A picture containing drawing

Description automatically generated

DS Assignment 2 – Fire Alarm Monitoring System

Module : Distributed Systems

Assignment 2

|  |  |
| --- | --- |
| IT 17167406 Rupasinghe T D | IT 18029864 Dhananjana A M |
| IT 17164122 Satharasinghe S A R L P | IT 18027020 Labeeshan A |

Contents

1. Introduction
2. Workflow
   1. Desktop
   2. Web

3. List of service interface in web client

4. List of service interface in desktop client

5. Appendix

5.1 Desktop

5.2 Web

1. Introduction

This report is about 2nd assignment of DS module which is about implementing a fire alarm monitoring system . This application contains two sub components which are Desktop client and web client those are connected to MYSQL database using RMI server and RSET API by depending on SOA principles.

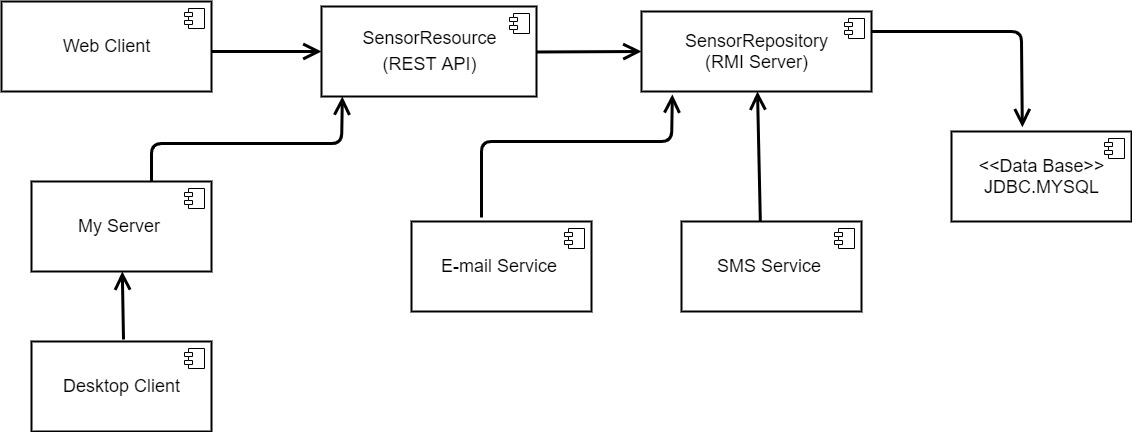
1.Desktop Client Application

* In this application admin can enter the fire alarm sensor details to the MYSQL database. And also admin can view , update and delete the fire alarm sensor details. Meantime it checks the sensor status by automatically refreshing the client application in every 10 seconds.

2.Web client Application

* In this application client can view the fire alarm sensor details through the web interface and also when co2 level and smoke level increased to 5 it will be notified as a warning via SMS and E mail to the admin. Meantime it checks the sensor status by automatically refreshing the web application in every 30 seconds.

**High Level Architecture Diagram of Fire Alarm Monitoring system**



**2.Workflow**

**2.Workflow**

**2.1 Desktop client application workflow**

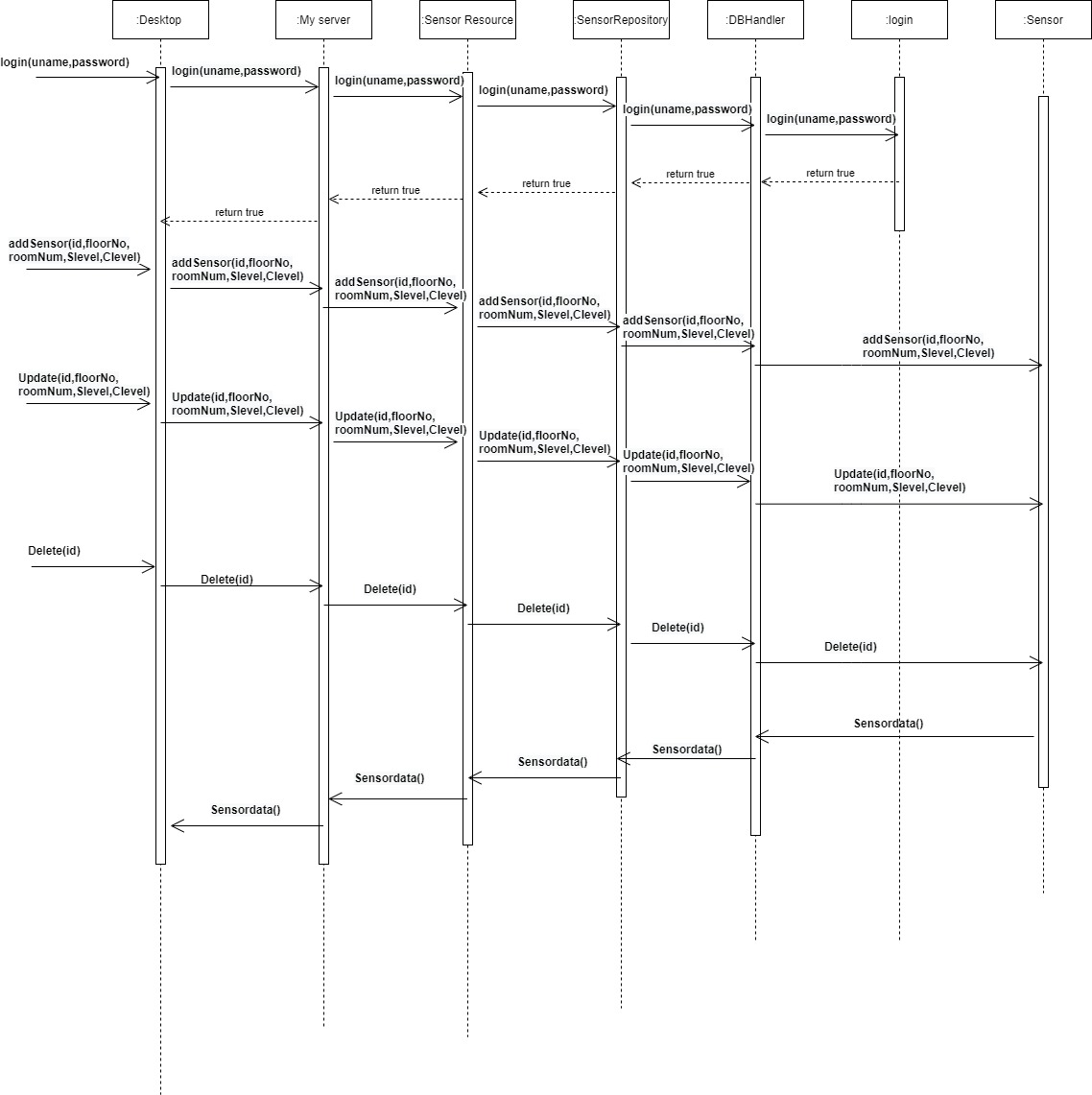
This desktop client application contains two interfaces named AddAlarm and Login. The “login” interface is connected to “AddAlarm” interface then it is connected to the “server”. After “server” is connected to the REST API known as SensorResource .After that to the SensorRepository (SQL Implementation).

When user starts the desktop client application user need to log into the system using his credentials. If credentials are correct user will redirect to Add alarm interface. When credentials are incorrect login system will notify the user to enter the correct information. Once user details are correct user can add new sensor to the system. Then Added information will save to database. And also all the information can be update and delete as well.

After adding and logout form the system user can see the added sensor information.

If user added CO2 and Smoke level values greater than 5, Then sensor status will change to the active state

**Sequence diagram to the workflow of Desktop Client**



**2.2 Web Client application workflow**

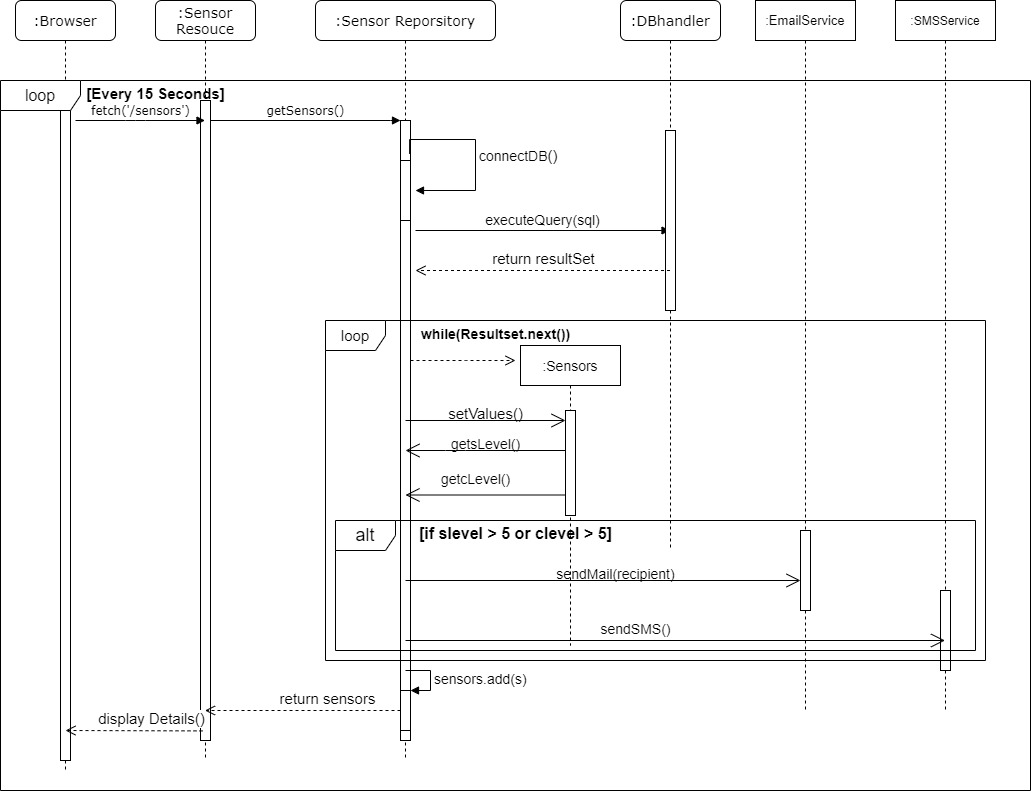
This web client application contains one jsp page named index.jsp(Browser). It act as a DOM of the web application. Also it connected to the REST API known as “SensorResource” and RMI server known as “SensorRepository” . There are two services Email service and SMS service both connected to the RMI server. SMTP email method is used as Email service and “Twilio”SMS service” used as SMS service

When user starts and load the web page , Browser(index.jsp) fetching the sensor resource through the path (/Sensors) . Sensor resource call the getSensor() method in the sensor repository via an object of Sensor Repository class sensor repository connect to the MYSQL database and it executes the particular query from the DB handler and returns the sensor details as a variable called resultSet.

In the senor repository it will check all the sensor results through the while loop and set the values to the relevant sensor objects and after it checks whether the CO2 level or Smoke level greater than 5. When one of the parameters satisfies the condition it will call the E-mail service the and the SMS service through the relevant methods. Finally the relevant sensor objects are added to the array list

In the sensor repository class and it will return to the sensor resource then to the browser

**Sequence diagram to the workflow of web client**



**3. List of service interface in web client**

**1. Index**

**2. Sensor**

**3. Sensor Repository**

**4. Sensor Resource**

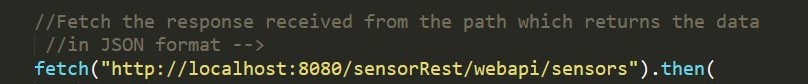


Figure 1 Index.jsp

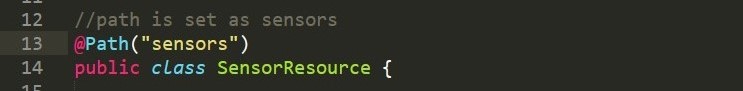
****

Figure 2 SensorResource.java

This is the way to connect the sensor resource (REST API) with index(DOM) by setting the path as “sensors” to access the relevant path in the index.

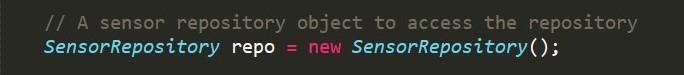
****

Figure 3 SensorResource.java

This is the object of sensor repository in the sensor resource which can access everything in the SensorRepository() constructor in sensor repository class.

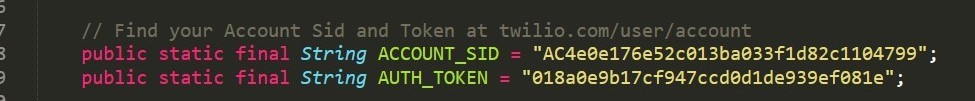


Figure 4 SensorRepository

This is the way of sending SMS using “Twilio” SMS service and these are the relevant account credentials of account Twilio

**4. List of service interface in desktop client**

**1.AddAlarm**

**2.Login**

**3.Sensor**

**4.Server**

**5.ServerInt**

A picture containing drawing

Description automatically generated

Figure AddAlarm.java

Get sensor data from admin and parse that data as sensor object in to the server.

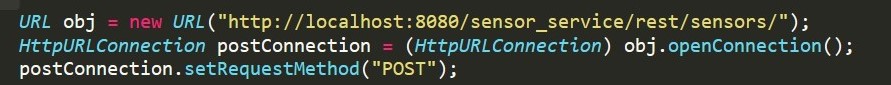


Figure Server.java

A picture containing drawing

Description automatically generatedGet sensor object from AddAlarm.java and parse that sensor object in to the POST method in SensorResource.java

Figure SensorResource.java

Get sensor object from Server.java class and parse that object in to the SensorRepository.java class



Figure SensorRepository.java

Get Sensor object from SensorResource.java class and save that Sensor object in database.

Above mentioned four code segments only related to the insert method. Other methods such as update, delete, check are also call according to the same steps.

**5.Appendix**

**5.1 Desktop client**

**AddAlarm.java**

import java.awt.BorderLayout;

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.border.EmptyBorder;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import java.awt.Font;

import javax.swing.SwingConstants;

import javax.swing.JTextField;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.io.IOException;

import java.rmi.RemoteException;

import java.awt.event.ActionEvent;

import java.sql.\*;

import java.awt.Color;

public class AddAlarm extends JFrame {

private JPanel contentPane;

private JTextField textField;

private JTextField textField\_1;

private JTextField textField\_2;

private JTextField textField\_4;

private JTextField textField\_3;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

//frame.dispose();

AddAlarm frame = new AddAlarm();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

public AddAlarm() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 860, 557);

contentPane = new JPanel();

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(null);

JLabel lblNewLabel = new JLabel("Add New Sensor");

lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 22));

lblNewLabel.setBounds(144, 13, 448, 49);

contentPane.add(lblNewLabel);

JLabel lblNewLabel\_1 = new JLabel("Sensor ID:");

lblNewLabel\_1.setHorizontalAlignment(SwingConstants.CENTER);

lblNewLabel\_1.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblNewLabel\_1.setBounds(106, 97, 137, 36);

contentPane.add(lblNewLabel\_1);

JLabel lblNewLabel\_2 = new JLabel("Floor Number");

lblNewLabel\_2.setHorizontalAlignment(SwingConstants.CENTER);

lblNewLabel\_2.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblNewLabel\_2.setBounds(106, 175, 137, 36);

contentPane.add(lblNewLabel\_2);

JLabel lblNewLabel\_3 = new JLabel("Room Number");

lblNewLabel\_3.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblNewLabel\_3.setBounds(129, 254, 100, 29);

contentPane.add(lblNewLabel\_3);

JLabel lblSmokeLevel = new JLabel("Smoke Level");

lblSmokeLevel.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblSmokeLevel.setHorizontalAlignment(SwingConstants.CENTER);

lblSmokeLevel.setBounds(90, 315, 180, 36);

contentPane.add(lblSmokeLevel);

JLabel lblCoLevel = new JLabel("Co2 Level");

lblCoLevel.setFont(new Font("Tahoma", Font.PLAIN, 15));

lblCoLevel.setHorizontalAlignment(SwingConstants.CENTER);

lblCoLevel.setBounds(106, 381, 151, 36);

contentPane.add(lblCoLevel);

textField = new JTextField();

textField.setBounds(374, 98, 213, 36);

contentPane.add(textField);

textField.setColumns(10);

textField\_1 = new JTextField();

textField\_1.setBounds(374, 176, 213, 36);

contentPane.add(textField\_1);

textField\_1.setColumns(10);

textField\_2 = new JTextField();

textField\_2.setBounds(374, 251, 213, 36);

contentPane.add(textField\_2);

textField\_2.setColumns(10);

textField\_4 = new JTextField();

textField\_4.setBounds(379, 375, 213, 36);

contentPane.add(textField\_4);

textField\_4.setColumns(10);

textField\_3 = new JTextField();

textField\_3.setBounds(374, 315, 213, 36);

contentPane.add(textField\_3);

textField\_3.setColumns(10);

JButton btnNewButton = new JButton("Add Sensor");

btnNewButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Sensor sen = new Sensor();

sen.setId(Integer.parseInt(textField.getText()));

sen.setStatus("Inactive");

sen.setFloorNum(Integer.parseInt(textField\_1.getText()));

sen.setRoomNum(Integer.parseInt(textField\_2.getText()));

sen.setsLevel(Integer.parseInt(textField\_3.getText()));

sen.setcLevel(Integer.parseInt(textField\_4.getText()));

try {

Server cp = new Server();

cp.addItem(sen);

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

});

btnNewButton.setBounds(422, 450, 130, 36);

contentPane.add(btnNewButton);

JButton btnUpdateSensor = new JButton("Update Sensor");

btnUpdateSensor.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

Sensor sen = new Sensor();

sen.setId(Integer.parseInt(textField.getText()));

sen.setFloorNum(Integer.parseInt(textField\_1.getText()));

sen.setRoomNum(Integer.parseInt(textField\_2.getText()));

sen.setsLevel(Integer.parseInt(textField\_3.getText()));

sen.setcLevel(Integer.parseInt(textField\_4.getText()));

try {

Server cp1 = new Server();

cp1.updateItem(sen);

} catch (IOException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

});

btnUpdateSensor.setBounds(564, 451, 122, 35);

contentPane.add(btnUpdateSensor);

JButton btnDeleteSensor = new JButton("Delete Sensor");

btnDeleteSensor.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

try {

Server cp = new Server();

cp.deleteItem(Integer.parseInt(textField.getText()));

} catch (RemoteException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

} catch (NumberFormatException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

} catch (Exception e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

}

});

btnDeleteSensor.setBounds(698, 450, 122, 36);

contentPane.add(btnDeleteSensor);

JButton btnLogout = new JButton("Logout");

btnLogout.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

dispose();

Login h = new Login();

h.main(null);

}

});

btnLogout.setFont(new Font("Tahoma", Font.PLAIN, 15));

btnLogout.setBounds(678, 78, 112, 43);

contentPane.add(btnLogout);

JLabel lblNewLabel\_4 = new JLabel("");

lblNewLabel\_4.setFont(new Font("Tahoma", Font.BOLD, 15));

lblNewLabel\_4.setForeground(new Color(255, 0, 0));

lblNewLabel\_4.setBounds(27, 417, 334, 67);

contentPane.add(lblNewLabel\_4);

}

}

**Login.java**

import java.awt.EventQueue;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JTextField;

import javax.swing.JPasswordField;

import javax.swing.JButton;

import java.awt.event.ActionListener;

import java.awt.event.ActionEvent;

import java.sql.\*;

import java.awt.Color;

import java.awt.Font;

import javax.swing.SwingConstants;

import javax.swing.table.DefaultTableModel;

import javax.swing.JTable;

import javax.swing.JScrollPane;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.rmi.RemoteException;

public class Login {

// private static Connection connection;

private JFrame frame;

private JTextField username;

private JPasswordField password;

private JTable table;

/\*\*

\* Launch the application.

\*/

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

Login window = new Login();

window.frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

public Login() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

private void initialize() {

frame = new JFrame();

frame.addWindowListener(new WindowAdapter() {

@Override

public void windowOpened(WindowEvent e) {

showData();

}

});

frame.setBounds(100, 100, 959, 571);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.getContentPane().setLayout(null);

JLabel lblUserName = new JLabel("User Name");

lblUserName.setHorizontalAlignment(SwingConstants.CENTER);

lblUserName.setFont(new Font("Tahoma", Font.BOLD, 15));

lblUserName.setBackground(Color.GRAY);

lblUserName.setForeground(Color.BLACK);

lblUserName.setBounds(0, 144, 148, 27);

frame.getContentPane().add(lblUserName);

JLabel lblPassword = new JLabel("Password");

lblPassword.setHorizontalAlignment(SwingConstants.CENTER);

lblPassword.setFont(new Font("Tahoma", Font.BOLD, 15));

lblPassword.setBounds(12, 209, 109, 27);

frame.getContentPane().add(lblPassword);

username = new JTextField();

username.setBounds(143, 144, 175, 38);

frame.getContentPane().add(username);

username.setColumns(10);

password = new JPasswordField();

password.setBounds(143, 204, 175, 38);

frame.getContentPane().add(password);

JButton btnLogin = new JButton("Login");

btnLogin.setFont(new Font("Tahoma", Font.BOLD, 15));

btnLogin.setBackground(Color.CYAN);

btnLogin.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

//

String uname = username.getText();

String pass = password.getText();

//

if(uname.equalsIgnoreCase("admin") && pass.equalsIgnoreCase("admin")){

frame.dispose();

AddAlarm h = new AddAlarm();

h.setVisible(true);

try {

Server s = new Server();

s.checkSensor();

} catch (RemoteException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

JOptionPane.showConfirmDialog(frame, "Your successfully Login");

}else {

JOptionPane.showConfirmDialog(frame, "Invalid username or password");

}

}

});

btnLogin.setBounds(143, 309, 135, 39);

frame.getContentPane().add(btnLogin);

JLabel lblNewLabel = new JLabel("Admin Login");

lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 25));

lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);

lblNewLabel.setBounds(12, 43, 281, 50);

frame.getContentPane().add(lblNewLabel);

JScrollPane scrollPane = new JScrollPane();

scrollPane.setBounds(330, 43, 599, 434);

frame.getContentPane().add(scrollPane);

table = new JTable();

scrollPane.setViewportView(table);

JLabel lblNewLabel\_1 = new JLabel("");

lblNewLabel\_1.setFont(new Font("Tahoma", Font.BOLD, 15));

lblNewLabel\_1.setForeground(new Color(255, 0, 0));

lblNewLabel\_1.setBounds(12, 383, 286, 100);

frame.getContentPane().add(lblNewLabel\_1);

}

private void showData() {

try {

DefaultTableModel model = new DefaultTableModel();

model.addColumn("Sensor ID");

model.addColumn("Status");

model.addColumn("Flow Num");

model.addColumn("Room Num");

model.addColumn("Smoke Level");

model.addColumn("CO2 Level");

Server s = new Server();

ResultSet rs = (ResultSet) s.getItems();

while(rs.next()) {

model.addRow(new Object[] {

rs.getString("id"),

rs.getString("status"),

rs.getString("floorNum"),

rs.getString("roomNum"),

rs.getString("sLevel"),

rs.getString("cLevel")

});

}

rs.close();

table.setModel(model);

table.setAutoResizeMode(0);

table.getColumnModel().getColumn(0).setPreferredWidth(80);

table.getColumnModel().getColumn(1).setPreferredWidth(120);

table.getColumnModel().getColumn(2).setPreferredWidth(80);

table.getColumnModel().getColumn(3).setPreferredWidth(80);

table.getColumnModel().getColumn(4).setPreferredWidth(80);

table.getColumnModel().getColumn(5).setPreferredWidth(80);

}catch(Exception e) {

System.out.println(e);

}

}

}

**Server.java**

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.net.HttpURLConnection;

import java.net.URL;

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

import org.json.JSONArray;

import org.json.JSONObject;

public class Server extends UnicastRemoteObject implements ServerInt {

protected Server() throws RemoteException {

super();

// TODO Auto-generated constructor stub

}

public void addItem(Sensor sensor) throws Exception {

final String POST\_PARAMS =

"{\n" + "\"id\": "+sensor.id+",\r\n" +

" \"floorNum\": "+sensor.floorNum+",\r\n" +

" \"roomNum\": "+sensor.roomNum+",\r\n" +

" \"sLevel\":"+sensor.sLevel+",\r\n" +

" \"cLevel\": "+sensor.cLevel+"" + "\n}";

System.out.println(POST\_PARAMS);

URL obj = new URL("http://localhost:8080/sensor\_service/rest/sensors/");

HttpURLConnection postConnection = (HttpURLConnection) obj.openConnection();

postConnection.setRequestMethod("POST");

postConnection.setRequestProperty("Content-Type", "application/json");

postConnection.setDoOutput(true);

OutputStream os = postConnection.getOutputStream();

os.write(POST\_PARAMS.getBytes());

os.flush();

os.close();

int responseCode = postConnection.getResponseCode();

System.out.println("POST Response Code : " + responseCode);

System.out.println("POST Response Message : " + postConnection.getResponseMessage());

int req=HttpURLConnection.HTTP\_CREATED ;

if (responseCode == (req-1)) { //success

BufferedReader in = new BufferedReader(new InputStreamReader(

postConnection.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in .readLine()) != null) {

response.append(inputLine);

} in .close();

// print result

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

} else {

System.out.println("Can not Insert");

}

}

public ArrayList<Sensor> getItems() throws Exception {

ArrayList<Sensor> faList = new ArrayList<Sensor>();

JSONArray myResponse ;

try {

String url = "http://localhost:8080/sensor\_service/rest/sensors/";

URL obj = new URL(url);

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

// optional default is GET

con.setRequestMethod("GET");

//add request header

con.setRequestProperty("User-Agent", "Mozilla/5.0");

int responseCode = con.getResponseCode();

System.out.println("\nSending 'GET' request to URL : " + url);

System.out.println("Response Code : " + responseCode);

BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

//print in String

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

//Read JSON response and print JSONArray

// JSONObject myResponse = new JSONObject(response.toString());

myResponse = new JSONArray(response.toString());

System.out.println("result after Reading JSON Response");

JSONObject sensor;

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

sensor = myResponse.getJSONObject(i);

Sensor currentSensor = new Sensor();

currentSensor.setId(Integer.valueOf(sensor.getInt("id")));

currentSensor.setStatus(sensor.getString("status"));

currentSensor.setFloorNum(Integer.valueOf(sensor.getInt("floorNum")));

currentSensor.setRoomNum(Integer.valueOf(sensor.getInt("roomNum")));

currentSensor.setsLevel(Integer.valueOf(sensor.getInt("sLevel")));

currentSensor.setcLevel(Integer.valueOf(sensor.getInt("cLevel")));

faList.add(currentSensor);

}

} catch (Exception e) {

// TODO: handle exception

}

return faList;

}

public void updateItem(Sensor sensor) throws Exception {

ArrayList<Sensor> faList = new ArrayList<Sensor>();

final String POST\_PARAMS =

"{\n" + "\"id\": "+sensor.id+",\r\n" +

" \"floorNum\": "+sensor.floorNum+",\r\n" +

" \"roomNum\": "+sensor.roomNum+",\r\n" +

" \"sLevel\":"+sensor.sLevel+",\r\n" +

" \"cLevel\": "+sensor.cLevel+"" + "\n}";

System.out.println(POST\_PARAMS);

URL obj = new URL("http://localhost:8080/sensor\_service/rest/sensors/"+sensor.id);

HttpURLConnection postConnection = (HttpURLConnection) obj.openConnection();

postConnection.setRequestMethod("DELETE");

postConnection.setRequestProperty("Content-Type", "application/json");

postConnection.setDoOutput(true);

OutputStream os = postConnection.getOutputStream();

os.write(POST\_PARAMS.getBytes());

os.flush();

os.close();

int responseCode = postConnection.getResponseCode();

System.out.println("POST Response Code : " + responseCode);

System.out.println("POST Response Message : " + postConnection.getResponseMessage());

int req=HttpURLConnection.HTTP\_CREATED ;

if (responseCode == (req-1)) { //success

BufferedReader in = new BufferedReader(new InputStreamReader(

postConnection.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in .readLine()) != null) {

response.append(inputLine);

} in .close();

// print result

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

}

}

public void checkSensor() throws Exception {

String url = ("http://localhost:8080/sensor\_service/rest/sensors/check");

URL obj = new URL(url);

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

// optional default is GET

con.setRequestMethod("GET");

//add request header

con.setRequestProperty("User-Agent", "Mozilla/5.0");

int responseCode = con.getResponseCode();

System.out.println("\nSending 'GET' request to URL : " + url);

System.out.println("Response Code : " + responseCode);

BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

//print in String

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

// TODO Auto-generated method stub

}

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

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

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

Server Hello = new Server();

Naming.rebind("ADD", Hello);

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

}

public void deleteItem(int id) throws Exception {

String url = ("http://localhost:8080/sensor\_service/rest/sensors/"+id);

URL obj = new URL(url);

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

// optional default is GET

con.setRequestMethod("GET");

//add request header

con.setRequestProperty("User-Agent", "Mozilla/5.0");

int responseCode = con.getResponseCode();

System.out.println("\nSending 'GET' request to URL : " + url);

System.out.println("Response Code : " + responseCode);

BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

while ((inputLine = in.readLine()) != null) {

response.append(inputLine);

}

in.close();

//print in String

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

// TODO Auto-generated method stub

}

}

**ServerInt.java**

import java.rmi.Remote;

import java.util.ArrayList;

public interface ServerInt extends Remote {

public void addItem(Sensor sensor) throws Exception;

public ArrayList<Sensor> getItems() throws Exception;

public void updateItem(Sensor sensor) throws Exception;

public void deleteItem(int id) throws Exception;

public void checkSensor() throws Exception;

}

**Sensor.java**

//A Sensor class which has all the required attributes and

//getters and setters.

public class Sensor {

// Attributes

int id;

String status;

int floorNum;

int roomNum;

int sLevel;

int cLevel;

// Getter for id

public int getId() {

return id;

}

// Setter for id

public void setId(int id) {

this.id = id;

}

// getter for status

public String isStatus() {

return status;

}

// setter for status

public void setStatus(String status) {

this.status = status;

}

// public String getStatus() {

//

// return status;

// }

// getter for floor number

public int getFloorNum() {

return floorNum;

}

// setter for floor number

public void setFloorNum(int floorNum) {

this.floorNum = floorNum;

}

// getter for room number

public int getRoomNum() {

return roomNum;

}

// setter for room number

public void setRoomNum(int roomNum) {

this.roomNum = roomNum;

}

// getter for smoke level

public int getsLevel() {

return sLevel;

}

// setter for smoke level

public void setsLevel(int sLevel) {

this.sLevel = sLevel;

}

// getter for co2 level

public int getcLevel() {

return cLevel;

}

// setter for co2 level

public void setcLevel(int cLevel) {

this.cLevel = cLevel;

}

// toString() method to get the details in the console

@Override

public String toString() {

return "Sensor [id=" + id + ", status=" + status + ", floorNum=" + floorNum + ", roomNum=" + roomNum

+ ", sLevel=" + sLevel + ", cLevel=" + cLevel + "]";

}

}

**5.1 Web Client**

**Sensor.java**

package com.sensor.sensorRest;

//A Sensor class which has all the required attributes and

//getters and setters.

public class Sensor {

// Attributes

private int id;

private String status;

private int floorNum;

private int roomNum;

private int sLevel;

private int cLevel;

// Getter for id

public int getId() {

return id;

}

// Setter for id

public void setId(int id) {

this.id = id;

}

// getter for status

public String isStatus() {

return status;

}

// setter for status

public void setStatus(String status) {

this.status = status;

}

// getter for floor number

public int getFloorNum() {

return floorNum;

}

// setter for floor number

public void setFloorNum(int floorNum) {

this.floorNum = floorNum;

}

// getter for room number

public int getRoomNum() {

return roomNum;

}

// setter for room number

public void setRoomNum(int roomNum) {

this.roomNum = roomNum;

}

// getter for smoke level

public int getsLevel() {

return sLevel;

}

// setter for smoke level

public void setsLevel(int sLevel) {

this.sLevel = sLevel;

}

// getter for co2 level

public int getcLevel() {

return cLevel;

}

// setter for co2 level

public void setcLevel(int cLevel) {

this.cLevel = cLevel;

}

// toString() method to get the details in the console

@Override

public String toString() {

return "Sensor [id=" + id + ", status=" + status + ", floorNum=" + floorNum + ", roomNum=" + roomNum

+ ", sLevel=" + sLevel + ", cLevel=" + cLevel + "]";

}

}

**SensorRepository.java**

package com.sensor.sensorRest;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.List;

import java.util.Properties;

import javax.mail.Authenticator;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.AddressException;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

import com.twilio.Twilio;

import com.twilio.type.PhoneNumber;

//Sensor Repository which deals with the back-end operations

public class SensorRepository {

// Find your Account Sid and Token at twilio.com/user/account

public static final String ACCOUNT\_SID = "AC4e0e176e52c013ba033f1d82c1104799";

public static final String AUTH\_TOKEN = "018a0e9b17cf947ccd0d1de939ef081e";

// Connection object that handles the connection for the database

Connection con = null;

public SensorRepository() {

// Attributes that has URL User-name and Password which helps

// to connect with the database

String url = "jdbc:mysql://localhost:3306/sensordb";

String username = "root";

String password = "";

try {

// Access for JDBC Driver

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

// Connect to the database

con = DriverManager.getConnection(url, username, password);

} catch (Exception e) {

System.out.println(e);

}

}

// Method which returns the sensor lists

public List<Sensor> getSensors() {

// Sensor lists objects

List<Sensor> sensors = new ArrayList<>();

// SQL query

String sql = "select \* from sensor";

try {

// statement object

Statement st = con.createStatement();

// query gets executed and the results get passed onto the ResultSet object

ResultSet rs = st.executeQuery(sql);

// each records are accessed and set into an object

while (rs.next()) {

Sensor s = new Sensor();

s.setId(rs.getInt(1));

s.setStatus(rs.getString(2));

s.setFloorNum(rs.getInt(3));

s.setRoomNum(rs.getInt(4));

s.setsLevel(rs.getInt(5));

s.setcLevel(rs.getInt(6));

// When a sensor has high smoke or co2 level the server automatically sends

// email

// to the recipient address.

if (s.getsLevel() > 5 || s.getcLevel() > 5) {

// sendMail function sends email to the recipient with the parameter

// id of the sensor.

sendMail("fashionhubafproject@gmail.com", s.getId());

// sendSMS function sends sms to the recipient with the parameter

// id of the sensor

sendSMS(s.getId());

}

// objects are added inside the sensors list

sensors.add(s);

}

} catch (Exception e) {

System.out.println(e);

}

// the list is returned

return sensors;

}

//Method to add the sensor details

public void add(Sensor sensor) throws SQLException {

String query = "insert into sensor (id,status,floorNum,roomNum,sLevel,cLevel) values (?,?,?,?,?,?)";

PreparedStatement stm = (PreparedStatement) con.prepareStatement(query);

stm.setInt(1, sensor.getId());

stm.setString(2, sensor.isStatus());

stm.setInt(3, sensor.getFloorNum());

stm.setInt(4, sensor.getRoomNum());

stm.setInt(5, sensor.getsLevel());

stm.setInt(6, sensor.getcLevel());

stm.execute();

stm.close();

//

}

//method to update the sensor details

public void update(Sensor newSensor) throws SQLException {

String query1 = "Update sensor set id = '" + newSensor.getId() + "', floorNum = '" + newSensor.getFloorNum()

+ "', roomNum = '" + newSensor.getRoomNum() + "', sLevel = '" + newSensor.getsLevel() + "', cLevel = '"

+ newSensor.getsLevel() + "' where id = '" + newSensor.getId() + "' ";

PreparedStatement stm = (PreparedStatement) con.prepareStatement(query1);

stm.execute();

stm.close();

}

//Method to delete the sensor details

public void delete(int id) throws SQLException {

String query2 = "delete from sensor where id = '" + id + "' ";

PreparedStatement stm = (PreparedStatement) con.prepareStatement(query2);

stm.execute();

stm.close();

}

//Method to check the details

public void check() throws InterruptedException, SQLException {

int i;

for (i = 0; i >= 0; i++) {

Thread.sleep(10000);

String query = "select \* from sensor";

Statement st = con.createStatement();

ResultSet rs = st.executeQuery(query);

while (rs.next()) {

int num1 = Integer.parseInt(rs.getString("sLevel"));

int num2 = Integer.parseInt(rs.getString("cLevel"));

if (num1 > 5 || num2 > 5) {

String query1 = "Update sensor set status = 'Active' where id = '" + rs.getString("id") + "' ";

PreparedStatement stm1 = (PreparedStatement) con.prepareStatement(query1);

stm1.execute();

i = -1;

} else {

String query3 = "Update sensor set status = 'Inactive' where id = '" + rs.getString("id") + "' ";

PreparedStatement stm3 = (PreparedStatement) con.prepareStatement(query3);

stm3.execute();

}

}

}

}

// method that helps to send sms.

public void sendSMS(int sensorID) {

Twilio.init(ACCOUNT\_SID, AUTH\_TOKEN);

com.twilio.rest.api.v2010.account.Message message = com.twilio.rest.api.v2010.account.Message

.creator(new PhoneNumber("+94775751416"), new PhoneNumber("+12057406151"),

"CAUTION !!! SENSOR " + sensorID + " is showing high smoke or CO2 Level.")

.create();

System.out.println(message.getSid());

System.out.println("DONE SENT SMS");

}

// method that helps to send email.

public static void sendMail(String recipient, int sensorID) {

System.out.println("Preparing to send email.");

// A properties object where properties are put inside to access with the

// session

Properties properties = new Properties();

properties.put("mail.smtp.auth", true);

properties.put("mail.smtp.starttls.enable", true);

properties.put("mail.smtp.host", "smtp.gmail.com");

properties.put("mail.smtp.port", "587");

final String myAccountEmail = "fashionhubafproject@gmail.com";

final String password = "Fashion@hub123";

Session session = Session.getInstance(properties, new Authenticator() {

@Override

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(myAccountEmail, password);

}

});

// preparing message that is going to be accessed with the transport method

Message message = prepareMessage(session, myAccountEmail, recipient, sensorID);

try {

Transport.send(message);

System.out.println("Message sent successfully.");

} catch (MessagingException e) {

e.printStackTrace();

}

}

// Prepare message method

private static Message prepareMessage(Session session, String myAccountEmail, String recipient, int sensorID) {

Message message = new MimeMessage(session);

try {

// From address, recipient address, subject and the body is set and then

// returned

message.setFrom(new InternetAddress(myAccountEmail));

message.setRecipient(Message.RecipientType.TO, new InternetAddress(recipient));

message.setSubject("CAUTION SENSOR " + sensorID + " IS AT RISK!");

message.setText(

"Hey, \nSensor " + sensorID + " is at risk!! \nDetected high smoke or co2 level.\nThank You.\n\n"

+ "~THIS IS AN AUTOGENERATED REPLY FROM THE SERVER DO NOT REPLY~");

return message;

} catch (AddressException e) {

e.printStackTrace();

} catch (MessagingException e) {

e.printStackTrace();

}

return null;

}

}

**SensorResourse.java**

package com.sensor.sensorRest;

import java.sql.SQLException;

import java.util.List;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

//path is set as sensors

@Path("sensors")

public class SensorResource {

// A sensor repository object to access the repository

SensorRepository repo = new SensorRepository();

// This is a GET request method which gives data type in a JSON format

@GET

@Produces(MediaType.APPLICATION\_JSON)

public List<Sensor> getSensors() {

System.out.println("Get Sensor Called.....");

// accesses the method getSensors() in the repository

return repo.getSensors();

}

@POST

@Produces(MediaType.APPLICATION\_JSON)

public void addSensor(Sensor newSensor) throws SQLException {

repo.add(newSensor);

}

@Path("{id}")

@POST

@Produces(MediaType.APPLICATION\_JSON)

public void updateSensor(Sensor newSensor) throws SQLException {

repo.update(newSensor);

}

@Path("{id}")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public void deleteSensor(@PathParam("id") int id) throws SQLException {

repo.delete(id);

}

@Path("check")

@GET

@Produces(MediaType.APPLICATION\_JSON)

public void ckeckSensor() throws SQLException, InterruptedException {

repo.check();

}

}

**Index.jsp**

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width-device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie-edge">

<link rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">

<style>

</style>

</head>

<body>

<h2>Fire Alarm Sensor Details</h2>

<table class="table table-stripped">

<!-- Table Header is set -->

<tr>

<th>Sensor ID</th>

<th>Status</th>

<th>Floor Number</th>

<th>Room Number</th>

<th>Smoke Level</th>

<th>CO2 Level</th>

</tr>

<!-- Table Body is set with an id as data -->

<tbody id="data">

</tbody>

</table>

<!-- Start of script tag -->

<script>

//The main function

function startFetchingData() {

//This function sets a time which re calls the same function again

setInterval(function () {

//Fetch the response received from the path which returns the data

//in JSON format -->

fetch("http://localhost:8080/sensorRest/webapi/sensors").then(

function(res) {

// getting the JSON type data from the response and passing

//it onto a function -->

res.json().then(

function(data) {

//To ensure that the data is getting retrieved -->

console.log(data);

// If data is not empty -->

if(data.length > 0) {

// creating a temporary variable for holding tags and data that

//is going to be overwritten-->

var temp = "";

// Accessing the data one by one and passing the content

//into an object as u -->

data.forEach(function(u) {

// checking if the smoke level or co2 level is above 5

//if yes all the text color in that section would have

//a style of red color -->

if((u.sLevel > 5) || (u.cLevel > 5)) {

temp += "<tr>";

temp += '<td style="color:red">' + u.id + "</td>";

temp += '<td style="color:red">' + u.status + "</td>";

temp += '<td style="color:red">' + u.floorNum + "</td>";

temp += '<td style="color:red">' + u.roomNum + "</td>";

temp += '<td style="color:red">' + u.sLevel + "</td>";

temp += '<td style="color:red">' + u.cLevel + "</td></tr>";

}

else {

// if there is no need of changing colors, it is set normally

//as a row without any beautification -->

temp += "<tr>";

temp += "<td>" + u.id + "</td>";

temp += "<td>" + u.status + "</td>";

temp += "<td>" + u.floorNum + "</td>";

temp += "<td>" + u.roomNum + "</td>";

temp += "<td>" + u.sLevel + "</td>";

temp += "<td>" + u.cLevel + "</td></tr>";

}

});

//----- close for loop

// the data element is overwritten with the codes in temp variable -->

document.getElementById("data").innerHTML = temp;

}

}

)

}

)

console.log('-----------15 seconds----------');

},15000);

}

//when the DOM is loaded, the main method is called

document.addEventListener('DOMContentLoaded', function () {

startFetchingData();

});

// End of script tag -->

</script>

</body>

</html>