Fire Alarm Monitoring System

DISTRIBUTE SYSTEMS

ASSIGNMENT – 02

(SRI LANKAN INSTITUTE OF INFORMATION TECHNOLOGY)

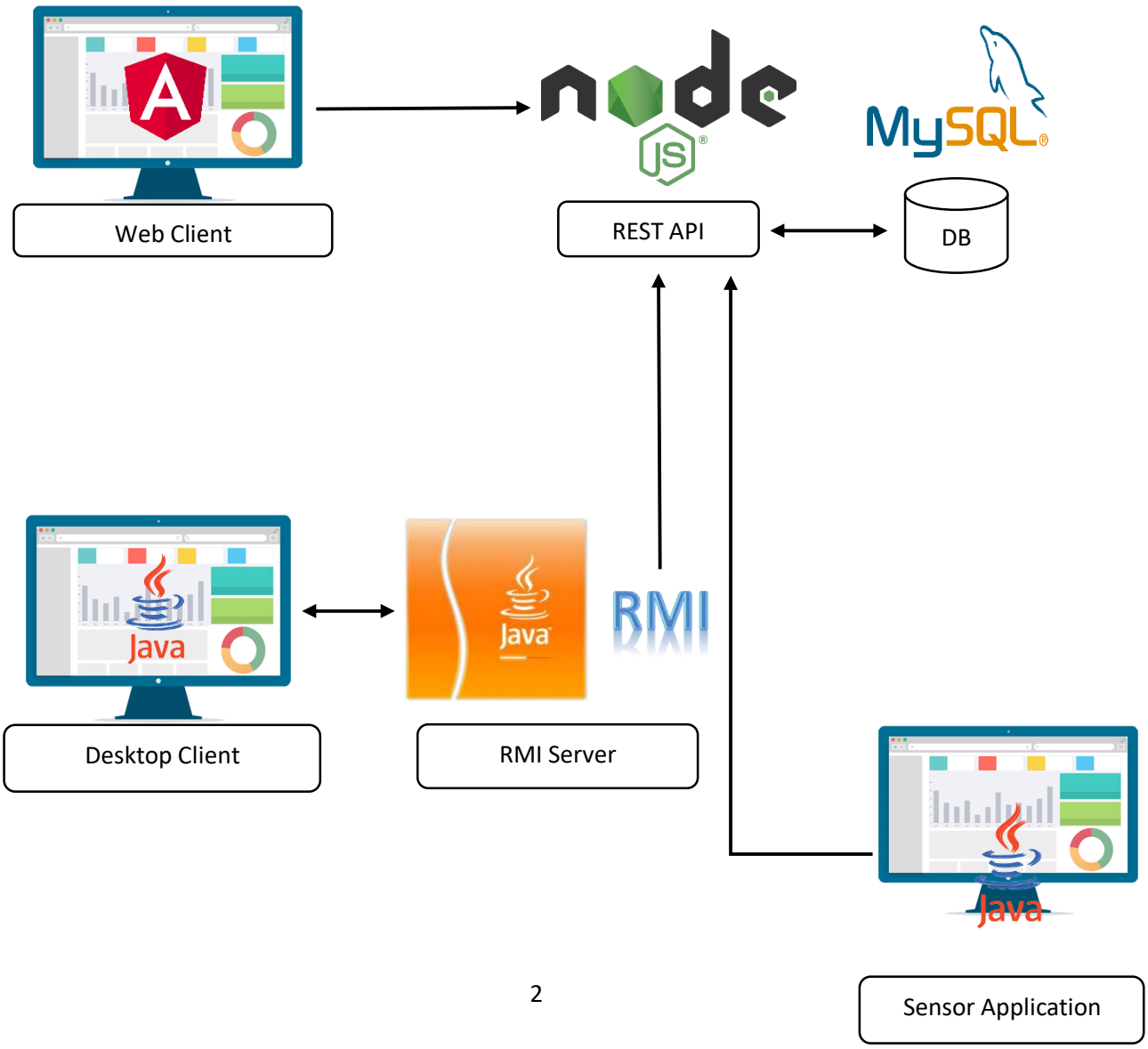
GROUP NO: 20.1 / 20.2

STUDENT ID	STUDENT NAME	CONTRIBUTION
IT18146202	Sudasinghe S.A.V.D	Contribute in developing RMI server, RMI desktop client, REST API and Web client.
IT18058642	Hansika P.D.U	Contribute in developing RMI server, RMI desktop client, REST API and Web client.
IT18049596	Silva P.H.D.D	Contribute in developing RMI server, RMI desktop client, REST API and Web client.
IT18107760	Wanigasinghe W.W.L.S.N	Contribute in developing RMI server, RMI desktop client, REST API and Web client.

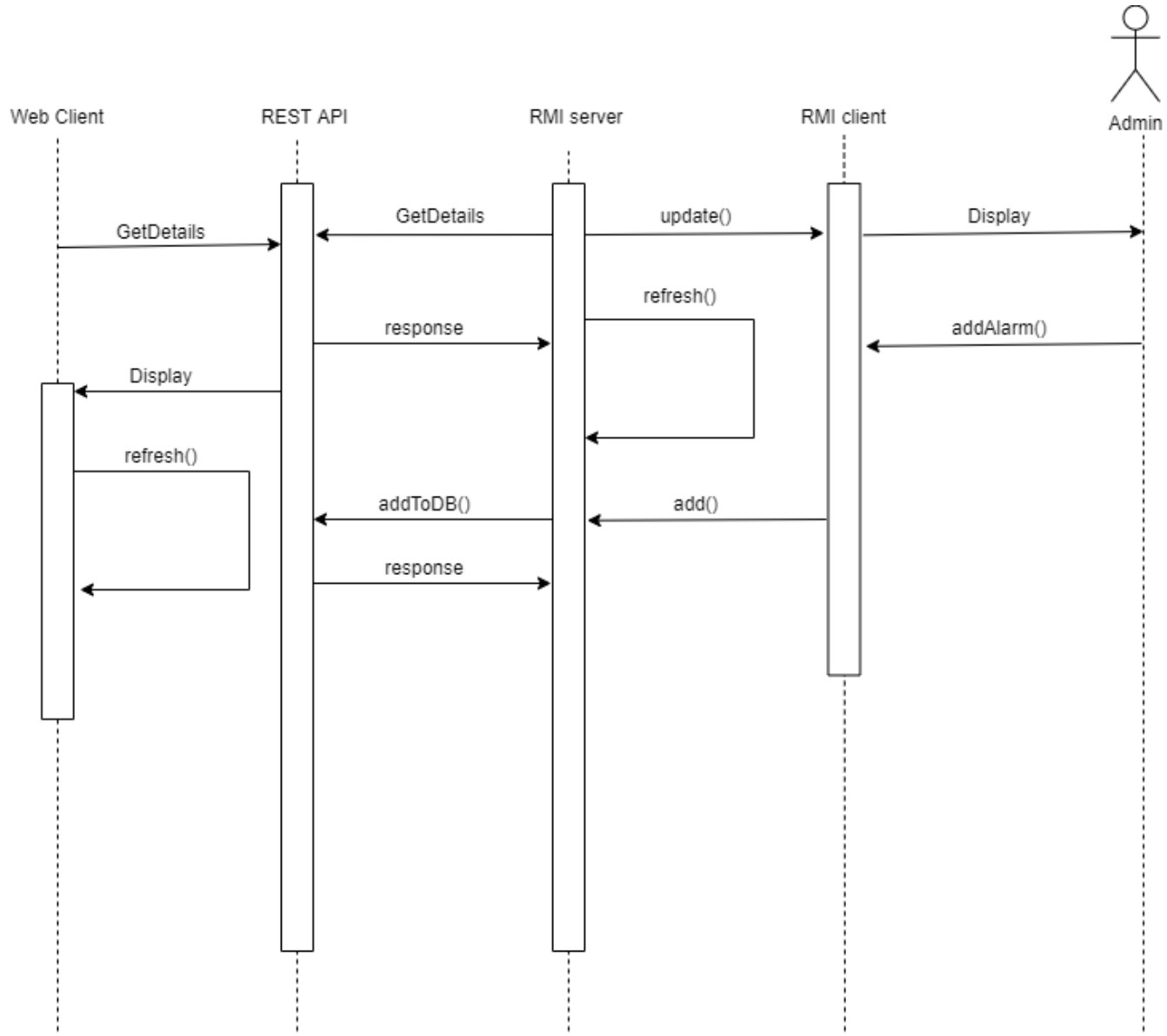
INTRODUCTION

This the assignment-02 for Distribute System module. In this assignment we had to develop a fire alarm monitoring system. We have developed RMI server, RMI desktop client application, REST API and Web client. RMI server and RMI desktop client application has developed by using JAVA programming language. The IDE we have used in here is NetBeans. We have developed REST API by using Node js programming language and Webstorm as the IDE. The webclient have developed by using Angular cli programming language and the IDE we used to develop this is Webstorm. MySQL is the database we have used in here. The admin can enter the floor number, room number, smoke level and status. The sensor ID is auto generated.

ARCHITECTURAL DESIGN



SEEQUENCE DIAGRAM



REST API

The programming language we have used to develop REST API is Nodejs and the IDE is Webstorm. We have used GET method to fetch the data from 'sensors' table in the 'firealarmdb' database. By using this address we could able to see the data in the database table 'http://127.0.0.1:3000/sensors '. We have used cors to enable (allow) the access of data to the web client.

Connect to the database,

```
1.  //Connect to MySQL
2.  var connection = mysql.createConnection({
3.      host: 'localhost',
4.      user: 'root',
5.      password: '',
6.      database: 'firealarmdb'
7.  });
8.
9.
10. connection.connect(function(err) {
11.     if (err) throw err
12.     console.log('Connected with mysql database')
13. })
```

Cors,

```
1.  //Enable the access of the database from different domain
2.
3.  app.use(cors({
4.
5.      origin: "*"
6.
7.  }));
```

GET method to fetch data,

```
8.  //REST API - fetch the data using GET method
9.  app.get('/sensors', function (req, res) {
10.     connection.query('select * from sensors', function (error, results, fields) {
11.         if (error) throw error;
12.         res.end(JSON.stringify(results));

});
```

WEB CLIENT

The web client has developed by using Angular cli programming language and Webstorm as the IDE. Start the web client application by typing 'ng serve --open' in the terminal. It will open the client web page via 'http://localhost:4200/'. The web page will display the fire alarm sensor details which include sensor ID, floor number, room number, smoke level and status.

Access data from the REST API to the web client.

```
1.  //Acess data from the API to the web client
2.
3.    ngOnInit() {
4.      let resp = this.http.get("http://127.0.0.1:3000/sensors");
5.      resp.subscribe((data) => this.sensors=data);
6.    }
```

RMI

We have developed the RMI server and RMI desktop client application from JAVA and NetBeans as IDE. In here admin can login to the system by providing username and password. Then the admin could able to see the database table with fire alarm sensor information. Admin can add new fire alarm sensors, edit and delete existing fire alarms. When admin selected a firealarm by ID from the table it will display on the input fields. If the smoke level greater than 5 the textfield related to status will appear in red colour. Otherwise it will appear in green colour.

Admin login,

```
1.  try{
2.              if(user.equals("admin") && pass.equals("123")){
3.
4.                  return found = true;
5.              }
6.              else{
7.                  return found = false;
8.              }
9.          }
10.         catch(Exception ex){
11.
12.             ex.printStackTrace();
13.         }
```

Database connection,

```
1.  private DatabaseConnection(){
2.
3.      try {
4.
5.          //Connecting to the MySQL Database
6.
7.         Class.forName("com.mysql.jdbc.Driver");
8.         System.out.println("Driver loaded");
9.         con = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/fire
    alarmdb","root","");
10.        System.out.println("Connection Established");
11.      }
12.      catch (Exception ex) {
13.
14.              System.out.println(ex);
15.
16.     }
17.
18.   }
```

Add new fire alarm sensor function,

```
1.      private void addActionPerformed(java.awt.event.ActionEvent evt) {

2.
3.          int floorNos = Integer.parseInt(jComboBox1.getSelectedItem().toString());
4.          String roomNos = jComboBox2.getSelectedItem().toString();
5.          int smokeLevels = Integer.parseInt(smoke_level.getText());
6.          String sensorStatus = status.getText();
7.
8.          try {
9.
10.             //Adding new fire alarm sensors to the database.
11.             //Admin need to provide the floor no (Integer value), room no(String value)
    , smoke level(Integer value) and status(String value) of the firelalrams.
12.             //No need to provide Sensor ID as it is autogenerated.
13.
14.             Statement s = con.createStatement();
15.             s.execute("INSERT INTO sensors(floorNo,roomNo,smokeLevel,status) values(" +
     floorNos + ",'" + roomNos + "'," + smokeLevels + ",'" + sensorStatus + "')");
16.
17.             JOptionPane.showMessageDialog(this, "Records Are Submitted!");
18.
19.             setalarmTable();
20.             s.close();
21.             resetData();
22.
23.         } catch (Exception ex) {
24.
25.             JOptionPane.showMessageDialog(this, ex);
26.
27.         }
28.
29.     }
```

Edit fire alarm sensor details,

```
30. private void editActionPerformed(java.awt.event.ActionEvent evt) {

31.
32.         if (alarmId != 0) {
33.
34.             int floorNos = Integer.parseInt(jComboBox1.getSelectedItem().toString());
35.             String roomNos = jComboBox2.getSelectedItem().toString();
36.             int smokeLevels = Integer.parseInt(smoke_level.getText());
37.             String sensorStatus = status.getText();
38.
39.             try{
40.
41.                 //Edit the fire alarm sensor details in the database
42.
43.                 Statement s = con.createStatement();
44.                 s.execute("update sensors set floorNo="+floorNos+",roomNo='"+roomNos+"'
    ,smokeLevel = "+smokeLevels+",status='"+sensorStatus+"' where id="+alarmId);
45.                 JOptionPane.showMessageDialog(this, "Records Are updated!");
46.                 setalarmTable();
47.                 resetData();
48.                 alarmId=0;
49.
50.             }
51.             catch(Exception ex){
52.
53.                 JOptionPane.showMessageDialog(this, "Cannot Update Records!");
54.
55.             }
56.
57.         }
58.

 }
```

Delete fire alarm,

```
1.  private void deleteActionPerformed(java.awt.event.ActionEvent evt) {

2.
3.          if(alarmId!=0){
4.
5.              try{
6.
7.                  //Delete the firealarm sensor details from the database when selected f
    rom the table  by Id
8.
9.                  Statement s = con.createStatement();
10.                 s.execute("delete from sensors where id="+alarmId);
11.                 JOptionPane.showMessageDialog(this, "Record Deleted!");
12.                 setalarmTable();
13.                 resetData();
14.                 alarmId=0;
15.
16.             }
17.             catch(Exception ex){
```

```
                JOptionPane.showMessageDialog(this, "Cannot Delete Records!");
            }
        }


 }
```

Change colour to red if the smoke level is greater than 5, if not change to green colour,

```
                if(smokeLevels>=5){

                    //When the smoke level is greater than or equal to 5, then the stat
    us textfield should appear in red colour

                    status.setBackground(Color.red);

                }

                else{

                    //When the smoke level is less than 5, then the status textfield sh
    ould appear in green colour

                    status.setBackground(Color.GREEN);

                }
```

Auto refresh the table in every 15 seconds,

```
 //Auto refresh the table in every 15 seconds

        setalarmTable();

        Thread t = new Thread(new Runnable() {
            @Override
            public void run() {
                for(;;){
                    try {
                        int time = 15000;
                        Thread.sleep(time);
                        System.out.println("Refreshed in 15 seconds!");
                        setalarmTable();

                    } catch (Exception ex) {
                        ex.printStackTrace();
                    }
                }
            }

        });

        t.start();
```

Appendices

We have removed the auto generated codes.

REST API

1. restapi.js

```
13. var http = require("http");
14. var express = require('express');
15. var mysql = require('mysql');
16. var bodyParser = require('body-parser');
17. var cors = require('cors')
18. 
19. var app = express();
20. 
21. 
22. //Enable the access of the database from different domain
23. 
24. app.use(cors({
25. 
26.     origin: "*"
27. 
28. }));
29. 
30. 
31. //Connect to MySQL
32. var connection = mysql.createConnection({
33.     host: 'localhost',
34.     user: 'root',
35.     password: '',
36.     database: 'firealarmdb'
37. });
38. 
39. 
40. connection.connect(function(err) {
41.     if (err) throw err
42.     console.log('Connected with mysql database')
43. })
44. 
45. 
46. //Body parser configuration
47. app.use( bodyParser.json() );       // to support JSON-encoded bodies
48. app.use(bodyParser.urlencoded({     // to support URL-encoded bodies
49.     extended: true
50. }));
51. 
52. 
53. //App server
54. const server = app.listen(3000, "127.0.0.1", function () {
```

```
55.
56.     var host = server.address().address
57.     var port = server.address().port
58.
59.     console.log("Example app listening at http://%s:%s", host, port)
60.
61. });
62.
63. //REST API - fetch the data using GET method
64. app.get('/sensors', function (req, res) {
65.     connection.query('select * from sensors', function (error, results, fields) {
66.         if (error) throw error;
67.         res.end(JSON.stringify(results));
68.     });
69. });
```

2. app.js

```
1.  var createError = require('http-errors');
2.  var express = require('express');
3.  var path = require('path');
4.  var cookieParser = require('cookie-parser');
5.  var logger = require('morgan');
6.  var indexRouter = require('./routes/index');
7.  var usersRouter = require('./routes/users');
8.
9.  var app = express();
10.
11. app.set('views', path.join(__dirname, 'views'));
12. app.set('view engine', 'pug');
13.
14. app.use(logger('dev'));
15. app.use(express.json());
16. app.use(express.urlencoded({ extended: false }));
17. app.use(cookieParser());
18. app.use(express.static(path.join(__dirname, 'public')));
19. app.use('/', indexRouter);
20. app.use('/users', usersRouter);
21.
22.
23. module.exports = app;
```

WEB CLIENT

1. user.component.html

```
1.  <!DOCTYPE html>
2.  <html lang="en">
3.  <head>
4.    <meta charset="utf-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <table class="table">

    <!--Display data of firealarm sensors-->

    <thead>
    <tr>
      <th>Sensor ID</th>
      <th>Floor Number</th>
      <th>Room Number</th>
      <th>Smoke Level</th>
      <th>Status</th>
    </tr>
    </thead>
    <tbody>
    <tr *ngFor="let user of sensors">
      <td>{{user.id}}</td>
      <td>{{user.floorNo}}</td>
      <td>{{user.roomNo}}</td>
      <td>{{user.smokeLevel}}</td>
      <td>{{user.status}}</td>
    </tr>
    </tbody>
  </table>
</div>

</body>
</html>
```

2. user.component.ts

```
import { Component, OnInit } from '@angular/core';
import {HttpClient} from '@angular/common/http';
import {delay} from "rxjs/operators";
import {of} from "rxjs";

@Component({
  selector: 'app-user',
  templateUrl: './user.component.html',
  styleUrls: ['./user.component.css']
})
export class UserComponent implements OnInit {

  sensors: any;

```

```
  constructor(private http:HttpClient) { }

  //Acess data from the API to the web client

  ngOnInit() {
    let resp = this.http.get("http://127.0.0.1:3000/sensors");
    resp.subscribe((data) => this.sensors=data);
  }

}
```

RMI

1. LoginServer.java

```
package rmi;

import java.rmi.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

//RMI server

public class LoginServer {

    public static void main(String[] args){

        try{

            Registry reg = LocateRegistry.createRegistry(1099);
            LogingImplementation lp = new LogingImplementation();
            reg.rebind("login", lp);
            System.out.println("Server Is Ready!");

        }
        catch(Exception ex){

            ex.printStackTrace();

        }


    }

}
```

2. LoginService.java

```
package rmi;

```

```
3.  import java.rmi.*;
4.
5.  public interface LoginService extends Remote{
6.
7.      public boolean getLogin(String user,String pass) throws RemoteException;
8.
9.  }
```

3. LogingImplementation.java

```
14. package rmi;
15.
16. import java.rmi.*;
17. import java.rmi.server.UnicastRemoteObject;
18.
19. public class LogingImplementation extends UnicastRemoteObject implements LoginService
    {
20.
21.     public LogingImplementation() throws RemoteException {
22.
23.
24.     }
25.
26.     //Implementing the admin logging function. Admin can able to log to the system via
    below logging credentials
27.     //Username:admin
28.     //Password:123
29.
30.     public boolean getLogin(String user, String pass) throws RemoteException {
31.
32.         boolean found = false;
33.
34.         try{
35.             if(user.equals("admin") && pass.equals("123")){
36.
37.                 return found = true;
38.             }
39.             else{
40.                 return found = false;
41.             }
42.         }
43.         catch(Exception ex){
44.
45.             ex.printStackTrace();
46.         }
47.
48.         return found;
49.
50.     }
51.
52.
53. }
```

4. Client.java

```
1.  package rmi;
2.
3.  import java.rmi.registry.LocateRegistry;
4.  import java.rmi.registry.Registry;
5.  import javax.swing.JOptionPane;
6.
7.  //RMI Client
8.
9.  public class Client extends javax.swing.JFrame {
10.
11.     public Client() {
12.         initComponents();
13.     }
14.
15.     @SuppressWarnings("unchecked")
16.     // <editor-
    fold defaultstate="collapsed" desc="Generated Code">
17.     private void initComponents() {
18.
19.     }// </editor-fold>
20.
21.     private void username_txtActionPerformed(java.awt.event.ActionEvent evt) {

22.         // TODO add your handling code here:
23.     }
24.
25.     private void password_txtActionPerformed(java.awt.event.ActionEvent evt) {

26.         // TODO add your handling code here:
27.     }
28.
29.     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

30.
31.         boolean fail = false;
32.
33.         try{
34.
35.             Registry reg = LocateRegistry.getRegistry("127.0.0.1",1099);
36.             LoginService i = (LoginService)reg.lookup("login");
37.             fail = i.getLogin(username_txt.getText(), password_txt.getText());
38.
39.             if(fail==true){
40.
41.                 //When admin logged to system it should display "Login sucessful!" mess
    age
42.
43.                 JOptionPane.showMessageDialog(null,"Login sucessful!");
44.
45.                 FireAlarmSensor alarmdata = new FireAlarmSensor();
46.                 alarmdata.setVisible(true);
47.
48.             }else{
49.
50.                 //If admin unable to logged to the system it should display "Login fail
    ed!" message
51.
52.                 JOptionPane.showMessageDialog(null,"Login failed!");
```

```
53.
54.             }
55.
56.         }
57.         catch(Exception ex){
58.
59.             ex.printStackTrace();
60.
61.         }
62.     }
63.
64.     /**
65.      * @param args the command line arguments
66.      */
67.     public static void main(String args[]) {
68.         /* Set the Nimbus look and feel */
69.         //<editor-
    fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
70.         /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
    look and feel.
71.          * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfe
    el/plaf.html
72.          */
73.         try {
74.             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.get
    InstalledLookAndFeels()) {
75.                 if ("Nimbus".equals(info.getName())) {
76.                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
77.                     break;
78.                 }
79.             }
80.         } catch (ClassNotFoundException ex) {
81.             java.util.logging.Logger.getLogger(Client.class.getName()).log(java.util.lo
    gging.Level.SEVERE, null, ex);
82.         } catch (InstantiationException ex) {
83.             java.util.logging.Logger.getLogger(Client.class.getName()).log(java.util.lo
    gging.Level.SEVERE, null, ex);
84.         } catch (IllegalAccessException ex) {
85.             java.util.logging.Logger.getLogger(Client.class.getName()).log(java.util.lo
    gging.Level.SEVERE, null, ex);
86.         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
87.             java.util.logging.Logger.getLogger(Client.class.getName()).log(java.util.lo
    gging.Level.SEVERE, null, ex);
88.         }
89.         //</editor-fold>
90.         //</editor-fold>
91.
92.         /* Create and display the form */
93.         java.awt.EventQueue.invokeLater(new Runnable() {
94.             public void run() {
95.                 new Client().setVisible(true);
96.             }
97.         });
98.     }
99.
100.        // Variables declaration - do not modify
101.        private javax.swing.JButton jButton1;
102.        private javax.swing.JLabel jLabel1;
103.        private javax.swing.JLabel jLabel2;
104.        private javax.swing.JLabel jLabel3;
105.        private javax.swing.JPasswordField password_txt;
```

```
106.           private javax.swing.JTextField username_txt;
107.           // End of variables declaration
108.       }
```

5. DatabaseConnection.java

```
19. package rmi;
20. 
21. import com.mysql.jdbc.Connection;
22. import static java.lang.Class.forName;
23. import java.sql.DriverManager;
24. import java.util.logging.Level;
25. import java.util.logging.Logger;
26. 
27. 
28. public class DatabaseConnection {
29. 
30.     private Connection con;
31.     private static DatabaseConnection db;
32. 
33.     private DatabaseConnection(){
34. 
35.     try {
36. 
37.         //Connecting to the MySQL Database
38. 
39.        Class.forName("com.mysql.jdbc.Driver");
40.        System.out.println("Driver loaded");
41.        con = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/fire
    alarmdb","root","");
42.        System.out.println("Connection Established");
43.      }
44.      catch (Exception ex) {
45. 
46.              System.out.println(ex);
47. 
48.     }
49. 
50.   }
51. 
52.     public static DatabaseConnection getDatabaseConnection(){
53. 
54.         if(db == null){
55.            db =  new DatabaseConnection();
56.         }
57.         return db;
58.     }
59. 
60.     public Connection getConnection(){
61.         return con;
62.     }
63. 
64. 
65. }
```

6. FireAlarmSensor.java

```
24. package rmi;
25. 
26. import java.awt.Color;
27. import java.sql.Connection;
28. import java.sql.DriverManager;
29. import java.sql.ResultSet;
30. import java.sql.Statement;
31. import javax.swing.JOptionPane;
32. import javax.swing.table.DefaultTableModel;
33. 
34. 
35. public class FireAlarmSensor extends javax.swing.JFrame {
36. 
37.     private Connection con;
38.     private int alarmId;
39. 
40. 
41.     public FireAlarmSensor() {
42.         initComponents();
43. 
44.         try {
45. 
46.             //Connection to the MySQL Database
47. 
48.             Class.forName("com.mysql.jdbc.Driver");
49.             System.out.println("Driver loaded");
50.             con = DriverManager.getConnection("jdbc:mysql://localhost:3306/firealarmdb"
    , "root", "");
51.             System.out.println("Connection Established!");
52. 
53.         } catch (Exception ex) {
54. 
55.             System.out.println(ex);
56. 
57.         }
58. 
59.         DatabaseConnection db = DatabaseConnection.getDatabaseConnection();
60.         con = db.getConnection();
61. 
62.         //Auto refresh the table in every 15 seconds
63. 
64.         setalarmTable();
65. 
66.         Thread t = new Thread(new Runnable() {
67.             @Override
68.             public void run() {
69.                 for(;;){
70.                     try {
71.                         int time = 15000;
72.                         Thread.sleep(time);
73.                         System.out.println("Refreshed in 15 seconds!");
74.                         setalarmTable();
75. 
76.                     } catch (Exception ex) {
77.                         ex.printStackTrace();
78.                     }
79.                 }
80.             }
```

```
81.
82.         });
83.
84.         t.start();
85.
86.     }
87.
88.     @SuppressWarnings("unchecked")
89.     // <editor-
    fold defaultstate="collapsed" desc="Generated Code">
90.     private void initComponents() {
91.
92.     }// </editor-fold>
93.
94.     private void deleteActionPerformed(java.awt.event.ActionEvent evt) {

95.
96.         if(alarmId!=0){
97.
98.             try{
99.
100.                         //Delete the firealarm sensor details from the database when sel
    ected from the table  by Id
101.
102.                        Statement s = con.createStatement();
103.                        s.execute("delete from sensors where id="+alarmId);
104.                        JOptionPane.showMessageDialog(this, "Record Are Deleted!");
105.
106.                        setalarmTable();
107.                        resetData();
108.                        alarmId=0;
109.
110.                    }
111.                    catch(Exception ex){
112.
113.                        JOptionPane.showMessageDialog(this, "Can't Delete Records!");
114.                    }
115.                }
116.
117.            }
118.
119.            private void addActionPerformed(java.awt.event.ActionEvent evt) {

120.
121.                int floorNos = Integer.parseInt(Floor_no.getSelectedItem().toString());

122.                String roomNos = Room_no.getSelectedItem().toString();
123.                int smokeLevels = Integer.parseInt(smoke_level.getText());
124.                String sensorStatus = status.getText();
125.
126.                try {
127.
128.                    //Adding new fire alarm sensors to the database.
129.                    //Admin need to provide the floor no (Integer value), room no(String
     value), smoke level(Integer value) and status(String value) of the firelalrams.
130.                    //No need to provide Sensor ID as it is autogenerated.
131.
132.                    Statement s = con.createStatement();
133.                    s.execute("INSERT INTO sensors(floorNo,roomNo,smokeLevel,status) val
    ues(" + floorNos + ",'" + roomNos + "'," + smokeLevels + ",'" + sensorStatus + "')");
134.
```

```
135.                     JOptionPane.showMessageDialog(this, "Records Are Submitted!");
136.
137.                     setalarmTable();
138.                     s.close();
139.                     resetData();
140.
141.                 } catch (Exception ex) {
142.
143.                     JOptionPane.showMessageDialog(this, ex);
144.
145.                 }
146.
147.             }
148.
149.             private void editActionPerformed(java.awt.event.ActionEvent evt) {

150.
151.                 if (alarmId != 0) {
152.
153.                     int floorNos = Integer.parseInt(Floor_no.getSelectedItem().toString(
    ));
154.                     String roomNos = Room_no.getSelectedItem().toString();
155.                     int smokeLevels = Integer.parseInt(smoke_level.getText());
156.                     String sensorStatus = status.getText();
157.
158.                     try{
159.
160.                         //Edit the fire alarm sensor details in the database
161.
162.                         Statement s = con.createStatement();
163.                         s.execute("update sensors set floorNo="+floorNos+",roomNo='"+roo
    mNos+"',smokeLevel = "+smokeLevels+",status='"+sensorStatus+"' where id="+alarmId);
164.                         JOptionPane.showMessageDialog(this, "Records Are Updated!");
165.
166.                         setalarmTable();
167.                         resetData();
168.                         alarmId=0;
169.
170.                     }
171.                     catch(Exception ex){
172.
173.                         JOptionPane.showMessageDialog(this, "Can't Update Records!");
174.
175.                     }
176.
177.                 }
178.
179.             }
180.
181.             private void alarmTableMouseClicked(java.awt.event.MouseEvent evt) {

182.
183.                 try {
184.
185.                     //When selected a row from the table, the details of the fire alarm
    sensor should display on the text boxes and combo boxes
186.
187.                     alarmId = Integer.parseInt(alarmTable.getValueAt(alarmTable.getSelec
    tedRow(), 0).toString());
188.                     Statement s = con.createStatement();
```

```
189.                     ResultSet rs = s.executeQuery("select * from sensors where id=" + al
    armId);
190.
191.                     if (rs.next()) {
192.
193.                         sensor_id.setText(rs.getInt(1) + "");
194.                         Floor_no.setSelectedItem(rs.getString(2));
195.                         Room_no.setSelectedItem(rs.getString(3));
196.                         smoke_level.setText(rs.getString(4) + "");
197.                         status.setText(rs.getString(5));
198.
199.
200.                         int smokeLevels = Integer.parseInt(smoke_level.getText());
201.
202.                         if(smokeLevels>=5){
203.
204.                             //When the smoke level is greater than or equal to 5, then t
    he status textfield should appear in red colour
205.
206.                             status.setBackground(Color.red);
207.
208.                         }
209.
210.                         else{
211.
212.                             //When the smoke level is less than 5, then the status textf
    ield should appear in green colour
213.
214.                             status.setBackground(Color.GREEN);
215.
216.                         }
217.
218.                     }
219.
220.                     rs.close();
221.                     s.close();
222.
223.                 } catch (Exception ex) {
224.
225.                     JOptionPane.showMessageDialog(this, ex);
226.                 }
227.             }
228.
229.             /**
230.              * @param args the command line arguments
231.              */
232.             public static void main(String args[]) {
233.                 /* Set the Nimbus look and feel */
234.                 //<editor-
    fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
235.                 /* If Nimbus (introduced in Java SE 6) is not available, stay with the d
    efault look and feel.
236.                  * For details see http://download.oracle.com/javase/tutorial/uiswing/lo
    okandfeel/plaf.html
237.                  */
238.                 try {
239.                     for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIMana
    ger.getInstalledLookAndFeels()) {
240.                         if ("Nimbus".equals(info.getName())) {
241.                             javax.swing.UIManager.setLookAndFeel(info.getClassName());
242.                             break;
```

```
243.                         }
244.                     }
245.                 } catch (ClassNotFoundException ex) {
246.                     java.util.logging.Logger.getLogger(FireAlarmSensor.class.getName()).
    log(java.util.logging.Level.SEVERE, null, ex);
247.                 } catch (InstantiationException ex) {
248.                     java.util.logging.Logger.getLogger(FireAlarmSensor.class.getName()).
    log(java.util.logging.Level.SEVERE, null, ex);
249.                 } catch (IllegalAccessException ex) {
250.                     java.util.logging.Logger.getLogger(FireAlarmSensor.class.getName()).
    log(java.util.logging.Level.SEVERE, null, ex);
251.                 } catch (javax.swing.UnsupportedLookAndFeelException ex) {
252.                     java.util.logging.Logger.getLogger(FireAlarmSensor.class.getName()).
    log(java.util.logging.Level.SEVERE, null, ex);
253.                 }
254.                 //</editor-fold>
255.
256.                 /* Create and display the form */
257.                 java.awt.EventQueue.invokeLater(new Runnable() {
258.                     public void run() {
259.                         new FireAlarmSensor().setVisible(true);
260.                     }
261.                 });
262.             }
263.
264.             // Variables declaration - do not modify
265.             private javax.swing.JComboBox<String> Floor_no;
266.             private javax.swing.JComboBox<String> Room_no;
267.             private javax.swing.JButton add;
268.             private javax.swing.JTable alarmTable;
269.             private javax.swing.JButton delete;
270.             private javax.swing.JButton edit;
271.             private javax.swing.JLabel jLabel1;
272.             private javax.swing.JLabel jLabel2;
273.             private javax.swing.JLabel jLabel3;
274.             private javax.swing.JLabel jLabel4;
275.             private javax.swing.JLabel jLabel5;
276.             private javax.swing.JLabel jLabel6;
277.             private javax.swing.JPanel jPanel1;
278.             private javax.swing.JScrollPane jScrollPane1;
279.             private javax.swing.JTextField sensor_id;
280.             private javax.swing.JTextField smoke_level;
281.             private javax.swing.JTextField status;
282.             // End of variables declaration
283.
284.             private void resetData() {
285.
286.         //        sensor_id.setText("");
287.         //        floor_no.setText("");
288.         //        room_no.setText("");
289.                 smoke_level.setText("");
290.                 status.setText("");
291.
292.             }
293.
294.             private void setalarmTable() {
295.
296.                 try {
297.
298.                     int rows = 0;
299.                     int rowIndex = 0;
```

```
300.
301.                     Statement s = con.createStatement();
302.                     ResultSet rs = s.executeQuery("select * from sensors order by id des
    c");
303.
304.                     if (rs.next()) {
305.
306.                         rs.last();
307.                         rows = rs.getRow();
308.                         rs.beforeFirst();
309.                     }
310.
311.                     String[][] data = new String[rows][5];
312.
313.                     while (rs.next()) {
314.
315.                         data[rowIndex][0] = rs.getInt(1) + "";
316.                         data[rowIndex][1] = rs.getInt(2) + "";
317.                         data[rowIndex][2] = rs.getString(3);
318.                         data[rowIndex][3] = rs.getInt(4) + "";
319.                         data[rowIndex][4] = rs.getString(5);
320.
321.                         rowIndex++;
322.
323.                     }
324.
325.                     String[] cols = {"ID", "Floor No", "Room No", "Smoke Level", "Status
    "};
326.                     DefaultTableModel model = new DefaultTableModel(data, cols);
327.                     alarmTable.setModel(model);
328.
329.
330.                     rs.close();
331.                     s.close();
332.
333.                 } catch (Exception ex) {
334.
335.                     JOptionPane.showMessageDialog(this, "Can't Retrieve Data!");
336.
337.                 }
338.
339.             }
340.
341.
342.         }
```

7. Emails.java

```
1.  package Mails;
2.
3.  import java.util.Properties;
4.  import javax.mail.Message;
5.  import javax.mail.MessagingException;
6.  import javax.mail.PasswordAuthentication;
7.  import javax.mail.Session;
8.  import javax.mail.Transport;
```

```
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

/**
 *
 * @author hp
 */
public class Emails {

    public static void main(String[] args) throws AddressException, MessagingException{


        final String username = "******@gmail.com";
        final String password = "*****";

        Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.port", "587");

        Session session = Session.getInstance(props, new javax.mail.Authenticator() {

                    protected PasswordAuthentication gePasswordAuthentication(){
                       return new PasswordAuthentication(username, password);

                    }
        });

        try{

            Message message = new MimeMessage(session);
            message.setFrom(new InternetAddress("****@gmail.com"));
            message.setRecipients(Message.RecipientType.TO,
                    InternetAddress.parse("****@gmail.com"));
            message.setSubject("FireSensor Alarm is active!!!");

            Transport.send(message);

            System.out.println("was the mail sent:Done");


        }
        catch(MessagingException e){

            throw new RuntimeException(e);
           //ex.printStackTrace();
        }


    }

}
```

THANK YOU