BSc (Hons) in Information Technology Year 3

DS Assignment 2 Report

Semester 1

SE3020 – DS

Registration No.	Student Name
IT18025354	U. R.P. Pramuditha
IT18018042	G. G. K. M. Rathnakumara
IT18511680	D. D. M. S. M. Dissanayake
IT17148450	D. H. U. Sandamini

Introduction

A fire Alarm Monitoring system is developed for Distributed System module of third year first semester as undergraduates of Bsc.(sp) in Software Engineering of SLIIT.

The system consists of 4 subsystems.

- Sensor
- Rest API
- RMI Server/RMI Desktop client
- Web Application

Sensor updates CO_2 and smoke level of each fire alarm. Desktop client connects to the API through RMI. Desktop client is responsible of registering users, login users, adding deleting updating fire alarms. Both the web client and the desktop client are able to view the status of fire alarms which are updated time to time. SMS and email are automatically send to the user by the RMI.

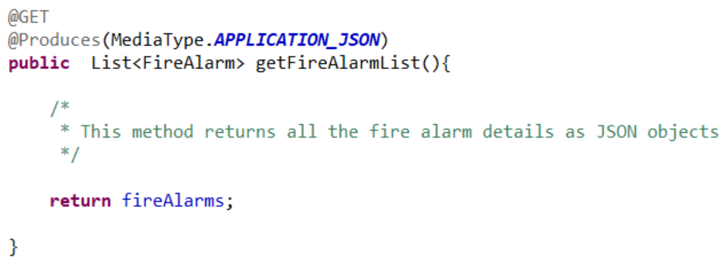
Database firealarmmonitor comprises with two tables firealarm and user. firealarm table contains sensor id, location (floor number and room number), active/inactive status smoke level and CO_2 level of the respective alarms. user table stores details of the users of desktop application (name, email, password).

Methodology

Whole project was developed using java language and in order to build the REST API, Jersey RESTful web services framework is used. Apache Tomcat v9 is used to deploy the system. Mysql server is used as the DBMS. To send mails Java mail API is used while Twillio API is used for sending SMS.

Service interfaces of the API

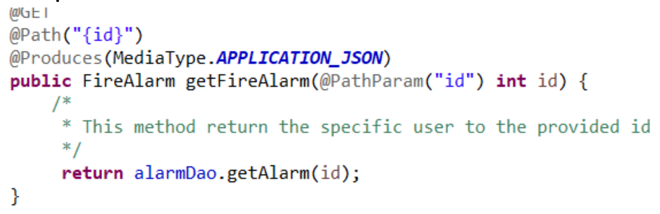
- getFireAlarmList()
    - ➢ Provide interface to get the status of all the fireAlarms



```
@GET
@Produces(MediaType.APPLICATION_JSON)
public  List<FireAlarm> getFireAlarmList(){

    /*
     * This method returns all the fire alarm details as JSON objects
     */

    return fireAlarms;

}
```

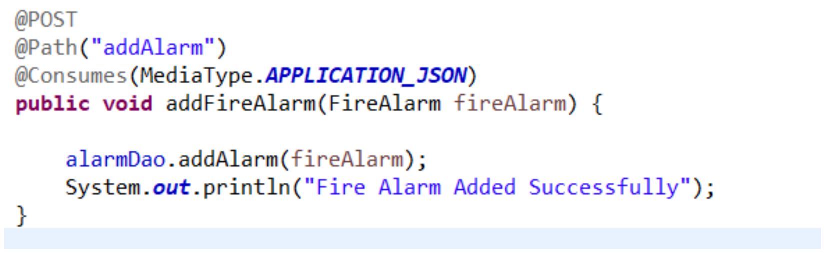
- getFireAlarm()

- Provide the details of a specific fire alarm when the id of the fire alarm is provided as a pathParam



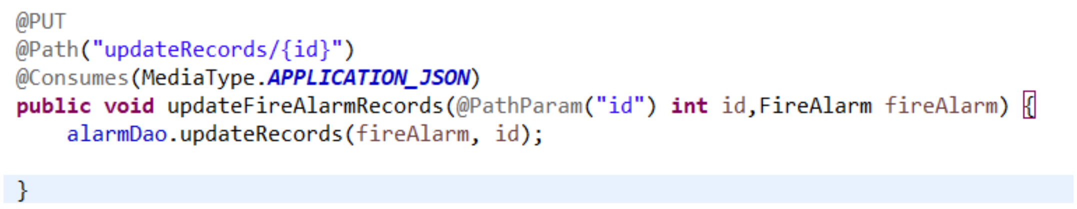
```
@GET
@Path("{id}")
@Produces(MediaType.APPLICATION_JSON)
public FireAlarm getFireAlarm(@PathParam("id") int id) {
    /*
     * This method return the specific user to the provided id
     */
     return alarmDao.getAlarm(id);
}
```

- addFireAlarm()
    - Used as the interface to add new fire alarms to the database



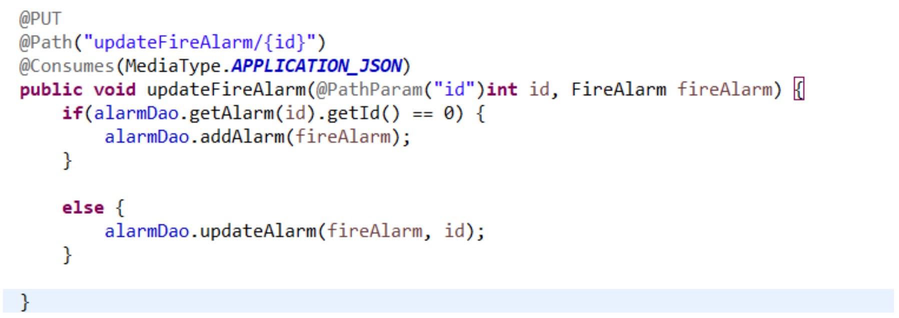
```
@POST
@Path("addAlarm")
@Consumes(MediaType.APPLICATION_JSON)
public void addFireAlarm(FireAlarm fireAlarm) {

    alarmDao.addAlarm(fireAlarm);
    System.out.println("Fire Alarm Added Successfully");
}
```

- updateFireAlarmRecords()
    - Provide interface for updating a fire alarm's status, CO_2 level, smoke level of when the specific alarm id is provided as path param



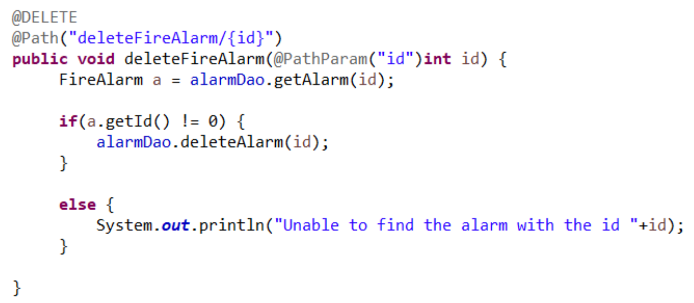
```
@PUT
@Path("updateRecords/{id}")
@Consumes(MediaType.APPLICATION_JSON)
public void updateFireAlarmRecords(@PathParam("id") int id,FireAlarm fireAlarm) {
    alarmDao.updateRecords(fireAlarm, id);

}
```

- updateFireAlarm()
    - Provide interface for update service of the fire alarms where location details of a specific alarm is send as a JSON object along with the specific alarm,s id as a path param



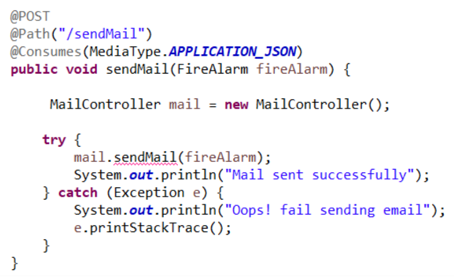
```
@PUT
@Path("updateFireAlarm/{id}")
@Consumes(MediaType.APPLICATION_JSON)
public void updateFireAlarm(@PathParam("id")int id, FireAlarm fireAlarm) {
    if(alarmDao.getAlarm(id).getId() == 0) {
        alarmDao.addAlarm(fireAlarm);
    }

    else {
        alarmDao.updateAlarm(fireAlarm, id);
    }

}
```

- deleteFireAlarm()
    - Interface for deleting a fire alarm when a specific alarm's id is provided.



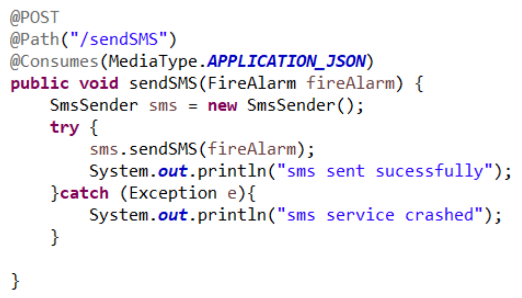
```
@DELETE
@Path("deleteFireAlarm/{id}")
public void deleteFireAlarm(@PathParam("id")int id) {
     FireAlarm a = alarmDao.getAlarm(id);

     if(a.getId() != 0) {
          alarmDao.deleteAlarm(id);
     }

     else {
          System.out.println("Unable to find the alarm with the id "+id);
     }

}
```

- sendMail()
    - ➢ Interface for mail sending service where fire alarm details are provided as JSON object



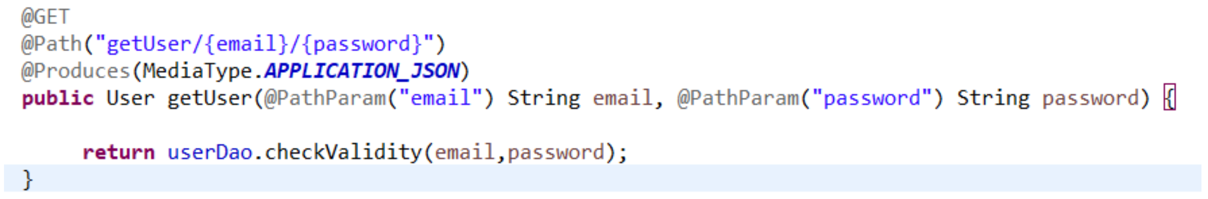
```
@POST
@Path("/sendMail")
@Consumes(MediaType.APPLICATION_JSON)
public void sendMail(FireAlarm fireAlarm) {

     MailController mail = new MailController();

    try {
        mail.sendMail(fireAlarm);
        System.out.println("Mail sent successfully");
    } catch (Exception e) {
        System.out.println("Oops! fail sending email");
        e.printStackTrace();
    }
}
```

- sendSMS()
    - ➢ Interface SMS sending service where required fire alarm details are provided as JSON object



```
@POST
@Path("/sendSMS")
@Consumes(MediaType.APPLICATION_JSON)
public void sendSMS(FireAlarm fireAlarm) {
    SmsSender sms = new SmsSender();
    try {
        sms.sendSMS(fireAlarm);
        System.out.println("sms sent sucessfully");
    }catch (Exception e){
        System.out.println("sms service crashed");
    }

}
```

- getUser()
    - ➢ Interface for the service checking validity of the users when the email and password are provided.



```
@GET
@Path("getUser/{email}/{password}")
@Produces(MediaType.APPLICATION_JSON)
public User getUser(@PathParam("email") String email, @PathParam("password") String password) {

    return userDao.checkValidity(email,password);
}
```

- addUser()
  - ➢ Interface for adding new user when email, password and name are provided as a JSON object

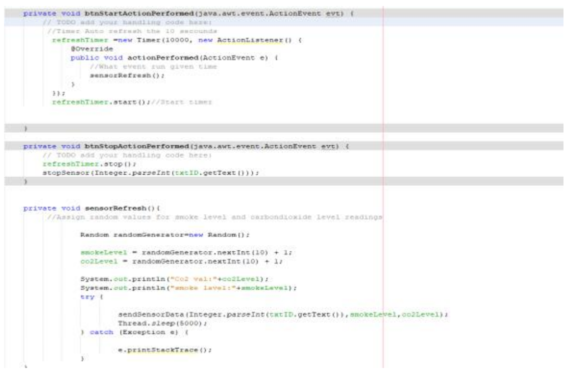
```java
@POST
@Path("addUser")
@Consumes(MediaType.APPLICATION_JSON)
public void registerUser(User user) {
    userDao.addUser(user);
    System.out.println("New User Added Successfully");
}
```

➢ Updating sensor Details

Desktop application user update the specific sensor's active status, CO_2 level, smoke level of the particular fire Alarm.

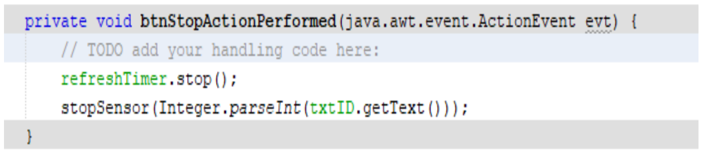
User can start a sensor by providing its ID. When the sensor starts the status is set to active. sensorRefresh() function is invoked when the start button is clicked where random values will be set to CO_2 level periodically with a time interval of 15s, smoke level and the status is set to 1 and id field is set from the text field value . This method call sendSensorData() where the above data is send as a fireAlarm JSON object to the API using the URL `"http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/updateRecords/"+id.`

Sensor Start Implementation and Refresh



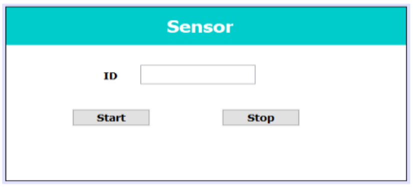
```java
private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
     //Timer Auto refresh the 10 seccunds
      refreshTimer =new Timer(10000, new ActionListener() {
          @Override
          public void actionPerformed(ActionEvent e) {
              //What event run given time
              sensorRefresh();
          }
      });
      refreshTimer.start();//Start timer


}

private void btnStopActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    refreshTimer.stop();
    stopSensor(Integer.parseInt(txtID.getText()));
}


private void sensorRefresh(){
     //Assign random values for smoke level and carbondioxide level readings

            Random randomGenerator=new Random();

            smokeLevel = randomGenerator.nextInt(10) + 1;
            co2Level = randomGenerator.nextInt(10) + 1;

            System.out.println("Co2 val:"+co2Level);
            System.out.println("smoke level:"+smokeLevel);
            try {

                    sendSensorData(Integer.parseInt(txtID.getText()),smokeLevel,co2Level);
                    Thread.sleep(5000);
            } catch (Exception e) {

                    e.printStackTrace();
            }
}
```

Sensor Stop



```java
private void btnStopActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    refreshTimer.stop();
    stopSensor(Integer.parseInt(txtID.getText()));
}
```

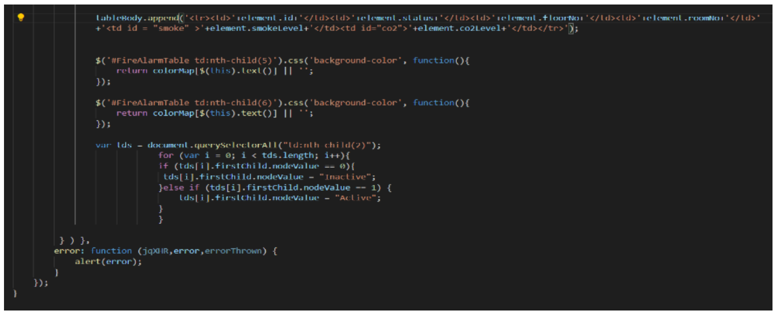
Sensor UI



When the user click stop button stopSensor() method is called, where the status, CO_2 level, smoke level is set to 0 and send to the API using the above URL. The URL will directed to the updateFireAlarmRecords() service interface. This will call the updateRecords() method in the FIreAlarmDao class where the status, co2Level,smokeLevel columns of the firealarm table of database is updated.

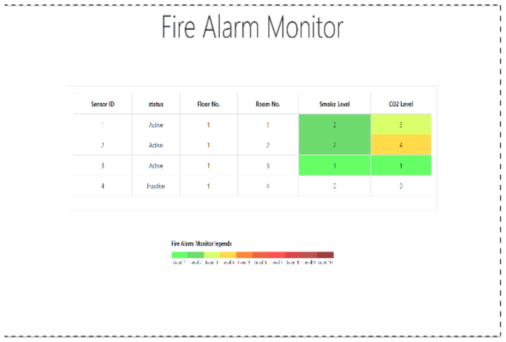
➢ Retrieving Sensor Status

Status of all fire alarms are displayed in the web and the desktop application which can be viewed by a guest user. Details in the web page is updated every after 40s. When the CO_2 or smoke level is more than 5 specific cell is colored in different shades of red. (cells are colored based on the cell value). From the web application a request for retrieving data is send to the REST API every after 40s through the URL http://localhost:8081/FireAlarmMonitor/rest/fireAlarms



```
tableBody.append('<tr><td>'+element.id+'</td><td>'+element.status+'</td><td>'+element.floorNo+'</td><td>'+element.roomNo+'</td>'
+'<td id = "smoke" >'+element.smokeLevel+'</td><td id="co2">'+element.co2Level+'</td></tr>');


$('#FireAlarmTable td:nth-child(5)').css('background-color', function(){
    return colorMap[$(this).text()] || '';
});

$('#FireAlarmTable td:nth-child(6)').css('background-color', function(){
    return colorMap[$(this).text()] || '';
});

var tds = document.querySelectorAll("td:nth-child(2)");
            for (var i = 0; i < tds.length; i++){
            if (tds[i].firstChild.nodeValue == 0){
             tds[i].firstChild.nodeValue = "Inactive";
            }else if (tds[i].firstChild.nodeValue == 1) {
                tds[i].firstChild.nodeValue = "Active";
            }
            }

    } ) },
  error: function (jqXHR,error,errorThrown) {
      alert(error);
  }
 });
}
```

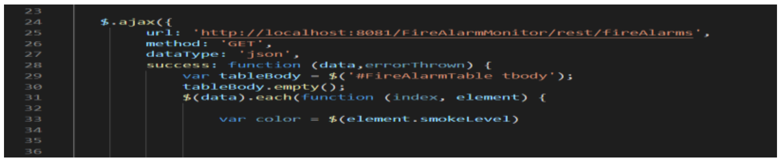
Web UI



Fire Alarm Monitor

Sensor ID	status	Floor No.	Room No.	Smoke Level	CO2 Level
1	Active	1	1	2	3
2	Active	1	2	2	4
3	Active	1	3	1	1
4	Inactive	1	4	0	0

Fire Alarm Monitor legends



```

    $.ajax({
          url: 'http://localhost:8081/FireAlarmMonitor/rest/fireAlarms',
          method: 'GET',
          dataType: 'json',
          success: function (data,errorThrown) {
               var tableBody = $('#FireAlarmTable tbody');
               tableBody.empty();
               $(data).each(function (index, element) {

                    var color = $(element.smokeLevel)



```

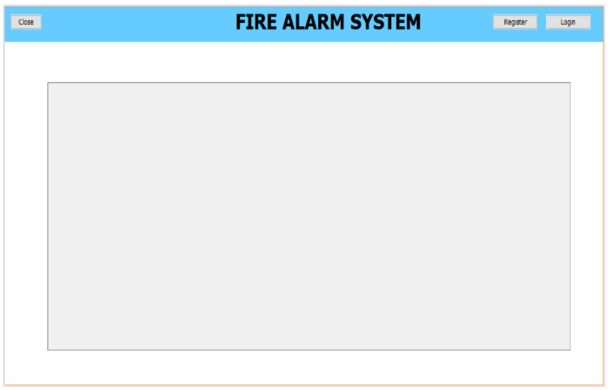
This will direct to the getFireAlarmList service interface which returns details of all fire alarms as json object array which are fetched from the database through getFireAlarms() method of theFireAlarmDao .

Retrieving Sensor Status (Desktop Client)

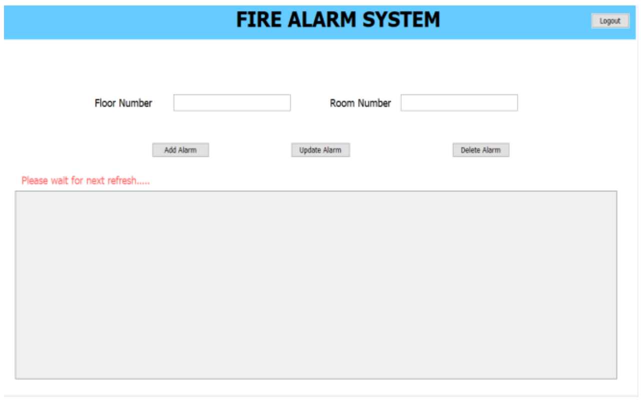
In the RMI , there is getstatus() which calls using http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/ in every 15s as a GET request .

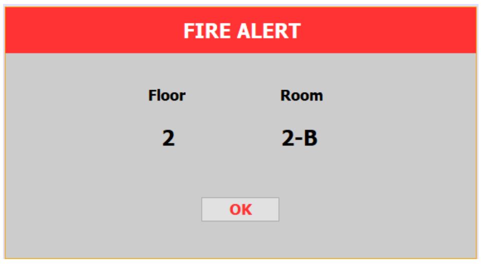
The RMI will store the fire alarm list in every 15 seconds. Desktop client will call showfirealarm() method in RMI interface in every 30 seconds. After that desktop will return the response and set table. If CO_2 level or smoke level is greater than 5 system will display fire alarm alert.

Guest UI



Admin UI

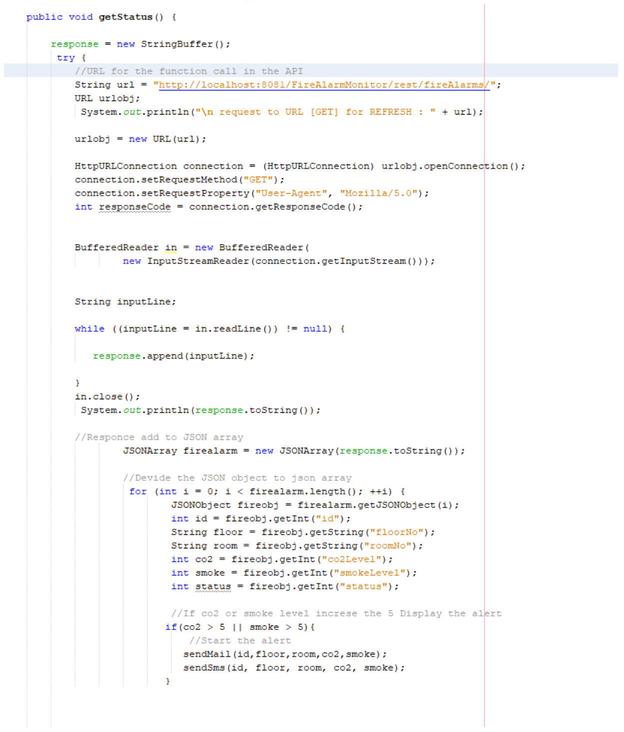




Fire Alarm Alert UI

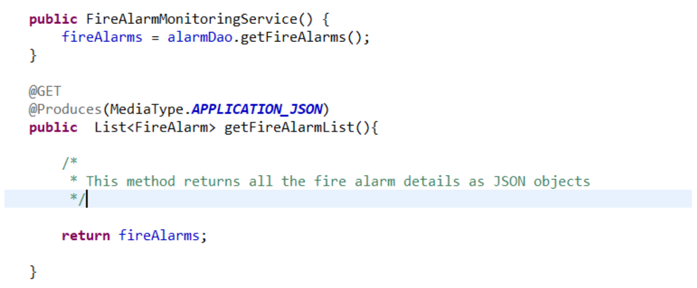
This will direct to getFireAlarmList() service interface which call the getFireAlarms() method of the API . All fire alarm data in the fire alarm table in the firealarmMonitor database are retrieved.

Refresh Method Implementation In RMI



```
public void getStatus() {

    response = new StringBuffer();
     try {
        //URL for the function call in the API
        String url = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/";
        URL urlobj;
         System.out.println("\n request to URL [GET] for REFRESH : " + url);

        urlobj = new URL(url);

        HttpURLConnection connection = (HttpURLConnection) urlobj.openConnection();
        connection.setRequestMethod("GET");
        connection.setRequestProperty("User-Agent", "Mozilla/5.0");
        int responseCode = connection.getResponseCode();


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


        String inputLine;

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

           response.append(inputLine);

        }
        in.close();
         System.out.println(response.toString());

        //Responce add to JSON array
                JSONArray firealarm = new JSONArray(response.toString());

                //Devide the JSON object to json array
                 for (int i = 0; i < firealarm.length(); ++i) {
                        JSONObject fireobj = firealarm.getJSONObject(i);
                        int id = fireobj.getInt("id");
                        String floor = fireobj.getString("floorNo");
                        String room = fireobj.getString("roomNo");
                        int co2 = fireobj.getInt("co2Level");
                        int smoke = fireobj.getInt("smokeLevel");
                        int status = fireobj.getInt("status");

                        //If co2 or smoke level increse the 5 Display the alert
                       if(co2 > 5 || smoke > 5){
                          //Start the alert
                         sendMail(id,floor,room,co2,smoke);
                         sendSms(id, floor, room, co2, smoke);
                       }
```

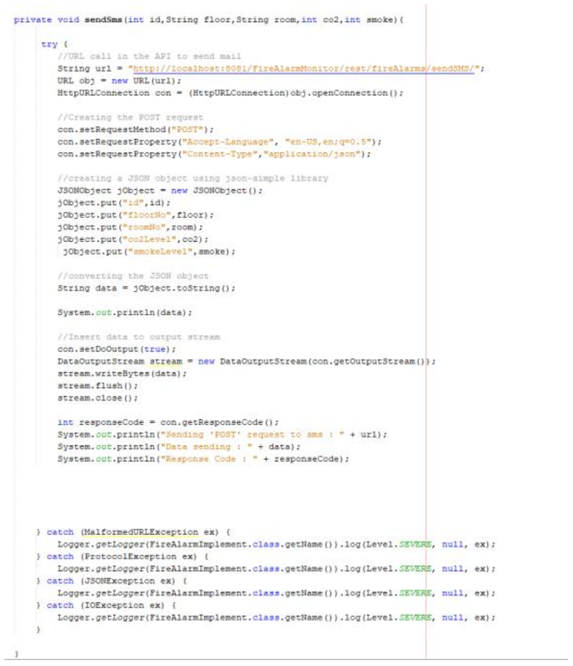
Get Method Implementation In API



```
public FireAlarmMonitoringService() {
    fireAlarms = alarmDao.getFireAlarms();
}

@GET
@Produces(MediaType.APPLICATION_JSON)
public  List<FireAlarm> getFireAlarmList(){

    /*
     * This method returns all the fire alarm details as JSON objects
     */

    return fireAlarms;

}
```

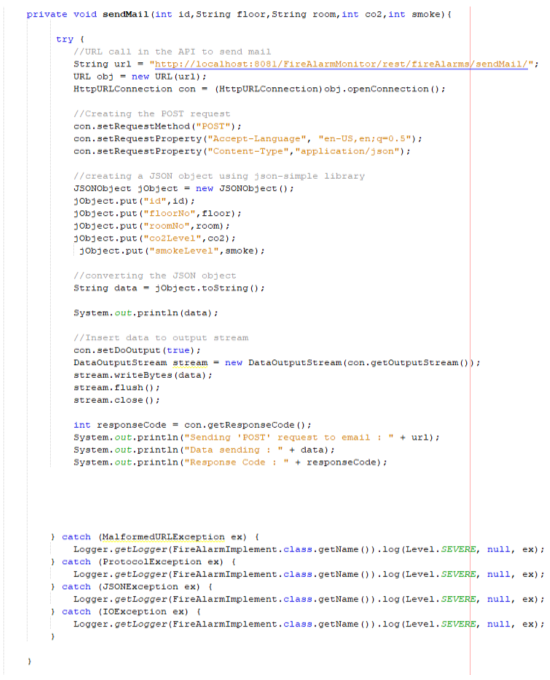
RMI will check the retrievedfire alarm list, if one of them have a CO_2 value greater than 5 or smoke level greater than 5, then RMI will call sendSms() function and sendMail() function. After executing these functions, admin will receive a mail and sms.

Send Mail Method Implementation In RMI



```
private void sendSms(int id,String floor,String room,int co2,int smoke){

    try {
        //URL call in the API to send mail
        String url = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/sendSMS/";
        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection)obj.openConnection();

        //Creating the POST request
        con.setRequestMethod("POST");
        con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
        con.setRequestProperty("Content-Type","application/json");

        //creating a JSON object using json-simple library
        JSONObject jObject = new JSONObject();
        jObject.put("id",id);
        jObject.put("floorNo",floor);
        jObject.put("roomNo",room);
        jObject.put("co2Level",co2);
         jObject.put("smokeLevel",smoke);

        //converting the JSON object
        String data = jObject.toString();

        System.out.println(data);

        //Insert data to output stream
        con.setDoOutput(true);
        DataOutputStream stream = new DataOutputStream(con.getOutputStream());
        stream.writeBytes(data);
        stream.flush();
        stream.close();

        int responseCode = con.getResponseCode();
        System.out.println("Sending 'POST' request to sms : " + url);
        System.out.println("Data sending : " + data);
        System.out.println("Response Code : " + responseCode);



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

}
```

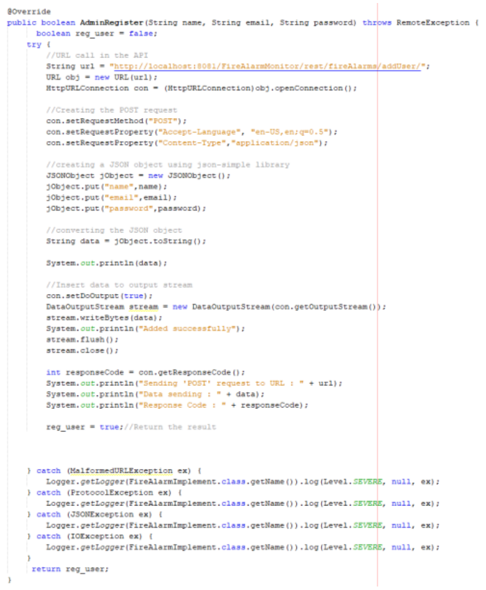
Send SMS Method Implementation In RMI



```
private void sendMail(int id,String floor,String room,int co2,int smoke){

    try {
        //URL call in the API to send mail
        String url = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/sendMail/";
        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection)obj.openConnection();

        //Creating the POST request
        con.setRequestMethod("POST");
        con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
        con.setRequestProperty("Content-Type","application/json");

        //creating a JSON object using json-simple library
        JSONObject jObject = new JSONObject();
        jObject.put("id",id);
        jObject.put("floorNo",floor);
        jObject.put("roomNo",room);
        jObject.put("co2Level",co2);
         jObject.put("smokeLevel",smoke);

        //converting the JSON object
        String data = jObject.toString();

        System.out.println(data);

        //Insert data to output stream
        con.setDoOutput(true);
        DataOutputStream stream = new DataOutputStream(con.getOutputStream());
        stream.writeBytes(data);
        stream.flush();
        stream.close();

        int responseCode = con.getResponseCode();
        System.out.println("Sending 'POST' request to email : " + url);
        System.out.println("Data sending : " + data);
        System.out.println("Response Code : " + responseCode);



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

}
```

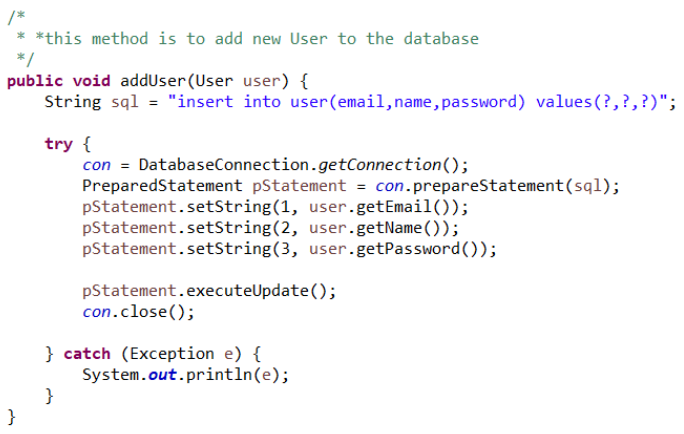
In desktop, when register button clicked by guest client, registration form will be displayed. Then client should enter name, email and password. After entering the data, user click register button. Then these data will be sent through http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/addUser/ as a POST request using AdminRegister() method in RMI. The registerUser() method in userDao is invoked by the addUser() service Interface of the API. Here new user object is passed as a parameter. AddUser() method is responsible in adding new user to the database.

Admin Registration Method Implementation In RMI



```
@Override
public boolean AdminRegister(String name, String email, String password) throws RemoteException {
     boolean reg_user = false;
    try {
        //URL call in the API
        String url = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/addUser/";
        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection)obj.openConnection();

        //Creating the POST request
        con.setRequestMethod("POST");
        con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
        con.setRequestProperty("Content-Type","application/json");

        //creating a JSON object using json-simple library
        JSONObject jObject = new JSONObject();
        jObject.put("name",name);
        jObject.put("email",email);
        jObject.put("password",password);

        //converting the JSON object
        String data = jObject.toString();

        System.out.println(data);

        //Insert data to output stream
        con.setDoOutput(true);
        DataOutputStream stream = new DataOutputStream(con.getOutputStream());
        stream.writeBytes(data);
        System.out.println("Added successfully");
        stream.flush();
        stream.close();

        int responseCode = con.getResponseCode();
        System.out.println("Sending 'POST' request to URL : " + url);
        System.out.println("Data sending : " + data);
        System.out.println("Response Code : " + responseCode);

        reg_user = true;//Return the result


    } catch (MalformedURLException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    } catch (ProtocolException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    } catch (JSONException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    }
     return reg_user;
}
```

Admin Registration Method Implementation In API



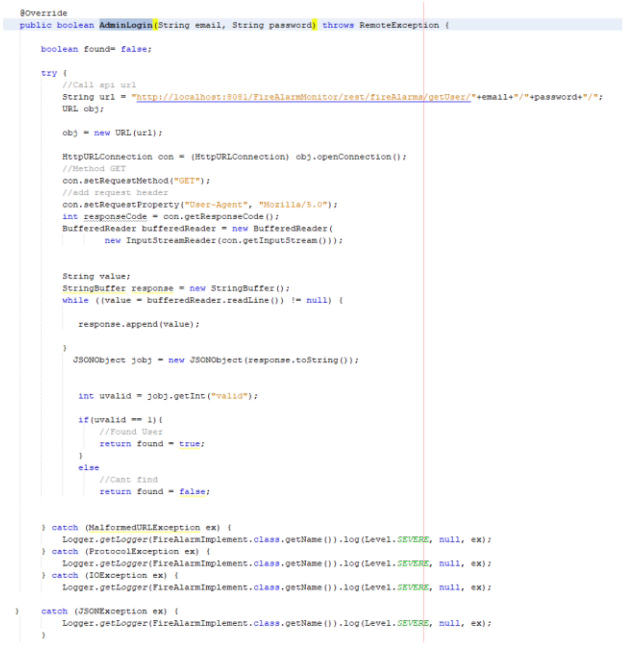
```
/*
 * *this method is to add new User to the database
 */
public void addUser(User user) {
    String sql = "insert into user(email,name,password) values(?,?,?)";

    try {
        con = DatabaseConnection.getConnection();
        PreparedStatement pStatement = con.prepareStatement(sql);
        pStatement.setString(1, user.getEmail());
        pStatement.setString(2, user.getName());
        pStatement.setString(3, user.getPassword());

        pStatement.executeUpdate();
        con.close();

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

➢ Admin Login

In desktop, when login button clicked by guest client; login form will be displayed. Then client should enter email and password. After entering the data, login button should be clicked.

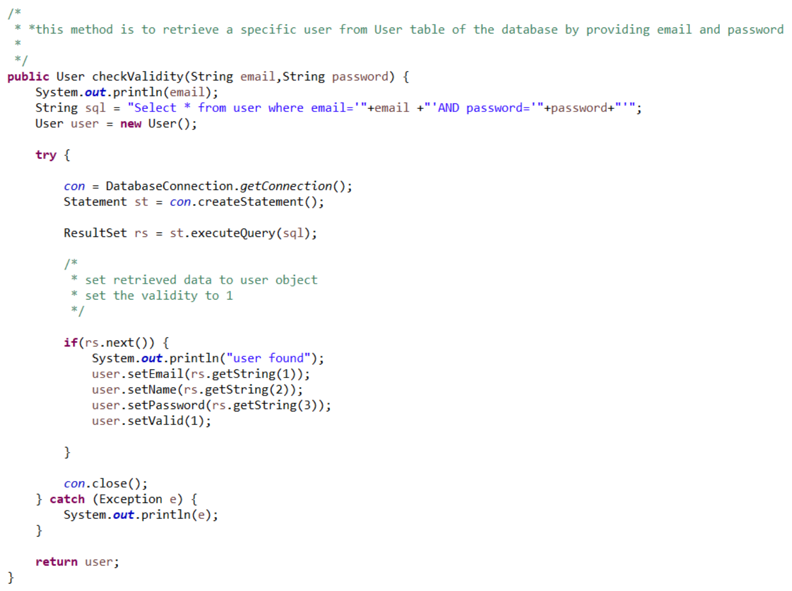
Then these data will be sent through http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/getUser/"+email+"/"+password+"/ as a PUT request using AdminLogin() method in RMI. The checkValidity() method in userDao is invoked by the getUser() service Interface of the API. Here mail and the password are passed as parameters.

Admin Login Method Implementation In RMI



```
@Override
public boolean AdminLogin(String email, String password) throws RemoteException {

    boolean found= false;

    try {
        //Call api url
        String url = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/getUser/"+email+"/"+password+"/";
        URL obj;

        obj = new URL(url);

        HttpURLConnection con = (HttpURLConnection) obj.openConnection();
        //Method GET
        con.setRequestMethod("GET");
        //add request header
        con.setRequestProperty("User-Agent", "Mozilla/5.0");
        int responseCode = con.getResponseCode();
        BufferedReader bufferedReader = new BufferedReader(
                new InputStreamReader(con.getInputStream()));


        String value;
        StringBuffer response = new StringBuffer();
        while ((value = bufferedReader.readLine()) != null) {

           response.append(value);

        }
          JSONObject jobj = new JSONObject(response.toString());


           int uvalid = jobj.getInt("valid");

           if(uvalid == 1){
               //Found User
               return found = true;
           }
           else
               //Cant find
               return found = false;


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

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

Admin Login Method Implementation In API

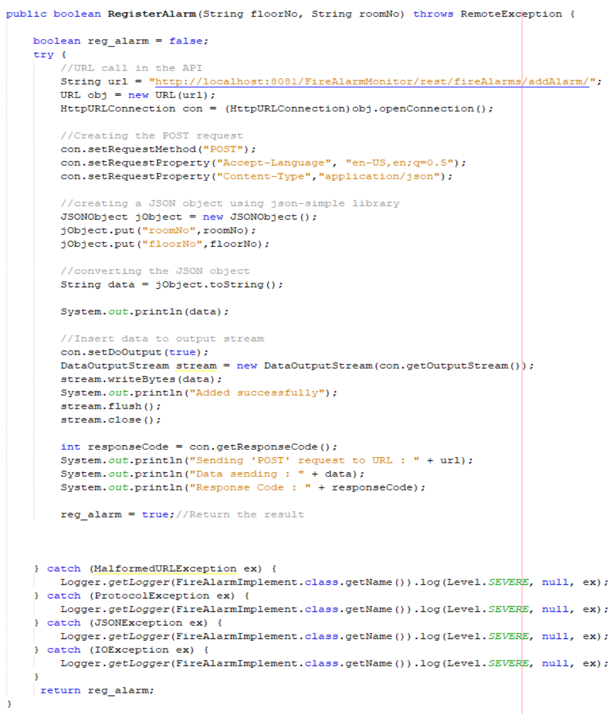


```
/*
 * *this method is to retrieve a specific user from User table of the database by providing email and password
 *
 */
public User checkValidity(String email,String password) {
    System.out.println(email);
    String sql = "Select * from user where email='"+email +"'AND password='"+password+"'";
    User user = new User();

    try {

        con = DatabaseConnection.getConnection();
        Statement st = con.createStatement();

        ResultSet rs = st.executeQuery(sql);

        /*
         * set retrieved data to user object
         * set the validity to 1
         */

        if(rs.next()) {
            System.out.println("user found");
            user.setEmail(rs.getString(1));
            user.setName(rs.getString(2));
            user.setPassword(rs.getString(3));
            user.setValid(1);

        }

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

    return user;
}
```

Add Fire Alarm

A valid user is able to register new fire alarm to the system, only after he is logging to the desktop application. He should enter floor number and room number and click add button for this where data is send to the API through the URL http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/addAlarm/ as a POST request using the RegisterAlarm() method of the RMI server. The addAlarm() method of the FireAlarmDao is invoked by the addFireAlarm service interface of the API where a new FireAlarm object is passed as the parameter. addFireAmarm() method is responsible of adding new fire alarms to the database.

Alarm Registration Method Implementation In RMI



```
public boolean RegisterAlarm(String floorNo, String roomNo) throws RemoteException {

    boolean reg_alarm = false;
    try {
        //URL call in the API
        String url = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/addAlarm/";
        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection)obj.openConnection();

        //Creating the POST request
        con.setRequestMethod("POST");
        con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
        con.setRequestProperty("Content-Type","application/json");

        //creating a JSON object using json-simple library
        JSONObject jObject = new JSONObject();
        jObject.put("roomNo",roomNo);
        jObject.put("floorNo",floorNo);

        //converting the JSON object
        String data = jObject.toString();

        System.out.println(data);

        //Insert data to output stream
        con.setDoOutput(true);
        DataOutputStream stream = new DataOutputStream(con.getOutputStream());
        stream.writeBytes(data);
        System.out.println("Added successfully");
        stream.flush();
        stream.close();

        int responseCode = con.getResponseCode();
        System.out.println("Sending 'POST' request to URL : " + url);
        System.out.println("Data sending : " + data);
        System.out.println("Response Code : " + responseCode);

        reg_alarm = true;//Return the result



    } catch (MalformedURLException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    } catch (ProtocolException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    } catch (JSONException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    }
     return reg_alarm;
}
```

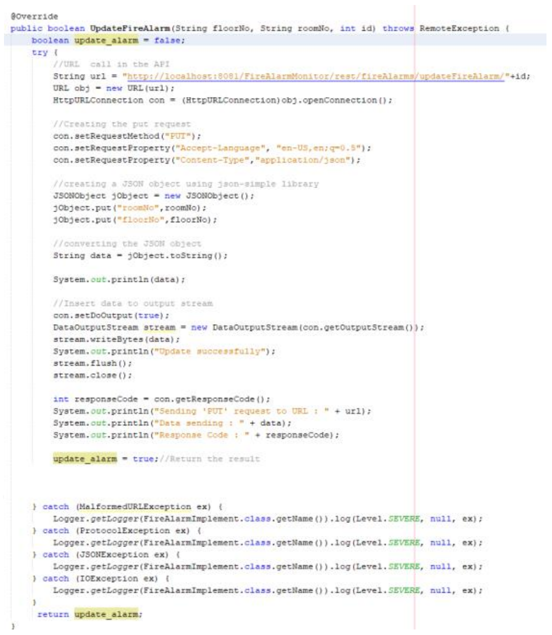
Alarm Registration Method Implementation In API

```
    /*
     * THis method is to add new fire Alaarm to the database
     */
    public void addAlarm(FireAlarm fireAlarm) {
        String sql = "insert into firealarm(roomNo,floorNo) values(?,?)";

try {
            con = DatabaseConnection.getConnection();
            PreparedStatement pStatement = con.prepareStatement(sql);
            pStatement.setString(1, fireAlarm.getRoomNo());
            pStatement.setString(2, fireAlarm.getFloorNo());

            pStatement.executeUpdate();
            con.close();
        }catch (Exception e) {
            System.out.println(e);
        }
}
```

➢ Update Fire Alarms

A user who is logged in to the system can update fire alarm by selecting a particular alarm, entering the data (floor number, room no) want to update and clicking the update button. This will call the UpdateFireAlarm() method of the RMI server. UpdateFireAlarm() method will send a PUT request to the REST API along with the particular alarm's id using the URL `"http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/updateFireAlarm/"+id` where the data of the particular alarm will be updated through the updateAlarm() method of the FireAlarmDao which is accessed through the updateFireAlarm service interface of the API.

Alarm Update Method Implementation In RMI



```
@Override
public boolean UpdateFireAlarm(String floorNo, String roomNo, int id) throws RemoteException {
    boolean update_alarm = false;
    try {
        //URL  call in the API
        String url = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/updateFireAlarm/"+id;
        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection)obj.openConnection();

        //Creating the put request
        con.setRequestMethod("PUT");
        con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
        con.setRequestProperty("Content-Type","application/json");

        //creating a JSON object using json-simple library
        JSONObject jObject = new JSONObject();
        jObject.put("roomNo",roomNo);
        jObject.put("floorNo",floorNo);

        //converting the JSON object
        String data = jObject.toString();

        System.out.println(data);

        //Insert data to output stream
        con.setDoOutput(true);
        DataOutputStream stream = new DataOutputStream(con.getOutputStream());
        stream.writeBytes(data);
        System.out.println("Update successfully");
        stream.flush();
        stream.close();

        int responseCode = con.getResponseCode();
        System.out.println("Sending 'PUT' request to URL : " + url);
        System.out.println("Data sending : " + data);
        System.out.println("Response Code : " + responseCode);

        update_alarm = true;//Return the result



    } catch (MalformedURLException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    } catch (ProtocolException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    } catch (JSONException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    }
     return update_alarm;
}
```

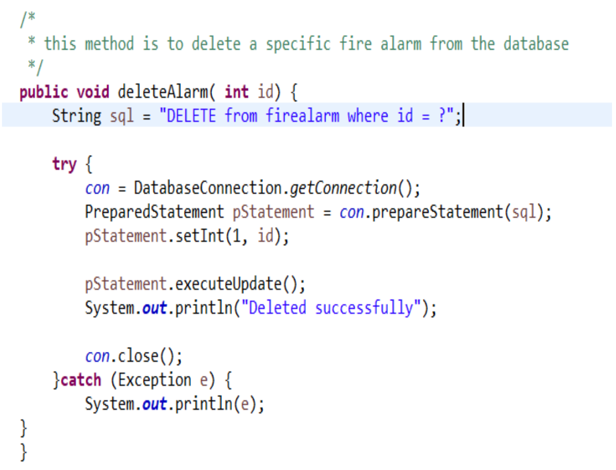
Alarm Update Method Implementation In API

```
    /*
     * * THis method is to update fire alarm details
     */
    public void updateAlarm(FireAlarm fireAlarm, int id) {
        String sql = "update firealarm set roomNo = ?, floorNo=? where id=?";

        try {
                    con = DatabaseConnection.getConnection();
                    PreparedStatement pStatement = con.prepareStatement(sql);
                    pStatement.setInt(3, id);
                    pStatement.setString(1, fireAlarm.getRoomNo());
                    pStatement.setString(2, fireAlarm.getFloorNo());
                    pStatement.executeUpdate();
                    System.out.println("Updated successfully");

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

When a valid user click the delete button of the admin page of the desktop application DeleteFireAlarm() method of the RMI is invoked where a DELETE request is send along with the fire alarm id to the URL "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/deleteFireAlarm/"+id.

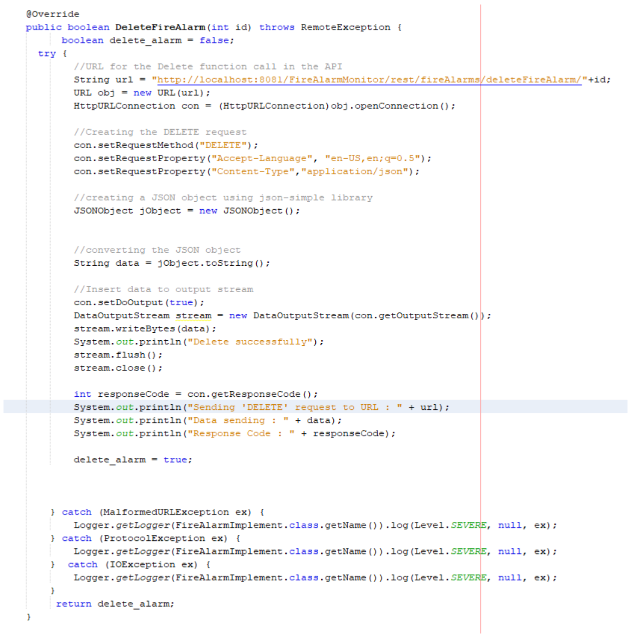
This will direct to the deleteFireAlarm() service interface which call the deleteAlarm() method of the FIreAlarmDao. This will delete the relavent data of in the firealarm table of the database.

Alarm Delete Method Implementation In API



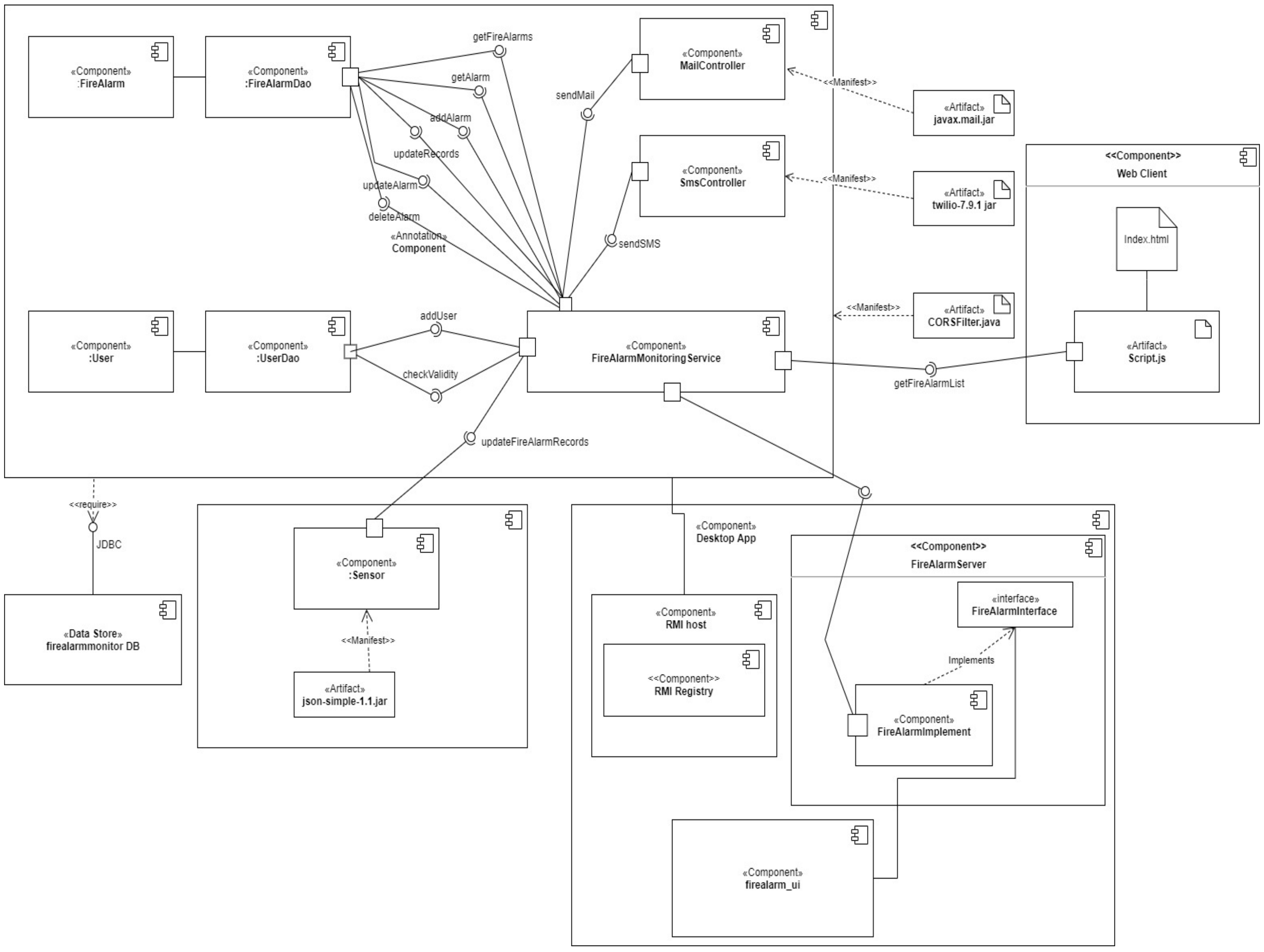
```java
/*
 * this method is to delete a specific fire alarm from the database
 */
public void deleteAlarm( int id) {
    String sql = "DELETE from firealarm where id = ?";

    try {
        con = DatabaseConnection.getConnection();
        PreparedStatement pStatement = con.prepareStatement(sql);
        pStatement.setInt(1, id);

        pStatement.executeUpdate();
        System.out.println("Deleted successfully");

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

Alarm Delete Method Implementation In RMI

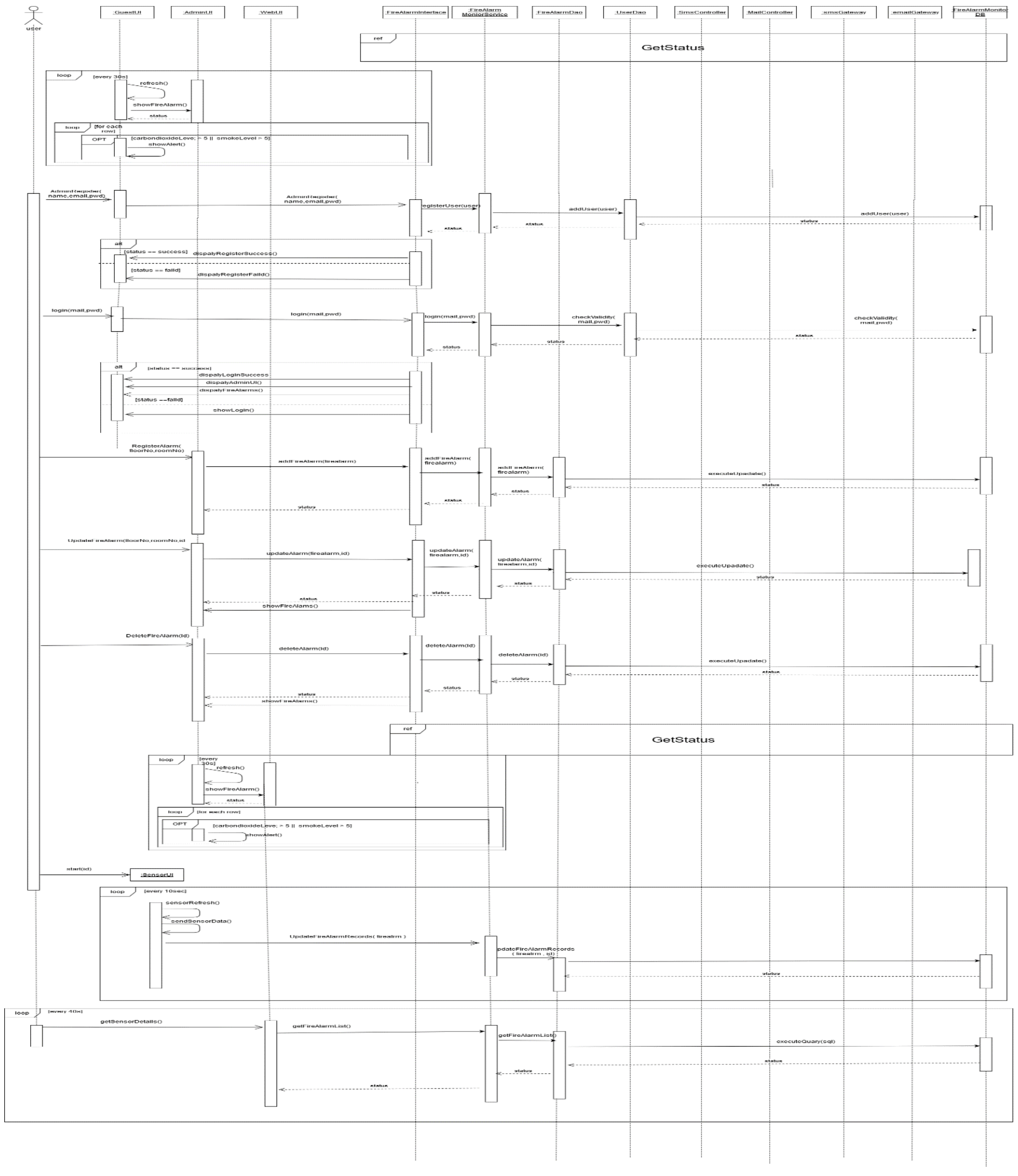


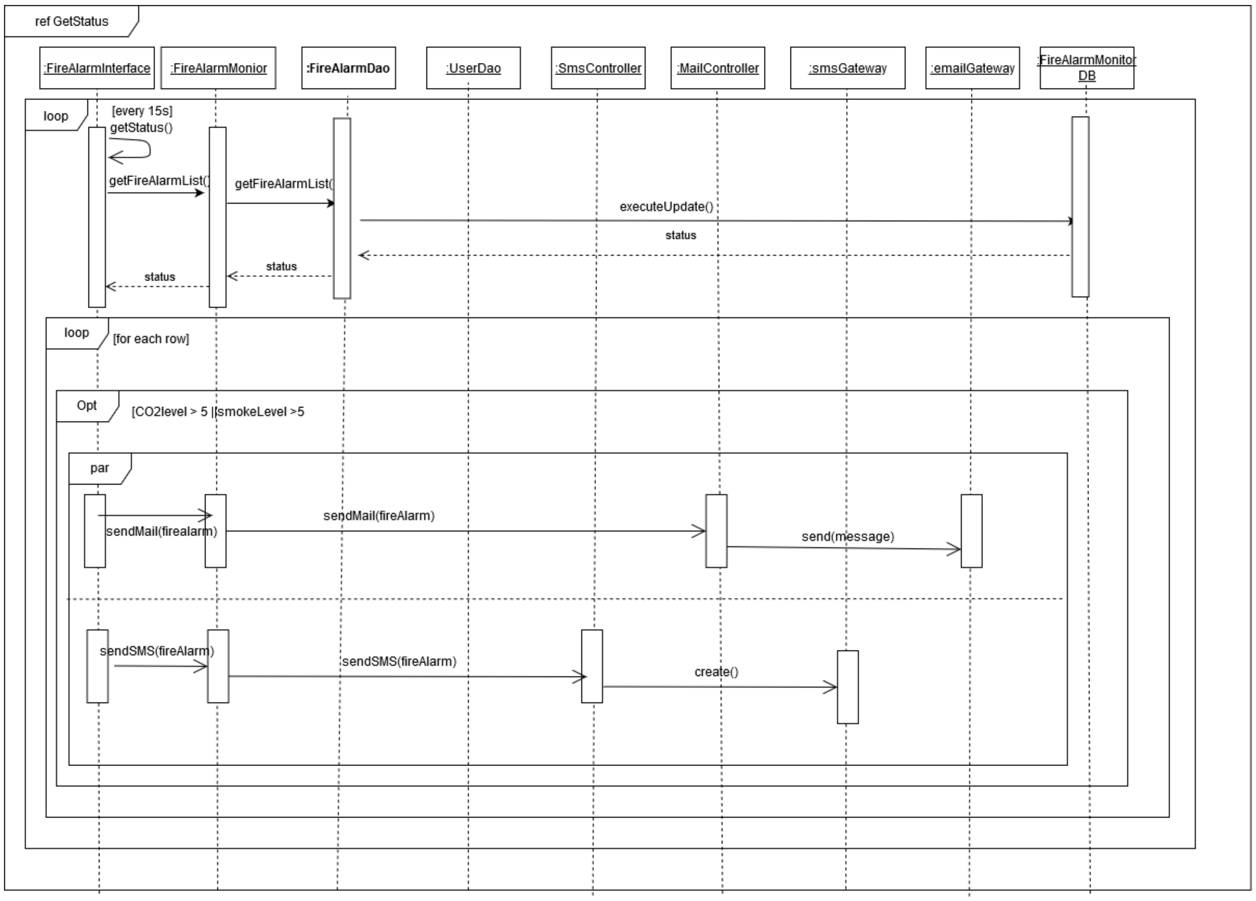
```java
@Override
public boolean DeleteFireAlarm(int id) throws RemoteException {
      boolean delete_alarm = false;
  try {
        //URL for the Delete function call in the API
        String url = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/deleteFireAlarm/"+id;
        URL obj = new URL(url);
        HttpURLConnection con = (HttpURLConnection)obj.openConnection();

        //Creating the DELETE request
        con.setRequestMethod("DELETE");
        con.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
        con.setRequestProperty("Content-Type","application/json");

        //creating a JSON object using json-simple library
        JSONObject jObject = new JSONObject();


        //converting the JSON object
        String data = jObject.toString();

        //Insert data to output stream
        con.setDoOutput(true);
        DataOutputStream stream = new DataOutputStream(con.getOutputStream());
        stream.writeBytes(data);
        System.out.println("Delete successfully");
        stream.flush();
        stream.close();

        int responseCode = con.getResponseCode();
        System.out.println("Sending 'DELETE' request to URL : " + url);
        System.out.println("Data sending : " + data);
        System.out.println("Response Code : " + responseCode);

        delete_alarm = true;



    } catch (MalformedURLException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    } catch (ProtocolException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    }  catch (IOException ex) {
        Logger.getLogger(FireAlarmImplement.class.getName()).log(Level.SEVERE, null, ex);
    }
     return delete_alarm;
}
```

➢ **Component Diagram**

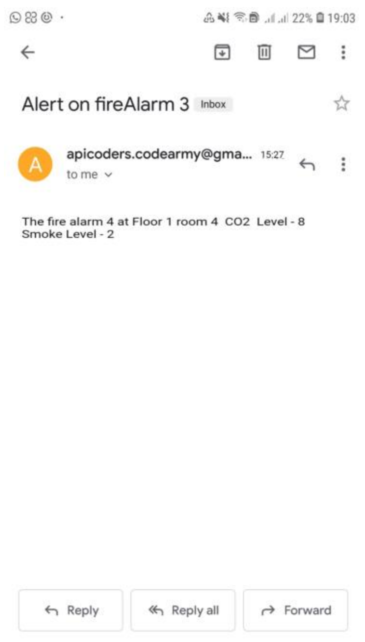


➢ Sequence Diagram





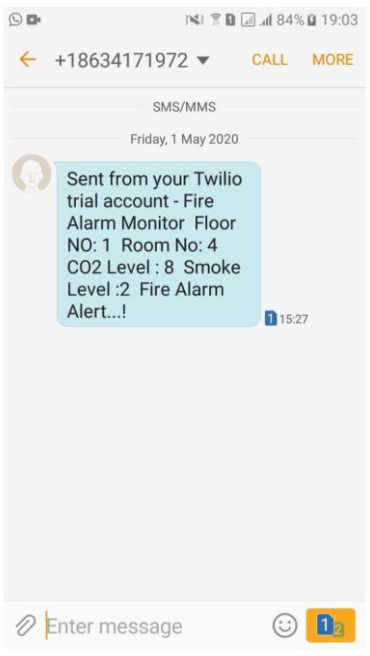
Appendix


Alert on fireAlarm 3 Inbox

apicoders.codearmy@gma... 15:27

to me

The fire alarm 4 at Floor 1 room 4 CO2 Level - 8
Smoke Level - 2


+18634171972

SMS/MMS

Friday, 1 May 2020

Sent from your Twilio trial account - Fire Alarm Monitor Floor NO: 1 Room No: 4 CO2 Level : 8 Smoke Level :2 Fire Alarm Alert...!

15:27

Sample mail & SMS

Code

Admin UI

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

     //Define the table and timer
    DefaultTableModel table;
    Timer refreshTimer;

   public Admin() {
      //Form Load Event
      initComponents();
      centerPanel();
      createCloums();
      autoRefresh();

   }

   private void centerPanel() {

  //JFrom Center the compter screen
         Dimension windowSize = getSize();//Get the form size
         GraphicsEnvironment graphicsEnvironment =
GraphicsEnvironment.getLocalGraphicsEnvironment();
         Point centerPoint = graphicsEnvironment.getCenterPoint();

         int hor = centerPoint.x - windowSize.width / 2;//From Horisontal center
```

```
        int ver = centerPoint.y - windowSize.height / 2;//From vertical center
        setLocation(hor, ver);//Set the From center in the computer screen

}



private void autoRefresh(){
   //Timer Auto refresh the 15 secounds
     refreshTimer =new Timer(15000, new ActionListener() {
         @Override
         public void actionPerformed(ActionEvent e) {
            //What event run given time
            Refresh();
         }
     });
     refreshTimer.start();//Start timer

  }

private void createCloums(){

   //In form load that columns are create
   table = (DefaultTableModel) tbladmin.getModel();
   table.addColumn("ID");
   table.addColumn("Floor No");
   table.addColumn("Room No");
   table.addColumn("CO2");
   table.addColumn("Smoke");
   table.addColumn("Status");
```

```
}


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



      boolean registerAlarm=false;
        //Register the RMI
        Registry register=LocateRegistry.getRegistry("127.0.0.1",1098);
        FireAlarmInterface firealarm=(FireAlarmInterface)register.lookup("firealarm");

        //Passing parameters
        registerAlarm=firealarm.RegisterAlarm(txtFloor.getText(),txtRoom.getText());

        if(registerAlarm==true)
        {
            //User Create Succuess
            JOptionPane.showMessageDialog(null, "Alarm Register Success.");
            txtInfo.setVisible(true);
            clearText();
        }
        else {
            //User Create Fail
            JOptionPane.showMessageDialog(null, "Alarm Register Fail.");
        }
```

```
    }

}

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

            //Form open Methods
            Refresh();
}


private void tbladminMouseClicked(java.awt.event.MouseEvent evt) {

    tbladmin.setCellSelectionEnabled(false);
  //Cell click, go that values to text boxes
      int r = tbladmin.getSelectedRow();//Get selected ro

    String id = table.getValueAt(r, 0).toString();//Get id
    String floor = table.getValueAt(r, 1).toString();//Get floor
    String room = table.getValueAt(r, 2).toString();//Get room


    //Set vales to text boxes
    txtFloor.setText(floor);
    txtRoom.setText(room);
    lblid.setText(id);
}

private void clearText(){
        txtFloor.setText("");

        txtRoom.setText("");

    }

    private void Refresh(){


          table =(DefaultTableModel) tbladmin.getModel();

            int rowCount = table.getRowCount();//Get row count

            //Remove rows one by one from table

            for (int i = rowCount - 1; i >= 0; i--) {

                table.removeRow(i);

        }


        JSONArray firealarm = new JSONArray();

        StringBuffer newresponse = new StringBuffer();

            //Rgister the RMI

            Registry reg=LocateRegistry.getRegistry("127.0.0.1",1098);

            FireAlarmInterface fireAlarm=(FireAlarmInterface)reg.lookup("firealarm");

            //Get responce from the server

            newresponse =  fireAlarm.showfirealarm();






            //Responce add to JSON array

             firealarm = new JSONArray(newresponse.toString());


                  //Devide the JSON object to json array

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

                         JSONObject fireobj = firealarm.getJSONObject(i);
```

```
                    int id = fireobj.getInt("id");
                    String floor = fireobj.getString("floorNo");
                    String room = fireobj.getString("roomNo");
                    int co2 = fireobj.getInt("co2Level");
                    int smoke = fireobj.getInt("smokeLevel");
                    int status = fireobj.getInt("status");

                    //Adding to Table that values
                    InsertRow(id,floor,room,co2,smoke, status);

                    //If co2 or smoke level increse the 5 Display the alert
                   if(co2 > 5 || smoke > 5){
                      //Start the alert
                      Alert frmAlert = new Alert(floor,room);
                      frmAlert.show();
                   }



         }
         txtInfo.setVisible(false);

      }

   }

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

   //Show update confirm alert
```

```
int x = JOptionPane.showConfirmDialog(null, "Do you want update");

  boolean updateAlarm=false;

  //Click Ok
    if(x == 0){

  //Regsiter RMI
  Registry register=LocateRegistry.getRegistry("127.0.0.1",1098);
  FireAlarmInterface firealarm=(FireAlarmInterface)register.lookup("firealarm");

  int id =Integer.parseInt(lblid.getText());
  updateAlarm=firealarm.UpdateFireAlarm(txtFloor.getText(),txtRoom.getText(),id);

  if(updateAlarm==true)
  {
      //Update Success
      JOptionPane.showMessageDialog(null, "Alarm Update Success.");
      txtInfo.setVisible(true);
      clearText();



  }
  else {
      //Update Fail
      JOptionPane.showMessageDialog(null, "Alarm Update Fail.");
  }
}
```

```
    }

}

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


        boolean deleteAlarm=false;
        //Register RMI
        Registry register=LocateRegistry.getRegistry("127.0.0.1",1098);
        FireAlarmInterface firealarm=(FireAlarmInterface)register.lookup("firealarm");

        int id =Integer.parseInt(lblid.getText());
        //Get responce from the RMI
        deleteAlarm=firealarm.DeleteFireAlarm(id);

        if(deleteAlarm==true)
        {
            //Update Success
            JOptionPane.showMessageDialog(null, "Alarm Delete Success.");
            txtInfo.setVisible(true);
            clearText();


        }
        else {
            //Update Fail
            JOptionPane.showMessageDialog(null, "Alarm Delete Fail.");
        }
    }
}

private void btnLogoutActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Guest frmGuest = new Guest();
    frmGuest.show();
   dispose();

}

private void InsertRow(int id,String floor,String room,int co2,int smoke,int status){

    //Responce data add to table
    table =(DefaultTableModel) tbladmin.getModel();
     String nStatus="";
    String sid = String.valueOf(id);
    if(status == 1){
        nStatus = "Active";
    }else if(status == 0){
        nStatus = "Inactive";
    }

    //Create
    String[] rowData ={ sid,floor,room,String.valueOf(co2),String.valueOf(smoke),nStatus};

    //Adding rows
    table.addRow(rowData);
}
```

```
    public static void main(String args[]) {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        }

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

Admin Login

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


    public AdminLogin() {
        initComponents();
        centerPanel();
    }


  private void centerPanel() {
  //JFrom Center the compter screen
```

```
        Dimension windowSize = getSize();//Get the form size
        GraphicsEnvironment graphicsEnvironment =
GraphicsEnvironment.getLocalGraphicsEnvironment();
        Point centerPoint = graphicsEnvironment.getCenterPoint();

        int hor = centerPoint.x - windowSize.width / 2;//From Horisontal center
        int ver = centerPoint.y - windowSize.height / 2;//From vertical center
        setLocation(hor, ver);//Set the From center in the computer screen
    }


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

      boolean f=false;

      if(txtEmail.getText().isEmpty()){//Check Email is empty
            JOptionPane.showMessageDialog(null, "Please enter the Email");
      }else if(txtPassword.getText().isEmpty()){//Check password is empty
             JOptionPane.showMessageDialog(null, "Please enter the Password");
      }else{//Details Filled


        Registry reg=LocateRegistry.getRegistry("127.0.0.1",1098);
        FireAlarmInterface i=(FireAlarmInterface)reg.lookup("firealarm");
        f=i.AdminLogin(txtEmail.getText(),txtPassword.getText());
        if(f==true)
        {
           //Login Success
           JOptionPane.showMessageDialog(null, "Login Success");
```

```
            //Open Admin Form

            Admin frmAdmin = new Admin();

            frmAdmin.show();

            dispose();


        }

        else {

            //Login Fail

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

        }

    }

    }

}


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



    //Open Guest Form

    Guest frmGuest = new Guest();

    frmGuest.show();

    this.hide();

}


public static void main(String args[]) {

            if ("Nimbus".equals(info.getName())) {

                javax.swing.UIManager.setLookAndFeel(info.getClassName());

                break;

            }

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

Admin Register

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

    public AdminRegister() {
        initComponents();
        centerPanel();
    }


private void centerPanel() {

  //JFrom Center the compter screen
            Dimension windowSize = getSize();//Get the form size
            GraphicsEnvironment graphicsEnvironment =
GraphicsEnvironment.getLocalGraphicsEnvironment();
            Point centerPoint = graphicsEnvironment.getCenterPoint();

            int hor = centerPoint.x - windowSize.width / 2;//From Horisontal center
            int ver = centerPoint.y - windowSize.height / 2;//From vertical center
        setLocation(hor, ver);//Set the From center in the computer screen
    }

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

        //Display the Guest Form
        Guest frmguest = new Guest();
        frmguest.show();
        this.hide();

    }

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

      if(txtName.getText().isEmpty()){
          JOptionPane.showMessageDialog(null, "Please enter the Name");
      }else if(txtEmail.getText().isEmpty()){
          JOptionPane.showMessageDialog(null, "Please enter the Email");
      }else if(txtPassword.getText().isEmpty()){
          JOptionPane.showMessageDialog(null, "Please enter the Password");
      }else if(txtConPassword.getText().isEmpty()){
          JOptionPane.showMessageDialog(null, "Please enter the Confirm password");
      }else{

      String pass = txtPassword.getText();
      String conpass = txtConPassword.getText();
      if(pass.equals(conpass)){
        //Register RMI
```

```
          Registry register=LocateRegistry.getRegistry("127.0.0.1",1098);
          FireAlarmInterface firealarm=(FireAlarmInterface)register.lookup("firealarm");

          //Pass parameters
registerUser=firealarm.AdminRegister(txtName.getText(),txtEmail.getText(),txtPassword.getText());

          if(registerUser==true)
          {
            //User register Success
            JOptionPane.showMessageDialog(null, "User Register Success.");
                Guest frmguest = new Guest();
                frmguest.show();
                this.hide();

          }
          else {
            //User register fail
            JOptionPane.showMessageDialog(null, "User Register Fail.");
          }
        }


        }else
          JOptionPane.showMessageDialog(null, "Password Does Not Match");

        }
      }
```

Alert

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




  public Alert(String floor,String room) {


    initComponents();

    centerPanel();

    lblFloor.setText(floor);

    lblRoom.setText(room);

  }


  private void centerPanel() {

      Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

      Dimension windowSize = getSize();

      GraphicsEnvironment graphicsEnvironment =
GraphicsEnvironment.getLocalGraphicsEnvironment();

      Point centerPoint = graphicsEnvironment.getCenterPoint();


      int x = (int) (screenSize.width - windowSize.getWidth());

      int y = (int) (screenSize.height - windowSize.getHeight() - 25);

      setLocation(x, y);

     // panel.setLocation(screenSize.width - panel.getWidth(), screenSize.height - panel.getHeight() -
25);

  }


  private Alert() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated
methods, choose Tools | Templates.
    }



    private void btnOkActionPerformed(java.awt.event.ActionEvent evt) {
        dispose();
    }




                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }

        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {

                new Alert().setVisible(true);
            }
        });
    }
```

Guest

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

```
//Define the table and timer
DefaultTableModel table;
Timer refreshTimer;

public Guest() {
   //Form Load Event
   initComponents();
   centerPanel();
   createCloums();
   autoRefresh();
}



private void createCloums(){

   //In form load that columns are create
   table = (DefaultTableModel) tblGuest.getModel();
   table.addColumn("ID");
   table.addColumn("Floor No");
   table.addColumn("Room No");
   table.addColumn("CO2");
   table.addColumn("Smoke");
   table.addColumn("Status");




}
```

```
  private void centerPanel() {

 //JFrom Center the compter screen
        Dimension windowSize = getSize();//Get the form size
        GraphicsEnvironment graphicsEnvironment =
GraphicsEnvironment.getLocalGraphicsEnvironment();
        Point centerPoint = graphicsEnvironment.getCenterPoint();

        int hor = centerPoint.x - windowSize.width / 2;//From Horisontal center
        int ver = centerPoint.y - windowSize.height / 2;//From vertical center
        setLocation(hor, ver);//Set the From center in the computer screen

  }

  private void autoRefresh(){
     //Timer Auto refresh the 15 secounds
      refreshTimer =new Timer(15000, new ActionListener() {
         public void actionPerformed(ActionEvent e) {
           //What event run given time
           Refresh();
         }
      });
      refreshTimer.start();//Start timer

   }

 private void Refresh(){
  table =(DefaultTableModel) tblGuest.getModel();
    int rowCount = table.getRowCount();//Get row count
    //Remove rows one by one from table
    for (int i = rowCount - 1; i >= 0; i--) {
        table.removeRow(i);
}

JSONArray firealarm = new JSONArray();
StringBuffer response = new StringBuffer();
try
{
    //Rgister the RMI
    Registry reg=LocateRegistry.getRegistry("127.0.0.1",1098);
    FireAlarmInterface fireAlarm=(FireAlarmInterface)reg.lookup("firealarm");
    //Get responce from the server
    response =  fireAlarm.showfirealarm();

    //Responce add to JSON array
     firealarm = new JSONArray(response.toString());

            //Devide the JSON object to json array
             for (int i = 0; i < firealarm.length(); ++i) {
                    JSONObject fireobj = firealarm.getJSONObject(i);
                    int id = fireobj.getInt("id");
                    String floor = fireobj.getString("floorNo");
                    String room = fireobj.getString("roomNo");
                    int co2 = fireobj.getInt("co2Level");
                    int smoke = fireobj.getInt("smokeLevel");
                    int status = fireobj.getInt("status");
                    //Adding to Table that values
                    InsertRow(id,floor,room,co2,smoke, status);


                    //If co2 or smoke level increse the 5 Display the alert
                   if(co2 > 5 || smoke > 5){
                      //Start the alert
                      Alert frmAlert = new Alert(floor,room);
                      frmAlert.show();
                   }




        }



    }


}


 private void InsertRow(int id,String floor,String room,int co2,int smoke,int status){


    //Responce data add to table
    table =(DefaultTableModel) tblGuest.getModel();
     String nStatus="";
    String sid = String.valueOf(id);
    if(status == 1){
        nStatus = "Active";
    }else if(status == 0){
            nStatus = "Inactive";
        }

        //Create
        String[] rowData ={ sid,floor,room,String.valueOf(co2),String.valueOf(smoke),nStatus};

        //Adding rows
        table.addRow(rowData);
    }

    private void formWindowOpened(java.awt.event.WindowEvent evt) {
        // TODO add your handling code here:
        //Form open method calling
       Refresh();
    }

    private void btnAdminRegisterActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //If register button click open the register form
        AdminRegister frmreg = new AdminRegister();
        frmreg.show();
        dispose();
    }

    private void btnAdminLoginActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //If the login button click open the login form
        AdminLogin frmLogin = new AdminLogin();
        frmLogin.show();
```

```
      dispose();
    }

    private void btnCloseActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
      //Close
      dispose();
    }
```

Firealarm server

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



    try
    {
         Registry register=LocateRegistry.createRegistry(1098);

         FireAlarmImplement fireAlarmImplement=new FireAlarmImplement();

         register.rebind("firealarm", fireAlarmImplement);
         System.out.println("FireAlarm Server Start.");



    }

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

FireAlarm Implementation

```
public class FireAlarmImplement extends UnicastRemoteObject implements FireAlarmInterface{

      //Intialize Variable
      StringBuffer response;
      Timer refreshTimer;

   public FireAlarmImplement() throws RemoteException{

      getStatus();
      //Set Refresh
       refreshTimer =new Timer(15000, new ActionListener() {
           @Override
           public void actionPerformed(ActionEvent e) {
              getStatus();
           }
        });
        refreshTimer.start();
   }
```

```
    public boolean AdminLogin(String email, String password) throws RemoteException {


        boolean found= false;


            //Call api url

            String url =
"http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/getUser/"+email+"/"+password+"/";

            URL UrlObj;


            UrlObj = new URL(url);


            HttpURLConnection connection = (HttpURLConnection) UrlObj.openConnection();

            //Method GET

            connection.setRequestMethod("GET");

            //add request header

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

            int responseCode = connection.getResponseCode();

            BufferedReader bufferedReader = new BufferedReader(

                    new InputStreamReader(connection.getInputStream()));



            String value;

            StringBuffer userResponse = new StringBuffer();

            while ((value = bufferedReader.readLine()) != null) {


                userResponse.append(value);
        }
         JSONObject jobj = new JSONObject(userResponse.toString());



          int uvalid = jobj.getInt("valid");


          if(uvalid == 1){
             //Found User
             return found = true;
          }
          else
             //Cant find
             return found = false;






    return found;//Return the result


}
public boolean AdminRegister(String name, String email, String password) throws RemoteException {
     boolean reg_user = false;


        //URL call in the API
        String RequestUrl = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/addUser/";
        URL obj = new URL(RequestUrl);
        HttpURLConnection httpcon = (HttpURLConnection)obj.openConnection();
```

```
//Creating the POST request
httpcon.setRequestMethod("POST");
httpcon.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
httpcon.setRequestProperty("Content-Type","application/json");

//creating a JSON object using json-simple library
JSONObject adminObject = new JSONObject();
adminObject.put("name",name);
adminObject.put("email",email);
adminObject.put("password",password);

//converting the JSON object
String data = adminObject.toString();

System.out.println(data);

//Insert data to output stream
httpcon.setDoOutput(true);
DataOutputStream stream = new DataOutputStream(httpcon.getOutputStream());
stream.writeBytes(data);
System.out.println("Added successfully");
stream.flush();
stream.close();

int responseCode = httpcon.getResponseCode();
System.out.println(" 'POST' request to URL : " + RequestUrl);

reg_user = true;//Return the result
```

```
     return reg_user;
}


public boolean RegisterAlarm(String floorNo, String roomNo) throws RemoteException {

    boolean reg_alarm = false;

        //URL call in the API
        String RequestUrl = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/addAlarm/";
        URL obj = new URL(RequestUrl);
        HttpURLConnection httpcon = (HttpURLConnection)obj.openConnection();

        //Creating the POST request
        httpcon.setRequestMethod("POST");
        httpcon.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
        httpcon.setRequestProperty("Content-Type","application/json");

        //creating a JSON object using json-simple library
        JSONObject alarmObject = new JSONObject();
        alarmObject.put("roomNo",roomNo);
        alarmObject.put("floorNo",floorNo);

        //converting the JSON object
        String data = alarmObject.toString();

        System.out.println(data);
```

```
            //Insert data to output stream
            httpcon.setDoOutput(true);
            DataOutputStream stream = new DataOutputStream(httpcon.getOutputStream());
            stream.writeBytes(data);
            System.out.println("Added successfully");
            stream.flush();
            stream.close();

            int responseCode = httpcon.getResponseCode();
            System.out.println("'POST' request to URL : " + RequestUrl);



            reg_alarm = true;//Return the result




         return reg_alarm;
     }



     public boolean UpdateFireAlarm(String floorNo, String roomNo, int id) throws RemoteException {
         boolean update_alarm = false;


            //URL  call in the API
            String RequestUrl =
"http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/updateFireAlarm/"+id;
            URL obj = new URL(RequestUrl);
            HttpURLConnection httpcon = (HttpURLConnection)obj.openConnection();
```

```
//Creating the put request
httpcon.setRequestMethod("PUT");
httpcon.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
httpcon.setRequestProperty("Content-Type","application/json");

//creating a JSON object using json-simple library
JSONObject alarmObject = new JSONObject();
alarmObject.put("roomNo",roomNo);
alarmObject.put("floorNo",floorNo);

//converting the JSON object
String data = alarmObject.toString();

System.out.println(data);

//Insert data to output stream
httpcon.setDoOutput(true);
DataOutputStream stream = new DataOutputStream(httpcon.getOutputStream());
stream.writeBytes(data);
System.out.println("Update successfully");
stream.flush();
stream.close();

int responseCode = httpcon.getResponseCode();
System.out.println("'PUT' request to URL : " + RequestUrl);



update_alarm = true;//Return the result
```

```
        return update_alarm;
    }



    public boolean DeleteFireAlarm(int id) throws RemoteException {
         boolean delete_alarm = false;
          //URL for the Delete function call in the API
          String RequestUrl =
"http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/deleteFireAlarm/"+id;
          URL obj = new URL(RequestUrl);
          HttpURLConnection httpcon = (HttpURLConnection)obj.openConnection();

          //Creating the DELETE request
          httpcon.setRequestMethod("DELETE");
          httpcon.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
          httpcon.setRequestProperty("Content-Type","application/json");

          //creating a JSON object using json-simple library
          JSONObject jObject = new JSONObject();



          //converting the JSON object
          String data = jObject.toString();

          //Insert data to output stream
          httpcon.setDoOutput(true);
        DataOutputStream stream = new DataOutputStream(httpcon.getOutputStream());
        stream.writeBytes(data);
        System.out.println("Delete successfully");
        stream.flush();
        stream.close();

        int responseCode = httpcon.getResponseCode();
        System.out.println("Sending 'DELETE' request to URL : " + RequestUrl);
        System.out.println("Data sending : " + data);
        System.out.println("Response Code : " + responseCode);

        delete_alarm = true;



     return delete_alarm;
}



public StringBuffer showfirealarm() throws RemoteException {



  return response;//Retrn the result


}



public void getStatus() {


    response = new StringBuffer();
```

```
//URL for the function call in the API
String RequestUrl = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/";
URL urlobj;
 System.out.println("\n request to URL [GET] for REFRESH : " + RequestUrl);

urlobj = new URL(RequestUrl);

HttpURLConnection connection = (HttpURLConnection) urlobj.openConnection();
connection.setRequestMethod("GET");
connection.setRequestProperty("User-Agent", "Mozilla/5.0");
int responseCode = connection.getResponseCode();


BufferedReader alarmBuffer = new BufferedReader(
        new InputStreamReader(connection.getInputStream()));


String inputLine;

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

   response.append(inputLine);

}
alarmBuffer.close();
 System.out.println(response.toString());

//Responce add to JSON array
```

```
            JSONArray firealarm = new JSONArray(response.toString());


            //Devide the JSON object to json array
             for (int i = 0; i < firealarm.length(); ++i) {
                    JSONObject fireobj = firealarm.getJSONObject(i);
                    int id = fireobj.getInt("id");
                    String floor = fireobj.getString("floorNo");
                    String room = fireobj.getString("roomNo");
                    int co2 = fireobj.getInt("co2Level");
                    int smoke = fireobj.getInt("smokeLevel");
                    int status = fireobj.getInt("status");


                    //If co2 or smoke level increse the 5 Display the alert
                   if(co2 > 5 || smoke > 5){
                      //Send sms and mail
                     sendMail(id,floor,room,co2,smoke);
                     sendSms(id, floor, room, co2, smoke);
                   }




        }




}


private void sendMail(int id,String floor,String room,int co2,int smoke){
```

```
//URL call in the API to send mail

String RequestUrl = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/sendMail/";

URL obj = new URL(RequestUrl);

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


//Creating the POST request

httpcon.setRequestMethod("POST");

httpcon.setRequestProperty("Accept-Language", "en-US,en;q=0.5");

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


//creating a JSON object using json-simple library

JSONObject alarmObject = new JSONObject();

alarmObject.put("id",id);

alarmObject.put("floorNo",floor);

alarmObject.put("roomNo",room);

alarmObject.put("co2Level",co2);

 alarmObject.put("smokeLevel",smoke);


//converting the JSON object

String data = alarmObject.toString();


System.out.println(data);


//Insert data to output stream

httpcon.setDoOutput(true);

DataOutputStream stream = new DataOutputStream(httpcon.getOutputStream());

stream.writeBytes(data);

stream.flush();

stream.close();
```

```
        int responseCode = httpcon.getResponseCode();
        System.out.println("'POST' request to email : " + RequestUrl);









}

private void sendSms(int id,String floor,String room,int co2,int smoke){

        //URL call in the API to send mail
        String RequestUrl = "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/sendSMS/";
        URL UrlObj = new URL(RequestUrl);
        HttpURLConnection httpcon = (HttpURLConnection)UrlObj.openConnection();

        //Creating the POST request
        httpcon.setRequestMethod("POST");
        httpcon.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
        httpcon.setRequestProperty("Content-Type","application/json");

        //creating a JSON object using json-simple library
        JSONObject alarmObject = new JSONObject();
        alarmObject.put("id",id);
        alarmObject.put("floorNo",floor);
        alarmObject.put("roomNo",room);
```

```
          alarmObject.put("co2Level",co2);
           alarmObject.put("smokeLevel",smoke);


          //converting the JSON object
          String data = alarmObject.toString();


          System.out.println(data);


          //Insert data to output stream
          httpcon.setDoOutput(true);
          DataOutputStream stream = new DataOutputStream(httpcon.getOutputStream());
          stream.writeBytes(data);
          stream.flush();
          stream.close();


          int responseCode = httpcon.getResponseCode();
          System.out.println("'POST' request to sms : " + RequestUrl);




}
```

Firealarminterface

```
public interface FireAlarmInterface extends Remote{
    public boolean AdminLogin(String email,String password) throws RemoteException;

    public boolean AdminRegister(String name,String email,String password) throws RemoteException;

    public boolean RegisterAlarm(String floorNo,String roomNo) throws RemoteException;

    public boolean UpdateFireAlarm(String floorNo,String roomNo,int id) throws RemoteException;

    public boolean DeleteFireAlarm(int id) throws RemoteException;

    public StringBuffer showfirealarm() throws RemoteException;


}
```

Sensor

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

    /**
     * Creates new form Sensor
     */

    int smokeLevel = 0;
    int co2Level = 0;
    int command = 0;
     Timer refreshTimer;
    public Sensor() {
        initComponents();
    }

    private void stopSensor(int id) {

            //URL for the updateFireAlarmRecords function call in the API
            String RequestUrl =
"http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/updateRecords/"+id;
            URL obj = new URL(RequestUrl);
            HttpURLConnection httpCon = (HttpURLConnection)obj.openConnection();

            //Creating the put request
            httpCon.setRequestMethod("PUT");
            httpCon.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
            httpCon.setRequestProperty("Content-Type","application/json");

            //creating a JSON object using json-simple library
            JSONObject jObject = new JSONObject();
            jObject.put("id",id);
            jObject.put("status",0);
            jObject.put("smokeLevel",0);
            jObject.put("co2Level",0);

            //converting the JSON object
            String data = jObject.toString();

            System.out.println(data);
            //Insert data to output stream
            httpCon.setDoOutput(true);
            DataOutputStream stream = new DataOutputStream(httpCon.getOutputStream());
            stream.writeBytes(data);
            System.out.println("update successfully");
            stream.flush();
            stream.close();

            int responseCode = httpCon.getResponseCode();
            System.out.println("'PUT' request to URL : " + RequestUrl);



            BufferedReader reader = new BufferedReader(
                    new InputStreamReader(httpCon.getInputStream()));
            String output;
            StringBuffer response = new StringBuffer();

            while ((output = reader.readLine()) != null) {
               response.append(output);
            }



            reader.close();

            //printing result from response
            System.out.println(response.toString());

              }
```

```
   private void sendSensorData(int id,int smokeLevel, int co2Level) throws Exception{
                     //URL for the updateFireAlarmRecords function call in the API
                     String RequestUrl =
"http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/updateRecords/"+id;
                     URL obj = new URL(RequestUrl);
                     HttpURLConnection httpCon = (HttpURLConnection)obj.openConnection();

                     //Creating the put request
                     httpCon.setRequestMethod("PUT");
                     httpCon.setRequestProperty("Accept-Language", "en-US,en;q=0.5");
                     httpCon.setRequestProperty("Content-Type","application/json");

                     //creating a JSON object using json-simple library
                     JSONObject jObject = new JSONObject();
                     jObject.put("id",id);
                     jObject.put("status",1);
                     jObject.put("smokeLevel",smokeLevel);
                     jObject.put("co2Level",co2Level);

                     //converting the JSON object
                     String data = jObject.toString();

                     System.out.println(data);

                      //Insert data to output stream
                     httpCon.setDoOutput(true);
                     DataOutputStream stream = new DataOutputStream(httpCon.getOutputStream());
                     stream.writeBytes(data);
```

```
                System.out.println("update successfully");
                stream.flush();
                stream.close();

                int responseCode = httpCon.getResponseCode();
                 System.out.println("'PUT' request to URL : " + RequestUrl);



                 BufferedReader reader = new BufferedReader(
                         new InputStreamReader(httpCon.getInputStream()));
                 String output;
                 StringBuffer response = new StringBuffer();

                 while ((output = reader.readLine()) != null) {
                  response.append(output);
                 }



                 reader.close();

                 //printing result from response
                 System.out.println(response.toString());
        }
```

FireAlarmDao

```
public class FireAlarmDao {

        private static Connection con;
```

```
	//Fetching all the data from the firealarm table
	public  List<FireAlarm> getFireAlarms(){

		String sql = "select id,floorNo,roomNo,co2Level,smokeLevel,status from
firealarm";
		ArrayList<FireAlarm>fireAlarmList  = new ArrayList<FireAlarm>();


			con =
com.ds.FireAlarmMonitor.util.DatabaseConnection.getConnection();
			Statement st = con.createStatement();
			ResultSet rs = st.executeQuery(sql);

			while(rs.next()) {
				FireAlarm fireAlarm = new FireAlarm();
				fireAlarm.setId(rs.getInt(1));
				fireAlarm.setFloorNo(rs.getString(2));
				fireAlarm.setRoomNo(rs.getString(3));
				fireAlarm.setCo2Level(rs.getInt(4));
				fireAlarm.setSmokeLevel(rs.getInt(5));
				fireAlarm.setStatus(rs.getInt(6));




				System.out.println(fireAlarm);
				fireAlarmList.add(fireAlarm);

			}
			con.close();
		}

		return fireAlarmList;
	}

	/*
	 ** THis method is to fetch a specific fire alarm from the db
	 */
	public FireAlarm getAlarm(int id) {
		String sql = "Select id,floorNo,roomNo,co2Level,smokeLevel,status from
firealarm where id="+id;
		FireAlarm fireAlarm = new FireAlarm();
		ArrayList<FireAlarm>fireAlarmList  = new ArrayList<FireAlarm>();
```

```
            con = DatabaseConnection.getConnection();
                Statement st = con.createStatement();
                ResultSet rs = st.executeQuery(sql);

                if(rs.next()) {

                    fireAlarm.setId(rs.getInt(1));
                    fireAlarm.setFloorNo(rs.getString(2));
                    fireAlarm.setRoomNo(rs.getString(3));
                    fireAlarm.setCo2Level(rs.getInt(4));
                    fireAlarm.setSmokeLevel(rs.getInt(5));
                    fireAlarm.setStatus(rs.getInt(6));



                }
                con.close();

            return fireAlarm;

        }

        /*
         * THis method is to add new fire Alaarm to the database
         */
        public void addAlarm(FireAlarm fireAlarm) {
            String sql = "insert into firealarm(roomNo,floorNo) values(?,?)";

                con = DatabaseConnection.getConnection();
                PreparedStatement pStatement = con.prepareStatement(sql);
                pStatement.setString(1, fireAlarm.getRoomNo());
                pStatement.setString(2, fireAlarm.getFloorNo());

                pStatement.executeUpdate();
                con.close();

    }

        /*
         * * This method is to update sensor records of a specific fire alarm of the database
         */
        public void updateRecords(FireAlarm fireAlarm, int id) {
            String sql = "update firealarm set status = ?, smokeLevel =?, co2Level =? where
    id=?";
```

```
			con = DatabaseConnection.getConnection();
			PreparedStatement pStatement = con.prepareStatement(sql);
			pStatement.setInt(4, id);
			pStatement.setInt(1, fireAlarm.getStatus());
			pStatement.setInt(2, fireAlarm.getSmokeLevel());
			pStatement.setInt(3, fireAlarm.getCo2Level());
			pStatement.executeUpdate();
			System.out.println("Updated successfully");

			con.close();

	}

	/*
	 * * THis method is to update fire alarm details
	 */
	public void updateAlarm(FireAlarm fireAlarm, int id) {
		String sql = "update firealarm set roomNo = ?, floorNo=? where id=?";


					con = DatabaseConnection.getConnection();
					PreparedStatement pStatement = 
con.prepareStatement(sql);
					pStatement.setInt(3, id);
					pStatement.setString(1, fireAlarm.getRoomNo());
					pStatement.setString(2, fireAlarm.getFloorNo());
					pStatement.executeUpdate();
					System.out.println("Updated successfully");

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

	/*
	 * this method is to delete a specific fire alarm from the database
	 */
	public void deleteAlarm( int id) {
		String sql = "DELETE from firealarm where id = ?";


			con = DatabaseConnection.getConnection();
```

```
                    PreparedStatement pStatement = con.prepareStatement(sql);
                    pStatement.setInt(1, id);

                    pStatement.executeUpdate();
                    System.out.println("Deleted successfully");

                    con.close();

        }
}
```

<u>MailController</u>

```
public class MailController {


        public  void sendMail(FireAlarm f) throws AddressException,MessagingException{

                String  sendermail = "apicoders.codeArmy@gmail.com";

                String senderPassword = "p5@k@a9833";

                String recipientAddress = "pasanpramuditha97@gmail.com";

                String subject = "Alert on fireAlarm "+f.getId();

                String emailBody = "The fire alarm "+f.getId()+" at Floor "+ f.getFloorNo()+" room 
"+f.getRoomNo()+"  CO2  Level - "+f.getCo2Level()+"  Smoke Level - "+f.getSmokeLevel();




                //set SMTP server properties

                 Properties properties = new Properties();

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

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

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

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

```
            //creating new session using authenticator
            Authenticator auth = new Authenticator() {
                public PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication(sendermail, senderPassword);
                }
             };

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

            //create new email
            Message message = new MimeMessage(session);

            message.setFrom(new InternetAddress(sendermail));
            InternetAddress[] toAddresses = { new InternetAddress(recipientAddress) };
            message.setRecipients(Message.RecipientType.TO, toAddresses);
            message.setSubject(subject);
            message.setSentDate(new Date());
            message.setText(emailBody);

            Transport.send(message);

        }
}
```

SMSSender

```
public class SmsSender {
 // Find your Account Sid and Auth Token at twilio.com/console
 public static final String ACCOUNT_SID =
        "ACae7f0f69b34d4c8724d932ff15b746d7";
 public static final String AUTH_TOKEN =
        "3fc0afae787af199383976d18075044f";

 public void sendSMS(FireAlarm sms) {
    Twilio.init(ACCOUNT_SID, AUTH_TOKEN);

    Message message = Message
            .creator(new PhoneNumber("+94710729569"), // to
                    new PhoneNumber("+18634171972"), // from
                    "Fire Alarm Monitor  Floor NO: " +sms.getFloorNo()+
                     "  Room No: "+sms.getRoomNo()+"  CO2 Level : "+sms.getCo2Level()+"  Smoke
Level :"+sms.getSmokeLevel()+"  Fire Alarm Alert...! ")
            .create();

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

UserDao

```
public class UserDao {
            private static Connection con;
```

```
	/*
	 * *this method is to retrieve a specific user from User table of the database by
providing email and password
	 *
	 */
	public User checkValidity(String email,String password) {
		System.out.println(email);
		String sql = "Select * from user where email='"+email +"'AND
password='"+password+"'";
		User user = new User();




			con = DatabaseConnection.getConnection();
			Statement st = con.createStatement();

			ResultSet rs = st.executeQuery(sql);

			/*
			 * set retrieved data to user object
			 * set the validity to 1
			 */

			if(rs.next()) {
				System.out.println("user found");
				user.setEmail(rs.getString(1));
				user.setName(rs.getString(2));
```

```
                    user.setPassword(rs.getString(3));
                    user.setValid(1);

                }

                con.close();
            }

            return user;
    }

    /*
     * *this method is to add new User to the database
     */
    public void addUser(User user) {
            String sql = "insert into user(email,name,password) values(?,?,?)";


                con = DatabaseConnection.getConnection();
                PreparedStatement pStatement = con.prepareStatement(sql);
                pStatement.setString(1, user.getEmail());
                pStatement.setString(2, user.getName());
                pStatement.setString(3, user.getPassword());

                pStatement.executeUpdate();
                con.close();
```

```
			}


		}



}
```

FireAlarm

```
package com.ds.FireAlarmMonitor.model;

public class FireAlarm {

		private int id;
		private   String roomNo;
		private String floorNo;
		private int smokeLevel;
		private int co2Level;
		private int status;

		public FireAlarm() {
			super();
		}

		public FireAlarm(int id, String roomNo, String floorNo, int smokeLevel, int co2Level, int status) {
			super();
			this.id = id;
			this.roomNo = roomNo;
			this.floorNo = floorNo;
			this.smokeLevel = smokeLevel;
			this.co2Level = co2Level;
			this.status = status;
		}

		public int getId() {
			return id;
		}

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

		public String getRoomNo() {
			return roomNo;
		}
```

```
	public void setRoomNo(String roomNo) {
		this.roomNo = roomNo;
	}

	public String getFloorNo() {
		return floorNo;
	}

	public void setFloorNo(String floorNo) {
		this.floorNo = floorNo;
	}

	public int getSmokeLevel() {
		return smokeLevel;
	}

	public void setSmokeLevel(int smokeLevel) {
		this.smokeLevel = smokeLevel;
	}

	public int getCo2Level() {
		return co2Level;
	}

	public void setCo2Level(int co2Level) {
		this.co2Level = co2Level;
	}

	public int getStatus() {
		return status;
	}

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


}
```

User

```
package com.ds.FireAlarmMonitor.model;

/*
 * Model class for user
 */
public class User {

	private String name;
	private String email;
	private String password;
	private int valid;

	public User() {
		super();
	}

	public User(String name, String email, String password) {

		this.name = name;
		this.email = email;
		this.password = password;
	}

	public int getValid() {
		return valid;
	}

	public void setValid(int valid) {
		this.valid = valid;
	}

	public String getName() {
		return name;
	}

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

	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public String getPassword() {
		return password;
	}

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


}
```

CORSFilter

```
package com.ds.FireAlarmMonitor.service;
```

```
public class CORSFilter implements ContainerResponseFilter {
          public void filter(ContainerRequestContext requestContext,
            ContainerResponseContext responseContext) throws IOException {
                responseContext.getHeaders().add(
                  "Access-Control-Allow-Origin", "*");
                responseContext.getHeaders().add(
                  "Access-Control-Allow-Credentials", "true");
                responseContext.getHeaders().add(
                 "Access-Control-Allow-Headers",
                 "origin, content-type, accept, authorization");
                responseContext.getHeaders().add(
                  "Access-Control-Allow-Methods",
                  "GET, POST, PUT, DELETE, OPTIONS, HEAD");
          }
}
```

FireAlarmMonitor

```
package com.ds.FireAlarmMonitor.service;


@Path("/fireAlarms")
public class FireAlarmMonitoringService {
	List<FireAlarm> fireAlarms;
	FireAlarmDao alarmDao = new FireAlarmDao();
	UserDao userDao = new UserDao();


	public FireAlarmMonitoringService() {
		fireAlarms = alarmDao.getFireAlarms();
	}
```

```
@GET
@Produces(MediaType.APPLICATION_JSON)
public  List<FireAlarm> getFireAlarmList(){

            /*
             * This method returns all the fire alarm details as JSON objects
             */

            return fireAlarms;

}

@GET
@Path("{id}")
@Produces(MediaType.APPLICATION_JSON)
public FireAlarm getFireAlarm(@PathParam("id") int id) {
            /*
             * This method return the specific user to the provided id
             */
             return alarmDao.getAlarm(id);
}

@POST
@Path("addAlarm")
@Consumes(MediaType.APPLICATION_JSON)
public void addFireAlarm(FireAlarm fireAlarm) {

            alarmDao.addAlarm(fireAlarm);
```

```
		System.out.println("Fire Alarm Added Successfully");
	}

	@PUT
	@Path("updateRecords/{id}")
	@Consumes(MediaType.APPLICATION_JSON)
	public void updateFireAlarmRecords(@PathParam("id") int id,FireAlarm fireAlarm) {
		alarmDao.updateRecords(fireAlarm, id);

	}

	@PUT
	@Path("updateFireAlarm/{id}")
	@Consumes(MediaType.APPLICATION_JSON)
	public void updateFireAlarm(@PathParam("id")int id, FireAlarm fireAlarm) {
		if(alarmDao.getAlarm(id).getId() == 0) {
			alarmDao.addAlarm(fireAlarm);
		}

		else {
			alarmDao.updateAlarm(fireAlarm, id);
		}

	}

	@DELETE
	@Path("deleteFireAlarm/{id}")
	public void deleteFireAlarm(@PathParam("id")int id) {
		 FireAlarm a = alarmDao.getAlarm(id);
```

```
            if(a.getId() != 0) {
                        alarmDao.deleteAlarm(id);
            }

            else {
                        System.out.println("Unable to find the alarm with the id "+id);
            }

}

@POST
@Path("/sendMail")
@Consumes(MediaType.APPLICATION_JSON)
public void sendMail(FireAlarm fireAlarm) {

            MailController mail = new MailController();

            try {
                        mail.sendMail(fireAlarm);
                        System.out.println("Mail sent successfully");
            } catch (Exception e) {
                        System.out.println("Oops! fail sending email");
                        e.printStackTrace();
            }
}

@POST
@Path("/sendSMS")
```

```
	@Consumes(MediaType.APPLICATION_JSON)
	public void sendSMS(FireAlarm fireAlarm) {
		SmsSender sms = new SmsSender();
		try {
			sms.sendSMS(fireAlarm);
			System.out.println("sms sent sucessfully");
		}catch (Exception e){
			System.out.println("sms service crashed");
		}

	}

	@POST
	@Path("addUser")
	@Consumes(MediaType.APPLICATION_JSON)
	public void registerUser(User user) {
		userDao.addUser(user);
		System.out.println("New User Added Successfully");
	}



	@GET
	@Path("getUser/{email}/{password}")
	@Produces(MediaType.APPLICATION_JSON)
	public User getUser(@PathParam("email") String email, @PathParam("password") String
password) {

		 return userDao.checkValidity(email,password);
	}
```

```
}
```

DatabaseConnection

```
public class DatabaseConnection {
	/*
	 * * This method is to build the connection with the database
	 */
	public static Connection getConnection() {
		Connection con = null;

		String url = "jdbc:mysql://localhost:3306/firealarmmonitor";
		String username = "root";
		String password = "";


			Class.forName("com.mysql.jdbc.Driver");
			con = DriverManager.getConnection(url,username,password);
			System.out.println("Connected successfully");
		}

		return con;
	}

}
```

index.html

```
<html>

  <head>

    <meta charset="utf-8" />

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

    <meta http-equiv="refresh" content="5;">

    <title>Page Title</title>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

    <script type = "text/javascript" src="./Script.js"></script>


    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"></script>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"/>

    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" ></script>

    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>

    <link rel="stylesheet" href="./style.css">


  </head>

  <body>

   <h1 class="display-3 text-center">Fire Alarm Monitor</h1>

<br><br><br>

   <div class="container p-3 my-3 border">

    <table border="1" style="border-collapse:collapse" class="table table-bordered text-center"
id="FireAlarmTable">


        <thead>

         <tr>

          <th scope="col">Sensor ID</th>
```

```
            <th scope="col">status</th>

            <th scope="col">Floor No.</th>

            <th scope="col">Room No.</th>

            <th scope="col">Smoke Level</th>

            <th scope="col">CO2 Level</th>

          </tr>

        </thead>

        <tbody></tbody>

      </table>

</div>

<br><br>

<div class="d-flex justify-content-center">

<div class='my-legend align-self-center'>

  <div class='legend-title'>Fire Alarm Monitor legends</div>

  <div class='legend-scale'>

    <ul class='legend-labels'>

      <li><span style='background:#00ff009a;'></span>Level 1</li>

      <li><span style='background:#33cc33b6;'></span>Level 2</li>

      <li><span style='background:#ccff33b9;'></span>Level 3</li>

      <li><span style='background:#ffcc00b6;'></span>Level 4</li>

      <li><span style='background:#ff6600c7;'></span>Level 5</li>

      <li><span style='background:#e62e00be;'></span>Level 6</li>

      <li><span style='background:#ff0000b0;'></span>Level 7</li>

      <li><span style='background:#cc0000b9;'></span>Level 8</li>

      <li><span style='background:#990000ad;'></span>Level 9</li>

      <li><span style='background:#800000c2;'></span>Level 10</li>

    </ul>

  </div>

</div>
```

```
</div>


        <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.js"></script>


        <script type="text/javascript" src="./Script.js"></script>

        <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>

    </body>

</html>
```

Script

```
$(document).ready(function () {

    getSensorDetails();


})



function getSensorDetails() {


    var colorMap = {

        1 : '#00ff009a',

        2 : '#33cc33b6',

        3 : '#ccff33b9',

        4 : '#ffcc00b6',

        5 : '#ff6600c7',

        6 : '#e62e00be',

        7 : '#ff0000b0',

        8 : '#cc0000b9',
    9 : '#990000ad',
    10: '#800000c2'
  };


  $.ajax({
    url: 'http://localhost:8081/FireAlarmMonitor/rest/fireAlarms',
    method: 'GET',
    dataType: 'json',
    success: function (data,errorThrown) {
      var tableBody = $('#FireAlarmTable tbody');
      tableBody.empty();
      $(data).each(function (index, element) {

        var color = $(element.smokeLevel)





tableBody.append('<tr><td>'+element.id+'</td><td>'+element.status+'</td><td>'+element.roo
mNo+'</td><td>'+element.floorNo+'</td><td id = "smoke" >'+element.smokeLevel+'</td><td
id="co2">'+element.co2Level+'</td></tr>');


        $('#FireAlarmTable td:nth-child(5)').css('background-color', function(){
          return colorMap[$(this).text()] || '';
        });
```

```
              $('#FireAlarmTable td:nth-child(6)').css('background-color', function(){
                 return colorMap[$(this).text()] || '';
              });

              var tds = document.querySelectorAll("td:nth-child(2)");
                        for (var i = 0; i < tds.length; i++){
                        if (tds[i].firstChild.nodeValue ==0){
                         tds[i].firstChild.nodeValue = "Inactive";
                        }else{
                           tds[i].firstChild.nodeValue = "Active";
                        }
                        }

         } ) },
        error: function (jqXHR,error,errorThrown) {
           alert(error);
        }
     });
  }



// function getSensorDetails(){

//     fetch('http://localhost:8081/FireAlarmMonitor/rest/fireAlarms')
//     .then((res) => res.json())
//     .then((data) => {
```

```
//      data.forEach(function(element){
//        output +=
'<tr><td>'+element.id+'</td><td>'+element.roomNo+'</td><td>'+element.floorNo+'</td><td>'
+element.status+'</td><td>'+element.smokeLevel+'</td><td>'+element.co2Level+'</td></tr>';
//      });
//      document.getElementById('output').innerHTML = output;
//    })
//  }

$(document).ready(function(){
  var fireAlarm = {};
  $('#add').click(function(){
    fireAlarm.id = 1;
    fireAlarm.roomNo = 1;
    fireAlarm.floorNo = 20;
    fireAlarm.status = 1;
    fireAlarm.smokeLevel = 2;
    fireAlarm.co2Level = 1;
    var fireSMS = JSON.stringify(fireAlarm);
    $.ajax({
      url: "http://localhost:8081/FireAlarmMonitor/rest/fireAlarms/sendSMS",
      type: "POST",
      data:fireSMS,
      contentType: "application/json",

      success:function(){
        alert('Saved Successfully')
      },
            error:function(error){
                alert(error);
            }


        })
    })
})
```

Style.css

```
.my-legend .legend-title {
    text-align: left;
    margin-bottom: 8px;
    font-weight: bold;
    font-size: 90%;
    }
  .my-legend .legend-scale ul {
    margin: 0;
    padding: 0;
    float: left;
    list-style: none;
    }
  .my-legend .legend-scale ul li {
    display: block;
    float: left;
    width: 50px;
    margin-bottom: 6px;
```

```
 text-align: center;
 font-size: 80%;
 list-style: none;
 }
.my-legend ul.legend-labels li span {
 display: block;
 float: left;
 height: 15px;
 width: 50px;
 }
```