

**Sri Lanka Institute of Information Technology**

Fire Alarm Monitoring System (Rest API)

**Report**

**Distribution System Assignment 02**

**2020**

**Submitted by:**

1. IT18144772: NIROSHAN K.
2. IT18001976: VARNIAH K.
3. IT18068610: PIRATHIKARAN V.
4. IT18152074: SANGEETH RAJ A.

TABLE OF CONTENT

* Introduction ………………………………………… 3
* Workflow Diagram ………………………………………… 4
* System workflow scenario execution ………………………………………… 5-6
* Source code and binaries ………………………………………. 7-41

PROJECT DESCRIPTION

INTRODUCTION

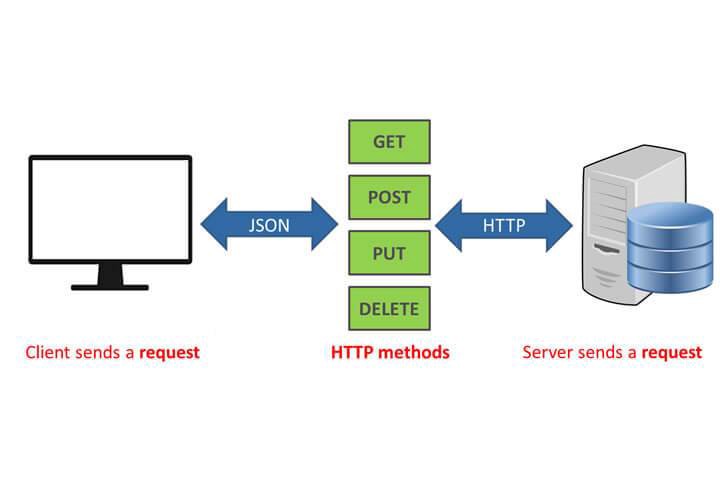
This project is regarding fire alarm monitoring system is implemented using REST API, web client, and RMI server & client. Below mentioned technologies which are used to implement the system as follows;

* REST API – ExpressJS
* Web client - react, bootstrap
* RMI server and client – java swing
* MongoDB : Mongoose - is used for store sensor, floor, room and user details.

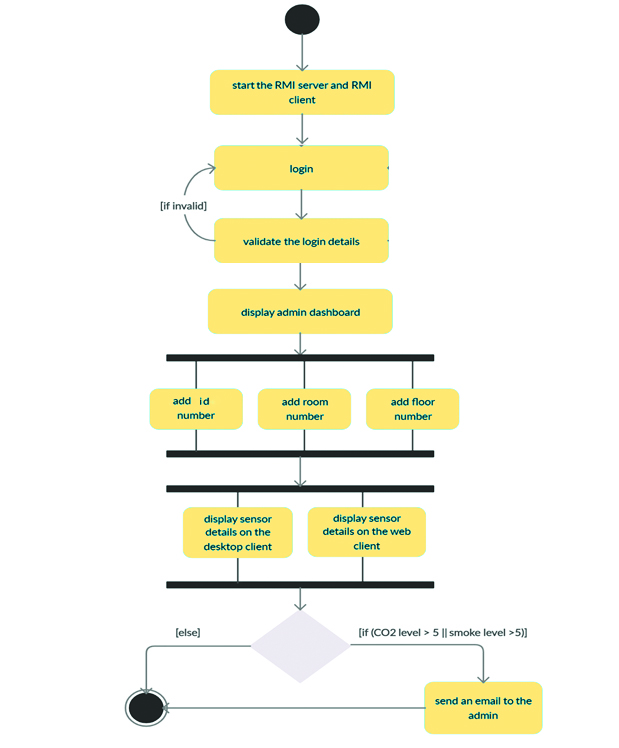
The system administrator will add the sensors to the relevant floor and room, and when the level of co2 and smoke rises, the assigned sensors will be activated.

The fire alarm monitoring system has a web client and a desktop client that can display sensor data such as room number, floor number, co2 level, smoke level and activated / deactivated status for both users.

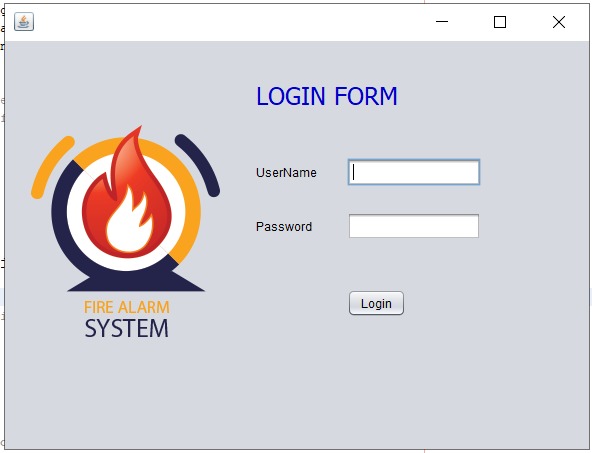
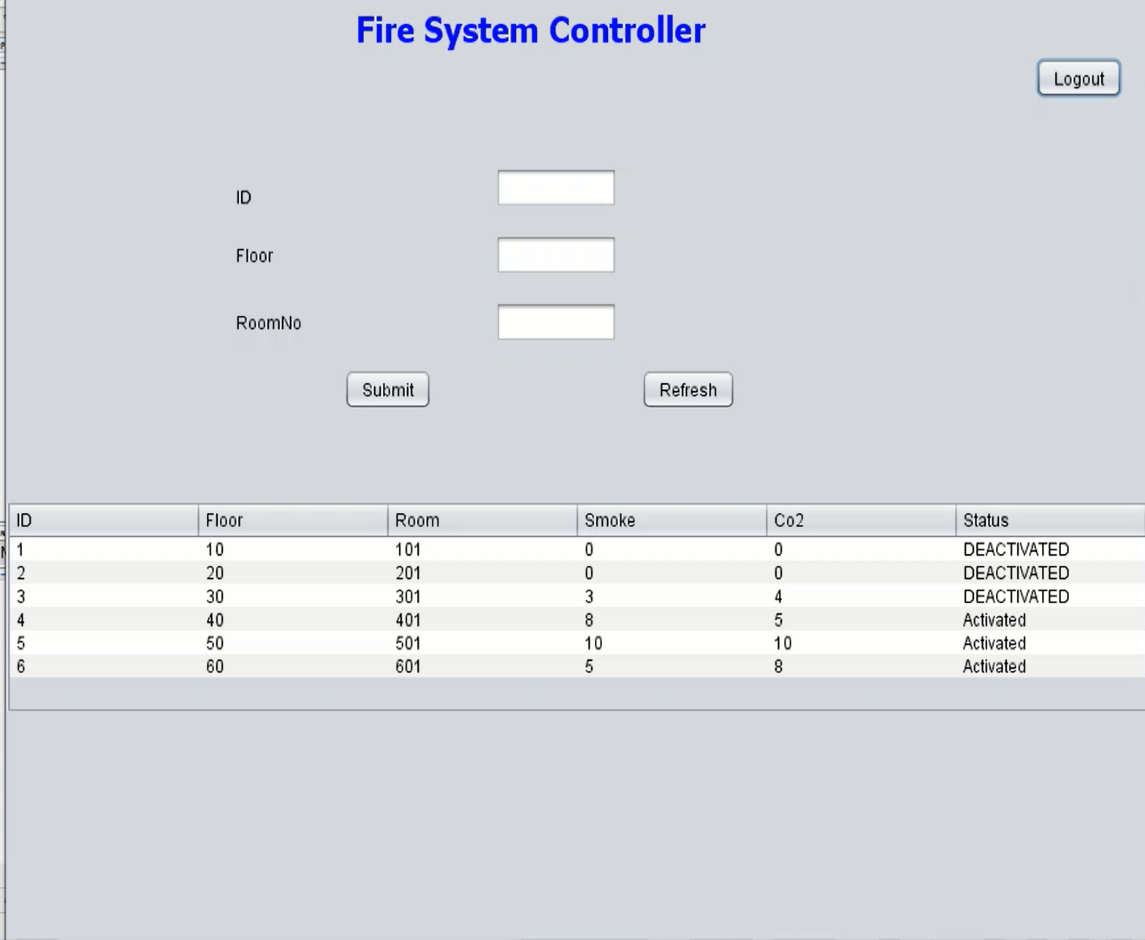
A dummy alarm sensor is implemented to manually control the level of co2 and smoke by using random variable calling in the function. The Admin will receive an email from the system when the sensor is activated when the co2 level or smoke level is above 5.

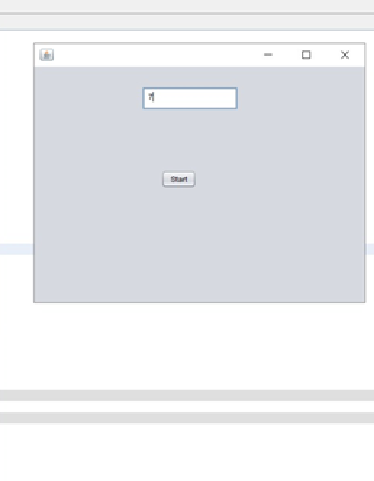
The structure and the process of the project,

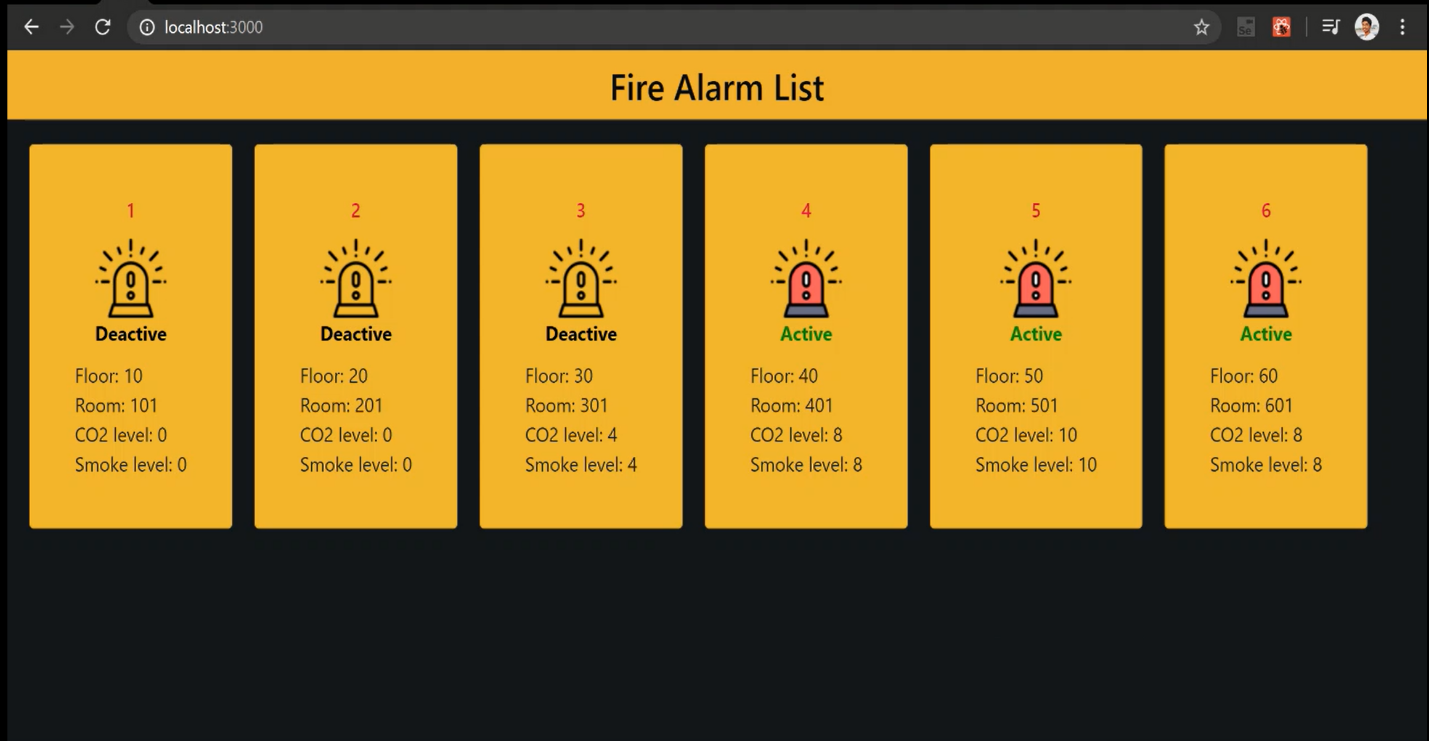
Workflow Diagram



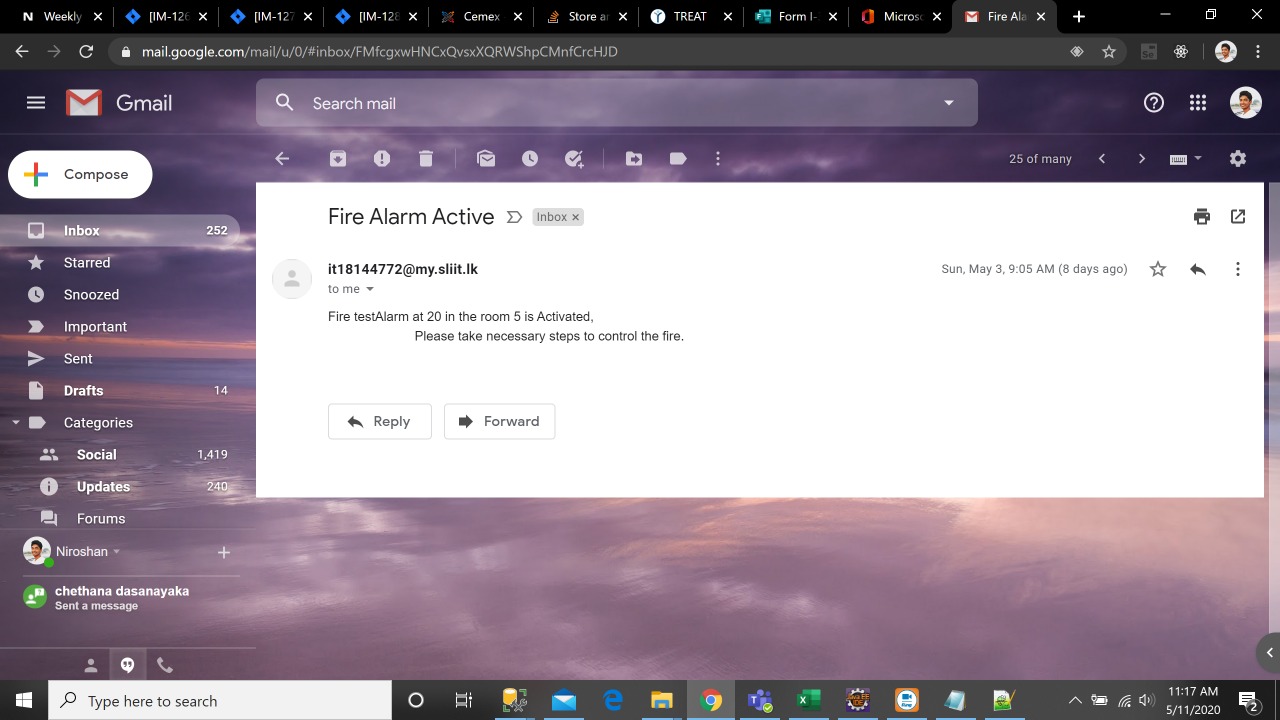
System workflow scenario execution

1. login
2. Admin Dashboard
3. Dummy alarm sensor



1. Client Dashboard 

5. Received Email



Source code and binaries

1. MERN stack

Server (ExpressJS)

* (Server.js)

const express=require('express');

const cors=require('cors');

const mongoose=require('mongoose');

require('dotenv').config();

const app= express();

const port=process.env.PORT || 5001;

app.use(cors());

app.use(express.json());

//connecting to the database using Connection String used as an environment Variable

const uri = process.env.FIREALARM\_URI;

mongoose.connect(uri,{useNewUrlParser:true,useCreateIndex:true});

const connection=mongoose.connection;

connection.once('open',()=>{

console.log("MongoDB connected successfully");

})

//

//Using the Routes for the Project in the Server

const alarmRouter = require('./routes/alarmRoute');

const userRounter = require('./routes/userRoute')

app.use('/alarm',alarmRouter);

app.use('/user',userRounter);

//

app.listen(port,()=>{

console.log(`Server is runnig on port: ${port}`);

})

* (alarmRoute.js)

const router = require('express').Router();

const nodemailer = require('nodemailer');

const transporter = nodemailer.createTransport({

service:'outlook',

auth:{

user:'/Email/',

pass:'/password/'

}

});

let Alarm = require('../models/alarm.module')

router.route('/').get((req,res)=>{

Alarm.find()

.then(alarms=>res.json(alarms))

.catch(err=>err.status(400).json('Error:'+err))

})

router.route('/:id').get((req,res)=>{

Alarm.findById(req.params.id)

.then(alarm=>res.json(alarm))

.catch(err=>res.status(400).json("Err:"+err));

})

router.route('/add').post((req,res)=>{

const name = req.body.name;

const floor = req.body.floor;

const room = req.body.room;

const co2 = req.body.co2;

const smoke = req.body.smoke;

const isAlarmActive = req.body.isAlarmActive;

const newAlarm= new Alarm({

name,

floor,

room,

co2,

smoke,

isAlarmActive

})

newAlarm.save()

.then(()=>res.json("Alarm Added"))

.catch(err=>res.status(400).json('Err:'+err))

})

router.route('/update/:id').put((req,res)=>{

Alarm.findById(req.params.id)

.then(alarm=>{

alarm.name=req.body.name;

alarm.floor=req.body.floor;

alarm.room=req.body.room;

alarm.co2=req.body.co2;

alarm.smoke=req.body.smoke;

alarm.isAlarmActive=req.body.isAlarmActive;

if(alarm.co2 >5 || alarm.smoke > 5){

alarm.isAlarmActive = true;

let mailOptions={

from:'it18144772@my.sliit.lk',

to: 'niroshantrinity@gmail.com',

subject:'Fire Alarm Active',

text:`Fire ${alarm.name} at floor ${alarm.floor} in the room ${alarm.room} is Activated,Please take necessary steps to control the fire.`

}

transporter.sendMail(mailOptions, function(error, info){

if (error) {

console.log(error);

} else {

console.log('Email sent: ' + info.response);

}

});

}

else{

alarm.isAlarmActive = false;

}

alarm.save()

.then(()=>res.json("Alarm updated!"))

.catch(err=>res.status(400).json("Err:"+ err));

})

.catch(err=>res.status(400).json("Err:"+err))

})

module.exports = router;

* (userRoute.js)

const router = require('Express').Router()

const User = require('../models/user.model')

router.route('/').get((req,res)=>{

User.find()

.then(users=>res.json(users))

.catch((err)=>res.status(400).json("Error:"+err))

})

router.route('/add').post((req,res)=>{

const userName = req.body.username;

const password = req.body.password;

const newUser= new User({

username:userName,

password:password

})

newUser.save()

.then(()=>res.json("User Added!"))

.catch(err=>res.status(400).json("Error:"+err))

})

router.route('/:name').get((req,res)=>{

User.find({username:req.params.name})

.then(user=>res.json(user))

.catch((err)=>res.status(400).json("Err:"+err))

})

module.exports = router

Web Client (React)

* (App.js)

import React from 'react';

import './App.css';

import AlarmList from './components/alarm-list.component'

function App() {

return (

<div className="App">

<h1 className="banner">Fire Alarm List</h1>

<AlarmList/>

</div>

);

}

export default App;

* (alarm-list.component.js)

import React, { Component } from 'react'

import AlarmItem from './alarm-item.components'

export default class AlarmList extends Component{

constructor(props){

super(props);

this.state={

alarms:[]

}

}

componentDidMount=()=>{

this.interval=setInterval(()=>fetch("http://localhost:5001/alarm/")

.then(res=> res.json())

.then(data=> {

this.setState({

alarms:data

})

}),10000);

}

renderAlarm=()=>{

return this.state.alarms.map(Currentalarm=>{

console.log(Currentalarm.isAlarmActive);

return <AlarmItem alarmData={Currentalarm} key={Currentalarm.\_id}/>

})

}

render(){

return(

<div className="flex-container">

{this.renderAlarm()}

</div>

);

}

componentWillUnmount(){

clearInterval(this.interval);

}

}

* (alarm-item.components.js)

import React, { Component } from 'react'

import {Card} from 'react-bootstrap'

import '../Card-Style.css'

export default class AlarmItem extends Component{

constructor(props){

super(props)

this.state={

status:""

}

}

alarmStatus=(props)=>{

if(this.props.alarmData.isAlarmActive===true){

this.state.status="Active"

return <img src={ require('../images/true.png') } />

}

else{

this.state.status="Deactive"

return <img src={ require('../images/false.png') } />

}

}

render(){

const {isAlarmActive} = this.props.alarmData;

return(

<Card className="card-item">

<div className="card-title">{this.props.alarmData.name}</div>

<div className="status">{this.alarmStatus()}</div>

<div className="status" style={{color:isAlarmActive ? "rgb(8, 138, 8)":"black"}}>{this.state.status}</div>

<div className="floor-text">Floor: {this.props.alarmData.floor}</div>

<div className="room-text">Room: {this.props.alarmData.room}</div>

<div className="co2-text">CO2 level: {this.props.alarmData.co2}</div>

<div className="smoke-test">Smoke level: {this.props.alarmData.co2}</div>

</Card>

);

}

}

2.RMI

* Sensor class

//co2 sensor

package main;

import java.util.Random;

import java.util.TimerTask;

public class Co2Pack extends TimerTask {

String coid;

boolean fire=false;

public int x;

public void run() {

Random r=new Random();

if(x>=5) {

fire=true;

//System.out.println("Fire Sensor is above level 5");

//System.exit(0);

System.out.println("Final co2 :" +x);

}

else {

x=r.nextInt(11);

}

getNum(x);

//System.out.println("The level is : "+x);

}

public void getNum(int a){

this.x = a;

}

public int setNum() {

return x;

}

}

// smoke sensor

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package main;

import java.util.Random;

import java.util.TimerTask;

/\*\*

\*

\* @author User

\*/

public class SmokeSensor extends TimerTask {

String id;

boolean fire=false;

public int x;

public void run() {

Random r=new Random();

if(x>=5) {

fire=true;

//System.out.println("Fire Sensor is above level 5");

//System.exit(0);

System.out.println("Final value :" +x);

}

else {

x=r.nextInt(11);

}

getNum(x);

//System.out.println("The level is : "+x);

}

public void getNum(int a){

this.x = a;

}

public int setNum() {

return x;

}

}

* Server

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package login.server;

/\*\*

\*

\* @author User

\*/

import java.rmi.\*;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import login.Implement.LoginImplement;

import login.Interface.LoginInterface;

import main.PackSensor;

import main.PackSensorI;

public class LoginServer {

public static void main(String[] args)throws Exception {

try{

Registry reg=LocateRegistry.createRegistry(1050);

LoginInterface obj=new LoginImplement();

PackSensorI obj2=new PackSensor();

reg.rebind("login", obj);

reg.rebind("sensor", obj2);

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

}

catch(Exception e){

e.printStackTrace();

}

}

}

* Login Interface

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package login.Implement;

/\*\*

\*

\* @author User

\*/

import java.rmi.\*;

import java.rmi.server.UnicastRemoteObject;

import java.util.Scanner;

import org.apache.http.HttpResponse;

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

import org.apache.http.impl.client.CloseableHttpClient;

import org.apache.http.impl.client.HttpClients;

public class LoginImplement extends UnicastRemoteObject implements login.Interface.LoginInterface{

public LoginImplement() throws RemoteException{

}

@Override

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

boolean found= false;

try{

// Class.forName("com.mysql.jdbc.Driver");

// java.sql.Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/firealarm", "root" , "");

// String sql= "select \*from logindb where username=? and password=?";

//

//// http('/localhost:5001:add/user/')

//// response.

// PreparedStatement pst=con.prepareStatement(sql);

// pst.setString(1,user);

// pst.setString(2, pass);

// ResultSet rs=pst.executeQuery();

CloseableHttpClient httpclient = HttpClients.createDefault();

//Creating a HttpGet object

HttpGet httpget = new HttpGet("https://jsonplaceholder.typicode.com/posts/1");

//Printing the method used

System.out.println("Request Type: "+httpget.getMethod());

//Executing the Get request

HttpResponse httpresponse = httpclient.execute(httpget);

Scanner sc = new Scanner(httpresponse.getEntity().getContent());

//Printing the status line

// System.out.println(httpresponse.getStatusLine());

while(sc.hasNext()) {

System.out.println(sc.nextLine());

// if(rs.next()){

// return found=true;

//

// }

// else{

// return found=false;

// }

// con.close();

//

// if(user.equals("admin") && pass.equals("123")){

// return found=true;

// }

// else{

// return found=false;

// }

}

}

catch(Exception e){

e.printStackTrace();

}

return found;

}

}

* Login Implemention

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package login.Interface;

/\*\*

\*

\* @author User

\*/

import java.rmi.\*;

public interface LoginInterface extends Remote{

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

}

* Login class

package main;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import javax.swing.JOptionPane;

import login.Interface.LoginInterface;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author User

\*/

public class Login extends javax.swing.JFrame {

/\*\*

\* Creates new form Login

\*/

public Login() {

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() {

jPanel1 = new javax.swing.JPanel();

jLabel4 = new javax.swing.JLabel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

txtusername = new javax.swing.JTextField();

loginbtn = new javax.swing.JButton();

txtpassword = new javax.swing.JPasswordField();

jLabel3 = new javax.swing.JLabel();

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

jLabel4.setFont(new java.awt.Font("Tahoma", 0, 24)); // NOI18N

jLabel4.setForeground(new java.awt.Color(0, 0, 204));

jLabel4.setText("LOGIN FORM");

jLabel1.setText("UserName");

jLabel2.setText("Password");

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

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

txtusernameActionPerformed(evt);

}

});

loginbtn.setText("Login");

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

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

loginbtnActionPerformed(evt);

}

});

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

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

txtpasswordActionPerformed(evt);

}

});

jLabel3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/main/Fire-Alarm-Systems.png"))); // NOI18N

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 233, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED\_SIZE, 167, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(loginbtn)

.addComponent(txtpassword, javax.swing.GroupLayout.PREFERRED\_SIZE, 134, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel1)

.addComponent(jLabel2))

.addGap(30, 30, 30)

.addComponent(txtusername, javax.swing.GroupLayout.PREFERRED\_SIZE, 134, javax.swing.GroupLayout.PREFERRED\_SIZE))))

.addContainerGap(108, Short.MAX\_VALUE))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(30, 30, 30)

.addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED\_SIZE, 48, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(39, 39, 39)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1)

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

.addGap(26, 26, 26)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel2)

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

.addGap(49, 49, 49)

.addComponent(loginbtn))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(73, 73, 73)

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 241, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addContainerGap(94, Short.MAX\_VALUE))

);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

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

boolean f=false;

try{

Registry reg=LocateRegistry.getRegistry("127.0.0.1",1050);

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

f=i.getLogin(txtusername.getText(), txtpassword.getText());

if(f == true){

JOptionPane.showMessageDialog(null, "Login successfully");

new AdminPanel().setVisible(true);

this.setVisible(false);

}

else{

JOptionPane.showMessageDialog(null, "user and password Do not matched");

txtusername.setText("");

txtpassword.setText("");

}

}catch(Exception e){

e.printStackTrace();

}

}

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

// TODO add your handling code here:

}

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

// TODO add your handling code here:

}

/\*\*

\* @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);

}

});

}

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JPanel jPanel1;

private javax.swing.JButton loginbtn;

private javax.swing.JPasswordField txtpassword;

private javax.swing.JTextField txtusername;

// End of variables declaration

}

* Database class

package main;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author User

\*/

class Database {

private int sno, smoke, co2;

private String floor, room,status;

public Database(int sno, String floor, String room,int smoke, int co2,String status) {

this.sno = sno;

this.smoke = smoke;

this.co2 = co2;

this.floor = floor;

this.room = room;

this.status = status;

}

public int getSno() {

return sno;

}

public int getSmoke() {

return smoke;

}

public int getCo2() {

return co2;

}

public String getFloor() {

return floor;

}

public String getRoom() {

return room;

}

public String getStatus(){

return status;

}

}

* Sensor Controller

package main;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author User

\*/

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.util.Timer;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.http.HttpResponse;

import org.apache.http.client.methods.HttpPut;

import org.apache.http.entity.StringEntity;

import org.apache.http.impl.client.CloseableHttpClient;

import org.apache.http.impl.client.HttpClients;

public class PackSensor extends UnicastRemoteObject implements PackSensorI{

public PackSensor() throws RemoteException{

super();

}

@Override

public int startSensor(String id) throws RemoteException {

new Thread(() -> {

Timer t1 = new Timer();

SmokeSensor s1 = new SmokeSensor();

Co2Pack co1=new Co2Pack();

t1.schedule(s1, 0, 15000);

t1.schedule(co1, 0, 15000);

for(; ;) {

System.out.println("the sensor smoke value = "+s1.setNum());

System.out.println("the co2 value = " +co1.setNum());

try {

Thread.sleep(5000);

} catch (InterruptedException ex) {

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

}

int a =s1.setNum();

int b=co1.setNum();

String c = "Deactivated";

if( a>=5 || b>=5){

c = "Activated";

};

int ID=Integer.parseInt(id);

try{

// Class.forName("com.mysql.jdbc.Driver");

// java.sql.Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/firealarm", "root" , "");

// String query = "update newsensordata set smoke = ? , co2 = ?, status = ? where sdataid = ?";

// PreparedStatement preparedStmt = con.prepareStatement(query);

// preparedStmt.setInt(1, a);

// preparedStmt.setInt(2, b);

// preparedStmt.setString(3, c);

// preparedStmt.setInt(4, ID);

//

//

// // execute the java preparedstatement

// preparedStmt.executeUpdate();

//

String putEndpoint = "http://dummy.restapiexample.com/api/v1/update/4710";

CloseableHttpClient httpclient = HttpClients.createDefault();

HttpPut httpPut = new HttpPut(putEndpoint);

httpPut.setHeader("Accept", "application/json");

httpPut.setHeader("Content-type", "application/json");

String inputJson = "{\n" +

" \"name\": \"put\_test\_employee\",\n" +

" \"salary\": \"1123\",\n" +

" \"age\": \"23\"\n" +

"}";

StringEntity stringEntity = new StringEntity(inputJson);

httpPut.setEntity(stringEntity);

System.out.println("Executing request " + httpPut.getRequestLine());

HttpResponse response = httpclient.execute(httpPut);

BufferedReader br = new BufferedReader(new InputStreamReader((response.getEntity().getContent())));

//Throw runtime exception if status code isn't 200

if (response.getStatusLine().getStatusCode() != 200) {

throw new RuntimeException("Failed : HTTP error code : " + response.getStatusLine().getStatusCode());

}

//Create the StringBuffer object and store the response into it.

// StringBuffer result = new StringBuffer();

// String line = "";

// while ((line = br.readLine()) != null) {

// System.out.println("Response : \n" + result.append(line));

// }

}

catch(Exception e){

e.printStackTrace();

}

System.out.println(a);

// if(a>= 5){

// return a;

// }

} }).start();

return 0;

}

}

* Sensor implentation

package main;

import java.rmi.Remote;

import java.rmi.RemoteException;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author User

\*/

public interface PackSensorI extends Remote {

public int startSensor(String id)throws RemoteException ;

}

* Dummy sensor

package main;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.util.logging.Level;

import java.util.logging.Logger;

import login.Interface.LoginInterface;

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\*

\* @author User

\*/

public class Sensorstart extends javax.swing.JFrame {

/\*\*

\* Creates new form NewJFrame

\*/

public Sensorstart() {

initComponents();

}

String id;

String Fl;

String rm;

String status= "Deactivated";

public Sensorstart(String ID,String floor,String room) {

this.id=ID;

this.Fl=floor;

this.rm=room;

initComponents();

sensorid.setText(ID);

}

/\*\*

\* 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() {

jPanel1 = new javax.swing.JPanel();

jScrollPane1 = new javax.swing.JScrollPane();

sensorid = new javax.swing.JTextPane();

jButton1 = new javax.swing.JButton();

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

jScrollPane1.setViewportView(sensorid);

jButton1.setText("Start");

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

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

jButton1ActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(164, 164, 164)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 145, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(193, 193, 193)

.addComponent(jButton1)))

.addContainerGap(193, Short.MAX\_VALUE))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(33, 33, 33)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 36, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(97, 97, 97)

.addComponent(jButton1)

.addContainerGap(184, Short.MAX\_VALUE))

);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

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

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

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

try{

String msg= id;

// psmt.executeUpdate();

// new PackSensor().startSensor(id);

Registry reg=LocateRegistry.getRegistry("127.0.0.1",1050);

PackSensorI i=(PackSensorI)reg.lookup("sensor");

i.startSensor(id);

}

catch(Exception e){

e.printStackTrace();

}

}

/\*\*

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

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

//</editor-fold>

/\* Create and display the form \*/

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

public void run() {

// new Sensorstart().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JPanel jPanel1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTextPane sensorid;

// End of variables declaration

}

* j