(30, 30, 30)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 481, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel1)))

.addGroup(layout.createSequentialGroup()

.addGap(43, 43, 43)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel2)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(txtPassword, javax.swing.GroupLayout.PREFERRED\_SIZE, 101, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel3)

.addGap(18, 18, 18)

.addComponent(txtUsername, javax.swing.GroupLayout.PREFERRED\_SIZE, 101, javax.swing.GroupLayout.PREFERRED\_SIZE)))))

.addGroup(layout.createSequentialGroup()

.addGap(255, 255, 255)

.addComponent(jButton1)))

.addContainerGap(614, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(25, 25, 25)

.addComponent(jLabel1)

.addGap(30, 30, 30)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 96, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(29, 29, 29)

.addComponent(jLabel2)

.addGap(30, 30, 30)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jLabel3)

.addComponent(txtUsername, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel4)

.addComponent(txtPassword, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 35, Short.MAX\_VALUE)

.addComponent(jButton1)

.addGap(118, 118, 118))

);

pack();

}// </editor-fold>

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

// TODO add your handling code here:

boolean f = false;

try {

Registry reg = LocateRegistry.getRegistry("127.0.0.1",1099);

LoginInterface i = (LoginInterface)reg.lookup("login");

f = i.getlogin(txtUsername.getText(), txtPassword.getText());

if(f == true)

{

AddItem add = new AddItem();

add.setVisible(true);

}else{

JOptionPane.showMessageDialog(null,"Invalid Username Or Password");

}

} catch (Exception e) {

}

}

/\*\*

\* @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(Login.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Login().setVisible(true);

}

});

}

private void DisplaySensor(){

try {

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

HttpURLConnection 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);

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.setSensorId(jsonObj.getInt("sensorId"));

ob.setFloorNo(jsonObj.getString("floorNo"));

ob.setRoomNo(jsonObj.getString("roomNo"));

ob.setSmokeLevel(jsonObj.getDouble("smokeLevel"));

ob.setCoLevel(jsonObj.getDouble("coLevel"));

DefaultTableModel dmodel = new DefaultTableModel();

dmodel.addColumn("Sid");

dmodel.addColumn("room");

dmodel.addColumn("floor");

dmodel.addColumn("co2");

dmodel.addColumn("smoke");

dmodel.addColumn("state");

if(ob.coLevel >= 5.00 || ob.smokeLevel >= 5.00){

this.table.setBackground(Color.red);

dmodel.addRow(new Object[]{ob.getSensorId(),ob.getFloorNo(),ob.getRoomNo(),ob.getCoLevel(),ob.getSmokeLevel(),"Active"});

this.table.setModel(dmodel);

line = br.readLine();

}

else

dmodel.addRow(new Object[]{ob.getSensorId(),ob.getFloorNo(),ob.getRoomNo(),ob.getCoLevel(),ob.getSmokeLevel(),"Active"});

this.table.setModel(dmodel);

line = br.readLine();

}

}

}

} catch (MalformedURLException ex) {

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

} catch (ProtocolException ex) {

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

} catch (IOException ex) {

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

} catch (JSONException ex) {

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

} finally {

}

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable table;

private javax.swing.JPasswordField txtPassword;

private javax.swing.JTextField txtUsername;

// End of variables declaration

}

DesctopClient- Firealaremclient – AddItem

public class AddItem extends javax.swing.JFrame {

/\*\*

\* Creates new form AddItem

\*/

public AddItem() {

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

sensorId = new javax.swing.JTextField();

roomno = new javax.swing.JLabel();

roomNo = new javax.swing.JTextField();

floor = new javax.swing.JLabel();

floorNo = new javax.swing.JTextField();

jButton1 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 24)); // NOI18N

jLabel1.setText("Add Sensors");

jLabel2.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

jLabel2.setText("SensorID");

sensorId.setText(" ");

roomno.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

roomno.setText("Room No");

roomNo.setText(" ");

floor.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

floor.setText("Floor No");

floorNo.setText(" ");

jButton1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

jButton1.setText("Add");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(layout.createSequentialGroup()

.addGap(0, 0, Short.MAX\_VALUE)

.addComponent(jButton1))

.addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

.addGap(48, 48, 48)

.addComponent(jLabel1)

.addGap(0, 117, Short.MAX\_VALUE))

.addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

.addGap(38, 38, 38)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel2)

.addComponent(roomno)

.addComponent(floor))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(sensorId)

.addComponent(roomNo)

.addComponent(floorNo, javax.swing.GroupLayout.DEFAULT\_SIZE, 88, Short.MAX\_VALUE))))

.addGap(177, 177, 177))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(24, 24, 24)

.addComponent(jLabel1)

.addGap(26, 26, 26)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel2)

.addComponent(sensorId, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(roomno)

.addComponent(roomNo, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(floor)

.addComponent(floorNo, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 70, Short.MAX\_VALUE)

.addComponent(jButton1)

.addGap(42, 42, 42))

);

pack();

}// </editor-fold>

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

// TODO add your handling code here:

addSensor(roomNo.getText(),floorNo.getText(),sensorId.getText());

}

/\*\*

\* @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(AddItem.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new AddItem().setVisible(true);

}

});

}

private void addSensor(String room,String floor,String id){

try {

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

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

con.setRequestMethod("POST");

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

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

con.setDoOutput(true);

String passingData = "{\"roomNo\": \""+room+"\",\"floorNo\": \""+floor+"\"}";

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

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

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

}

int code = con.getResponseCode();

System.out.println(code);

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

StringBuilder response = new StringBuilder();

String responseLine = null;

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

response.append(responseLine.trim());

}

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

JOptionPane.showMessageDialog(null,"Successfully Added");

}

// checkStart=true;

} catch (IOException ex) {

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

}

}

// Variables declaration - do not modify

private javax.swing.JLabel floor;

private javax.swing.JTextField floorNo;

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JTextField roomNo;

private javax.swing.JLabel roomno;

private javax.swing.JTextField sensorId;

// End of variables declaration

}

DesctopClient- Firealaremclient – Sensor

@XmlRootElement(name = "sensor")

public class Sensor {

int sensorId;

String roomNo;

String floorNo;

double coLevel;

double smokeLevel;

@XmlElement

public int getSensorId() {

return sensorId;

}

public void setSensorId(int sensorId) {

this.sensorId = sensorId;

}

@XmlElement

public String getRoomNo() {

return roomNo;

}

public void setRoomNo(String roomNo) {

this.roomNo = roomNo;

}

@XmlElement

public String getFloorNo() {

return floorNo;

}

public void setFloorNo(String floorNo) {

this.floorNo = floorNo;

}

@XmlElement

public double getCoLevel() {

return coLevel;

}

public void setCoLevel(double coLevel) {

this.coLevel = coLevel;

}

@XmlElement

public double getSmokeLevel() {

return smokeLevel;

}

public void setSmokeLevel(double smokeLevel) {

this.smokeLevel = smokeLevel;

}

}

Sensor- Alarm – AlarmSensorServer

public class AlarmSensorServer extends UnicastRemoteObject implements TemperatureSensor, Runnable {

private volatile double CO2;

private volatile double Smoke;

private ArrayList<AlarmListener> list = new ArrayList<AlarmListener>();

//Set CO2 and Smoke in the constructor

public AlarmSensorServer() throws java.rmi.RemoteException {

CO2 = 4.0;

Smoke = 5.0;

}

//This method will return C02 level

public double getCo2() throws java.rmi.RemoteException {

return CO2;

}

//This method will return Smoke level

public double getSmoke() throws java.rmi.RemoteException {

return Smoke;

}

//This method is for Registering listners

public void addAlarmListener(AlarmListener listener)

throws java.rmi.RemoteException {

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

list.add(listener);

}

//This method is for Removing Registered listners

public void removeAlarmListener(AlarmListener listener)

throws java.rmi.RemoteException {

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

list.remove(listener);

}

//This method is use to Set the thread

public void run() {

Random r = new Random();

for (;;) {

try {

// Sleep for a random amount of time

int duration = r.nextInt() % 10000 + 200;

// Check to see if negative, if so, reverse

if (duration < 0) {

duration = duration \* -1;

Thread.sleep(duration);

}

} catch (InterruptedException ie) {

}

// Get a number, to see if CO2 and Smoke goes up or down

int num = r.nextInt();

if (num < 0) {

CO2 += 0.5;

Smoke += 0.5;

} else {

CO2 -= 0.5;

Smoke -= 0.5;

}

// Notify registered listeners

try {

notifyListeners();

updateSensor(Smoke, CO2);

if(Smoke > 5 || CO2 > 5){

// sendEmail("2");

}

} catch (RemoteException e) {

e.printStackTrace();

}

}

}

private void notifyListeners() throws RemoteException {

// Notifing

for(AlarmListener e : list){

e.Co2Changed(CO2);

e.SmokeChanged(Smoke);

}

}

public static void main(String[] args) {

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

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

try {

AlarmSensorServer sensor = new AlarmSensorServer();

Registry reg=LocateRegistry.createRegistry(9998);

reg.rebind("hi server", sensor);

Thread thread = new Thread(sensor);

thread.start();

} catch (RemoteException re) {

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

} catch (Exception e) {

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

}

}

private void updateSensor(double smoke,double co2){

try {

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

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

con.setRequestMethod("POST");

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

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

con.setDoOutput(true);

Sensor s = new Sensor();

s.setSensorId(3);

s.setFloorNo(10);

s.setRoomNo(452);

String passingData = "{\"sensorId\": \""+s.getSensorId()+"\",\"floorNo\": \""+s.getFloorNo()+"\",\"roomNo\": \""+s.getRoomNo()+"\" ,\"smokeLevel\": \""+smoke+"\",\"coLevel\": \""+co2+"\" }";

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

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

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

}

int code = con.getResponseCode();

System.out.println(code);

try(BufferedReader br = new BufferedReader(new InputStreamReader(con.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(AlarmSensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void sendEmail(String id){

try {

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

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 = "{\"id\": \""+id+"\"}";

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(AlarmSensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

Sensor- Alarm – AlarmListner

interface AlarmListener extends java.rmi.Remote

{

//Remort methods

public void Co2Changed(double CO2) throws java.rmi.RemoteException;

public void SmokeChanged(double Smoke) throws java.rmi.RemoteException;

}

Sensor- Alarm – AlarmMonitor

public class AlarmMonitor extends UnicastRemoteObject implements

AlarmListener, Runnable {

private int count = 0;

private int count1 =0;

//This method isfor monitoring CO2 and Smoke

public AlarmMonitor() throws RemoteException {

}

public static void main(String[] args) {

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

try {

Registry reg=LocateRegistry.getRegistry(9998);

Remote remoteService = reg.lookup("hi server");

TemperatureSensor sensor = (TemperatureSensor) remoteService;

double reading1 = sensor.getCo2();

System.out.println("Original CO2 : " + reading1);

double reading2 = sensor.getSmoke();

System.out.println("Original Smoke : " + reading2);

AlarmMonitor monitor = new AlarmMonitor();

//Adding the method call to register the listener in the server object

sensor.addAlarmListener(monitor);

monitor.run();

} catch (RemoteException re) {

} catch (NotBoundException nbe) {

}

}

//get a system output when the CO2 changes

public void Co2Changed(double Co2)

throws java.rmi.RemoteException {

System.out.println("\nCo2 change event : " + Co2);

count = 0;

}

public void SmokeChanged(double Smoke)

throws java.rmi.RemoteException {

System.out.println("\nSmoke change event : " + Smoke);

count1 = 0;

}

public void run() {

for (;;) {

count++;

count1++;

// note that this might only work on windows console

//System.out.print("\r" + count);

//System.out.print("\r" + count1);

try {

Thread.sleep(100);

} catch (InterruptedException ie) {

}

}

}

}

Sensor- Alarm – TemperatureSensor

public interface TemperatureSensor extends java.rmi.Remote

{

public double getCo2() throws

java.rmi.RemoteException;

public double getSmoke() throws

java.rmi.RemoteException;

public void addAlarmListener

(AlarmListener listener )

throws java.rmi.RemoteException;

public void removeAlarmListener

(AlarmListener listener )

throws java.rmi.RemoteException;

}

Sensor- Alarm – Sensor

@XmlRootElement(name = "sensor")

public class Sensor {

int sensorId;

int roomNo;

int floorNo;

double coLevel;

double smokeLevel;

@XmlElement

public int getSensorId() {

return sensorId;

}

public void setSensorId(int sensorId) {

this.sensorId = sensorId;

}

@XmlElement

public int getRoomNo() {

return roomNo;

}

public void setRoomNo(int roomNo) {

this.roomNo = roomNo;

}

@XmlElement

public int getFloorNo() {

return floorNo;

}

public void setFloorNo(int floorNo) {

this.floorNo = floorNo;

}

@XmlElement

public double getCoLevel() {

return coLevel;

}

public void setCoLevel(double coLevel) {

this.coLevel = coLevel;

}

@XmlElement

public double getSmokeLevel() {

return smokeLevel;

}

public void setSmokeLevel(double smokeLevel) {

this.smokeLevel = smokeLevel;

}

}