Sri Lanka Institute of Information Technology

Assignment 2

FIRE ALARM MONITORING SYSTEM

Y3S2.09(WE)

Distributed System

B.Sc. (Hons) in Computer Science and Software Engineering

BSc (Hons) in Computer Science and Software Engineering – Year 3

Assignment 2 (DS)

Semester 1, 2020

CONTENT

BSc (Hons) in Computer Science and Software Engineering– Year 3

Assignment 2 (DS)

Semester 1, 2020

Introduction

The Fire Alarm Monitoring System is implemented using technologies such as Java Spring Boot as Rest API architecture and Java Swing for RMI client and desktop Client, and ReactJS for Web Client, and MySQL for database.

In this system Desktop client has an admin login, through the that admin login admin can login to the system. Then he can register some sensor details like sensor floor Number and sensor room number [Location]. When the smoke lever or CO2 level getting more than five admin will be informed. Then he will send the email notification or SMS notification.

Web client application where users can view the status of all fire alarm sensors. For each sensor, details of specific sensor location, smoke level, CO2 level, whether the fire alarm sensor is active [When the smoke lever or CO2 level getting more than five] , the alert message will be received to admin. This system is implemented dummy sensor application, dummy SMS and email applications.

1.0 High Level Architectural Diagram

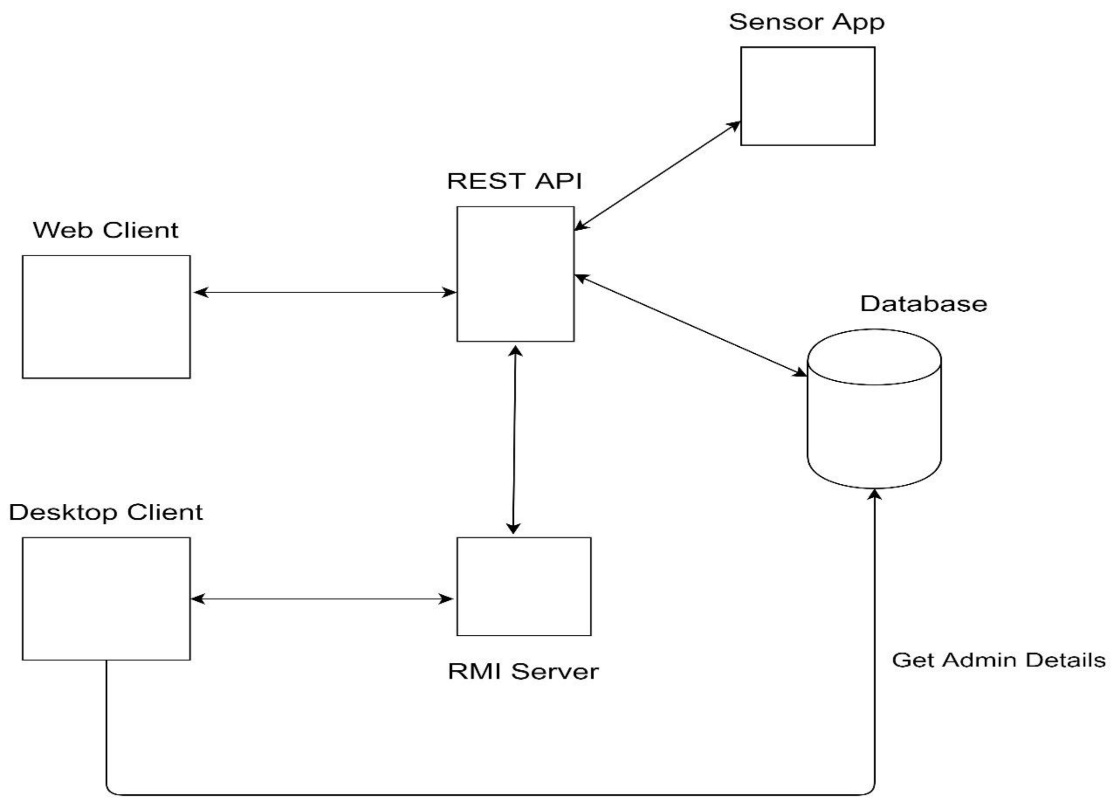


Figure 1 – The Fire Alarm Monitoring System Architecture

The Fire Alarm System is implemented above architectural style. Other components are connected to the RESTful web services API.

This System has main 7 Components as follows:

1. Web Client
2. RESTful Web Services API
3. RMI Server & Desktop Client
4. Fire Alarm Sensor
5. SMS Service

6. Email Service
7. Database

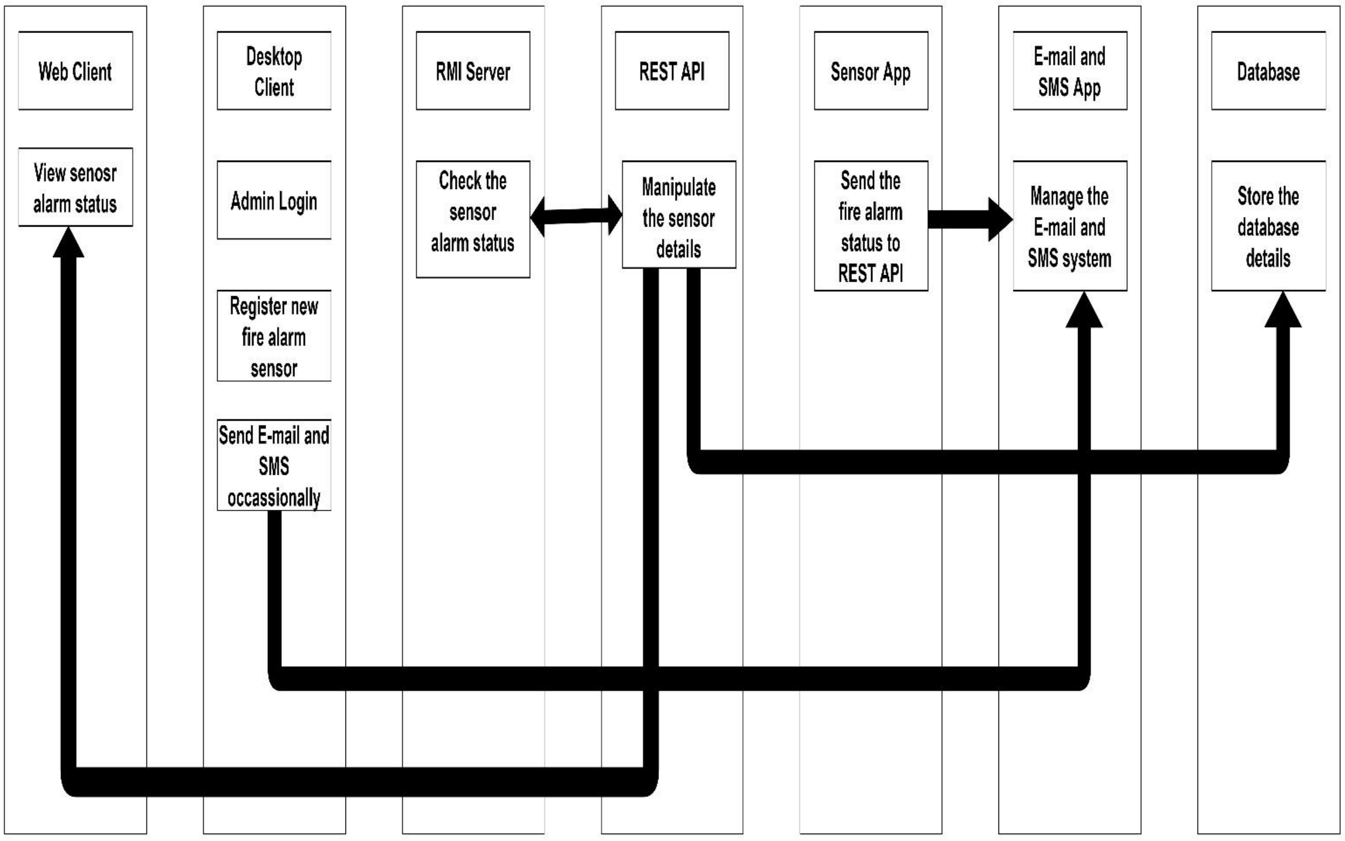

Web Client
View senosr alarm status
Desktop Client
Admin Login
Register new fire alarm sensor
Send E-mail and SMS occassionally
RMI Server
Check the sensor alarm status
REST API
Manipulate the sensor details
Sensor App
Send the fire alarm status to REST API
E-mail and SMS App
Manage the E-mail and SMS system
Database
Store the database details

Figure 2 - Connection between the components

2.0 Workflow of the Fire alarm system

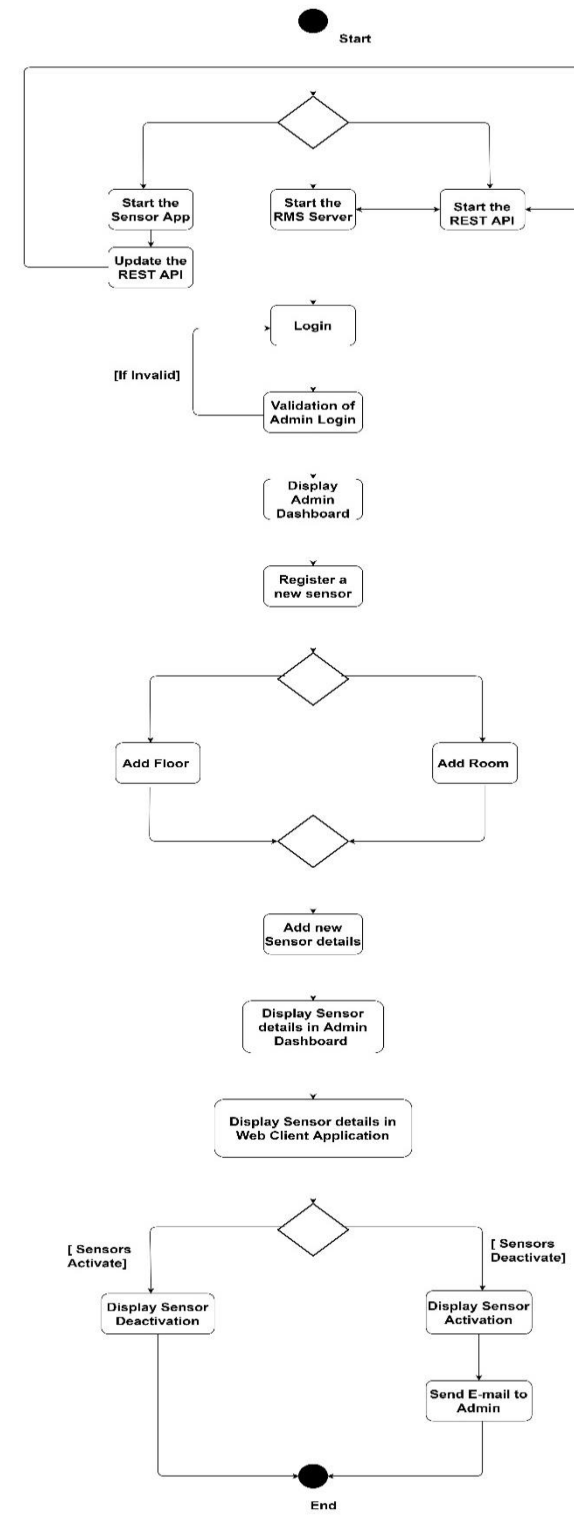
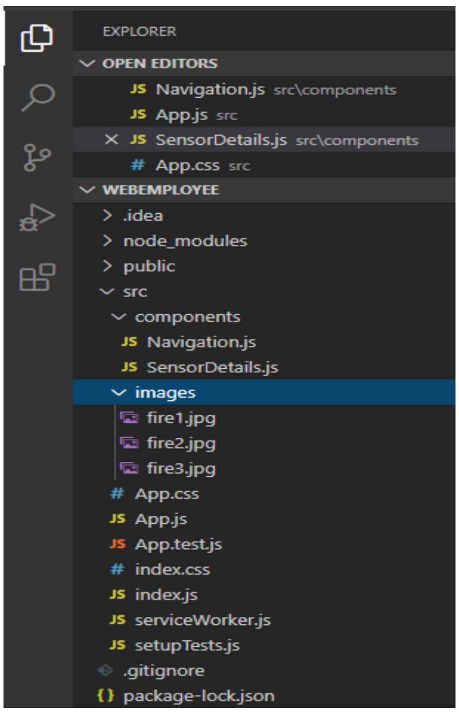


Figure 3- workflow of fire alarm monitoring system

3.0 System workflow – Interfaces

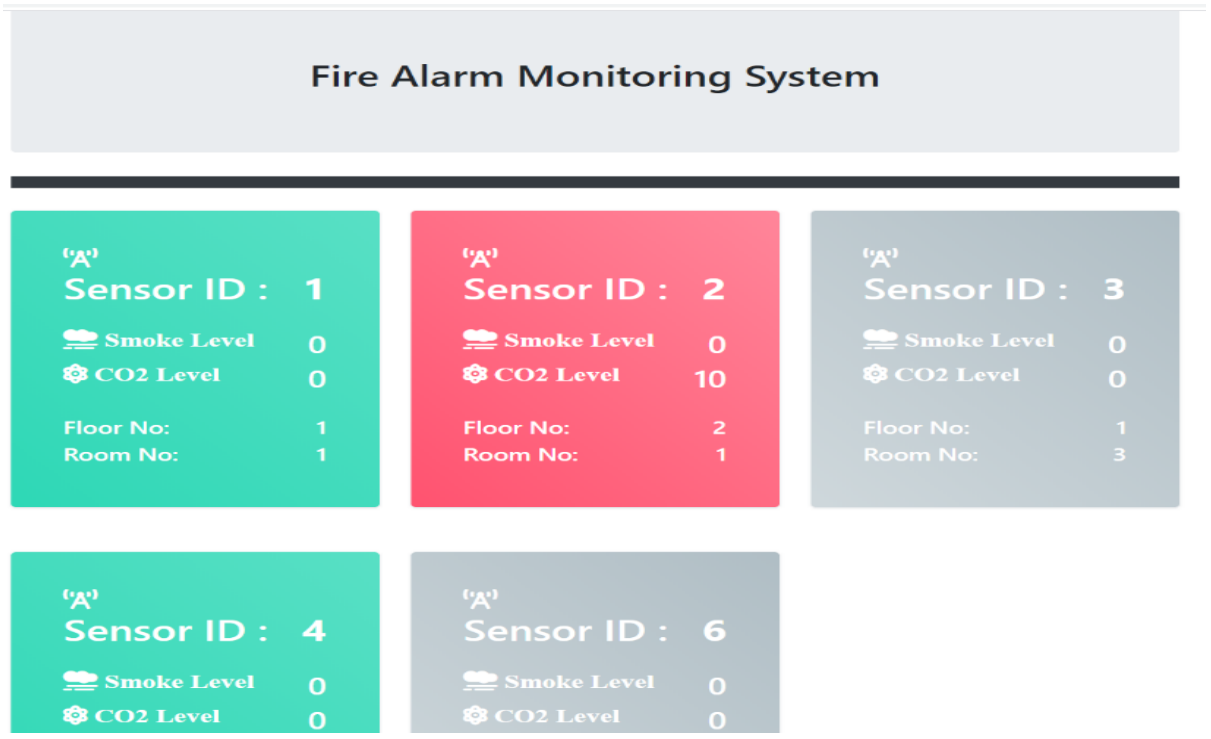
3.1 This is the folder structure of webclient interface.



```
EXPLORER
OPEN EDITORS
    Navigation.js src\components
    App.js src
  × SensorDetails.js src\components
    App.css src
WEBEMPLOYEE
  > .idea
  > node_modules
  > public
  ∨ src
    ∨ components
      Navigation.js
      SensorDetails.js
    ∨ images
      fire1.jpg
      fire2.jpg
      fire3.jpg
    App.css
    App.js
    App.test.js
    index.css
    index.js
    serviceWorker.js
    setupTests.js
  .gitignore
  package-lock.json
```

Figure 4 – folder structure of webclient

3.1.1 This is the webclient interface which is appear to the user. User can view sensor details and activation of the sensor. These details will update every 40 sec.



Fire Alarm Monitoring System

Sensor ID	Smoke Level	CO2 Level	Floor No:	Room No:
1	0	0	1	1
2	0	10	2	1
3	0	0	1	3
4	0	0		
6	0	0		

Figure 5 – Interface of webclient

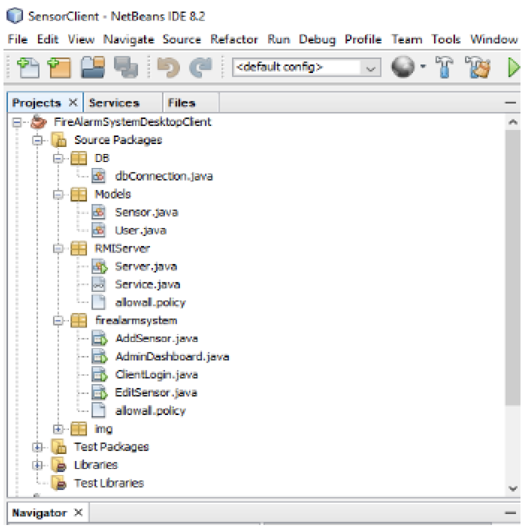
The Color coding of the cards are as follows,

• Green - CO2 and Smoke level under level 5

• Red - CO2 or Smoke or both levels have exceeded level 5

• Grey – Fire Alarm Sensor is Not Active

3.2 This is the desktop client folder structure.



```
SensorClient - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window
Projects  Services  Files
FireAlarmSystemDesktopClient
  Source Packages
    DB
      dbConnection.java
    Models
      Sensor.java
      User.java
    RMIServer
      Server.java
      Service.java
      allowall.policy
    firealarmsystem
      AddSensor.java
      AdminDashboard.java
      ClientLogin.java
      EditSensor.java
      allowall.policy
    img
  Test Packages
  Libraries
  Test Libraries
Navigator
```

3.2.1 This is the desktop client interface. This interface is only visible to the admin. First admin can login to the system. Admin can add a new sensor to the system. If necessary, admin can search, edit or remove sensor details. Sensor details will appear to the admin with the sensor activation status.

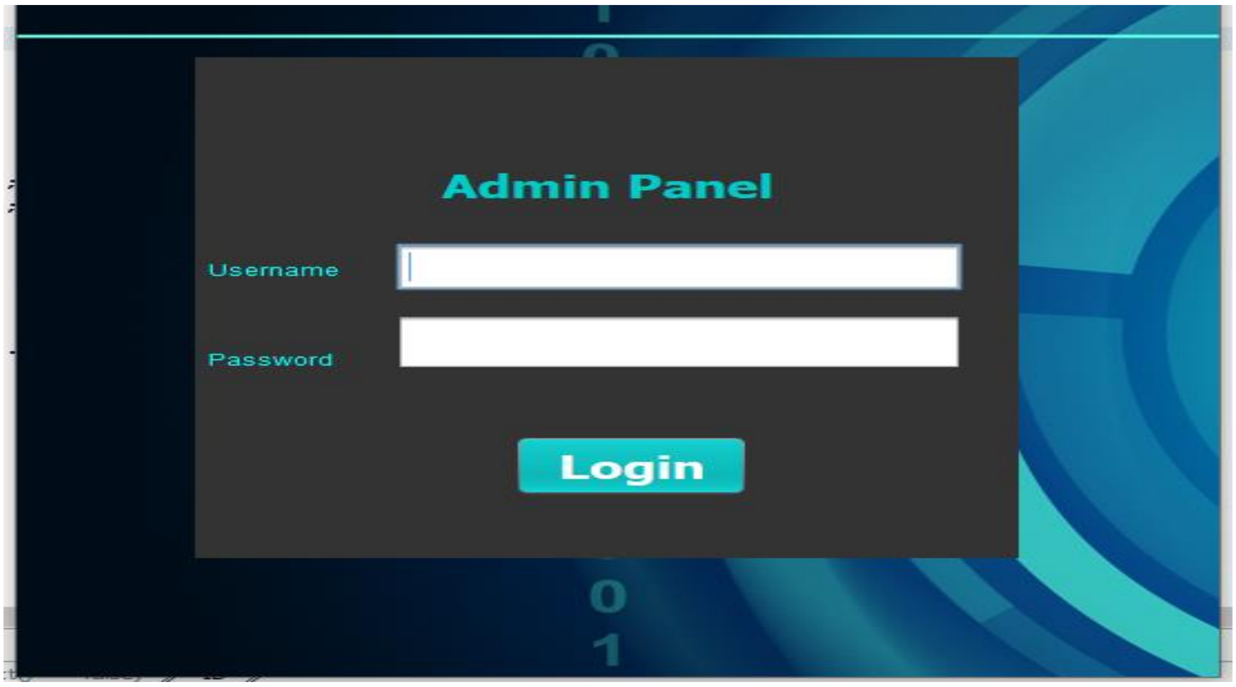


Figure 6 Admin Login

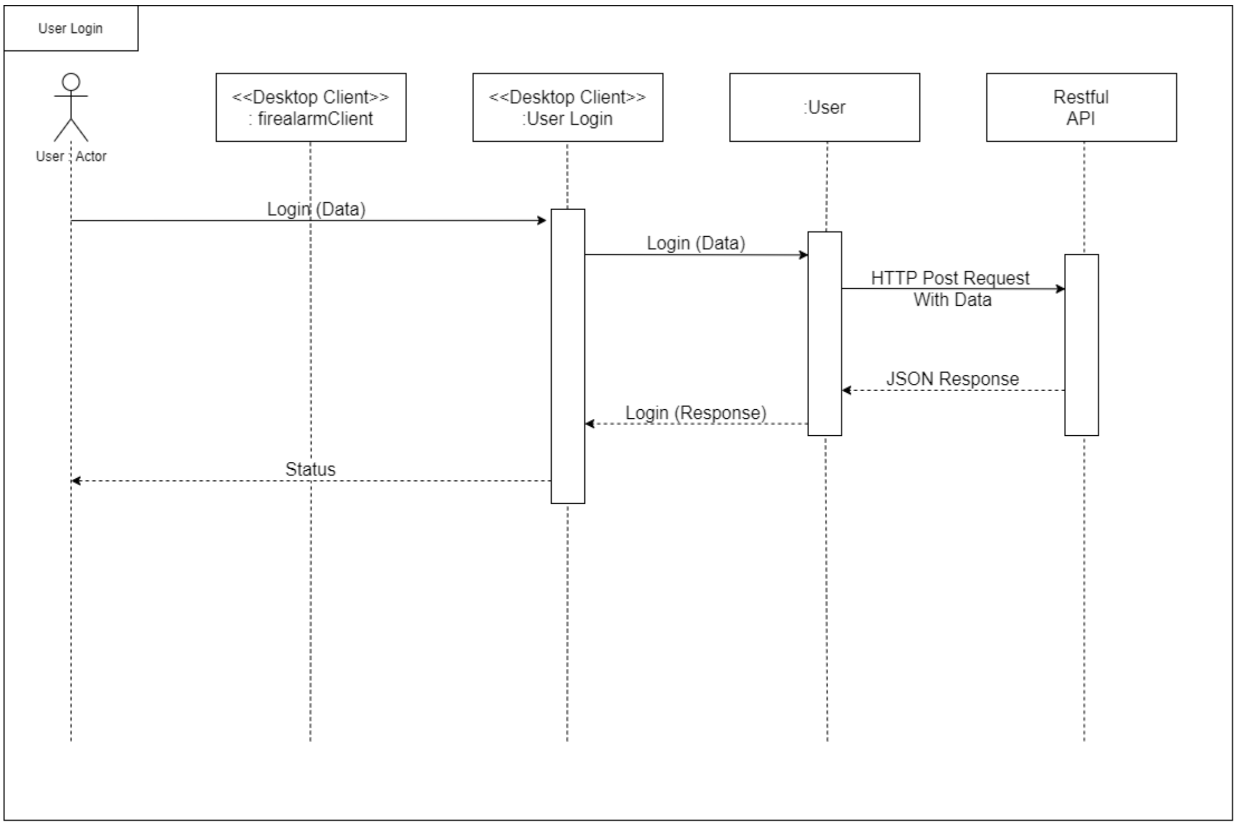


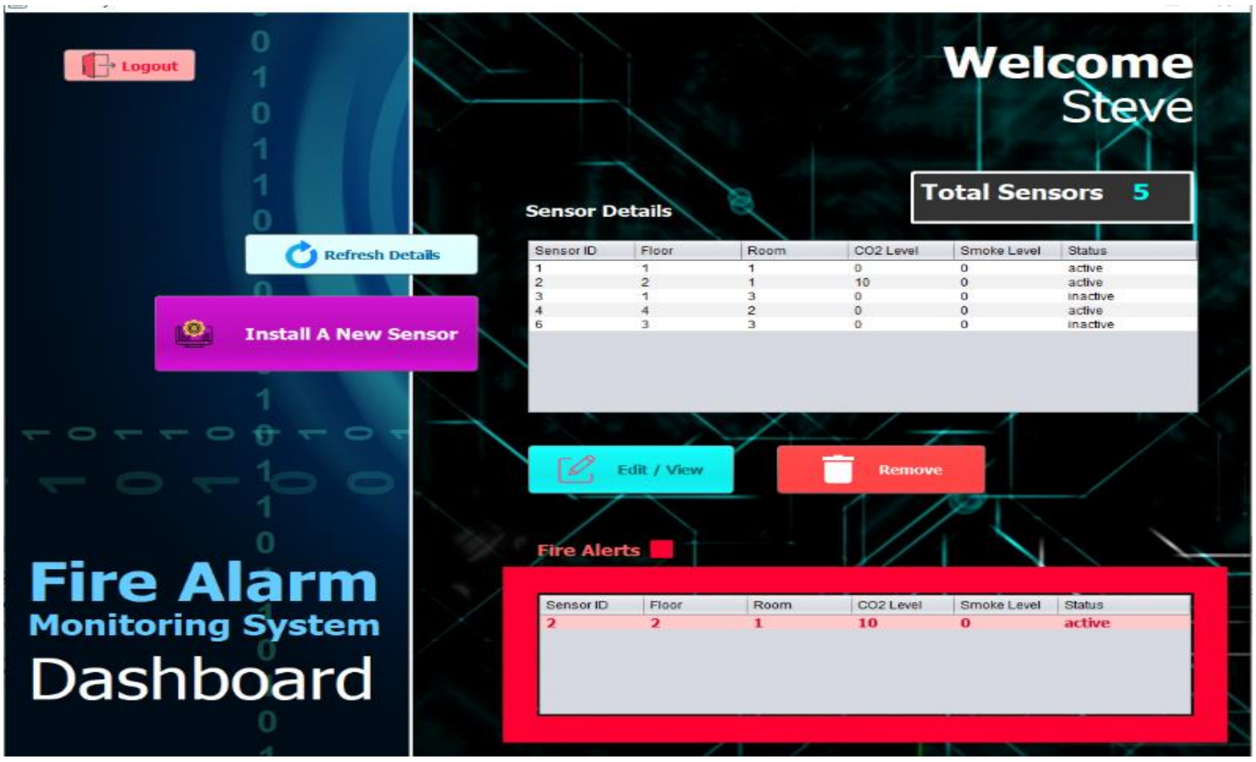
Figure 7 sequence diagram for login

Desktop Client

Fire Alarm System, desktop client is provided 5 major actions to the system. connection between REST API is needed to complete the below actions,

1. Add Fire Alarm Sensors
2. Edit Fire Alarm Sensors
3. Delete Fire Alarm Sensors
4. Update Sensor Data of Fire Alarm Sensors
5. Get Sensor Data of Fire Alarm Sensors

The number of sensors is displayed top of the sensor details. And especially fire alerts are figured into separate table .other than that admin can refresh the sensor details.



Total Sensors 5

Sensor Details

Sensor ID	Floor	Room	CO2 Level	Smoke Level	Status
1	1	1	0	0	active
2	2	1	10	0	active
3	1	3	0	0	inactive
4	4	2	0	0	active
6	3	3	0	0	inactive

Fire Alerts

Sensor ID	Floor	Room	CO2 Level	Smoke Level	Status
2	2	1	10	0	active

Figure 8 admin dashboard

- As mention above admin can be added to a sensor details as floor number of the sensor and room number of the sensor.

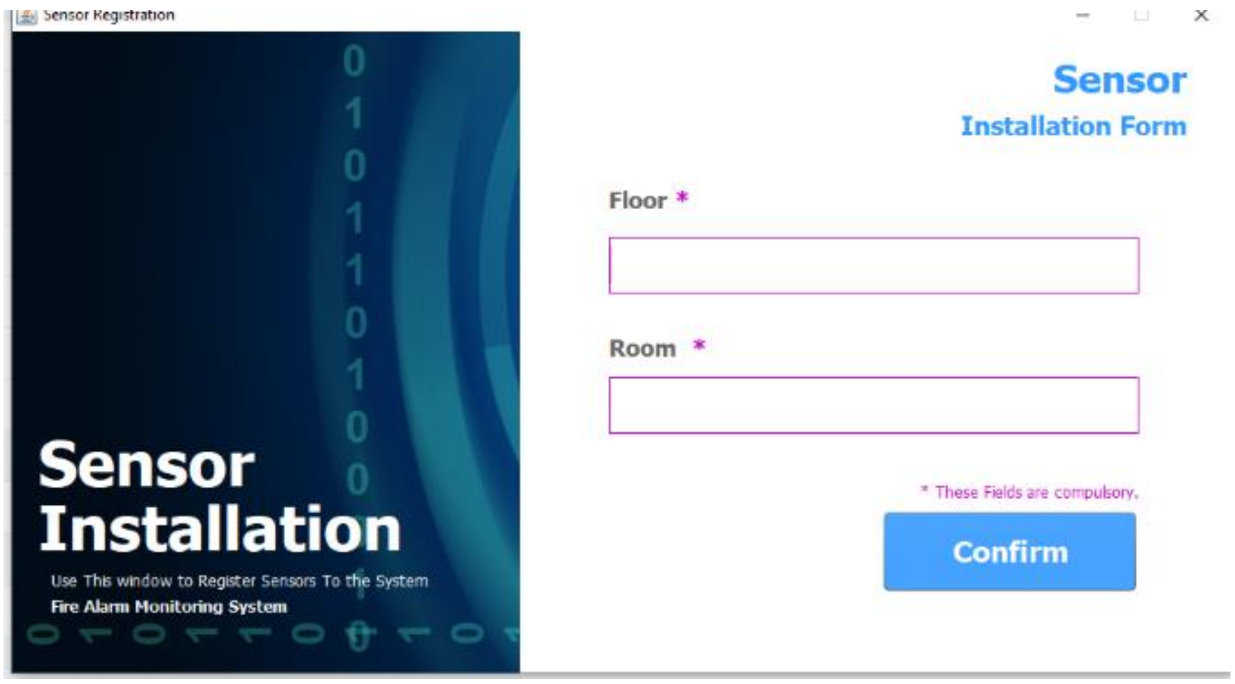

Sensor Registration
Sensor
Installation
Use This window to Register Sensors To the System
Fire Alarm Monitoring System
Sensor
Installation Form
Floor *
Room *
* These Fields are compulsory.
Confirm

Figure 9 Interface of add sensor details

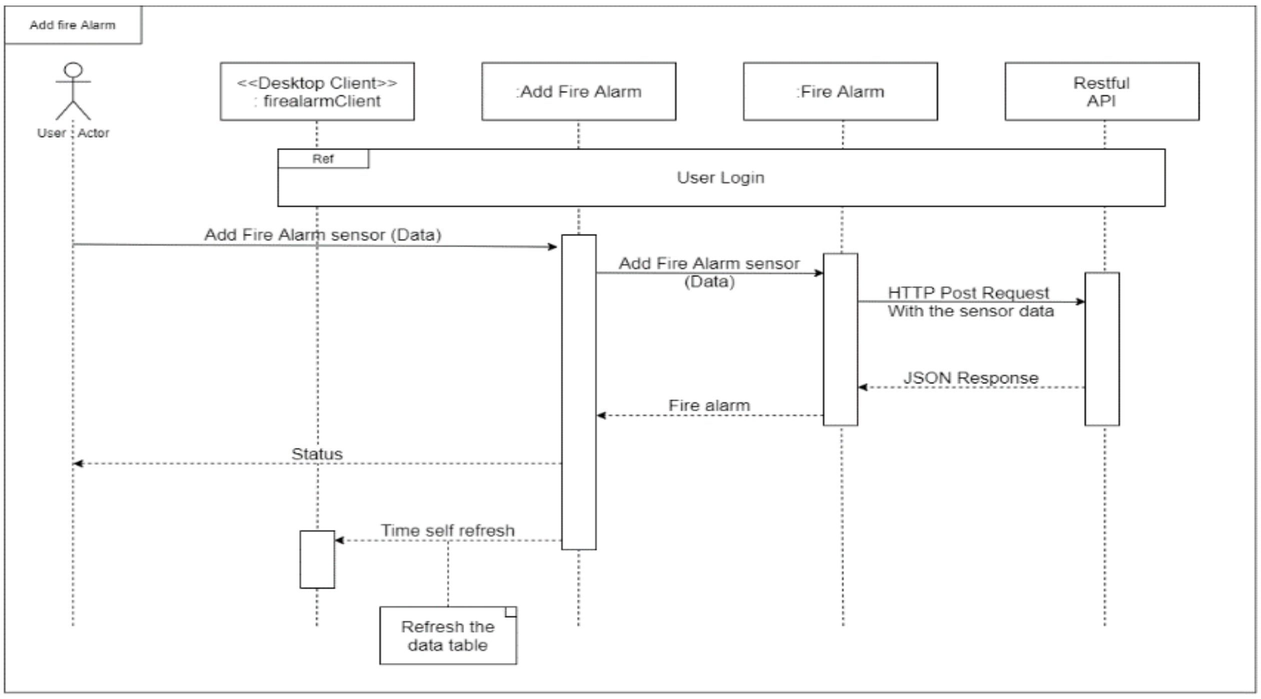
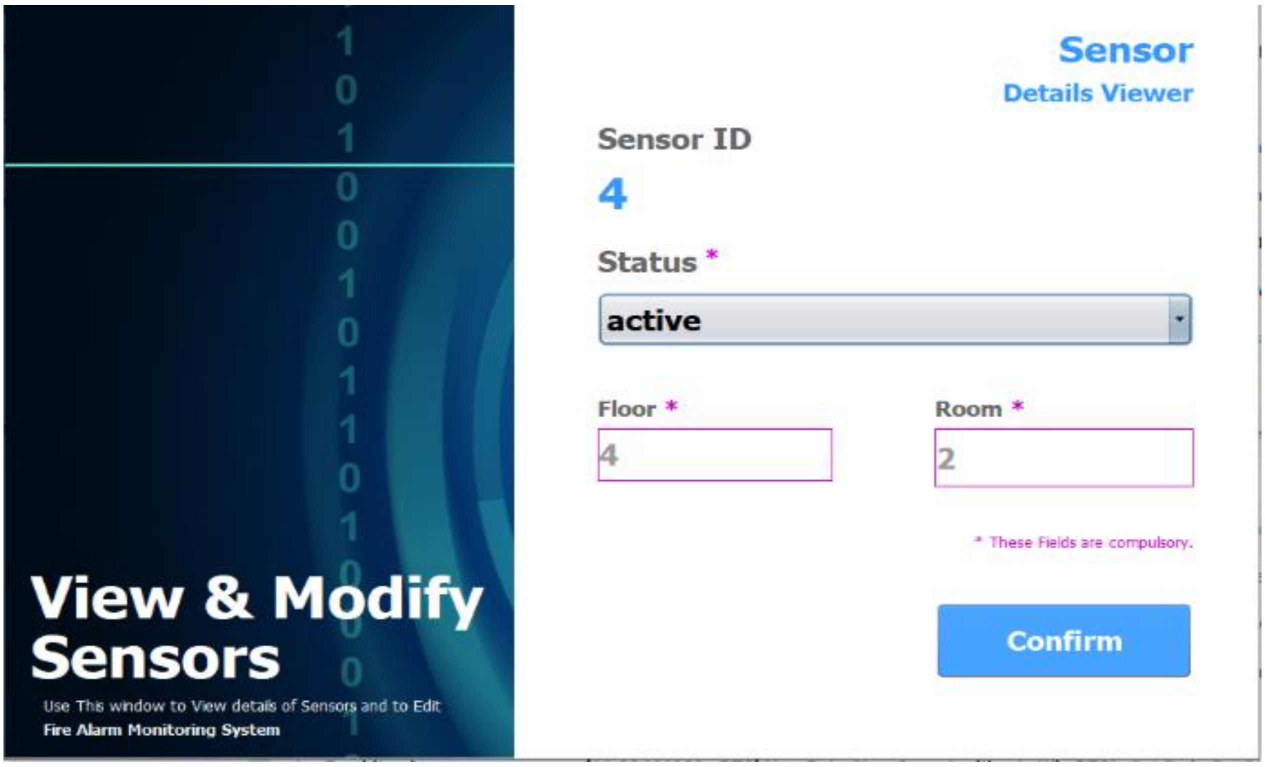


Figure 10 Sequence diagram for add fire alarm sensor


View & Modify Sensors
Use This window to View details of Sensors and to Edit Fire Alarm Monitoring System
Sensor
Details Viewer
Confirm
* These Fields are compulsory.

Sensor ID
4
Status *
active
Floor *
4
Room *
2

Figure 11 interface of edit and view sensor details

After adding the details, if necessary, admin can modify the sensor details and view those details from above interface. For each sensor: sensor Id, status, floor number and room number is available.

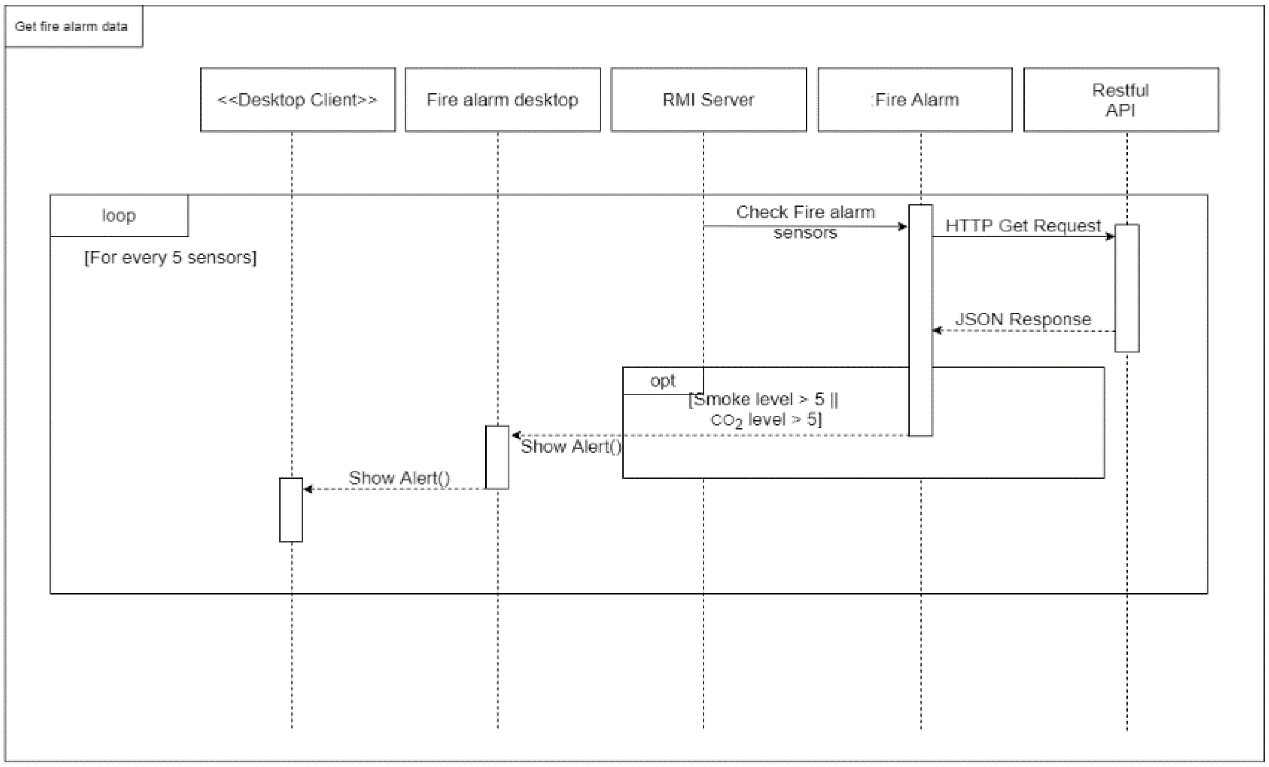
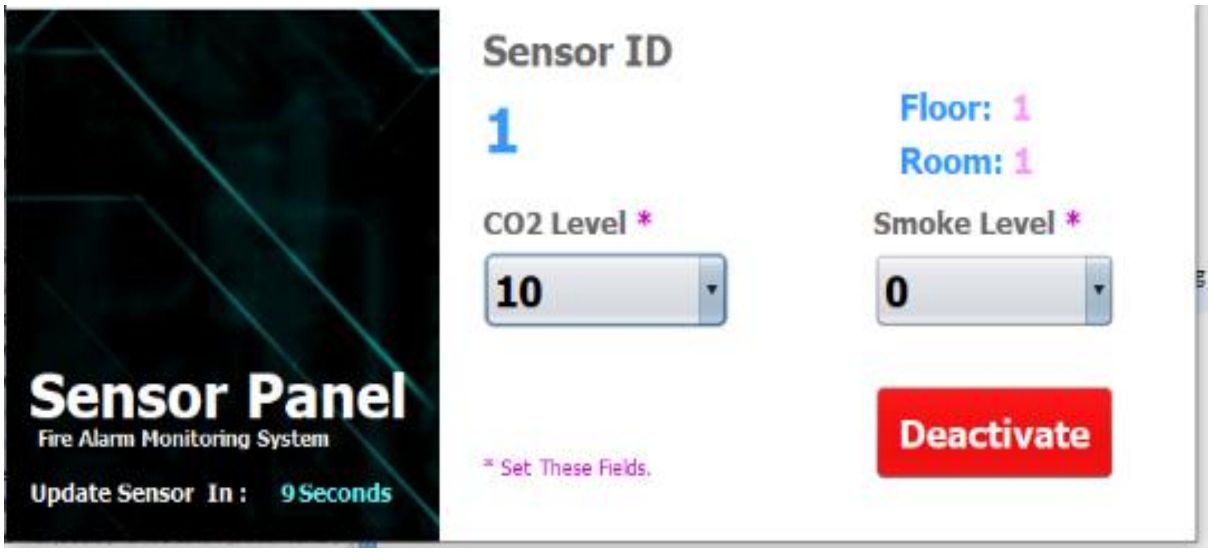


Figure 12 get fire alarm data from Rest Api

3.3 This is sensor application interface. This is a dummy sensor panel. From this data panel co2 level and smoke level can be added .It can be deactivate the particular sensor.



Update Sensor In : 9 Seconds

Sensor ID

1

Floor: 1

Room: 1

CO2 Level *

10

Smoke Level *

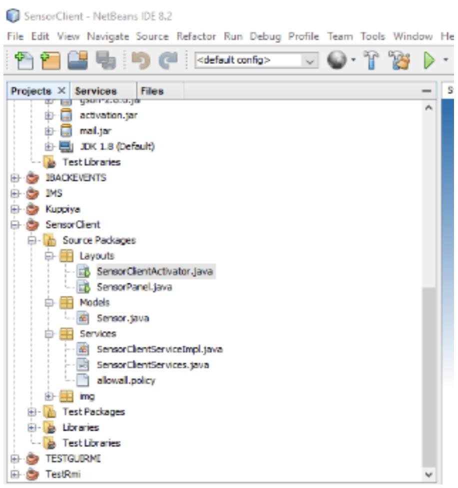
0

* Set These Fields.

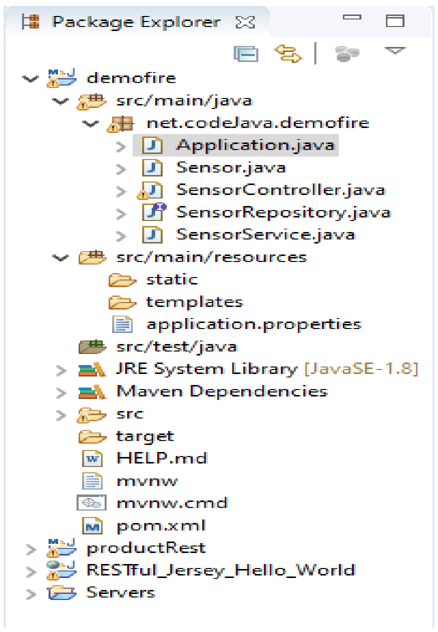
Figure 13 interface of sensor dummy app

Fire Alarm Sensor Application is a major component of this system as the most important data in this system is the processing of the data obtained using the Fire Alarm Sensors. The Fire Alarm sensors can use the Web interface provided by the REST API to update the data. The data needed to make the request for HTTP. A simple dummy program was built in the framework to simulate a Fire Alarm. This was developed using JAVA Swing. This dummy app transmits the data to the device within 10 seconds.

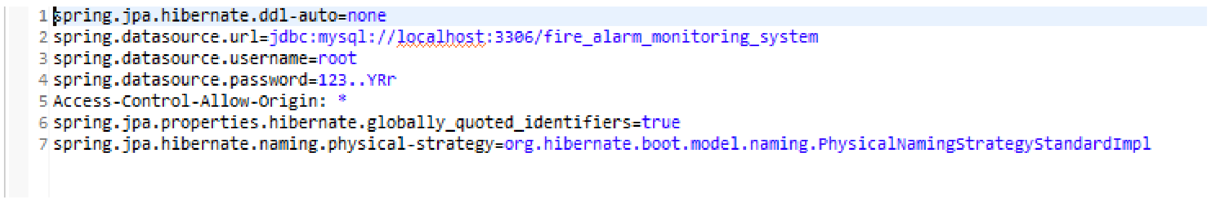
Folder structure of Sensor application



3.4 This is a REST API folder structure

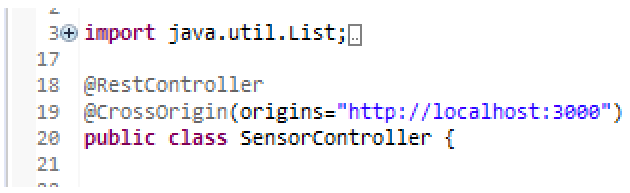


3.4.1 This is an application properties . In here , we can add connection to the MySQL database.



```
spring.jpa.hibernate.ddl-auto=none
spring.datasource.url=jdbc:mysql://localhost:3306/fire_alarm_monitoring_system
spring.datasource.username=root
spring.datasource.password=123..YRr
Access-Control-Allow-Origin: *
spring.jpa.properties.hibernate.globally_quoted_identifiers=true
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

3.4.2 This is cross origin code segment. It will initiate the URLs.

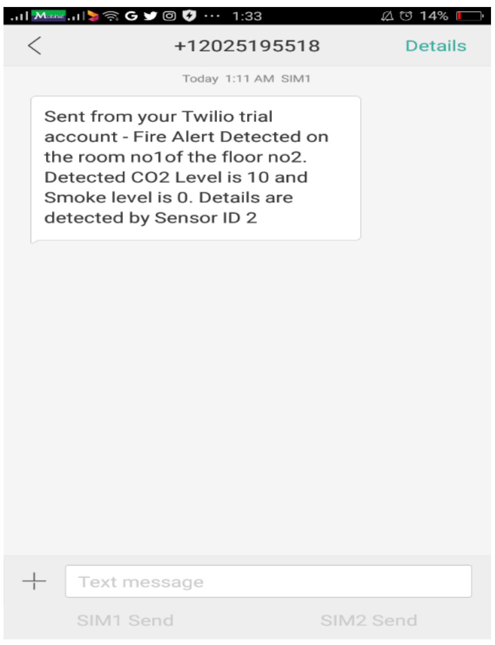


```java
import java.util.List;

@RestController
@CrossOrigin(origins="http://localhost:3000")
public class SensorController {

```

3.5 SMS Service

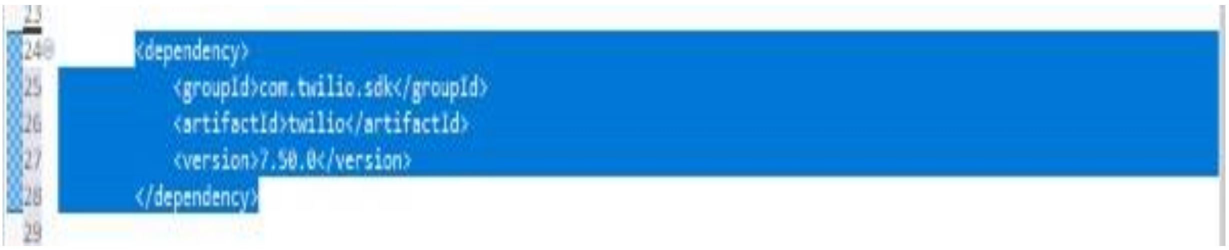


+12025195518

Today 1:11 AM SIM1

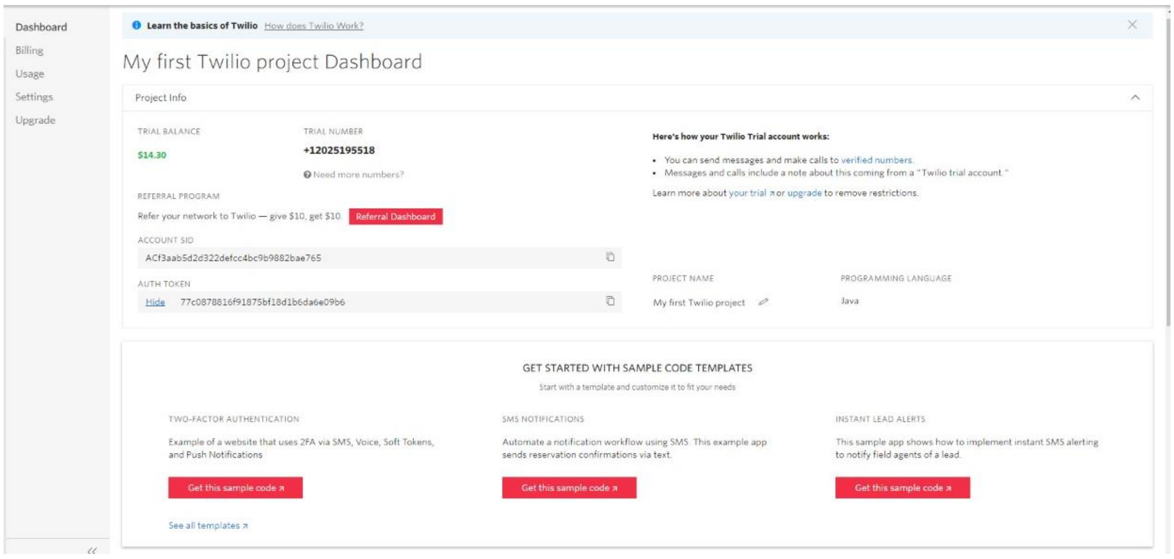
Sent from your Twilio trial account - Fire Alert Detected on the room no1of the floor no2. Detected CO2 Level is 10 and Smoke level is 0. Details are detected by Sensor ID 2

Figure 14 SMS from messenger service



```
    <dependency>
        <groupId>com.twilio.sdk</groupId>
        <artifactId>twilio</artifactId>
        <version>7.50.0</version>
    </dependency>
```

Dependencies for SMS service



My first Twilio project Dashboard

Project Info

TRIAL BALANCE
$14.30

TRIAL NUMBER
+12025195518

Need more numbers?

REFERRAL PROGRAM
Refer your network to Twilio — give $10, get $10 Referral Dashboard

ACCOUNT SID
ACf3aab5d2d322defcc4bc9b9882bae765

AUTH TOKEN
Hide 77c0878816f91875bf18d1b6da6e09b6

Here's how your Twilio Trial account works:

- You can send messages and make calls to verified numbers.
- Messages and calls include a note about this coming from a "Twilio trial account."

Learn more about your trial or upgrade to remove restrictions.

PROJECT NAME
My first Twilio project

PROGRAMMING LANGUAGE
Java

GET STARTED WITH SAMPLE CODE TEMPLATES
Start with a template and customize it to fit your needs

TWO-FACTOR AUTHENTICATION
Example of a website that uses 2FA via SMS, Voice, Soft Tokens, and Push Notifications
Get this sample code

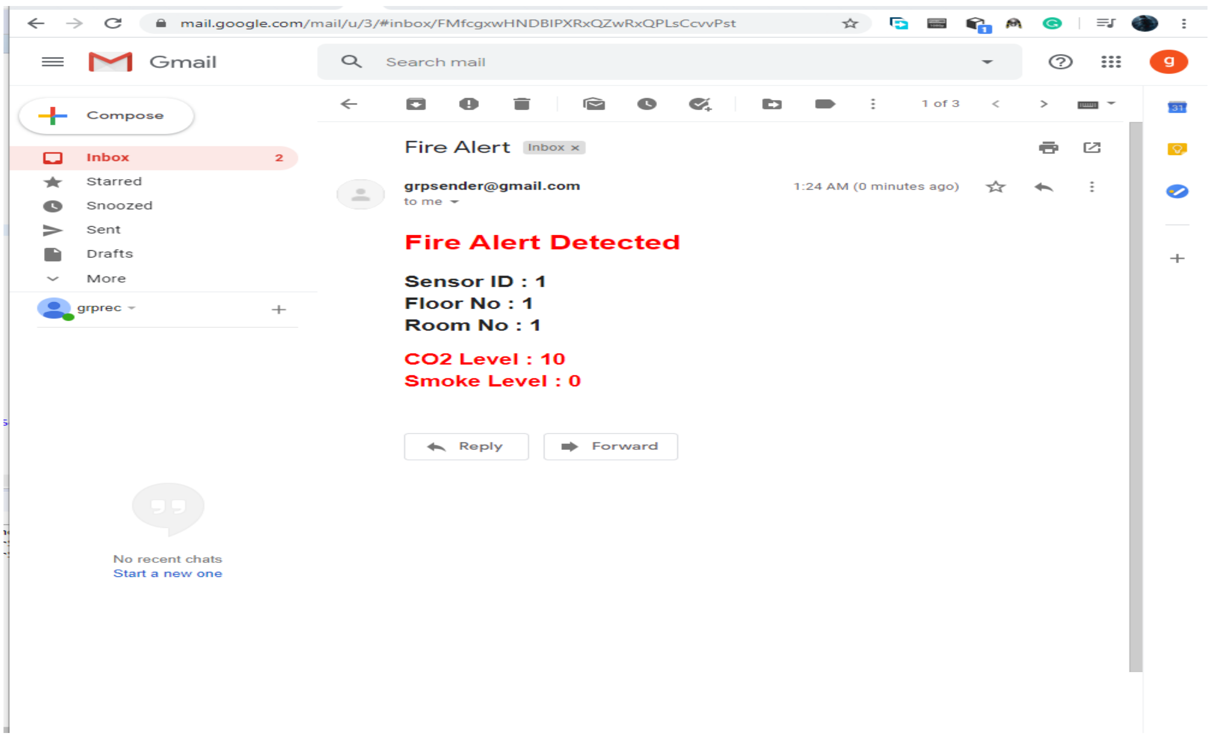
SMS NOTIFICATIONS
Automate a notification workflow using SMS. This example app sends reservation confirmations via text.
Get this sample code

INSTANT LEAD ALERTS
This sample app shows how to implement instant SMS alerting to notify field agents of a lead.
Get this sample code

See all templates

Figure 15 Message authentication details

3.6 Email Service



Fire Alert Inbox ×

grpsender@gmail.com
to me

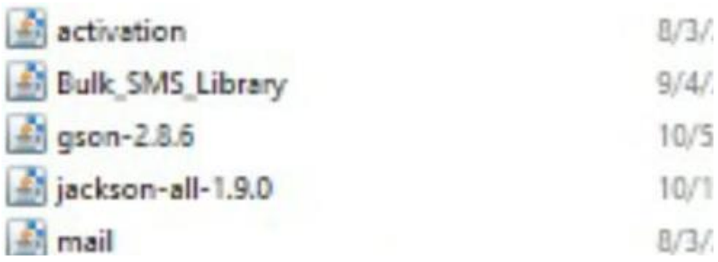
1:24 AM (0 minutes ago)

Fire Alert Detected

Sensor ID : 1
Floor No : 1
Room No : 1

CO2 Level : 10
Smoke Level : 0

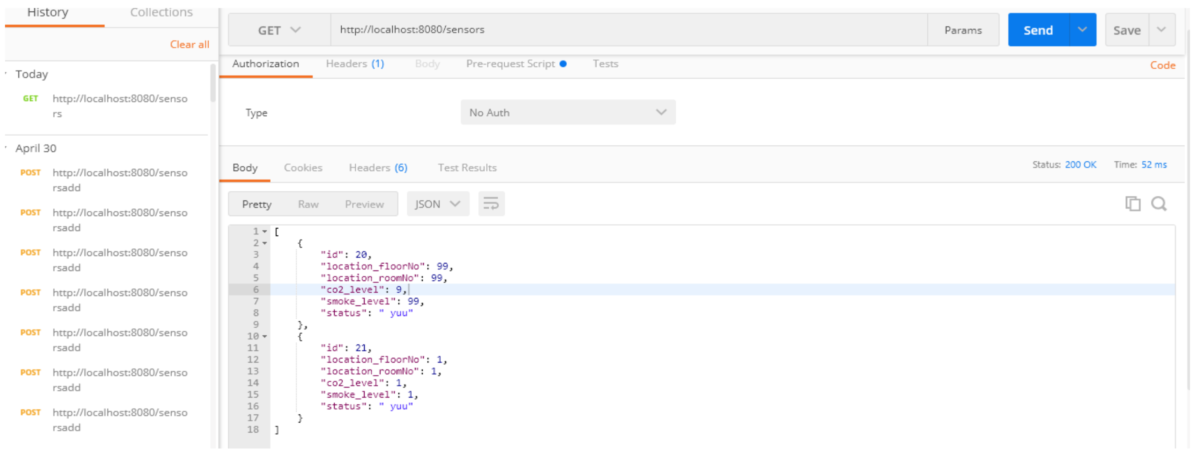
Figure 16 Email



activation	8/3/
Bulk_SMS_Library	9/4/
gson-2.8.6	10/5
jackson-all-1.9.0	10/1
mail	8/3/

Email sender libraries

RESTful Web Services API


History | Collections | Clear all

Today
GET http://localhost:8080/sensors

April 30
POST http://localhost:8080/sensorsadd
POST http://localhost:8080/sensorsadd
POST http://localhost:8080/sensorsadd
POST http://localhost:8080/sensorsadd
POST http://localhost:8080/sensorsadd
POST http://localhost:8080/sensorsadd
POST http://localhost:8080/sensorsadd

GET http://localhost:8080/sensors

Authorization | Headers (1) | Body | Pre-request Script | Tests

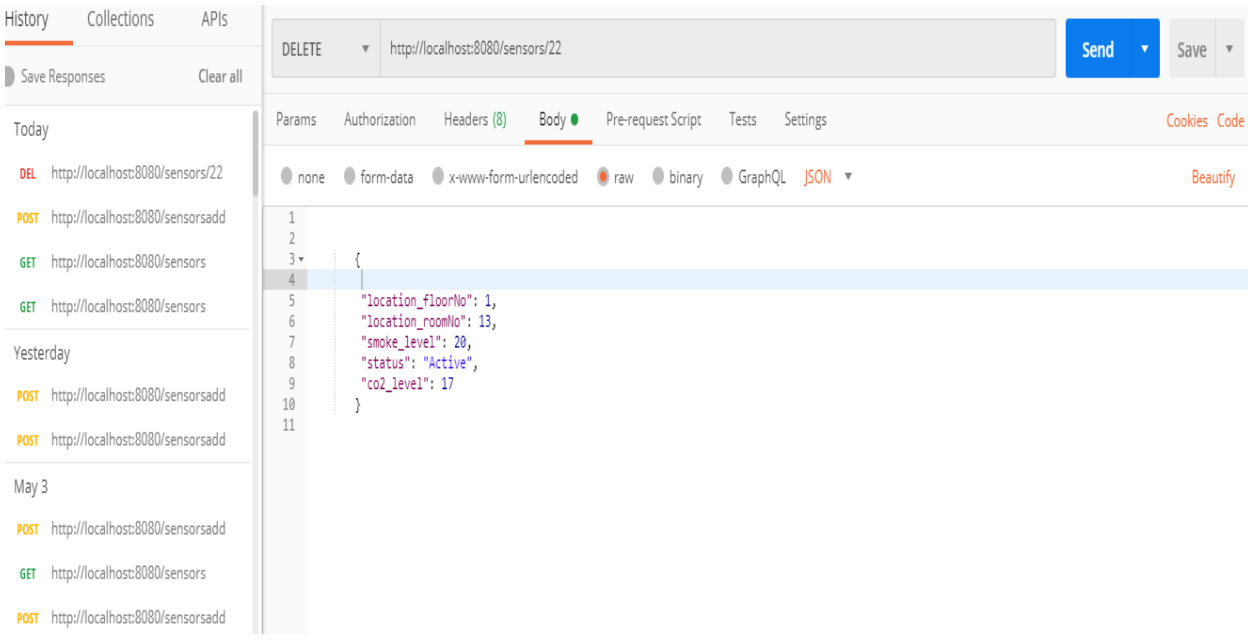
Type: No Auth

Body | Cookies | Headers (6) | Test Results — Status: 200 OK Time: 52 ms

Pretty | Raw | Preview | JSON

```
[
    {
        "id": 20,
        "location_floorNo": 99,
        "location_roomNo": 99,
        "co2_level": 9,
        "smoke_level": 99,
        "status": " yuu"
    },
    {
        "id": 21,
        "location_floorNo": 1,
        "location_roomNo": 1,
        "co2_level": 1,
        "smoke_level": 1,
        "status": " yuu"
    }
]
```

Figure 17 HTTP Get request


History | Collections | APIs

Save Responses | Clear all

Today
DEL http://localhost:8080/sensors/22
POST http://localhost:8080/sensorsadd
GET http://localhost:8080/sensors
GET http://localhost:8080/sensors

Yesterday
POST http://localhost:8080/sensorsadd
POST http://localhost:8080/sensorsadd

May 3
POST http://localhost:8080/sensorsadd
GET http://localhost:8080/sensors
POST http://localhost:8080/sensorsadd

DELETE http://localhost:8080/sensors/22

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

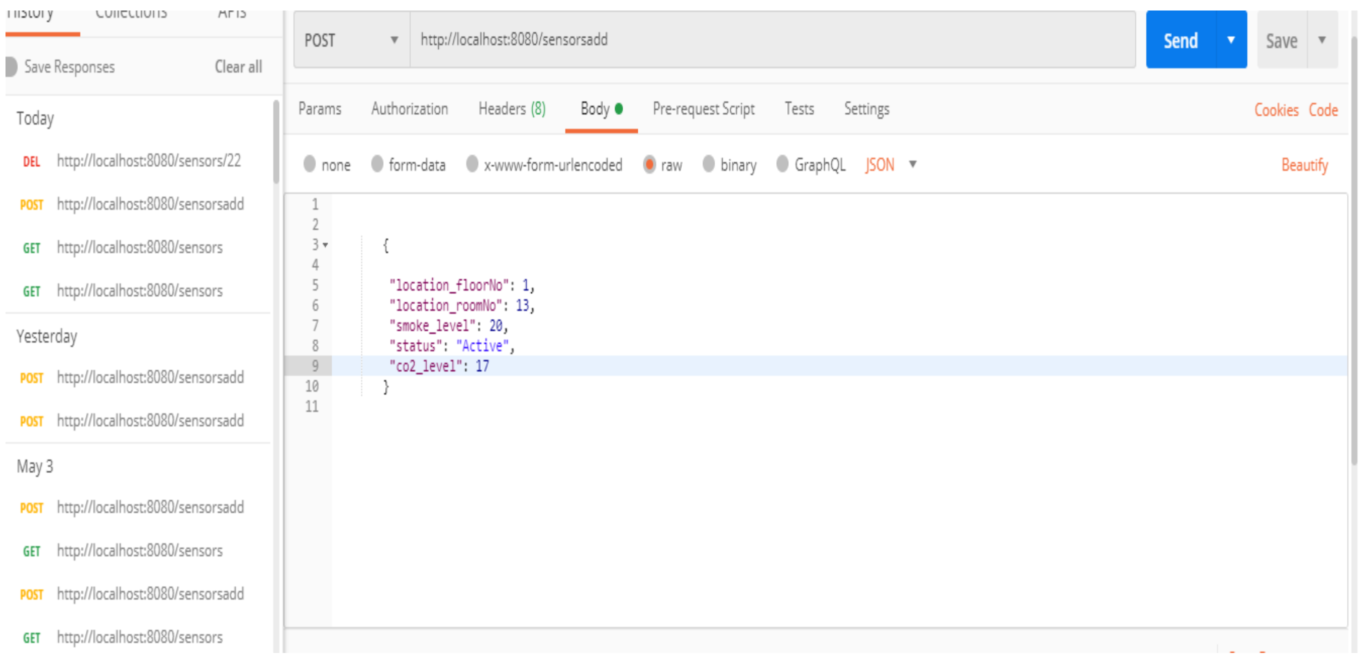
```


{

  "location_floorNo": 1,
  "location_roomNo": 13,
  "smoke_level": 20,
  "status": "Active",
  "co2_level": 17
}

```

Figure 18 HTTP Delete request

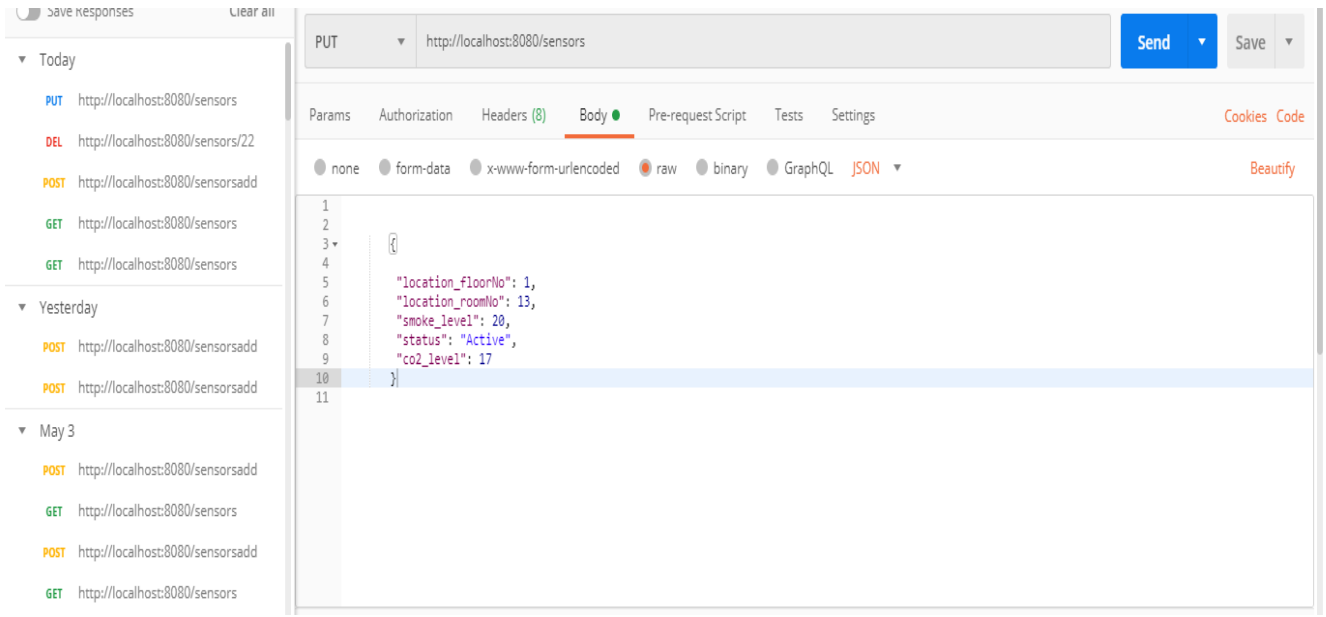


POST http://localhost:8080/sensorsadd

```
{

"location_floorNo": 1,
"location_roomNo": 13,
"smoke_level": 20,
"status": "Active",
"co2_level": 17
}
```

Figure 19 HTTP Post request



PUT http://localhost:8080/sensors

```
{

"location_floorNo": 1,
"location_roomNo": 13,
"smoke_level": 20,
"status": "Active",
"co2_level": 17
}
```

Figure 20 HTTP Put request

4.0 Appendix

Source Codes & Binaries

4.1 Web Client

Navigation.js

```
import React , {Component} from 'react';
import {NavLink} from 'react-router-dom';
import {Navbar,Nav} from 'react-bootstrap';

export class  Navigation extends Component{
    render(){
       return(
           <Navbar bg="dark" expand ="lg" >
             <Navbar.Toggle aria-controls="basic-navbar-nav"/>
             <Navbar.Collapse id = "basic-navbar-nav">
              <Nav>
                 <NavLink className="d-inline-p-2 bg-dark text-white" to="/SensorDetails" ></NavLink>
              </Nav>
             </Navbar.Collapse>
           </Navbar>
       )
    }
}
```

SensorDetails.js

```
import React , {Component} from 'react';
import {Table} from "react-bootstrap";

export class  SensorDetails extends Component{
    intervalID;
    constructor(props){
         super(props);
         this.state={sens:[]}
    }

    componentDidMount(){
        this.refreshList();
    }

    componentWillUnmount(){
        clearTimeout(this.intervalID);
    }

//get Sensor List
    refreshList(){
        fetch('http://localhost:8080/api/fireAlarmSystem/sensors')
        .then(response=> response.json())
        .then(data => {
            this.setState({sens:data});
//Re load data after 40seconds
            this.intervalID = setTimeout(this.refreshList.bind(this), 40000);
        })
    }
    render(){
        const {sens} =this.state;
        return(

         <div className="content">
             <div class="row">
               {this.state.sens.map(sensor=>(

<div class="col-md-4 col-xl-4">
    <div class={sensor.co2_level > 5 || sensor.smoke_level > 5 ? "card bg-c-pink order-card" : sensor.status ==
"active" ? "card bg-c-green order-card":"card bg-c-gray order-card"}>
        <div class="card-block">
        <i class="fas fa-broadcast-tower"></i>
            <h1 class="m-b-20">Sensor ID :   <b> <span class="f-right">{sensor.id}</span></b></h1><br/>
            <h2 class="text-right"><i class="fas fa-smog f-left"><span>  Smoke
Level</span></i><span>{sensor.smoke_level}</span></h2>
            <h2 class="text-right"><i class="fas fa-atom f-left"><span>    CO2
Level</span></i><span>{sensor.co2_level}</span></h2><br/>
            <h4 class="m-b-0">Floor No: <span class="f-right">{sensor.location_floorNo}</span></h4>
            <h4 class="m-b-0">Room No: <span class="f-right">{sensor.location_roomNo}</span></h4>
        </div>
    </div>
</div>

              ))}
         </div>
  </div>
            )
    }
}
```

App.js

```
import React from 'react';
import './App.css';

import {SensorDetails} from "./components/SensorDetails";
import {BrowserRouter,Route, Switch} from "react-router-dom";
import {Navigation} from "./components/Navigation";


function App() {
  return (
    <BrowserRouter>
      <div className="container">

      <div class="jumbotron">
          <h1 className="m-3 d-flex justify-content-center" >Fire Alarm Monitoring System</h1>

          </div>

          <Navigation/>
           <Switch>

               <Route path = '/' component={SensorDetails}  />

           </Switch>
      </div>
    </BrowserRouter>
  );
}

export default App;
```

4.2 Desktop Client

Models

Sensor.java

```
public class Sensor implements Serializable{

    private String id;
    private String location_floorNo;
    private String location_roomNo;
    private String co2_level;
    private String smoke_level;
    private String status;

    public Sensor(String id, String location_floorNo, String location_roomNo, String status) {
        this.id = id;
        this.location_floorNo = location_floorNo;
        this.location_roomNo = location_roomNo;
        this.status = status;
    }

    public Sensor(String location_floorNo, String location_roomNo, String status) {

        this.location_floorNo = location_floorNo;
        this.location_roomNo = location_roomNo;
        this.status = status;
    }

    public Sensor(String id, String location_floorNo, String location_roomNo, String co2_level, String
smoke, String status) {
        this.id = id;
        this.location_floorNo = location_floorNo;
        this.location_roomNo = location_roomNo;
        this.co2_level = co2_level;
        this.smoke_level = smoke;
        this.status = status;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getLocation_floorNo() {
        return location_floorNo;
    }

    public void setLocation_floorNo(String location_floorNo) {
        this.location_floorNo = location_floorNo;
    }

    public String getLocation_roomNo() {
        return location_roomNo;
    }

    public void setLocation_roomNo(String location_roomNo) {
        this.location_roomNo = location_roomNo;
    }

    public String getCo2_level() {
        return co2_level;
    }

    public void setCo2_level(String co2_level) {
        this.co2_level = co2_level;
    }

    public String getSmoke_level() {
        return smoke_level;
    }

    public void setSmoke_level(String smoke_level) {
        this.smoke_level = smoke_level;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }


}
```

User.java

```
public class User implements Serializable{

   private String Id;
   private String Name;
   private String Password;

    public User(String Id, String Name, String Password) {
        this.Id = Id;
        this.Name = Name;
        this.Password = Password;
    }

    public String getId() {
        return Id;
    }

    public void setId(String Id) {
        this.Id = Id;
    }

    public String getName() {
        return Name;
    }

    public void setName(String Name) {
        this.Name = Name;
    }

    public String getPassword() {
        return Password;
    }

    public void setPassword(String Password) {
        this.Password = Password;
    }


}
```

Database connection

```
package DB;
public class dbConnection {
    private static Connection con;
```

```
            public static Connection getConnection() {
                  if (con == null)
                       try {
                            Class.forName("com.mysql.jdbc.Driver");
               con = DriverManager.getConnection("jdbc:mysql://localhost:3306/fire_alarm_system", "root",
    "Homagama502");
             } catch (SQLException | ClassNotFoundException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
             }
          } return con;
        }
}


    public class AddSensor extends javax.swing.JFrame {


       public AddSensor() {
          initComponents();
       }
```

AddSensor.java

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



          //check if neccessary fields are empty
          if(Floor.getText().equals("") || Room.getText().equals("")){


             //diplay error
```

```
            JPanel panel = new JPanel();
            JOptionPane.showMessageDialog(panel, "Compulsory Fields are Empty. Please fill Compulsory
fields.", "Empty Fields", JOptionPane.ERROR_MESSAGE);
        }
        else {

            try {


                //initiate a Server Service instance

                System.setProperty("java.security.policy", "file:allowall.policy");
                Service service = (Service) Naming.lookup("rmi://localhost:1209/SensorServer");


                //set status
                String Status ="inactive";


                //initiate a sensor object from JFRAME label values
                Sensor sensor = new Sensor(Floor.getText(), Room.getText(), Status);

                //check if senso adding successfull
                if (service.addSensor(sensor)) {

                    //display success
                    JPanel panel = new JPanel();
                    JOptionPane.showMessageDialog(panel, "Sensor Installation is Successfull.", "Congrats!",
JOptionPane.INFORMATION_MESSAGE);

                    dispose();

                }
            } catch (SQLException ex) {
                Logger.getLogger(AddSensor.class.getName()).log(Level.SEVERE, null, ex);

                JPanel panel = new JPanel();
                JOptionPane.showMessageDialog(panel, "Registration is unsuccessfull. Please try again.",
"Error", JOptionPane.ERROR_MESSAGE);

            }catch (NotBoundException ex) {

                JPanel panel = new JPanel();
                JOptionPane.showMessageDialog(panel, "Registration is unsuccessfull. Server Error", "Error",
JOptionPane.ERROR_MESSAGE);

            }catch (MalformedURLException ex) {

                JPanel panel = new JPanel();
                JOptionPane.showMessageDialog(panel, "Registration is unsuccessfull. Server Error", "Error",
JOptionPane.ERROR_MESSAGE);

                System.out.println("Error 2");
                System.err.println(ex.getMessage());
            } catch (Exception ex) {

                JPanel panel = new JPanel();
                JOptionPane.showMessageDialog(panel, "Registration is unsuccessfull. Error Occured", "Error",
JOptionPane.ERROR_MESSAGE);

                System.out.println("Error 3");
                System.err.println(ex.getMessage());
            }

        }
      }
```

AdminDashboard.java

```
public class AdminDashboard extends javax.swing.JFrame {




    //properties
    String id;
    String name;
    User user;
    /**
     * Creates new form CustomerReg
     */
    public AdminDashboard() {

        //avoid unauthorized Access

        //dispose
        close();
        JPanel pane5 = new JPanel();
                JOptionPane.showMessageDialog(pane5, "Unauthorised Access. Please Log-in to the System", 
"ByPassed Attempt", JOptionPane.ERROR_MESSAGE);





                System.exit(-1);





    }

    //dispose frame
    public void close(){
```

```
    dispose();
  }

  public AdminDashboard(User user) {
    initComponents();
    setLabels(user);

    //set Properties
    this.user=user;
    this.name=user.getName();



    //Get Data to Tables and Sensor Count Label
    popTableAndCount();

    //Get Data to Alert Table
    popTableAlert();

    //Start Auto Updte Table after 10 Seconds
    autoRefrsh();







  }

  // set User Data to JFRAME labels
  public void setLabels(User u1){

    // set LAbel
    uname.setText(u1.getName());
    }



private void RemoveActionPerformed(java.awt.event.ActionEvent evt) {
        try {


            //Initiate Server Service instance
            System.setProperty("java.security.policy", "file:allowall.policy");
            Service service = (Service) Naming.lookup("rmi://localhost:1209/SensorServer");

           //Get Table model
            DefaultTableModel model = (DefaultTableModel) jTable1.getModel();

            // get Selected Row
            int selectrw = jTable1.getSelectedRow();
            id = model.getValueAt(selectrw, 0).toString();



            // delete SelectedSensor
            if(service.deleteSensor(id)){

                JPanel pane5 = new JPanel();
                JOptionPane.showMessageDialog(pane5, "Delete the event Successfully", "Deleted",
JOptionPane.INFORMATION_MESSAGE);
                popTableAndCount();
            }
        } catch (Exception ex) {
            Logger.getLogger(AdminDashboard.class.getName()).log(Level.SEVERE, null, ex);

            JPanel pane5 = new JPanel();
            JOptionPane.showMessageDialog(pane5, "Some Error Occured", "Error", 
JOptionPane.ERROR_MESSAGE);
        }



        //Re load data after Delete Sensor
        refresh();


    }


private void addSensorActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:


       //Shoe add sensor JFRAME
        new AddSensor().setVisible(true);
    }


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



        //get Table model
        DefaultTableModel model = (DefaultTableModel) jTable1.getModel();


        //get selected Row
        int selectrw = jTable1.getSelectedRow();
        id = model.getValueAt(selectrw, 0).toString();




       //Show sensor Editor/ Viewer to selected Sensor
        new EditSensor(id).setVisible(true);
    }

 private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:



        //log out from Dashboard
        dispose();
        new ClientLogin().setVisible(true);
    }

 private void refreshActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:



          // reload data
          refresh();



    }

public void refresh(){

        // reload data to main table
        popTableAndCount();

        // reload data to alert table
        popTableAlert();
    }

java.awt.EventQueue.invokeLater(new Runnable() {
          public void run() {
            new AdminDashboard().setVisible(true);
          }
        });
    }



 public void popTableAndCount(){

        try {

       //Initiate Server Service instance

          System.setProperty("java.security.policy", "file:allowall.policy");
          Service service = (Service) Naming.lookup("rmi://localhost:1209/SensorServer");






          //get table model
          DefaultTableModel model = (DefaultTableModel) jTable1.getModel();

          //set row count to 0
          model.setRowCount(0);


          // get all sensor List
          ArrayList<Sensor> array = service.getSensorList();


          //load data into table from array
          Object rawdata[] = new Object[6];
          for(int i=0; i<array.size();i++){

             rawdata[0]=array.get(i).getId();
             rawdata[1]=array.get(i).getLocation_floorNo();
             rawdata[2]=array.get(i).getLocation_roomNo();
             rawdata[3]=array.get(i).getCo2_level();
             rawdata[4]=array.get(i).getSmoke_level();
             rawdata[5]=array.get(i).getStatus();



             array.get(i).toString();

             //add row to table
             model.addRow(rawdata);


          }

          // get number of rows
          int coutInt= model.getRowCount();

          String ContS = String.valueOf(coutInt);

          //set count label to row count
           count.setText(ContS);



        } catch (SQLException ex) {
          Logger.getLogger(AdminDashboard.class.getName()).log(Level.SEVERE, null, ex);
```

```
            JPanel pane5 = new JPanel();
                  JOptionPane.showMessageDialog(pane5, "Error Loading the Sensor Contents", "Error",
JOptionPane.ERROR_MESSAGE);


            dispose();
            new ClientLogin().setVisible(true);
        }catch (NotBoundException ex) {


           System.out.println("Error 1");
           System.err.println(ex.getMessage());
        } catch (MalformedURLException ex) {
           System.out.println("Error 2");
           System.err.println(ex.getMessage());
        } catch (RemoteException ex) {
           System.out.println("Error 3");
           System.err.println(ex.getMessage());

        } catch (Exception ex) {



           JPanel pane5 = new JPanel();
                  JOptionPane.showMessageDialog(pane5, "Error Loading the Sensor Contents", "Error",
JOptionPane.ERROR_MESSAGE);


            dispose();
            new ClientLogin().setVisible(true);



        }



     }
```

```
//load data to Alert table
public void popTableAlert(){




   try {
      //Initiate Server Service instance

        System.setProperty("java.security.policy", "file:allowall.policy");
        Service service = (Service) Naming.lookup("rmi://localhost:1209/SensorServer");






        //get table model
        DefaultTableModel model = (DefaultTableModel) jTable2.getModel();
        // set row count to 0
        model.setRowCount(0);

        // get all sensor List
        ArrayList<Sensor> array = service.AlertSensorList();



        // load data into table from array
        Object rawdata[] = new Object[6];

        for(int i=0; i<array.size();i++){

            rawdata[0]=array.get(i).getId();
            rawdata[1]=array.get(i).getLocation_floorNo();
            rawdata[2]=array.get(i).getLocation_roomNo();
```

```
                rawdata[3]=array.get(i).getCo2_level();
                rawdata[4]=array.get(i).getSmoke_level();
                rawdata[5]=array.get(i).getStatus();



                array.get(i).toString();

                // add row to table
                model.addRow(rawdata);


              }


            } catch (SQLException ex) {
              Logger.getLogger(AdminDashboard.class.getName()).log(Level.SEVERE, null, ex);

               JPanel pane5 = new JPanel();
                    JOptionPane.showMessageDialog(pane5, "Error Loading the Alert Sensor Contents", "Error",
JOptionPane.ERROR_MESSAGE);

               dispose();
               new ClientLogin().setVisible(true);
            }catch (NotBoundException ex) {

              System.out.println("Error 1");
              System.err.println(ex.getMessage());
            } catch (MalformedURLException ex) {
              System.out.println("Error 2");
              System.err.println(ex.getMessage());
            } catch (RemoteException ex) {
              System.out.println("Error 3");
              System.err.println(ex.getMessage());
```

```
    } catch (Exception ex) {


        JPanel pane5 = new JPanel();
                JOptionPane.showMessageDialog(pane5, "Error Loading the Sensor Contents", "Error",
JOptionPane.ERROR_MESSAGE);

         dispose();
         new ClientLogin().setVisible(true);


    }
    //show red color Labels
     showAlert();


   }

   //show red color labels if alert table has data rows
   public void showAlert(){
   // set properties
      alert.setVisible(false);
      alert1.setVisible(false);


      //get table model
       DefaultTableModel model = (DefaultTableModel) jTable2.getModel();

       // get row count of the table
       int rowcount=  model.getRowCount();
```

```
        // show labels if fowcount is greater than 0
        if(rowcount>0){

            alert.setVisible(true);
            alert1.setVisible(true);

        }else if(rowcount==0){

            alert.setVisible(false);
            alert1.setVisible(false);



        }




    }
    //load data in 10 second priods
    public void autoRefrsh(){

        // initiate a timer
        Timer timer = new Timer(10000, new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent arg0) {

                // reload tables and data
                refresh();

            }
        });

          // set timer properties
          timer.setRepeats(true); // Only execute once
          timer.start();
      }
```

ClientLogin.java

```
public class ClientLogin extends javax.swing.JFrame {

   /**
    * Creates new form AdminLogin
    */
   public ClientLogin() {
      initComponents();
   }

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {




      try {

           //Initate a Sever Service Object

          System.setProperty("java.security.policy", "file:allowall.policy");
          Service service = (Service) Naming.lookup("rmi://localhost:1209/SensorServer");
    //Get User By User ID
    User user = service.getUser(Username.getText());




    //Validate User

      if(user.getPassword().equals(String.valueOf(pass.getPassword()))){


      //display success dialog
        JPanel pane3 = new JPanel();
        JOptionPane.showMessageDialog(pane3, "Login Successfull. You may re-directed to your
Dashboard Page", "Login Successfull", JOptionPane.INFORMATION_MESSAGE);

        dispose();

        //display admin Dashboard
        new AdminDashboard(user).setVisible(true);
      }

      else{

        //display login faild
        JPanel pane5 = new JPanel();
        JOptionPane.showMessageDialog(pane5, "Password Incorrect", "Login Faild",
JOptionPane.ERROR_MESSAGE);

      }


  } catch (NotBoundException ex) {
```

```
            System.out.println("Error 1");
            System.err.println(ex.getMessage());
        } catch (MalformedURLException ex) {
            System.out.println("Error 2");
            System.err.println(ex.getMessage());
        } catch (RemoteException ex) {
            System.out.println("Error 3");
            System.err.println(ex.getMessage());
            JPanel pane2 = new JPanel();
                JOptionPane.showMessageDialog(pane2, "Server Not Found.", " Server Error",
JOptionPane.ERROR_MESSAGE);

        } catch (NullPointerException ex1) {
            Logger.getLogger(ClientLogin.class.getName()).log(Level.SEVERE, null, ex1);
            JPanel pane2 = new JPanel();
                JOptionPane.showMessageDialog(pane2, "User ID Not Found or Empty. Please enter a valid ID.",
"ID Not Found", JOptionPane.ERROR_MESSAGE);
        } catch (Exception ex2) {
            Logger.getLogger(ClientLogin.class.getName()).log(Level.SEVERE, null, ex2);
            JPanel panel = new JPanel();
                JOptionPane.showMessageDialog(panel, "Some error occured. Please try again.", "Login Failed",
JOptionPane.ERROR_MESSAGE);
        }



    }

public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */


   if(System.getSecurityManager()==null){
   System.setProperty("java.security.policy", "file:allowall.policy");



      System.out.println("Here Done");
   }



   try {
      for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
         if ("Nimbus".equals(info.getName())) {
            javax.swing.UIManager.setLookAndFeel(info.getClassName());
            break;
         }
      }
   } catch (ClassNotFoundException ex) {
      java.util.logging.Logger.getLogger(ClientLogin.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
   } catch (InstantiationException ex) {
      java.util.logging.Logger.getLogger(ClientLogin.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
   } catch (IllegalAccessException ex) {
      java.util.logging.Logger.getLogger(ClientLogin.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
   } catch (javax.swing.UnsupportedLookAndFeelException ex) {
      java.util.logging.Logger.getLogger(ClientLogin.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
   }
   //</editor-fold>
   //</editor-fold>
   //</editor-fold>
   //</editor-fold>
```

```
        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ClientLogin().setVisible(true);
            }
        });
    }

}
```

EditSensor.java

```
public class EditSensor extends javax.swing.JFrame {

    //properties
    String id;
    Sensor sensor;



    /**
     * Creates new form CustomerReg
     */
    public EditSensor() {
        initComponents();

        //Avoid UnAuthorized Access
        dispose();
        JPanel pane5 = new JPanel();
                JOptionPane.showMessageDialog(pane5, "Unauthorised Access. Please Log-in to the System",
"ByPassed Attempt", JOptionPane.ERROR_MESSAGE);




                System.exit(-1);
    }

    public EditSensor(String ID) {


        initComponents();


        //set Properies
        id=ID;
        sensor=getSensor(ID);


        //Set JFRAME label values getting values from Seonso object
        setDetails(sensor);


    }


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





        //Check neccessary fields are not empty

        if(Floor.getText().equals("") || Room.getText().equals("")){

            JPanel panel = new JPanel();
            JOptionPane.showMessageDialog(panel, "Compulsory Fields are Empty. Please fill Compulsory
fields.", "Empty Fields", JOptionPane.ERROR_MESSAGE);
        }
    else {

        try {

            //Initate a Sever Service Object

            System.setProperty("java.security.policy", "file:allowall.policy");
            Service service = (Service) Naming.lookup("rmi://localhost:1209/SensorServer");

            //create Server object from values obtained from JFRAME
            Sensor sen = new Sensor(id, Floor.getText(),
Room.getText(),sensor.getCo2_level(),sensor.getSmoke_level(),  Status.getSelectedItem().toString());



            //check if UPDATE successful
            if (service.updateSensor(sen)) {

                //Display Success Message
                JPanel panel = new JPanel();
                JOptionPane.showMessageDialog(panel, "Sensor Update is Successfull.", "Congrats!",
JOptionPane.INFORMATION_MESSAGE);
                dispose();

            }

        //catch Exceptions
        } catch (SQLException ex) {
            Logger.getLogger(EditSensor.class.getName()).log(Level.SEVERE, null, ex);

            JPanel panel = new JPanel();
            JOptionPane.showMessageDialog(panel, "Update is unsuccessfull. Please try again.", "Error",
JOptionPane.ERROR_MESSAGE);
```

```
            }catch (NotBoundException ex) {

                JPanel panel = new JPanel();
                JOptionPane.showMessageDialog(panel, "Update is unsuccessfull. Server Error", "Error",
JOptionPane.ERROR_MESSAGE);

            }catch (MalformedURLException ex) {

                JPanel panel = new JPanel();
                JOptionPane.showMessageDialog(panel, "Update is unsuccessfull. Server Error", "Error",
JOptionPane.ERROR_MESSAGE);


        } catch (Exception ex) {


            JPanel panel = new JPanel();
            JOptionPane.showMessageDialog(panel, "Update is unsuccessfull. Error Occured", "Error",
JOptionPane.ERROR_MESSAGE);

        }

        }
    }

 public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
        /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
         * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
         */
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger(EditSensor.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger(EditSensor.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger(EditSensor.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {
            java.util.logging.Logger.getLogger(EditSensor.class.getName()).log(java.util.logging.Level.SEVERE,
null, ex);
        }
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>
        //</editor-fold>

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new EditSensor().setVisible(true);
            }
        });
    }


    // get Sensor Details by ID Through REST API
    public Sensor getSensor(String ID){

        Sensor sensor = null;
        try {

             //Initate a Sever Service Object

            System.setProperty("java.security.policy", "file:allowall.policy");
            Service service = (Service) Naming.lookup("rmi://localhost:1209/SensorServer");



            //get Sensor By ID
            sensor = service.getSensor(ID);





        } catch (NotBoundException ex) {
            Logger.getLogger(EditSensor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (MalformedURLException ex) {
            Logger.getLogger(EditSensor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (RemoteException ex) {
            Logger.getLogger(EditSensor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (Exception ex) {
            Logger.getLogger(EditSensor.class.getName()).log(Level.SEVERE, null, ex);
        }

        //return Server Object
        return sensor;
    }

    //Set  VAlues of the JFRAME
    public void setDetails(Sensor sensor){
        // set Label Values
        ID.setText(sensor.getId());
        Floor.setText(sensor.getLocation_floorNo());
        Room.setText(sensor.getLocation_roomNo());

        if(sensor.getStatus().equalsIgnoreCase("Inactive")){
            Status.setSelectedIndex(0);
        }else{
            Status.setSelectedIndex(1);
        }



    }
```

4.4 RMI Server

Server.java

```
package RMIServer;


public class Server extends UnicastRemoteObject implements Service {

    //Declare a Timer Oject
    Timer timer;
    //Declare HTTP URL Connectio object
    HttpURLConnection conn;
    //Declare an Object of authenticator class object
    Authenticator auth;
    //Create connection class by retrieving connection from dbConnection class object
```

```
Connection con = dbConnection.getConnection();
//REGISTRY PORT NUMBER
final static int REGISTRY=1209;
//Constructor
public Server() throws RemoteException, SQLException {

    super();
    //CREATE USER TABLE in database if not Exist
    createUserTable();

}
// Main Method of the Server
public static void main(String[] args) {
    // TODO code application logic here



    //set Security Policy
    System.setProperty("java.security.policy", "file:allowall.policy");

    try {

        //Create a Server Object
        Server svr = new Server();

        //create Registry
        Registry registry = LocateRegistry.createRegistry(REGISTRY);

        //Bind the Server to the Registry
        registry.bind("SensorServer", svr);

        System.out.println("Service Started.....!");

    //Catch Exceptions
    } catch (RemoteException re) {
        System.err.println(re.getMessage());
    } catch (AlreadyBoundException abe) {
        System.err.println(abe.getMessage());
    } catch (SQLException ex) {
        Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
    }

}
@Override
//get the user/admin details directly from database
public User getUser(String id) throws Exception {

    //create a user object
    User user = null;

    try {

        //Sql query to take a user from ID from user table
        String sql = "SELECT * from user where userId ='" + id + "'";

        // create statement object
        Statement stm = con.createStatement();

        //set results of the sql queries to a result set object
        ResultSet rlt = stm.executeQuery(sql);

        //read each result in result set
        while (rlt.next()) {

            //get vlues from the one object of the result set
            String ID = rlt.getString("userId");
            String name = rlt.getString("name");
            String password = rlt.getString("password");


            //Create new user from retrieved data
            user = new User(ID, name, password);

          }

          //catch any SQL and other exceptions
        } catch (Exception ex) {
          Logger.getLogger(ClientLogin.class.getName()).log(Level.SEVERE, null, ex);
          System.out.println("Error is in SQL");
        }

        //return a user object
        return user;

    }

    @Override
    //add sensors to the database through REST API
    public boolean addSensor(Sensor sensor) throws SQLException {

        //hold the success of failure of the method execution; true if success, false if fails
        boolean stat = false;

        try {

          //Set initial CO2 level of the the new sensor to 0
          sensor.setCo2_level("0");
           //Set initial Smoke level of the the new sensor to 0
          sensor.setSmoke_level("0");
```

```
//Create new object mapper Class
ObjectMapper mapper = new ObjectMapper();

//Set values of the Sensor Object in to a JSON String
String jsonString = mapper.writeValueAsString(sensor);

//Create URL object , targeting REST APIs Endpoint to Add Sensors
URL url = new URL("http://localhost:8080/api/fireAlarmSystem/Add");

// Create a HTTPURL Connection object and open a connection
HttpURLConnection conn = (HttpURLConnection) url.openConnection();

// Set request method into POST
conn.setRequestMethod("POST");

// Set the Request Content Property
conn.setRequestProperty("Content-Type", "application/json; charset=UTF-8");

// Set the Request Property to accept JSON responses
conn.setRequestProperty("Accept", "application/json");

// Set a output stream
conn.setDoOutput(true);

// Creating the Data in the request body and writing it to output stream
String data = jsonString;

//Get Output Stream into an OutputStreamObject
OutputStream stream = conn.getOutputStream();

//Set get data format
byte[] input = data.getBytes("utf-8");
```

```
    //Write inputs into OutputStream
    stream.write(input, 0, input.length);

    // Get response code
    int responseCode = conn.getResponseCode();

    // Read the response
    Reader reader = null;

    // check ResponseCode
    if (responseCode >= 200 && responseCode <= 299) {

      //read Buffer Data
       reader = new BufferedReader(new InputStreamReader(conn.getInputStream(), "utf-8"));
       //Set method execution success
       stat = true;

    } else {

       //read Buffer Data
       reader = new BufferedReader(new InputStreamReader(conn.getErrorStream(), "utf-8"));

       //Set method execution Fail
       stat = false;
    }

  //catch Exceptions
  } catch (IOException ex) {

    System.out.println("Json build error");
    Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
  }
```

```
        // return Method success state
        return stat;

    }

    @Override
    //Update Sensor Through REST API
    public boolean updateSensor(Sensor sensor) throws SQLException {

        //hold the success of failure of the method execution; true if success, false if fails
        boolean stat = false;

        try {

            //Hold Sensor ID
            String id = sensor.getId();

             //Create new object mapper Class
            ObjectMapper mapper = new ObjectMapper();

            //Set values of the Sensor Object in to a JSON String
            String jsonString = mapper.writeValueAsString(sensor);

           //Create URL object , targeting REST APIs Endpoint to Update Sensors
            URL url = new URL("http://localhost:8080/api/fireAlarmSystem/Update/" + id);
            // Create a HTTPURL Connection object and open a connection
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();

            // Set request method into PUT
            conn.setRequestMethod("PUT");

            // Set the Request Content Property
```

```
conn.setRequestProperty("Content-Type", "application/json; charset=UTF-8");

// Set the Request Property to accept JSON responses
conn.setRequestProperty("Accept", "application/json");

// Set a output stream
conn.setDoOutput(true);

// Creating the Data in the request body and writing it to output stream
String data = jsonString;

//Get Output Stream into an OutputStreamObject
OutputStream stream = conn.getOutputStream();

//Set get data format
byte[] input = data.getBytes("utf-8");

//Write inputs into OutputStream
stream.write(input, 0, input.length);

// Get response code
int responseCode = conn.getResponseCode();

// Read the response
Reader reader = null;

// check ResponseCode
if (responseCode >= 200 && responseCode <= 299) {

   //read Buffer Data
    reader = new BufferedReader(new InputStreamReader(conn.getInputStream(), "utf-8"));
    //Set method execution success
    stat = true;
```

```
        } else {

            //read Buffer Data
            reader = new BufferedReader(new InputStreamReader(conn.getErrorStream(), "utf-8"));


            //Set method execution Fail
            stat = false;
        }

    //catch Exceptions
    } catch (MalformedURLException ex) {
        Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
    }

    // return Method success state
    return stat;

}

@Override
//Delete Sensor Through REST API
public boolean deleteSensor(String Id) throws SQLException {

    try {

        //Create URL object , targeting REST APIs Endpoint to Delete Sensors
        URL url = new URL("http://localhost:8080/api/fireAlarmSystem/Delete/" + Id);

       // Create a HTTPURL Connection object and open a connection
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();

    // Set request method into DELETE
    conn.setRequestMethod("DELETE");

    // Set the Request Content Property
    conn.setRequestProperty("Content-Type", "application/json; charset=UTF-8");

    // Set the Request Property to accept JSON responses
    conn.setRequestProperty("Accept", "application/json");

    // Set a output stream
    conn.setDoOutput(true);

    //Get the Responsecode
    int responseCode = conn.getResponseCode();
    // Read the response
    Reader reader = null;
    // check ResponseCode
    if (responseCode >= 200 && responseCode <= 299) {

        //read Buffer Data
        reader = new BufferedReader(new InputStreamReader(conn.getInputStream(), "utf-8"));
    } else {

        //read Buffer Data
        reader = new BufferedReader(new InputStreamReader(conn.getErrorStream(), "utf-8"));
    }

//Check Exceptions
} catch (MalformedURLException ex) {
    Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
} catch (IOException ex) {
        Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
      }

      return true;

    }

    @Override
    //Get Sensor Through REST API
    public Sensor getSensor(String Id) throws Exception {

      //Sensor Object
      Sensor sensor = null;

      //Create URL object , targeting REST APIs Endpoint to Get Sensor BY ID
      URL url = new URL("http://localhost:8080/api/fireAlarmSystem/sensors/" + Id);

      // Create a HTTPURL Connection object and open a connection
      HttpURLConnection con = (HttpURLConnection) url.openConnection();

       // Set request method into GET
      con.setRequestMethod("GET");

      // Set the Request  Property to accept JSON
      con.setRequestProperty("Accept", "application/json");

      //Get the Responsecode
      int responseCode = con.getResponseCode();

      // Read the response
      Reader reader = null;

      //Check Responses
      if (responseCode >= 200 && responseCode <= 299) {
        reader = new BufferedReader(new InputStreamReader(con.getInputStream(), "utf-8"));
      } else {
        reader = new BufferedReader(new InputStreamReader(con.getErrorStream(), "utf-8"));
      }

      //Create GSON Object
      Gson gson = new Gson();
      try {

        //Create Sensor object from JSON Response
        sensor = gson.fromJson(reader, Sensor.class);

      } catch (Exception e) {
        System.out.println(e);
      }

      // check if sensor is null
      if (sensor != null) {

        //if not null return sensor
        return sensor;
      } else {
        return null;
      }

    }

    @Override
    //Get All Sensors  Through REST API
    public ArrayList<Sensor> getSensorList() throws SQLException, IOException {

      //ArrayList
```

```
ArrayList<Sensor> list = new ArrayList<>();

//Create URL object , targeting REST APIs Endpoint to Get Sensors
URL url = new URL("http://localhost:8080/api/fireAlarmSystem/sensors");
// Create a HTTPURL Connection object and open a connection
HttpURLConnection con = (HttpURLConnection) url.openConnection();

 // Set request method into GET
con.setRequestMethod("GET");

// Set the Request  Property to accept JSON
con.setRequestProperty("Accept", "application/json");

//Get the Responsecode
int responseCode = con.getResponseCode();

// Read the response
Reader reader = null;

//Check Responses
if (responseCode >= 200 && responseCode <= 299) {
   reader = new BufferedReader(new InputStreamReader(con.getInputStream(), "utf-8"));
} else {
   reader = new BufferedReader(new InputStreamReader(con.getErrorStream(), "utf-8"));
}

//Create GSON Object
Gson gson = new Gson();
try {

   //Create Sensor object Array from JSON Response
   Sensor[] temp = gson.fromJson(reader, Sensor[].class);
```

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

                //add to ArrayList
                list.add(temp[i]);
            }

        //catch Exceptions
        } catch (Exception e) {
            System.out.println(e);
        }

        //Return ArrayLists
        return list;

    }

    @Override
    //get Alert Triggered all Sensors Through REST API
    public ArrayList<Sensor> AlertSensorList() throws SQLException {

        //SEnsor object
        Sensor sensor = null;

        //ArrayList
        ArrayList<Sensor> list = new ArrayList<>();

        try {


            //Create URL object , targeting REST APIs Endpoint to Get Sensors
            URL url = new URL("http://localhost:8080/api/fireAlarmSystem/sensors");

            // Create a HTTPURL Connection object and open a connection
```

```
HttpURLConnection con = (HttpURLConnection) url.openConnection();

 // Set request method into GET
con.setRequestMethod("GET");

// Set the Request  Property to accept JSON
con.setRequestProperty("Accept", "application/json");

//Get the Responsecode
int responseCode = con.getResponseCode();

// Read the response
Reader reader = null;



//Check Responses
if (responseCode >= 200 && responseCode <= 299) {
    reader = new BufferedReader(new InputStreamReader(con.getInputStream(), "utf-8"));
} else {
    reader = new BufferedReader(new InputStreamReader(con.getErrorStream(), "utf-8"));
}

//Create GSON Object
Gson gson = new Gson();
try {

    //Create Sensor object Array from JSON Response
    Sensor[] temp = gson.fromJson(reader, Sensor[].class);

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

        //Check if Sensor is Alerted
        if (Integer.parseInt(temp[i].getCo2_level()) > 5 || Integer.parseInt(temp[i].getSmoke_level()) > 5) {
                //add to Arraylist
                list.add(temp[i]);
              }
            }

        //Catch Exceptions
          } catch (Exception e) {
            System.out.println(e);
          }

        } catch (MalformedURLException ex) {
          Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
          Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
        }

        //Return ArrayList
        return list;
      }

      //Create table USER and Add Data if not Exist
      private void createUserTable() throws SQLException {

        //Table Properties
        String TABLENAME = "user";
        String sqlCreate = "CREATE TABLE IF NOT EXISTS " + TABLENAME
            + "  (userId      INTEGER,"
            + "   Name     VARCHAR(50),"
            + "   Password VARCHAR(20))";

        //Get Database Connection
        Statement stmt = con.createStatement();
        //Execute Query
        stmt.execute(sqlCreate);

        String sqlIsEmpty = "SELECT * from " + TABLENAME;
        ResultSet rs = stmt.executeQuery(sqlIsEmpty);

        // checking if ResultSet is empty
        if (rs.next() == false) {

            //if ReultSet is Empty Insert Data to Table
            String sql = "INSERT INTO user VALUES (?,?,?)";

            //Get Database Connection
            PreparedStatement stm1 = con.prepareStatement(sql);

            int ID = 123;
            String NAME = "Steve";
            String PASS = "123";

            stm1.setObject(1, ID);
            stm1.setObject(2, NAME);
            stm1.setObject(3, PASS);

            //Execute Query
            boolean createStat = stm1.executeUpdate() > 0;

        }

    }
}
```

Service.java

```
package RMIServer;

import Models.Sensor;
import Models.User;
import java.rmi.Remote;
import java.sql.SQLException;
import java.util.ArrayList;

public interface Service extends Remote{

    User getUser(String id) throws Exception ;
    boolean addSensor(Sensor sensor) throws Exception;

    boolean updateSensor(Sensor cus)throws Exception;
    boolean deleteSensor(String Id)throws Exception;
    Sensor getSensor(String Id)throws Exception;
    ArrayList<Sensor> getSensorList()throws Exception;
   ArrayList<Sensor> AlertSensorList()throws Exception;

}
```

4.5 REST API

Application.java

```
package net.codeJava.demofire;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
@EnableAutoConfiguration
public class Application {

                                        public static void main(String[] args) {
```

```
                SpringApplication.run(Application.class, args);
               }


}
```

EmailGenerator.java

```
package net.codeJava.demofire;

import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;



//This class Hold functions and propertis of the Email Generator
public class EmailGenerator {

               //senders email
               final String senderEmail = "grpsender@gmail.com";
               //senders password
               final String senderPassword = "grpGRP123";
               //mail server
               final String Server = "smtp.gmail.com";
               //port number
               final String Port = "465";
               //Receiver Email
               String receiverEmail = null;
               //email subject
               static String Subject ;
               //email body
               static String Body;

               public EmailGenerator(String receiverEmail, String Subject, String Body) {
```

```
        this.receiverEmail = receiverEmail;
        this.Subject = Subject;
        this.Body = Body;

        try {

            //set properties

            Properties properties = new Properties();
            properties.put("mail.smtp.user", senderEmail);
            properties.put("mail.smtp.host", Server);
            properties.put("mail.smtp.port", Port);
            properties.put("mail.smtp.starttls.enable", "true");
            properties.put("mail.smtp.auth", "true");
            properties.put("mail.smtp.socketFactory.port", Port);
            properties.put("mail.smtp.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
            properties.put("mail.smtp.socketFactory.fallback", "false");
            SecurityManager security = System.getSecurityManager();

            //generate a authenticator

            Authenticator authentiator = new Authenticator();

            //get a session

            Session session = Session.getInstance(properties, authentiator);

            //generate message instance from session

            MimeMessage msg = new MimeMessage(session);

            //set message properties
```

```
                    msg.setContent(this.Body, "text/html");
                    msg.setSubject(this.Subject);
                    msg.setFrom(new InternetAddress(senderEmail));
                    msg.addRecipient(Message.RecipientType.TO, new
InternetAddress(this.receiverEmail));

                    //send Email
                    Transport.send(msg);

                    System.out.println("Email Sent");
               }

               catch (Exception e) {
                    System.out.println("Error");
               }

              }
//authenticator Class for sender email authentication
              public class Authenticator extends javax.mail.Authenticator {
               public PasswordAuthentication getPasswordAuthentication() {

                    //return authentication result status
                    return new PasswordAuthentication(senderEmail, senderPassword);
               }
              }
}
```

Sensor.java

```
package net.codeJava.demofire;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "sensor")
public class Sensor implements Serializable {

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)

        @Column(name = "id")
        private Integer id;

        @Column(name = "location_floorNo")
        private Integer location_floorNo;

        @Column(name = "location_roomNo")
        private Integer location_roomNo;

        @Column(name = "co2_level")
        private Integer co2_level;

        @Column(name = "smoke_level")
        private Integer smoke_level;

        @Column(name = "status")
        private String  status;
```

```
public Sensor() {
  super();


}


public Sensor(Integer id, Integer location_floorNo, Integer location_roomNo, Integer
co2_level, Integer smoke_level,
          String status) {


  this.id = id;
  this.location_floorNo = location_floorNo;
  this.location_roomNo = location_roomNo;
  this.co2_level = co2_level;
  this.smoke_level = smoke_level;
  this.status = status;
}




public Integer getId() {
  return id;
}


public void setId(Integer id) {
  this.id = id;
}


public Integer getLocation_floorNo() {
  return location_floorNo;
}


public void setLocation_floorNo(Integer location_floorNo) {
  this.location_floorNo = location_floorNo;
}
```

```
public Integer getLocation_roomNo() {
  return location_roomNo;
}

public void setLocation_roomNo(Integer location_roomNo) {
  this.location_roomNo = location_roomNo;
}

public Integer getCo2_level() {
  return co2_level;
}

public void setCo2_level(Integer co2_level) {
  this.co2_level = co2_level;
}

public Integer getSmoke_level() {
  return smoke_level;
}

public void setSmoke_level(Integer smoke_level) {
  this.smoke_level = smoke_level;
}

public String getStatus() {
  return status;
}

public void setStatus(String status) {
  this.status = status;
}
```

```
}
```

SensorController.java

```
@RestController
@CrossOrigin(origins="http://localhost:3000")
public class SensorController {


            //SMS Sender Properties

            String FROMNO="+12025195518";
            String TONO="+94714009185";

            String ACCOUNT_SID="ACf3aab5d2d322defcc4bc9b9882bae765";
            String AUTH_TOKEN="77c0878816f91875bf18d1b6da6e09b6";

            SMS sms= new SMS();



            //EMAIL Sender Properties
            final String EMAILRECIEVER="grpreceiver@gmail.com";
            String EMAILBODY;

            //array list to hold alert triggered sensors IDs
            ArrayList<Integer> temp =new ArrayList<Integer>();

            //holds the status of if a given id is in the "temp" array or not
            boolean stat=false;
```

```
@Autowired
private SensorService service;

@GetMapping("/api/fireAlarmSystem/sensors")
public List<Sensor> list() {
  return service.listAll();

}

@GetMapping("/api/fireAlarmSystem/sensors/{id}")
public ResponseEntity<Sensor> get(@PathVariable Integer id) {
try {
  Sensor sensor = service.get(id);
  return new ResponseEntity<Sensor>(sensor, HttpStatus.OK);

}catch (NoSuchElementException e) {
  return new ResponseEntity<Sensor>(HttpStatus.NOT_FOUND);

  }
}



@PostMapping("/api/fireAlarmSystem/Add")
public void add(@RequestBody Sensor sensor) {

  System.out.println("Sensor Recievedis :" + sensor);
  service.save(sensor);
}


@PutMapping("/api/fireAlarmSystem/Update/{id}")
```

```
public ResponseEntity<?> update(@RequestBody Sensor sensor,@PathVariable
Integer id) {

    try {

    //get sensor by ID
    Sensor existSensor = service.get(id);

    //update Sensor
    service.save(sensor);

    //set initial status true
    stat=true;

    //Check if the received sensor is a alerted(Smoke level or CO2 level >5) sensor
    if(sensor.getCo2_level()>5 || sensor.getSmoke_level()>5) {

        //check if temp array is empty
            if (temp.isEmpty()) {

                // if temp array is empty add the first sensor to temp array
                temp.add(sensor.getId());

            //Sms Message
            String MESSAGE="Fire Alert Detected on the room no
"+sensor.getLocation_roomNo() +"of the floor no " +  sensor.getLocation_floorNo()+ ". Detected CO2 Level is
"+ sensor.getCo2_level()+" and Smoke level is "+sensor.getSmoke_level()+". Details are detected by Sensor ID
"+ sensor.getId();

            //Authorized Message API Account
             Twilio.init(ACCOUNT_SID, AUTH_TOKEN);
```

```
                    //Send SMS

                     Message message = Message.creator(new
PhoneNumber(TONO),new PhoneNumber(FROMNO), MESSAGE).create();

                     //print message ID
                     System.out.println(message.getSid());



                         //Email Body
                         EMAILBODY="<h1 style=\"color:red;\">Fire Alert
Detected</h1><h2>Sensor ID : "+sensor.getId().toString()+"<br> Floor No : "+
sensor.getLocation_floorNo()+"<br> Room No : "+ sensor.getLocation_roomNo()+"</h2><h2
style=\"color:red;\"> CO2 Level : "+sensor.getCo2_level()+"<br>Smoke Level :
"+sensor.getSmoke_level()+"</h2>";
                         // send email
                         new EmailGenerator(EMAILRECIEVER, "Fire Alert",
EMAILBODY);
                    }

                    else
                        for (int i : temp)
                            if (sensor.getId() == i) {
                                stat = false;

                            }
                         }

                         if (stat) {
                             temp.add(sensor.getId());

                         // send SMS
```

```
                                        String MESSAGE = "Fire Alert Detected on the room no " +
sensor.getLocation_roomNo()+ "of the floor no " + sensor.getLocation_floorNo() + ". Detected CO2 Level is "+
sensor.getCo2_level() + " and Smoke level is " + sensor.getSmoke_level()+ ". Details are detected by Sensor ID
" + sensor.getId();


//Authorized Message API Account

Twilio.init(ACCOUNT_SID, AUTH_TOKEN)

//Send SMS

Message message = Message.creator(new PhoneNumber(TONO), new PhoneNumber(FROMNO),
MESSAGE).create();

//Print Message ID

System.out.println(message.getSid()

//Email Body

EMAILBODY="<h1 style=\"color:red;\">Fire Alert Detected</h1><h2>Sensor ID :
"+sensor.getId().toString()+"<br> Floor No : "+ sensor.getLocation_floorNo()+"<br> Room No : "+
sensor.getLocation_roomNo()+"</h2><h2 style=\"color:red;\"> CO2 Level :
"+sensor.getCo2_level()+"<br>Smoke Level : "+sensor.getSmoke_level()+"</h2>";

// send email

new EmailGenerator(EMAILRECIEVER, "Fire Alert", EMAILBODY);

}

}}


                                        return new ResponseEntity<>(HttpStatus.OK);

                                       }catch (NoSuchElementException e) {


                                        return new ResponseEntity<>(HttpStatus.NOT_FOUND);

                                        }

                                       }


                                       @DeleteMapping("/api/fireAlarmSystem/Delete/{id}")

                                       public void delete(@PathVariable Integer id) {

                                        service.delete(id)

}
```

SensorRepository.java

```
package net.codeJava.demofire;

import org.springframework.data.jpa.repository.JpaRepository;

public interface SensorRepository extends JpaRepository<Sensor,Integer>{}
```

SensorService.java

```
package net.codeJava.demofire;
import java.util.List;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class SensorService {

        @Autowired
        private SensorRepository repo;

        public List<Sensor> listAll(){
          return repo.findAll();
        }

        public void save(Sensor sensor) {
          repo.save(sensor);
        }

        public Sensor get(Integer id) {
          return repo.findById(id).get();

        }
```

```
                              public void delete(Integer id){
                                repo.deleteById(id);
                              }
}
```

4.6 Sensor Application

SensorClientServiceImpl.java

```
public class SensorClientServiceImpl implements SensorClientServices{

    //HTTPURL Connection Object
    HttpURLConnection conn;

    //Database Connection Object



    @Override
    //Check if Sensor is in InActive State
    public boolean checkStatusInActive(String ID) throws SQLException {

        //hold the status
        boolean availability =false;

        try {

            //Sensor Object
            Sensor sensor = getSensor(ID);

            //hold sensor initial status
            String status = sensor.getStatus();
            //check status
            if(status.equalsIgnoreCase("Inactive")){

                //set Sensor is open to Activate
                availability= true;

             //Check if sensor is already Activated
            }else if(status.equalsIgnoreCase("active")){
                    JPanel pane5 = new JPanel();
                    JOptionPane.showMessageDialog(pane5, "Sensor Is Already Activated", "Activation Faild",
JOptionPane.ERROR_MESSAGE);
                    availability= false;
            }
        } catch (Exception ex) {
            Logger.getLogger(SensorClientServiceImpl.class.getName()).log(Level.SEVERE, null, ex);
            JPanel pane3 = new JPanel();
                    JOptionPane.showMessageDialog(pane3, "Sensor is Not in the System. Please Enter a Valid Sensor
ID", "Activation Unsuccessfull", JOptionPane.ERROR_MESSAGE);
        }

        //return status
        return availability;

    }

    @Override

    //Update Status of Sensor To Active
    public boolean upadteStateActive(Sensor sensor) throws SQLException {

         //Method success status
         boolean stat=false;
```

```
try {

    //Get Sensor ID
    String id = sensor.getId();

    //Set Status to Active
    sensor.setStatus("active");

    //crete object mapper
    ObjectMapper mapper = new ObjectMapper();

    //covert sensor object into JSON String
    String jsonString = mapper.writeValueAsString(sensor);



    //Connect endpoint
    URL url = new URL("http://localhost:8080/api/fireAlarmSystem/Update/"+id);

    //intiate connection
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();

    //send put Request
    conn.setRequestMethod("PUT");

    // Set Request Property
    conn.setRequestProperty("Content-Type", "application/json; charset=UTF-8");

    //Set property to accept JSON
    conn.setRequestProperty("Accept", "application/json");

    //enable output Stream
    conn.setDoOutput(true);
```

```
String data = jsonString;

//Write data in to output object

OutputStream stream = conn.getOutputStream();
byte[] input = data.getBytes("utf-8");
stream.write(input, 0, input.length);

//get Response Code
int responseCode = conn.getResponseCode();

//initate a reader
Reader reader = null;


//Check Responses
if (responseCode >= 200 && responseCode <= 299) {


   reader = new BufferedReader(new InputStreamReader(conn.getInputStream(), "utf-8"));

   //update status state
   stat = true;

} else {
   reader = new BufferedReader(new InputStreamReader(conn.getErrorStream(), "utf-8"));


   //update status state
   stat = false;
}
```

```
        //catch exceptions
        } catch (MalformedURLException ex) {

        } catch (IOException ex) {

        }

        //return status
        return stat;

    }
    public boolean upadteStateInActive(Sensor sensor) throws SQLException {

        //Method success status
         boolean stat=false;

        try {

            //set Properties
            String id = sensor.getId();
            sensor.setStatus("Inactive");
            sensor.setCo2_level("0");
            sensor.setSmoke_level("0");

            //write data to json using object mapper

            ObjectMapper mapper = new ObjectMapper();
            String jsonString = mapper.writeValueAsString(sensor);

            System.out.println("Jason String: " + jsonString);

            //Connect endpoint
            URL url = new URL("http://localhost:8080/api/fireAlarmSystem/Update/"+id);
```

```
//intiate connection
HttpURLConnection conn = (HttpURLConnection) url.openConnection();

//send put PUT
conn.setRequestMethod("PUT");

// Set Request Property
conn.setRequestProperty("Content-Type", "application/json; charset=UTF-8");

//Set property to accept JSON
conn.setRequestProperty("Accept", "application/json");

//enable output Stream
conn.setDoOutput(true);

//Write data to output objects
String data = jsonString;
OutputStream stream = conn.getOutputStream();
byte[] input = data.getBytes("utf-8");
stream.write(input, 0, input.length);



//get Response Code
int responseCode = conn.getResponseCode();

//initaite reader
Reader reader = null;

//Check Responses
if (responseCode >= 200 && responseCode <= 299) {
```

```
            reader = new BufferedReader(new InputStreamReader(conn.getInputStream(), "utf-8"));
            stat = true;
        } else {
            reader = new BufferedReader(new InputStreamReader(conn.getErrorStream(), "utf-8"));



            stat = false;
        }

    //catch exceptions
    } catch (MalformedURLException ex) {

    } catch (IOException ex) {

    }

    //return status
    return stat;
}




public Sensor getSensor(String Id) throws Exception {

    //sensor object
    Sensor sensor = null;




    //Connect endpoint
    URL url = new URL("http://localhost:8080/api/fireAlarmSystem/sensors/"+Id);

    //intiate connection
```

```
        HttpURLConnection con = (HttpURLConnection) url.openConnection();

        //send put GET
        con.setRequestMethod("GET");

        //Set property to accept JSON
        con.setRequestProperty("Accept", "application/json");



        //get Response Code
        int responseCode = con.getResponseCode();

        //initaite reader
        Reader reader = null;

        //Check Responses
        if (responseCode >= 200 && responseCode <= 299) {
            reader = new BufferedReader(new InputStreamReader(con.getInputStream(), "utf-8"));
        } else {
            reader = new BufferedReader(new InputStreamReader(con.getErrorStream(), "utf-8"));
        }

// parsing the JSON response to a Java Object
        Gson gson = new Gson();
        try {
            sensor = gson.fromJson(reader, Sensor.class);



        } catch (Exception e) {
            System.out.println(e);
        }

                                if(sensor!=null){
```

```
        return sensor;
        }
        else{
            return null;
        }



    }

    @Override
    public boolean upadteSensor(Sensor sensor,String CO2,String Smoke) throws SQLException {

        //Method success status
        boolean stat=false;

        try {

            //set Properties

            String id = sensor.getId();
            sensor.setCo2_level(CO2);
            sensor.setSmoke_level(Smoke);

            //write data to json using object mapper
            ObjectMapper mapper = new ObjectMapper();
            String jsonString = mapper.writeValueAsString(sensor);

            System.out.println("Jason String: " + jsonString);

            //Connect endpoint
            URL url = new URL("http://localhost:8080/api/fireAlarmSystem/Update/"+id);
```

```
//intiate connection
HttpURLConnection conn = (HttpURLConnection) url.openConnection();

//send put PUT
conn.setRequestMethod("PUT");

// Set Request Property
conn.setRequestProperty("Content-Type", "application/json; charset=UTF-8");

//Set property to accept JSON
conn.setRequestProperty("Accept", "application/json");

//enable output Stream
conn.setDoOutput(true);

//write data into output object
String data = jsonString;
OutputStream stream = conn.getOutputStream();
byte[] input = data.getBytes("utf-8");
stream.write(input, 0, input.length);



//get Response Code
int responseCode = conn.getResponseCode();

//initaite reader
Reader reader = null;

//Check Responses
if (responseCode >= 200 && responseCode <= 299) {


    reader = new BufferedReader(new InputStreamReader(conn.getInputStream(), "utf-8"));
```

```
            stat = true;
        } else {
            reader = new BufferedReader(new InputStreamReader(conn.getErrorStream(), "utf-8"));


            stat = false;
        }



    } catch (MalformedURLException ex) {

    } catch (IOException ex) {

    }

    //return status
    return stat;

}
//Check if sensor is mannual Deactivated by Admin Panel
public boolean checkTimerInActive(String ID) throws SQLException {

    //Sensor Inactivity status
    boolean availability =false;

    try {

        //Get Properties
        Sensor sensor = getSensor(ID);
        String status = sensor.getStatus();


        //check inactivity
```

```
            if(status.equalsIgnoreCase("Inactive")){

                availability= true;
            }else {
                    availability= false;
            }

         //catch exceptions

        } catch (Exception ex) {
            System.out.println("Timer Active catcher occurs fails");
        }

        //return sensor status
        return availability;

    }
}
```

SensorClientServices.java

```
package Services;

import Models.Sensor;
import java.sql.SQLException;

public interface SensorClientServices {
    public boolean checkStatusInActive(String ID)throws SQLException;
    public boolean upadteStateActive(Sensor sensor)throws SQLException;
    public boolean upadteStateInActive(Sensor sensor)throws SQLException;
    public boolean upadteSensor(Sensor sensor,String CO2,String Smoke)throws SQLException;
    public Sensor getSensor(String Id) throws Exception;
    public boolean checkTimerInActive(String ID) throws SQLException;
}
```

SensorClientActivator.java

```
public class SensorClientActivator extends javax.swing.JFrame {



  public SensorClientActivator() {
    initComponents();
  }

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {





    try {
      //Get Sensor Service Instance
      SensorClientServiceImpl service = new SensorClientServiceImpl();



      //get properties
      String ID = id.getText();





        //check if sensor was inActive
        if(service.checkStatusInActive(ID)){



        //get Sensor ID
          Sensor sensor=service.getSensor(ID);


          //display success dialog
```

```
                JPanel pane3 = new JPanel();
                JOptionPane.showMessageDialog(pane3, "Sensor is Activated. You may re-directed to Server
Panel", "Activation Successfull", JOptionPane.INFORMATION_MESSAGE);

                //Update Sensor Status to Active
                service.upadteStateActive(sensor);

                //Display Sensor panel JFRAME
                new SensorPanel(sensor).setVisible(true);
              }
            //catch exceptions

        } catch (NotBoundException ex) {

            System.out.println("Error 1");
            System.err.println(ex.getMessage());
        } catch (MalformedURLException ex) {
            System.out.println("Error 2");
            System.err.println(ex.getMessage());
        } catch (RemoteException ex) {
            System.out.println("Error 3");
            System.err.println(ex.getMessage());
            JPanel pane2 = new JPanel();
              JOptionPane.showMessageDialog(pane2, "Server Not Found.", " Server Error",
JOptionPane.ERROR_MESSAGE);

        } catch (NullPointerException ex1) {
            Logger.getLogger(SensorClientActivator.class.getName()).log(Level.SEVERE, null, ex1);
            JPanel pane2 = new JPanel();
              JOptionPane.showMessageDialog(pane2, "Some error occured. Please try again.", "Activation
Failed", JOptionPane.ERROR_MESSAGE);
        } catch (Exception ex2) {
            Logger.getLogger(SensorClientActivator.class.getName()).log(Level.SEVERE, null, ex2);
            JPanel panel = new JPanel();
            JOptionPane.showMessageDialog(panel, "Some error occured. Please try again.", "Activation Failed",
JOptionPane.ERROR_MESSAGE);
        }




    }
```

SensorPanel.java

```
public class SensorPanel extends javax.swing.JFrame {

    SensorClientServiceImpl service = new SensorClientServiceImpl();

    //Set Timer to detectDeactivation
    Timer timer;

    //Set Timer to detect AutoUpdate
    Timer timer2;

    //Declare a Sensor Object
    Sensor sensor;

    //Sensor ID Property
    String SensorID;

    /**
     * Creates new form CustomerReg
     */
    public SensorPanel() {
        initComponents();

        //avoid unauthorized access
        dispose();
        JPanel pane5 = new JPanel();
        JOptionPane.showMessageDialog(pane5, "Unauthorised Access. Please Activate", "ByPassed Attempt",
JOptionPane.ERROR_MESSAGE);

        System.exit(-1);
    }

    public SensorPanel(Sensor s1) {

        initComponents();

        //set object
        sensor = s1;

        //set Jframe labels value according to the Sensor object properties
        setDetails(sensor);

        //Start detect Deactivations
        detectDeactiation();

        //start autoUpdate
        AutoUpdate();

        //set Sensor ID
        this.SensorID = sensor.getId();

    }

private void formWindowClosed(java.awt.event.WindowEvent evt) {

        //stop repeting timer
        timer.setRepeats(false);
        //stop timer , timer 2
        timer.stop();
        timer2.stop();
        try {

            //set Sensor Status "Inactive"
            service.upadteStateInActive(sensor);

        } catch (SQLException ex) {
            JPanel pane5 = new JPanel();
            JOptionPane.showMessageDialog(pane5, "Sensor Update Error Occured.", "Error",
JOptionPane.ERROR_MESSAGE);
        }

    }

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

        //stop repeting timer
        timer.setRepeats(false);

        //stop timer , timer 2
        timer.stop();
        timer2.stop();

        try {
            //set Sensor Status "Inactive"
            service.upadteStateInActive(sensor);
        } catch (SQLException ex) {
            JPanel pane5 = new JPanel();
            JOptionPane.showMessageDialog(pane5, "Sensor Update Error Occured.", "Error",
JOptionPane.ERROR_MESSAGE);
        }

        dispose();
    }

public void setDetails(Sensor sensor) {

        //set ID
        ID.setText(sensor.getId());
        //Set Room No
        room.setText(sensor.getLocation_roomNo());
        //set Floor No
        floor.setText(sensor.getLocation_floorNo());

        //set Co2 Level
        if (sensor.getCo2_level().equals("1")) {
            co2.setSelectedIndex(0);
        } else if (sensor.getCo2_level().equals("2")) {
            co2.setSelectedIndex(1);

        } else if (sensor.getCo2_level().equals("3")) {
            co2.setSelectedIndex(2);

        } else if (sensor.getCo2_level().equals("4")) {
            co2.setSelectedIndex(3);

        } else if (sensor.getCo2_level().equals("5")) {
            co2.setSelectedIndex(4);

        } else if (sensor.getCo2_level().equals("6")) {
            co2.setSelectedIndex(5);

        } else if (sensor.getCo2_level().equals("7")) {
            co2.setSelectedIndex(6);
```

```
} else if (sensor.getCo2_level().equals("8")) {
    co2.setSelectedIndex(7);

} else if (sensor.getCo2_level().equals("9")) {
    co2.setSelectedIndex(8);

} else if (sensor.getCo2_level().equals("10")) {
    co2.setSelectedIndex(9);

}

//Set Smoke Level
if (sensor.getSmoke_level().equals("1")) {
    smoke.setSelectedIndex(0);
} else if (sensor.getSmoke_level().equals("2")) {
    smoke.setSelectedIndex(1);
} else if (sensor.getSmoke_level().equals("3")) {
    smoke.setSelectedIndex(2);
} else if (sensor.getSmoke_level().equals("4")) {
    smoke.setSelectedIndex(3);
} else if (sensor.getSmoke_level().equals("5")) {
    smoke.setSelectedIndex(4);
} else if (sensor.getSmoke_level().equals("6")) {
    smoke.setSelectedIndex(5);
} else if (sensor.getSmoke_level().equals("7")) {
    smoke.setSelectedIndex(6);
} else if (sensor.getSmoke_level().equals("8")) {
    smoke.setSelectedIndex(7);
} else if (sensor.getSmoke_level().equals("9")) {
    smoke.setSelectedIndex(8);
} else if (sensor.getSmoke_level().equals("10")) {
    smoke.setSelectedIndex(9);
```

```
        }

    }

    //Detect if Admin Deactivted the Sensor by Admin Panel
    public void detectDeactiation() {

        //initiate timer
        timer = new Timer(100, new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent arg0) {

                try {
                    //Check if sensor is inActive
                    if (service.checkTimerInActive(sensor.getId())) {

                        //Stop Timer
                        timer.setRepeats(false);

                        timer.stop();

                        //diplay error dialog
                        JPanel pane5 = new JPanel();
                        JOptionPane.showMessageDialog(pane5, "Sensor ID " + sensor.getId() + "  is Deactivated by
Admin.", "Manual Deactivation Occurred ", JOptionPane.ERROR_MESSAGE);

                        dispose();

                    }

                    //catch exceptions
                } catch (SQLException ex) {
                    Logger.getLogger(SensorPanel.class.getName()).log(Level.SEVERE, null, ex);
            }

          }

        });

        //set Timer prperties
        timer.setRepeats(true); // Only execute once
        timer.start();
    }

    //Auto update Sensor Details after 10 seconds
    public void AutoUpdate() {

        //initiate timer
        timer2 = new Timer(1000, new ActionListener() {

            //Time to auto Update in seconds
            int initial = 10;

            //Counter
            int c = 0;

            @Override
            public void actionPerformed(ActionEvent arg0) {

                //Check counter equals to iniatial time
                if (c == initial) {

                    c = 0;
                }
                //increment counter
                c++;
            //seconds remaining
            int timeup = initial - c;

            //Set Seconds value to Display in the jframe Label
            seconds1.setText(String.valueOf((timeup)));

            try {

                //Check 10 seconds past
                if (timeup == 0) {

                    //update sensor details in the database through REST API
                    service.upadteSensor(sensor, co2.getSelectedItem().toString(),
smoke.getSelectedItem().toString());
                }

                //catch exceptions
            } catch (SQLException ex) {
                Logger.getLogger(SensorPanel.class.getName()).log(Level.SEVERE, null, ex);

                JPanel pane5 = new JPanel();
                JOptionPane.showMessageDialog(pane5, "Sensor Update Error Occured.", "Error",
JOptionPane.ERROR_MESSAGE);
            } catch (Exception ex) {
                Logger.getLogger(SensorPanel.class.getName()).log(Level.SEVERE, null, ex);
            }

        }

    });

    //set Timer Properties
    timer2.setRepeats(true); // Only execute once
    timer2.start();
}
```