

Fire Alarm Monitoring System

DISTRIBUTE SYSTEMS

ASSIGNMENT – 02

(SRI LANKAN INSTITUTE OF INFORMATION TECHNOLOGY)

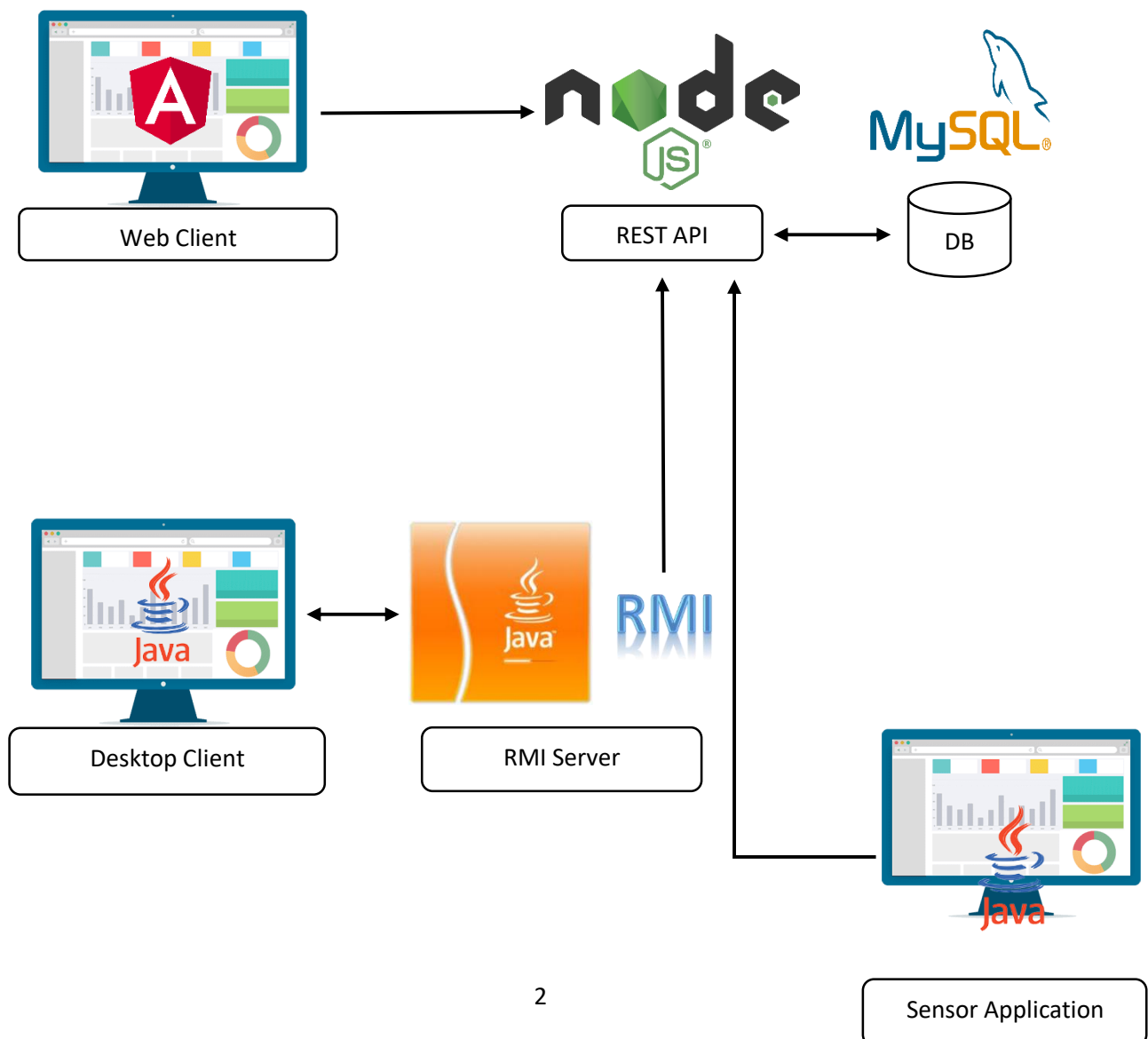
GROUP NO: 20.1 / 20.2

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.

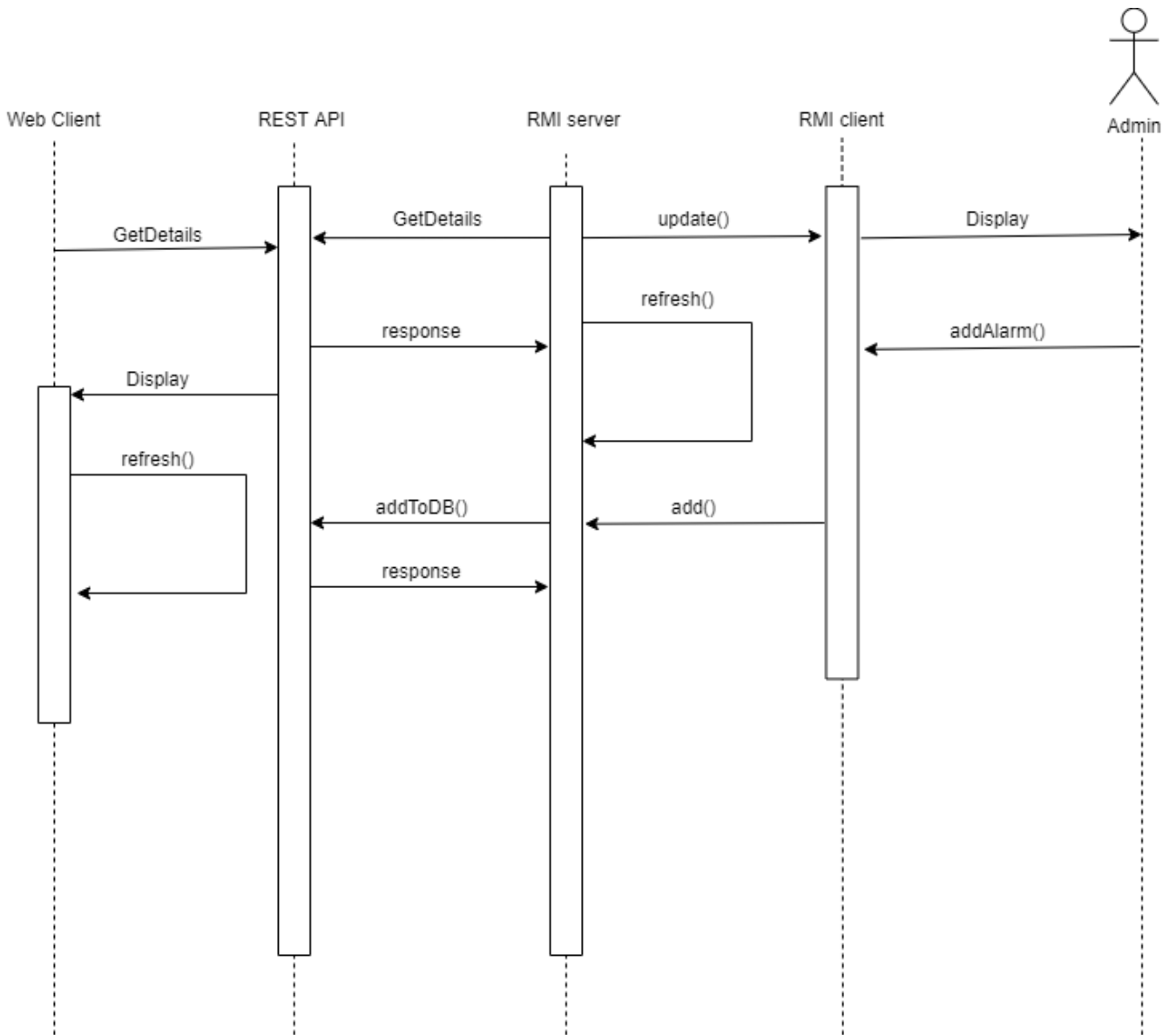
INTRODUCTION

This is the assignment-02 for Distributed 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));
13.   });
14. });
```

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. //Access 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 firealarms.
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+"',  
45.             ,smokeLevel = '"+smokeLevels+",status='"+sensorStatus+"' where id="+alarmId);  
46.             JOptionPane.showMessageDialog(this, "Records Are updated!");  
47.             setalarmTable();  
48.             resetData();  
49.             alarmId=0;  
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  
8.             rom the table by Id  
9.  
10.            Statement s = con.createStatement();  
11.            s.execute("delete from sensors where id="+alarmId);  
12.            JOptionPane.showMessageDialog(this, "Record Deleted!");  
13.            setalarmTable();  
14.            resetData();  
15.            alarmId=0;  
16.        }  
17.        catch(Exception ex){
```

```

18.
19.         JOptionPane.showMessageDialog(this, "Cannot Delete Records!");
20.     }
21. }
22.
23.
}

```

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

```

1.         if(smokeLevels>=5){
2.
3.             //When the smoke level is greater than or equal to 5, then the status textfield should appear in red colour
4.
5.             status.setBackground(Color.red);
6.
7.         }
8.
9.         else{
10.
11.             //When the smoke level is less than 5, then the status textfield should appear in green colour
12.
13.             status.setBackground(Color.GREEN);
14.
15.         }

```

Auto refresh the table in every 15 seconds,

```

1. //Auto refresh the table in every 15 seconds
2.
3.     setalarmTable();
4.
5.     Thread t = new Thread(new Runnable() {
6.         @Override
7.         public void run() {
8.             for(;;){
9.                 try {
10.                     int time = 15000;
11.                     Thread.sleep(time);
12.                     System.out.println("Refreshed in 15 seconds!");
13.                     setalarmTable();
14.
15.                 } catch (Exception ex) {
16.                     ex.printStackTrace();
17.                 }
18.             }
19.         }
20.     });
21.
22.
23.     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">

```

```

5.   <meta name="viewport" content="width=device-width, initial-scale=1">
6.   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/boot
strap.min.css">
7.   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></scri
pt>
8.   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.j
s"></script>
9.   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></s
cript>
10. </head>
11. <body>
12.
13. <div class="container">
14.   <table class="table">
15.
16.     <!--Display data of firealarm sensors-->
17.
18.     <thead>
19.       <tr>
20.         <th>Sensor ID</th>
21.         <th>Floor Number</th>
22.         <th>Room Number</th>
23.         <th>Smoke Level</th>
24.         <th>Status</th>
25.       </tr>
26.     </thead>
27.     <tbody>
28.       <tr *ngFor="let user of sensors">
29.         <td>{{user.id}}</td>
30.         <td>{{user.floorNo}}</td>
31.         <td>{{user.roomNo}}</td>
32.         <td>{{user.smokeLevel}}</td>
33.         <td>{{user.status}}</td>
34.       </tr>
35.     </tbody>
36.   </table>
37. </div>
38.
39. </body>
40. </html>

```

2. user.component.ts

```

7. import { Component, OnInit } from '@angular/core';
8. import { HttpClient } from '@angular/common/http';
9. import { delay } from "rxjs/operators";
10. import { of } from "rxjs";
11.
12. @Component({
13.   selector: 'app-user',
14.   templateUrl: './user.component.html',
15.   styleUrls: ['./user.component.css']
16. })
17. export class UserComponent implements OnInit {
18.
19.   sensors: any;
20.

```

```

21. constructor(private http:HttpClient) { }
22.
23. //Access data from the API to the web client
24.
25. ngOnInit() {
26.     let resp = this.http.get("http://127.0.0.1:3000/sensors");
27.     resp.subscribe((data) => this.sensors=data);
28. }
29.
30. }

```

RMI

1. LoginServer.java

```

2. package rmi;
3.
4. import java.rmi.*;
5. import java.rmi.registry.LocateRegistry;
6. import java.rmi.registry.Registry;
7.
8. //RMI server
9.
10. public class LoginServer {
11.
12.     public static void main(String[] args){
13.
14.         try{
15.
16.             Registry reg = LocateRegistry.createRegistry(1099);
17.             LoggingImplementation lp = new LoggingImplementation();
18.             reg.rebind("login", lp);
19.             System.out.println("Server Is Ready!");
20.
21.         }
22.         catch(Exception ex){
23.
24.             ex.printStackTrace();
25.
26.         }
27.
28.     }
29. }
30.
31. }

```

2. LoginService.java

```

1. package rmi;
2.

```

```

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. LoggingImplementation.java

```

14. package rmi;
15.
16. import java.rmi.*;
17. import java.rmi.server.UnicastRemoteObject;
18.
19. public class LoggingImplementation extends UnicastRemoteObject implements LoginService
20. {
21.     public LoggingImplementation() 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.
23.         // TODO add your handling code here:
24.     }
25.
26.     private void password_txtActionPerformed(java.awt.event.ActionEvent evt) {
27.
28.         // TODO add your handling code here:
29.     }
30.
31.     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
32.
33.         boolean fail = false;
34.
35.         try{
36.
37.             Registry reg = LocateRegistry.getRegistry("127.0.0.1",1099);
38.             LoginService i = (LoginService)reg.lookup("login");
39.             fail = i.getLogin(username_txt.getText(), password_txt.getText());
40.
41.             if(fail==true){
42.
43.                 //When admin logged to system it should display "Login sucessful!" message
44.                 JOptionPane.showMessageDialog(null,"Login sucessful!");
45.
46.                 FireAlarmSensor alarmdata = new FireAlarmSensor();
47.                 alarmdata.setVisible(true);
48.
49.             }else{
50.
51.                 //If admin unable to logged to the system it should display "Login failed!" message
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-
70.     fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
71.     /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
72.     look and feel.
73.     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfe
74.     el/plaf.html
75.     */
76.     try {
77.         for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.get
78.         InstalledLookAndFeels()) {
79.             if ("Nimbus".equals(info.getName())) {
80.                 javax.swing.UIManager.setLookAndFeel(info.getClassName());
81.                 break;
82.             }
83.         }
84.     } catch (ClassNotFoundException ex) {
85.         java.util.logging.Logger.getLogger(Client.class.getName()).log(java.util.lo
86.         gging.Level.SEVERE, null, ex);
87.     } catch (InstantiationException ex) {
88.         java.util.logging.Logger.getLogger(Client.class.getName()).log(java.util.lo
89.         gging.Level.SEVERE, null, ex);
90.     } catch (IllegalAccessException ex) {
91.         java.util.logging.Logger.getLogger(Client.class.getName()).log(java.util.lo
92.         gging.Level.SEVERE, null, ex);
93.     } catch (javax.swing.UnsupportedLookAndFeelException ex) {
94.         java.util.logging.Logger.getLogger(Client.class.getName()).log(java.util.lo
95.         gging.Level.SEVERE, null, ex);
96.     }
97.     //</editor-fold>
98.     //</editor-fold>
99.
100.    /* Create and display the form */
101.    java.awt.EventQueue.invokeLater(new Runnable() {
102.        public void run() {
103.            new Client().setVisible(true);
104.        }
105.    });
106. }
107.
108. // Variables declaration - do not modify
109. private javax.swing.JButton jButton1;
110. private javax.swing.JLabel jLabel1;
111. private javax.swing.JLabel jLabel2;
112. private javax.swing.JLabel jLabel3;
113. 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.
123.     String roomNos = Room_no.getSelectedItem().toString();
124.     int smokeLevels = Integer.parseInt(smoke_level.getText());
125.     String sensorStatus = status.getText();
126.
127.     try {
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 firelalarm.
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.                ));
155.                String roomNos = Room_no.getSelectedItem().toString();
156.                int smokeLevels = Integer.parseInt(smoke_level.getText());
157.                String sensorStatus = status.getText();
158.
159.                try{
160.
161.                    //Edit the fire alarm sensor details in the database
162.
163.                    Statement s = con.createStatement();
164.                    s.execute("update sensors set floorNo="+floorNos+",roomNo='"+roomNos+"',smokeLevel = "+smokeLevels+",status='"+sensorStatus+"' where id="+alarmId);
165.                    JOptionPane.showMessageDialog(this, "Records Are Updated!");
166.
167.                    setalarmTable();
168.                    resetData();
169.                    alarmId=0;
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
186.                //sensor should display on the text boxes and combo boxes
187.
188.                alarmId = Integer.parseInt(alarmTable.getValueAt(alarmTable.getSelectedRow(), 0).toString());
189.                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;

```

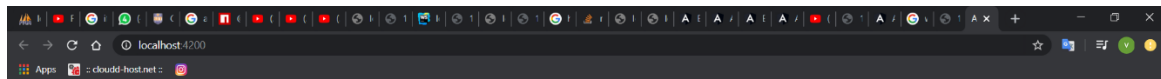
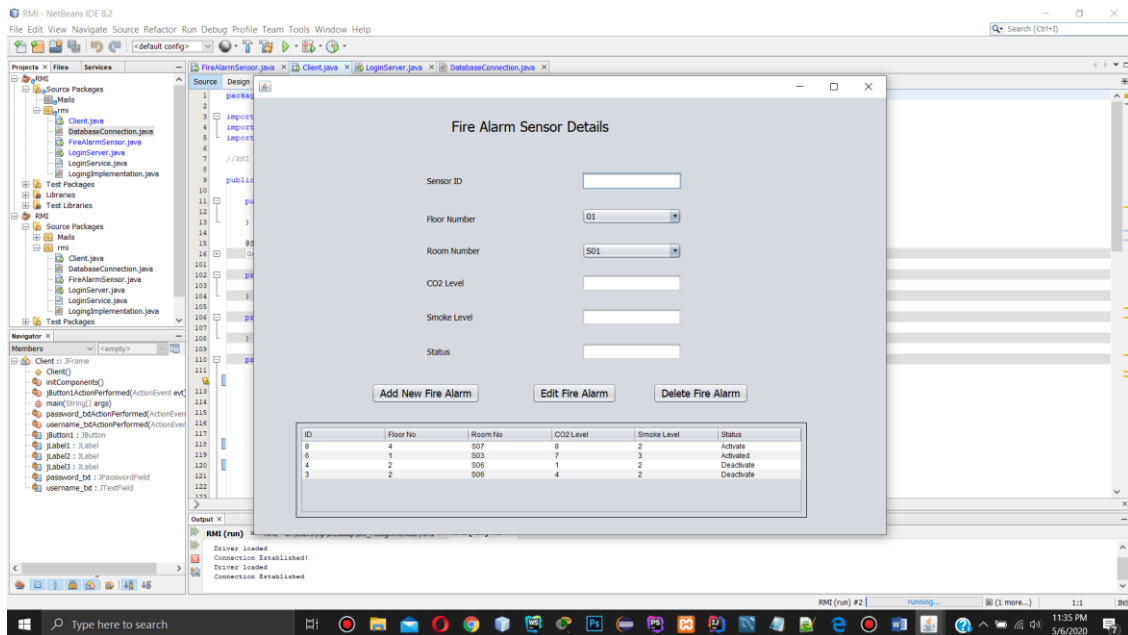
```

9. import javax.mail.internet.AddressException;
10. import javax.mail.internet.InternetAddress;
11. import javax.mail.internet.MimeMessage;
12.
13. /**
14.  *
15.  * @author hp
16.  */
17. public class Emails {
18.
19.     public static void main(String[] args) throws AddressException, MessagingException{
20.
21.         final String username = "*****@gmail.com";
22.         final String password = "*****";
23.
24.         Properties props = new Properties();
25.         props.put("mail.smtp.auth", "true");
26.         props.put("mail.smtp.starttls.enable", "true");
27.         props.put("mail.smtp.host", "smtp.gmail.com");
28.         props.put("mail.smtp.port", "587");
29.
30.         Session session = Session.getInstance(props, new javax.mail.Authenticator() {
31.
32.             protected PasswordAuthentication getPasswordAuthentication(){
33.                 return new PasswordAuthentication(username, password);
34.             }
35.         });
36.
37.
38.         try{
39.
40.             Message message = new MimeMessage(session);
41.             message.setFrom(new InternetAddress("*****@gmail.com"));
42.             message.setRecipients(Message.RecipientType.TO,
43.                 InternetAddress.parse("*****@gmail.com"));
44.             message.setSubject("FireSensor Alarm is active!!!");
45.
46.             Transport.send(message);
47.
48.             System.out.println("was the mail sent:Done");
49.
50.
51.         }
52.         catch(MessagingException e){
53.
54.             throw new RuntimeException(e);
55.             //ex.printStackTrace();
56.         }
57.
58.
59.     }
60.
61. }

```

IMPORTANT NOTE

We have missed adding of CO2 level when we do the project. We saw it later after uploading our video to the YouTube. So later we have done it again. Here are the codes and screen shots of the output.



Fire Alarm Sensor

Sensor ID	Floor Number	Room Number	CO2 Level	Smoke Level	Status
3	2	S06	4	2	Deactivate
4	2	S06	1	2	Deactivate
6	1	S03	7	3	Activated
8	4	S07	8	2	Activate



FireAlarmSensor.java

```
1. package rmi;
2.
3. import java.awt.Color;
4. import java.awt.Component;
5. import static java.lang.reflect.Array.set;
6. import java.sql.Connection;
7. import java.sql.DriverManager;
8. import java.sql.PreparedStatement;
9. import java.sql.ResultSet;
10. import java.sql.SQLException;
11. import java.sql.Statement;
12. import java.util.ArrayList;
13. import java.util.logging.Level;
14. import java.util.logging.Logger;
15. import javax.swing.JOptionPane;
16. import javax.swing.JTable;
17. import javax.swing.table.DefaultTableCellRenderer;
18. import javax.swing.table.DefaultTableModel;
19. import javax.swing.table.TableCellRenderer;
20. import javax.xml.registry.infomodel.User;
21. //import org.nit.instance.DtabaseConnection;
22.
23. public class FireAlarmSensor extends javax.swing.JFrame {
24.
25.     private Connection con;
26.     private int alarmId;
27.
28.
29.     public FireAlarmSensor() {
30.         initComponents();
31.
32.         try {
33.
34.             //Connection to the MySQL Database
35.
36.             Class.forName("com.mysql.jdbc.Driver");
37.             System.out.println("Driver loaded");
38.             con = DriverManager.getConnection("jdbc:mysql://localhost:3306/sensoralarm"
39. , "root", "");
40.             System.out.println("Connection Established!");
41.         } catch (Exception ex) {
42.
43.             System.out.println(ex);
44.
45.         }
46.
47.         DatabaseConnection db = DatabaseConnection.getDatabaseConnection();
48.         con = db.getConnection();
49.
50.         //Auto refresh the table in every 15 seconds
51.
52.         setalarmTable();
53.
54.         Thread t = new Thread(new Runnable() {
55.             @Override
56.             public void run() {
57.                 for(;;){
```

```

58.         try {
59.             int time = 15000;
60.             Thread.sleep(time);
61.             System.out.println("Refreshed in 15 seconds!");
62.             setalarmTable();
63.
64.         } catch (Exception ex) {
65.             ex.printStackTrace();
66.         }
67.     }
68. }
69.
70. });
71.
72.     t.start();
73.
74. }
75.
76.
77.
78. @SuppressWarnings("unchecked")
79. // <editor-
    fold defaultstate="collapsed" desc="Generated Code">
80.     private void initComponents() {
81.
82.         jPanel1 = new javax.swing.JPanel();
83.         jScrollPane1 = new javax.swing.JScrollPane();
84.         alarmTable = new javax.swing.JTable();
85.         jLabel1 = new javax.swing.JLabel();
86.         jLabel2 = new javax.swing.JLabel();
87.         jLabel3 = new javax.swing.JLabel();
88.         sensor_id = new javax.swing.JTextField();
89.         add = new javax.swing.JButton();
90.         edit = new javax.swing.JButton();
91.         delete = new javax.swing.JButton();
92.         jLabel4 = new javax.swing.JLabel();
93.         Floor_no = new javax.swing.JComboBox<>();
94.         Room_no = new javax.swing.JComboBox<>();
95.         jLabel5 = new javax.swing.JLabel();
96.         jLabel6 = new javax.swing.JLabel();
97.         jLabel7 = new javax.swing.JLabel();
98.         co2 = new javax.swing.JTextField();
99.         smoke_level = new javax.swing.JTextField();
100.        status = new javax.swing.JTextField();
101.
102.        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
103.
104.        jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.
t.Color(0, 0, 0)));
105.
106.        alarmTable.setModel(new javax.swing.table.DefaultTableModel(
107.            new Object [][] {
108.                {null, null, null, null},
109.                {null, null, null, null},
110.                {null, null, null, null},
111.                {null, null, null, null}
112.            },
113.            new String [] {
114.                "Title 1", "Title 2", "Title 3", "Title 4"
115.            }
116.        ));

```

```

117.         alarmTable.addMouseListener(new java.awt.event.MouseAdapter() {
118.             public void mouseClicked(java.awt.event.MouseEvent evt) {
119.                 alarmTableMouseClicked(evt);
120.             }
121.         });
122.         jScrollPane1.setViewportView(alarmTable);
123.
124.         javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPan
    e11);
125.         jPanel1.setLayout(jPanel1Layout);
126.         jPanel1Layout.setHorizontalGroup(
127.             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
128.                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.
    createSequentialGroup())
129.                 .addContainerGap()
130.                 .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE
    , 857, Short.MAX_VALUE)
131.                 .addContainerGap())
132.         );
133.         jPanel1Layout.setVerticalGroup(
134.             jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    LEADING)
135.                 .addGroup(jPanel1Layout.createSequentialGroup()
136.                     .addContainerGap()
137.                     .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE
    , 148, Short.MAX_VALUE)
138.                     .addContainerGap())
139.         );
140.
141.         jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
142.         jLabel1.setText("Sensor ID");
143.
144.         jLabel2.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
145.         jLabel2.setText("Floor Number");
146.
147.         jLabel3.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
148.         jLabel3.setText("Room Number");
149.
150.         sensor_id.setEditable(false);
151.         sensor_id.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
152.
153.         add.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
154.         add.setText("Add New Fire Alarm");
155.         add.addActionListener(new java.awt.event.ActionListener() {
156.             public void actionPerformed(java.awt.event.ActionEvent evt) {
157.                 addActionPerformed(evt);
158.             }
159.         });
160.
161.         edit.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
162.         edit.setText("Edit Fire Alarm");
163.         edit.addActionListener(new java.awt.event.ActionListener() {
164.             public void actionPerformed(java.awt.event.ActionEvent evt) {
165.                 editActionPerformed(evt);
166.             }
167.         });
168.
169.         delete.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
170.         delete.setText("Delete Fire Alarm");
171.         delete.addActionListener(new java.awt.event.ActionListener() {

```

```

172.         public void actionPerformed(java.awt.event.ActionEvent evt) {
173.             deleteActionPerformed(evt);
174.         }
175.     });
176.
177.     jLabel4.setFont(new java.awt.Font("Tahoma", 0, 24)); // NOI18N
178.     jLabel4.setText("Fire Alarm Sensor Details");
179.
180.     Floor_no.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
181.     Floor_no.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] {
182.         "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "14", "15" }));
183.
184.     Room_no.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
185.     Room_no.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "
186.         S01", "S02", "S03", "S05", "S06", "S07", "S08", "S09", "S10" }));
187.
188.     jLabel5.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
189.     jLabel5.setText("CO2 Level");
190.
191.     jLabel6.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
192.     jLabel6.setText("Smoke Level");
193.
194.     jLabel7.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
195.     jLabel7.setText("Status");
196.
197.     co2.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
198.
199.     smoke_level.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
200.
201.     status.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
202.
203.     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane()
204.         .getContentPane().setLayout(layout);
205.         layout.setHorizontalGroup(
206.             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
207.                 .addGroup(layout.createSequentialGroup()
208.                     .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
209.                         .addGroup(layout.createSequentialGroup()
210.                             .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
211.                                 .add(layout.createSequentialGroup()
212.                                     .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
213.                                         .add(layout.createSequentialGroup()
214.                                             .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
215.                                                 .add(layout.createSequentialGroup()
216.                                                     .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
217.                                                         .add(layout.createSequentialGroup()
218.                                                             .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
219.                                                                 .add(layout.createSequentialGroup()
220.                                                                     .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
221.                                                                         .add(layout.createSequentialGroup()
222.

```

```

223.                .addComponent(edit)
224.                .addGap(63, 63, 63)
225.                .addComponent(delete))
226.                .addGroup(layout.createSequentialGroup())
227.                .addGap(168, 168, 168)
228.                .addGroup(layout.createParallelGroup(javax.swing
    .GroupLayout.Alignment.LEADING, false)
229.                .addComponent(sensor_id, javax.swing.GroupLa
    yout.DEFAULT_SIZE, 169, Short.MAX_VALUE)
230.                .addComponent(Floor_no, 0, javax.swing.Group
    Layout.DEFAULT_SIZE, Short.MAX_VALUE)
231.                .addComponent(Room_no, 0, javax.swing.GroupL
    ayout.DEFAULT_SIZE, Short.MAX_VALUE)
232.                .addComponent(co2)
233.                .addComponent(smoke_level)
234.                .addComponent(status))))))
235.                .addGroup(layout.createSequentialGroup())
236.                .addGap(337, 337, 337)
237.                .addComponent(jLabel4))
238.                .addGroup(layout.createSequentialGroup())
239.                .addGap(72, 72, 72)
240.                .addComponent(jPanel11, javax.swing.GroupLayout.PREFERRED
    _SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
241.                .addContainerGap(123, Short.MAX_VALUE))
242.            );
243.            layout.setVerticalGroup(
244.                layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING
    )
245.                .addGroup(layout.createSequentialGroup())
246.                .addGap(34, 34, 34)
247.                .addComponent(jLabel4)
248.                .addGap(64, 64, 64)
249.                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
    gnment.BASELINE)
250.                .addComponent(jLabel11)
251.                .addComponent(sensor_id, javax.swing.GroupLayout.PREFERRED_S
    IZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
252.                .addGap(32, 32, 32)
253.                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
    gnment.TRAILING)
254.                .addComponent(jLabel12)
255.                .addComponent(Floor_no, javax.swing.GroupLayout.PREFERRED_SI
    ZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
256.                .addGap(32, 32, 32)
257.                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
    gnment.BASELINE)
258.                .addComponent(jLabel13)
259.                .addComponent(Room_no, javax.swing.GroupLayout.PREFERRED_SIZ
    E, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
260.                .addGap(27, 27, 27)
261.                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
    gnment.BASELINE)
262.                .addComponent(jLabel15)
263.                .addComponent(co2, javax.swing.GroupLayout.PREFERRED_SIZE, j
    avax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
264.                .addGap(29, 29, 29)
265.                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
    gnment.BASELINE)
266.                .addComponent(jLabel16)

```

```

267.                .addComponent(smoke_level, javax.swing.GroupLayout.PREFERRED
_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
268.                .addGap(30, 30, 30)
269.                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.BASELINE)
270.                    .addComponent(jLabel17)
271.                    .addComponent(status, javax.swing.GroupLayout.PREFERRED_SIZE
, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
272.                .addGap(39, 39, 39)
273.                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Ali
gnment.BASELINE)
274.                    .addComponent(add)
275.                    .addComponent(edit)
276.                    .addComponent(delete))
277.                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELA
TED, 54, Short.MAX_VALUE)
278.                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, j
avax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
279.                .addGap(29, 29, 29))
280.            );
281.
282.            setSize(new java.awt.Dimension(1096, 791));
283.            setLocationRelativeTo(null);
284.        }// </editor-fold>
285.
286.        private void deleteActionPerformed(java.awt.event.ActionEvent evt) {
287.
288.            if(alarmId!=0){
289.
290.                try{
291.
292.                    //Delete the firealarm sensor details from the database when sel
ected from the table by Id
293.
294.                    Statement s = con.createStatement();
295.                    s.execute("delete from sensor where id="+alarmId);
296.                    JOptionPane.showMessageDialog(this, "Record Are Deleted!");
297.
298.                    setalarmTable();
299.                    resetData();
300.                    alarmId=0;
301.
302.                }
303.                catch(Exception ex){
304.
305.                    JOptionPane.showMessageDialog(this, "Can't Delete Records!");
306.                }
307.            }
308.
309.
310.        }
311.
312.        private void addActionPerformed(java.awt.event.ActionEvent evt) {
313.
314.            int floorNos = Integer.parseInt(Floor_no.getSelectedItem().toString());
315.
316.            String roomNos = Room_no.getSelectedItem().toString();
317.            int co2level = Integer.parseInt(co2.getText());
318.            int smokeLevels = Integer.parseInt(smoke_level.getText());

```

```

318.         String sensorStatus = status.getText();
319.         //         int smokeLevels = Integer.parseInt(smoke_level.getText());
320.
321.         try {
322.
323.             //Adding new fire alarm sensors to the database.
324.             //Admin need to provide the floor no (Integer value), room no(String
value), smoke level(Integer value) and status(String value) of the firealarms.
325.             //No need to provide Sensor ID as it is autogenerated.
326.
327.             Statement s = con.createStatement();
328.             s.execute("INSERT INTO sensor(floorNo,roomNo,co2,smokeLevel,status)
values(" + floorNos + "','" + roomNos + "','" + co2level + "','" + smokeLevels + "','" + sensorS
tatus + "'");
329.
330.             JOptionPane.showMessageDialog(this, "Records Are Submitted!");
331.
332.             setalarmTable();
333.             s.close();
334.             resetData();
335.
336.         } catch (Exception ex) {
337.
338.             JOptionPane.showMessageDialog(this, ex);
339.
340.         }
341.
342.     }
343.
344.     private void editActionPerformed(java.awt.event.ActionEvent evt) {
345.
346.         if (alarmId != 0) {
347.
348.             int floorNos = Integer.parseInt(Floor_no.getSelectedItem().toString(
));
349.             String roomNos = Room_no.getSelectedItem().toString();
350.             int co2level = Integer.parseInt(co2.getText());
351.             int smokeLevels = Integer.parseInt(smoke_level.getText());
352.             String sensorStatus = status.getText();
353.
354.             try{
355.
356.                 //Edit the fire alarm sensor details in the database
357.
358.                 Statement s = con.createStatement();
359.                 s.execute("update sensor set floorNo="+floorNos+",roomNo='"+room
Nos+"',co2 = "+co2level+",smokeLevel = "+smokeLevels+",status='"+sensorStatus+"' where
id="+alarmId);
360.                 JOptionPane.showMessageDialog(this, "Records Are Updated!");
361.
362.                 setalarmTable();
363.                 resetData();
364.                 alarmId=0;
365.
366.             }
367.             catch(Exception ex){
368.
369.                 JOptionPane.showMessageDialog(this, "Can't Update Records!");
370.
371.             }

```

```

372.
373.         }
374.
375.
376.     }
377.
378.     private void alarmTableMouseClicked(java.awt.event.MouseEvent evt) {
379.
380.         try {
381.
382.             //When selected a row from the table, the details of the fire alarm
383.             sensor should display on the text boxes and combo boxes
384.             alarmId = Integer.parseInt(alarmTable.getValueAt(alarmTable.getSelectedRow(), 0).toString());
385.             Statement s = con.createStatement();
386.             ResultSet rs = s.executeQuery("select * from sensor where id=" + alarmId);
387.
388.             if (rs.next()) {
389.
390.                 sensor_id.setText(rs.getInt(1) + "");
391.                 Floor_no.setSelectedItem(rs.getString(2));
392.                 Room_no.setSelectedItem(rs.getString(3));
393.                 co2.setText(rs.getInt(4) + "");
394.                 smoke_level.setText(rs.getString(5) + "");
395.                 status.setText(rs.getString(6));
396.
397.
398.                 int smokeLevels = Integer.parseInt(smoke_level.getText());
399.                 int co2level = Integer.parseInt(co2.getText());
400.
401.                 if(smokeLevels>5 || co2level>5){
402.                     // if(smokeLevels>5){
403.
404.                         //When the smoke level is greater than or equal to 5, then the status textfield should appear in red colour
405.
406.                         status.setBackground(Color.red);
407.
408.                     }
409.
410.                     // else if(co2level>5){
411.                     //
412.                         status.setBackground(Color.red);
413.                     //
414.                     }
415.                 else{
416.
417.                     //When the smoke level is less than 5, then the status textfield should appear in green colour
418.
419.                     status.setBackground(Color.GREEN);
420.
421.                 }
422.
423.             }
424.
425.             rs.close();
426.             s.close();

```



```

427.
428.         } catch (Exception ex) {
429.
430.             JOptionPane.showMessageDialog(this, ex);
431.         }
432.
433.     }
434.
435.     /**
436.      * @param args the command line arguments
437.      */
438.     public static void main(String args[]) {
439.         /* Set the Nimbus look and feel */
440.         //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
441.         /* If Nimbus (introduced in Java SE 6) is not available, stay with the d
442.         efault look and feel.
443.          * For details see http://download.oracle.com/javase/tutorial/uiswing/lo
444.         okandfeel/plaf.html
445.          */
446.         try {
447.             for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIMana
448.             ger.getInstalledLookAndFeels()) {
449.                 if ("Nimbus".equals(info.getName())) {
450.                     javax.swing.UIManager.setLookAndFeel(info.getClassName());
451.                     break;
452.                 }
453.             } catch (ClassNotFoundException ex) {
454.                 java.util.logging.Logger.getLogger(FireAlarmSensor.class.getName()).
455.                 log(java.util.logging.Level.SEVERE, null, ex);
456.             } catch (InstantiationException ex) {
457.                 java.util.logging.Logger.getLogger(FireAlarmSensor.class.getName()).
458.                 log(java.util.logging.Level.SEVERE, null, ex);
459.             } catch (IllegalAccessException ex) {
460.                 java.util.logging.Logger.getLogger(FireAlarmSensor.class.getName()).
461.                 log(java.util.logging.Level.SEVERE, null, ex);
462.             } catch (javax.swing.UnsupportedLookAndFeelException ex) {
463.                 java.util.logging.Logger.getLogger(FireAlarmSensor.class.getName()).
464.                 log(java.util.logging.Level.SEVERE, null, ex);
465.             }
466.         } //</editor-fold>
467.
468.         /* Create and display the form */
469.         java.awt.EventQueue.invokeLater(new Runnable() {
470.             public void run() {
471.                 new FireAlarmSensor().setVisible(true);
472.             }
473.         });
474.
475.     }
476.
477.     // Variables declaration - do not modify
478.     private javax.swing.JComboBox<String> Floor_no;
479.     private javax.swing.JComboBox<String> Room_no;
480.     private javax.swing.JButton add;
481.     private javax.swing.JTable alarmTable;
482.     private javax.swing.JTextField co2;
483.     private javax.swing.JButton delete;
484.     private javax.swing.JButton edit;
485.     private javax.swing.JLabel jLabel1;
486.     private javax.swing.JLabel jLabel2;

```

```

480.         private javax.swing.JLabel jLabel3;
481.         private javax.swing.JLabel jLabel4;
482.         private javax.swing.JLabel jLabel5;
483.         private javax.swing.JLabel jLabel6;
484.         private javax.swing.JLabel jLabel7;
485.         private javax.swing.JPanel jPanel1;
486.         private javax.swing.JScrollPane jScrollPane1;
487.         private javax.swing.JTextField sensor_id;
488.         private javax.swing.JTextField smoke_level;
489.         private javax.swing.JTextField status;
490.         // End of variables declaration
491.
492.         private void resetData() {
493.
494.             sensor_id.setText("");
495.             // floor_no.setText("");
496.             // room_no.setText("");
497.             co2.setText("");
498.             smoke_level.setText("");
499.             status.setText("");
500.
501.         }
502.
503.         private void setalarmTable() {
504.
505.             try {
506.
507.                 int rows = 0;
508.                 int rowIndex = 0;
509.
510.                 Statement s = con.createStatement();
511.                 ResultSet rs = s.executeQuery("select * from sensor order by id desc
512. ");
513.
514.                 if (rs.next()) {
515.                     rs.last();
516.                     rows = rs.getRow();
517.                     rs.beforeFirst();
518.                 }
519.
520.                 String[][] data = new String[rows][6];
521.
522.                 while (rs.next()) {
523.
524.                     data[rowIndex][0] = rs.getInt(1) + "";
525.                     data[rowIndex][1] = rs.getInt(2) + "";
526.                     data[rowIndex][2] = rs.getString(3);
527.                     data[rowIndex][3] = rs.getInt(4) + "";
528.                     data[rowIndex][4] = rs.getInt(5) + "";
529.                     data[rowIndex][5] = rs.getString(6);
530.
531.                     rowIndex++;
532.                 }
533.
534.
535.                 String[] cols = {"ID", "Floor No", "Room No", "CO2 Level", "Smoke Le
536. vel", "Status"};
537.                 DefaultTableModel model = new DefaultTableModel(data, cols);
538.                 alarmTable.setModel(model);

```

```

539.
540.         rs.close();
541.         s.close();
542.
543.     } catch (Exception ex) {
544.
545.         JOptionPane.showMessageDialog(this, "Can't Retrieve Data!");
546.
547.     }
548.
549. }
550.
551.
552.
553. }
```

DatabaseConnection.java

We have created a separate database for this.

```

1. package rmi;
2.
3. import com.mysql.jdbc.Connection;
4. import static java.lang.Class.forName;
5. import java.sql.DriverManager;
6. import java.util.logging.Level;
7. import java.util.logging.Logger;
8.
9.
10. public class DatabaseConnection {
11.
12.     private Connection con;
13.     private static DatabaseConnection db;
14.
15.     private DatabaseConnection(){
16.
17.     try {
18.
19.         //Connecting to the MySQL Database
20.
21.         Class.forName("com.mysql.jdbc.Driver");
22.         System.out.println("Driver loaded");
23.         con = (Connection) DriverManager.getConnection("jdbc:mysql://localhost:3306/sens
24.         oralarm","root","");
25.         System.out.println("Connection Established");
26.     }
27.     catch (Exception ex) {
28.
29.         System.out.println(ex);
30.     }
31. }
```

```

32.  }
33.
34.  public static DatabaseConnection getDatabaseConnection(){
35.
36.      if(db == null){
37.          db = new DatabaseConnection();
38.      }
39.      return db;
40.  }
41.
42.  public Connection getConnection(){
43.      return con;
44.  }
45.
46.
47. }

```

WEB CLIENT

user.component.ts

```

1.  import { Component, OnInit } from '@angular/core';
2.  import {HttpClient} from '@angular/common/http';
3.  import {delay} from "rxjs/operators";
4.  import {of} from "rxjs";
5.
6.  @Component({
7.      selector: 'app-user',
8.      templateUrl: './user.component.html',
9.      styleUrls: ['./user.component.css']
10. })
11. export class UserComponent implements OnInit {
12.
13.     sensor: any;
14.
15.     constructor(private http:HttpClient) { }
16.
17.     //Access data from the API to the web client
18.
19.     ngOnInit() {
20.         let resp = this.http.get("http://127.0.0.3:3001/sensor");
21.         resp.subscribe((data) => this.sensor=data);
22.     }
23.
24. }

```

user.component.html

```
1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4.   <meta charset="utf-8">
5.   <meta name="viewport" content="width=device-width, initial-scale=1">
6.   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/boot
strap.min.css">
7.   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></scri
pt>
8.   <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.j
s"></script>
9.   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></s
cript>
10. </head>
11. <body>
12.
13. <div class="container">
14.   <table class="table">
15.
16.     <!--Display data of sensoralarm sensor-->
17.
18.     <thead>
19.       <tr>
20.         <th>Sensor ID</th>
21.         <th>Floor Number</th>
22.         <th>Room Number</th>
23.         <th>CO2 Level</th>
24.         <th>Smoke Level</th>
25.         <th>Status</th>
26.       </tr>
27.     </thead>
28.     <tbody>
29.       <tr *ngFor="let user of sensor">
30.         <td>{{user.id}}</td>
31.         <td>{{user.floorNo}}</td>
32.         <td>{{user.roomNo}}</td>
33.         <td>{{user.co2}}</td>
34.         <td>{{user.smokeLevel}}</td>
35.         <td>{{user.status}}</td>
36.       </tr>
37.     </tbody>
38.   </table>
39. </div>
40.
41. </body>
42. </html>
```

REST API

Restapi.js

```
1. var http = require("http");
2. var express = require('express');
3. var app = express();
4. var mysql = require('mysql');
5. var bodyParser = require('body-parser');
6. var cors = require('cors')
7.
8. var app = express();
9.
10.
11. //Enable the access of the database from different domain
12.
13. app.use(cors({
14.
15.     origin: "*"
16. }));
17. });
18.
19.
20. //Connect to MySQL
21. var connection = mysql.createConnection({
22.     host: 'localhost',
23.     user: 'root',
24.     password: '',
25.     database: 'sensoralarm'
26. });
27.
28.
29. connection.connect(function(err) {
30.     if (err) throw err
31.     console.log('Connected with mysql database')
32. })
33.
34.
35. //Body parser configuration
36. app.use( bodyParser.json() );           // to support JSON-encoded bodies
37. app.use(bodyParser.urlencoded({         // to support URL-encoded bodies
38.     extended: true
39. }));
40.
41.
42. //App server
43. const server = app.listen(3001, "127.0.0.3", function () {
44.
45.     var host = server.address().address
46.     var port = server.address().port
47.
48.     console.log("Example app listening at http://%s:%s", host, port)
49.
50. });
51.
52. //REST API - fetch the data using GET method
53. app.get('/sensor', function (req, res) {
54.     connection.query('select * from sensor', function (error, results, fields) {
55.         if (error) throw error;
```

```
56.         res.end(JSON.stringify(results));  
57.     });  
58. });
```

THANK YOU