Fire Alarm Monitoring System

Distributed Systems – Assignment 2

Group Members

Student ID	Name
IT 18 1208 44	Dissanayake D.M.T.C
IT 18 1237 46	Dayarathna L.B.P.M.
IT 18 1628 06	Ranaweera K.A.D.S
IT 18 1196 40	Kumarasena P.P.

Fire Alarm Monitoring System

A fire alarm system is a distributed system that has a set of components working together to detect and warn people when smoke level or carbon dioxide level goes up beyond the given range. It is one of the most important systems for the safety of a building.

Therefore, a fire alarm system was created in java. This Fire Alarm system consists of many components such as 'Fire Sensor', 'Fire Alarm Monitoring System Desktop Client', 'RMI Server', 'Web App', Rest API, and Database.

Instead of using actual Fire Sensor to get details, we implemented a FiresensorMain.java dummy class in the 'FireSensor Project' which generates sensor data and sends the status to the database via rest API in every 30 seconds.

The data is well defined and structured. So, we choose the MySQL database to store data. Sensor Details and admin details are stored in this database.

A Desktop client project is created separately ('Fire-Alarm-Monitoring-System-Desktop'). This Java Swing GUI Desktop client get sensor data from database and display it in a table. Users can view Sensor ID, room No, Floor No, Sensor status Co2 level, Smoke level, Date, and time. The information is refreshed every 30 seconds. When the application is launched, it creates a separate thread to invoke **getAllSensorDetails()** in Fire AlarmServer.java. Then it calls the **rest/sensor/all** endpoint in rest API. All sensor details will be returned, and it will be displayed on the main page.

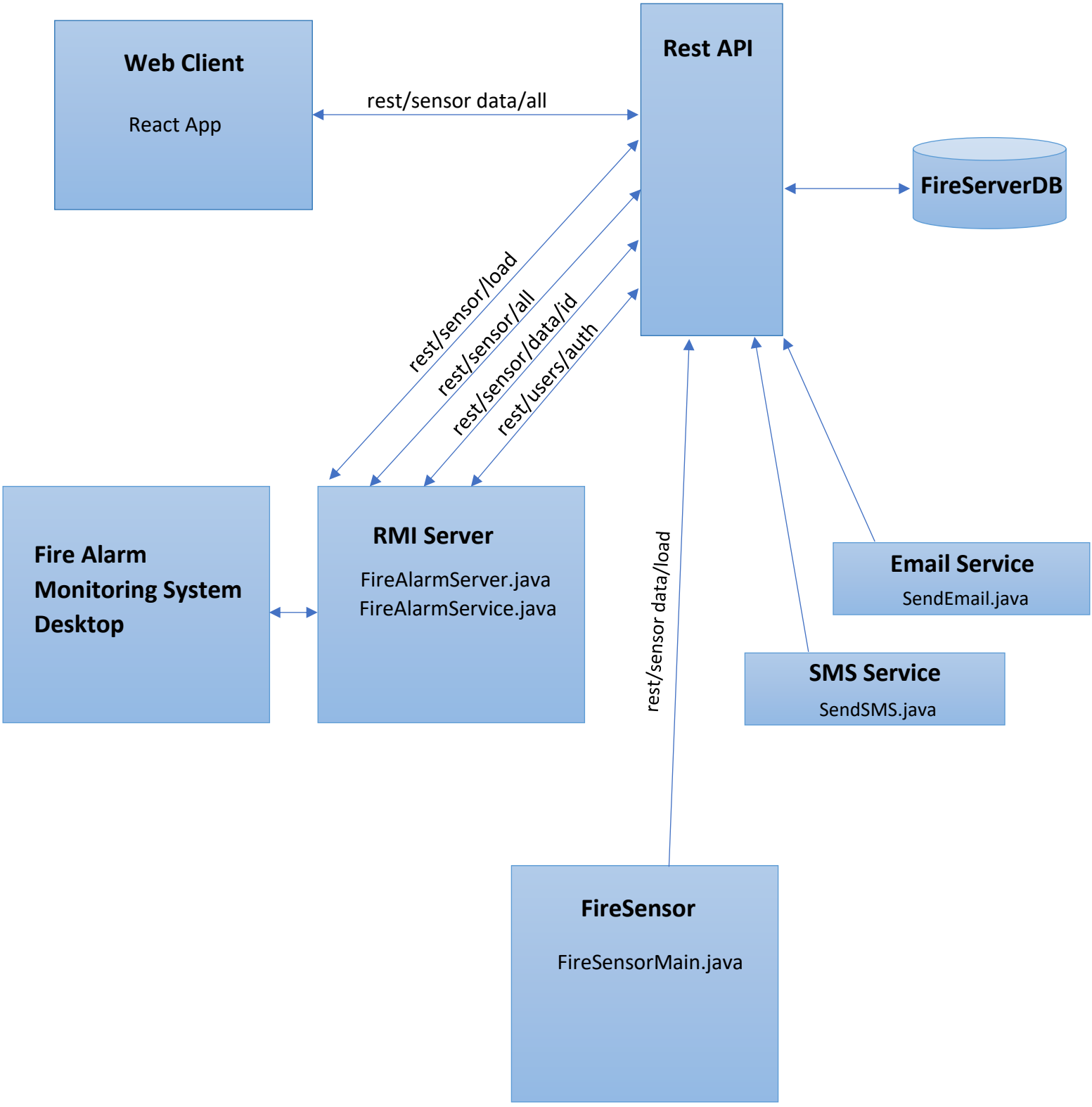
Admin can log in to the system using a given username and password. It is validating in the database and if it is correct admin can go to the admin dashboard. **adminLogin(username, password)** in FireAlarmServer.java Then it calls the **/rest/users/auth** endpoint in rest API and gets the result whether user credentials are correct or incorrect.

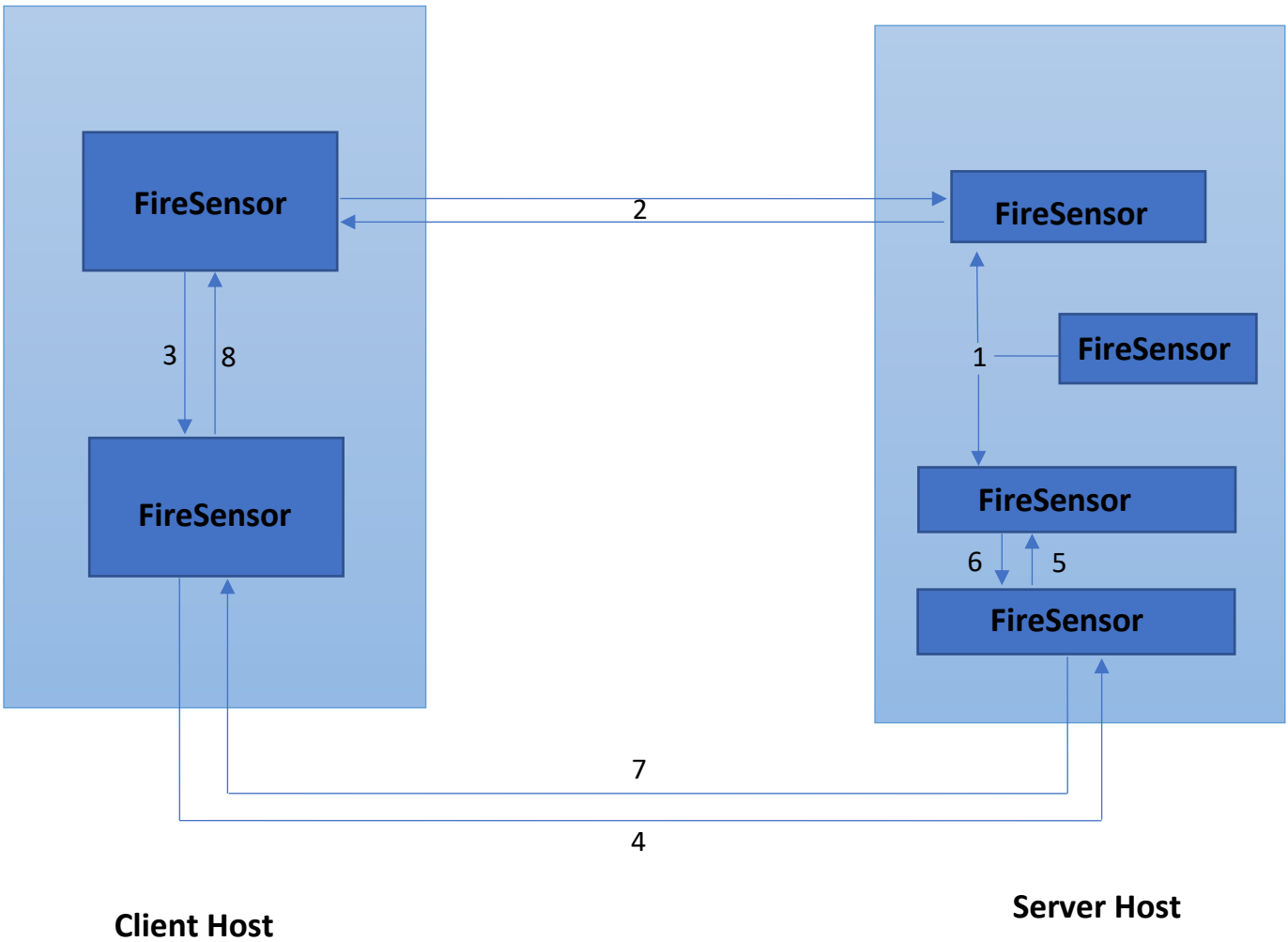
Admin can Add, Update Delete Sensor Details as well. Admin can add sensor details such as Sensor ID, Room No, Floor No, and Active Status in TextFields and press the 'Add New Sensor' button. It invokes **addSensor(sensorid, floor_no, room_no, is_active)** in FireAlarmServer.java . It calls the **/rest/sensor/load** endpoint in rest API and the data will be added to the database. On the right side Textfield, admin can provide sensor ID and press delete. It invokes the **deleteSensor(sensorid)** in FireAlarmServer.java. It calls the **rest/sensor/delete/{"+sensorid+"}"** endpoint in rest API and the sensor will be deleted.
Admin can also update sensor data. Admin can provide sensor id and press update. It invokes **getSensorDetails(sensorid)** in FireAlarmServer.java . It calls **rest/getsensor/{"+sensorid+"}** endpoint and return sensor data. These data set to the TextFields and the Add button will be changed to update. Then the user can modify these fields and press the update button. It invokes the **updateSensor(sensorid, floor_no, room_no, is_active)** in FireAlarmServer.java and calls to the **rest/sensor/load** endpoint in RestAPI and the database will be updated.

A Web client app also created using React. Any user can view Sensor details such as Sensor ID, Room No, Floor No, Smoke Level, Co2 level, Date, and Time. If the Co2 level or smoke level goes above 5, it will be marked in red, and otherwise, it will be marked in green. Sensor details are refreshed every 40 seconds.

Additionally, SMS and Email sending features added. Once the Co2 Level or smoke level goes above 5 Email and SMS will be sent to the user.

System Architecture of the Fire Alarm Monitoring System




FireSensor
FireSensor
FireSensor
FireSensor
FireSensor
FireSensor
1
2
3
4
5
6
7
8
Client Host
Server Host

getSensorDetails [AdminDashboard.java] → getSensorDetails [FireAlarmServer.java]

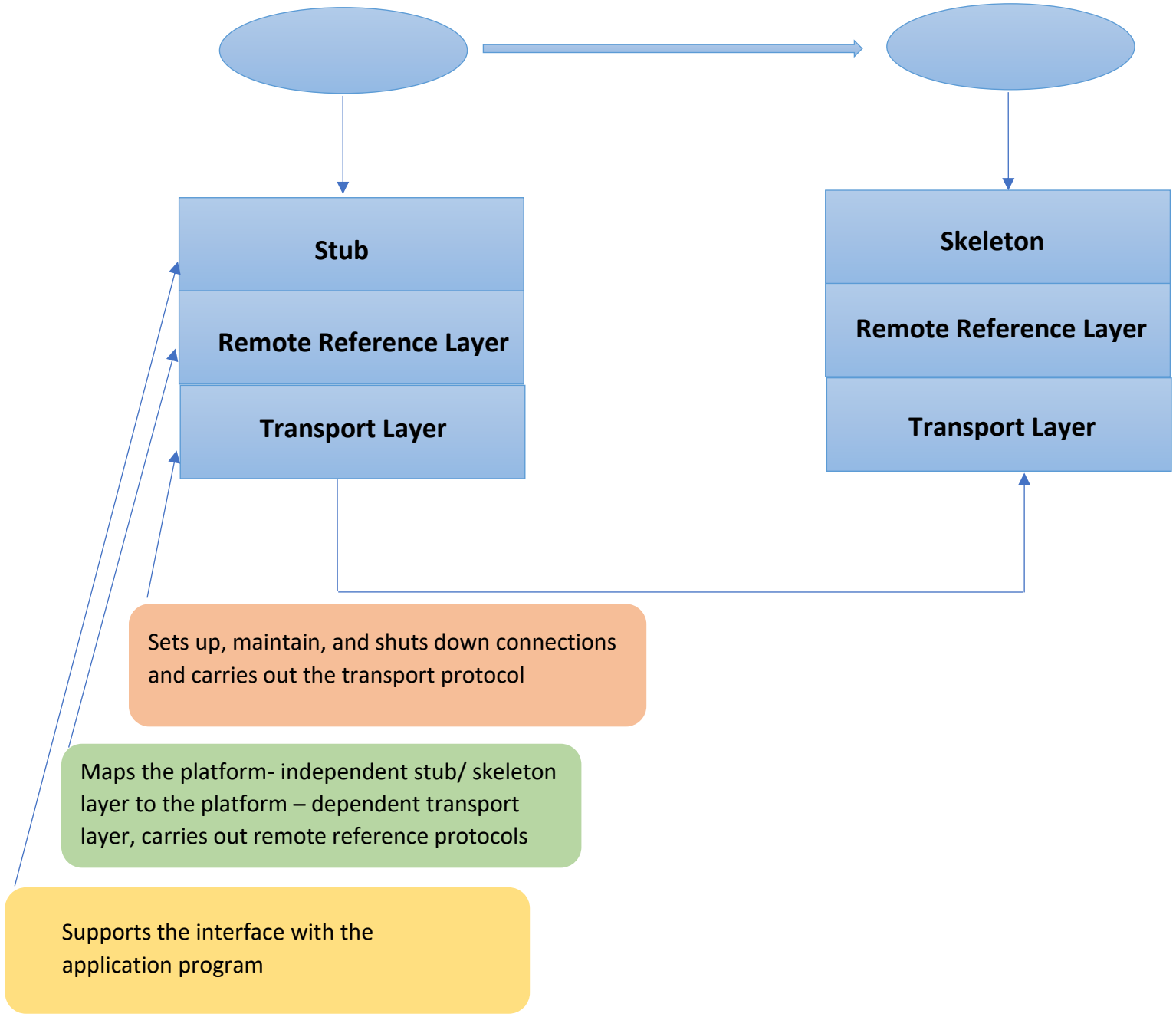
adminLogin [login.java] → adminLogin [FireAlarmServer.java]

getAllSensorDetails [ClientApp.java] → getAllSensorDetails [FireAlarmServer.java]

addSensor [AdminDashboard.java] → addSensor [FireAlarmServer.java]

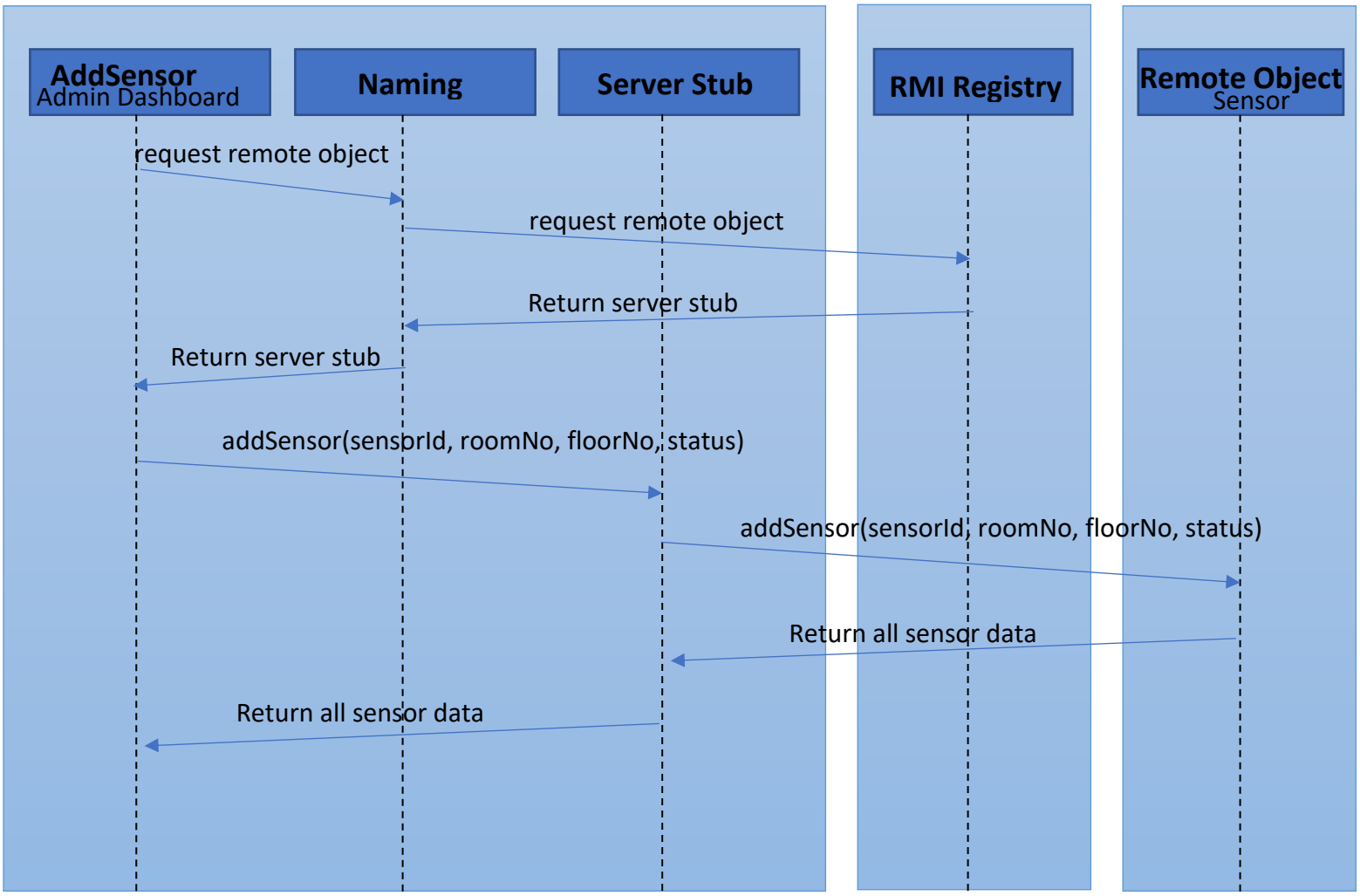
updateSensor [AdminDashboard.java] → updateSensor [FireAlarmServer.java]

deleteSensor [AdminDashboard.java] → deleteSensor [FireAlarmServer.java]

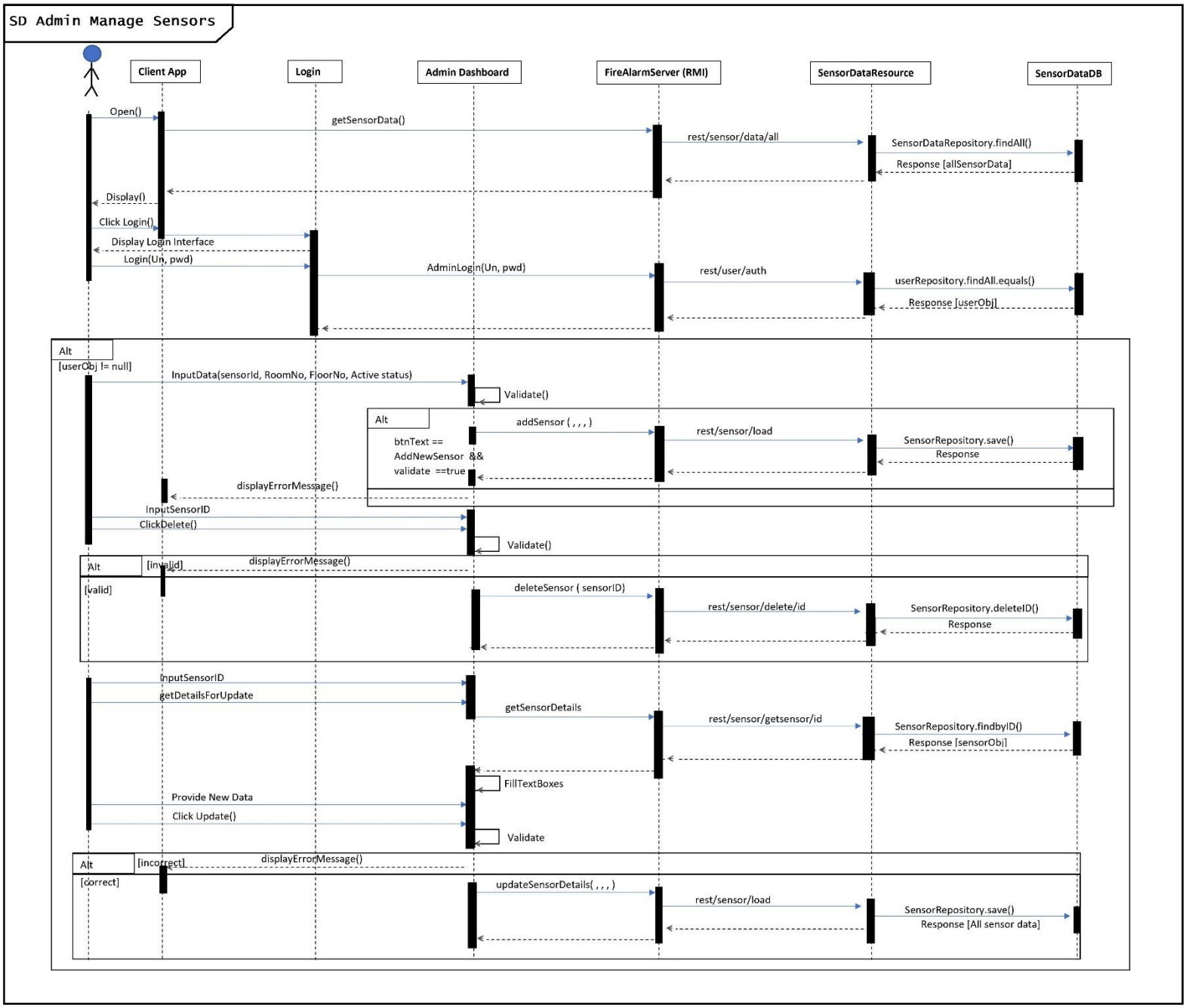


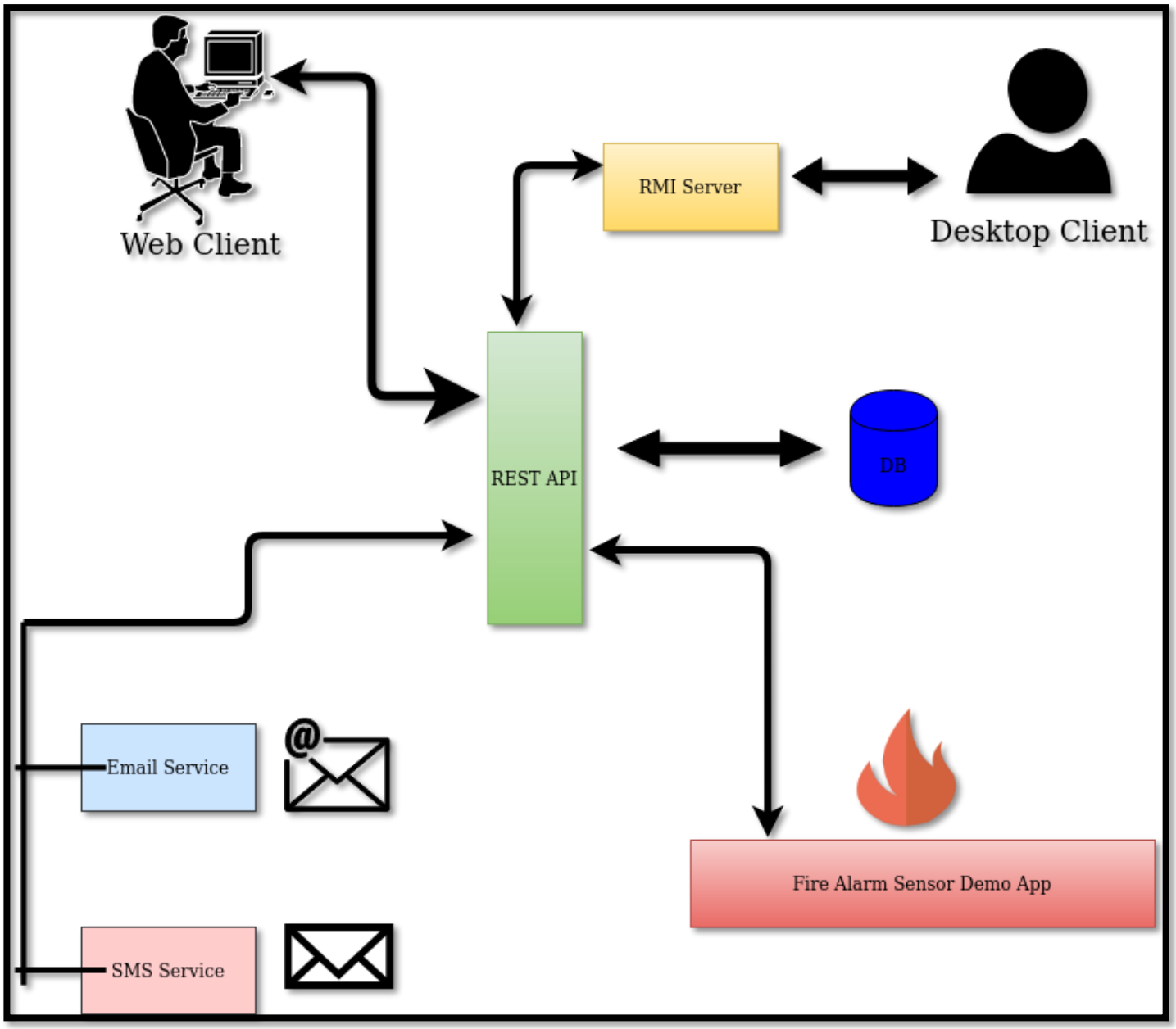
Remote Method Invocation

Example Scenario: Add Sensor

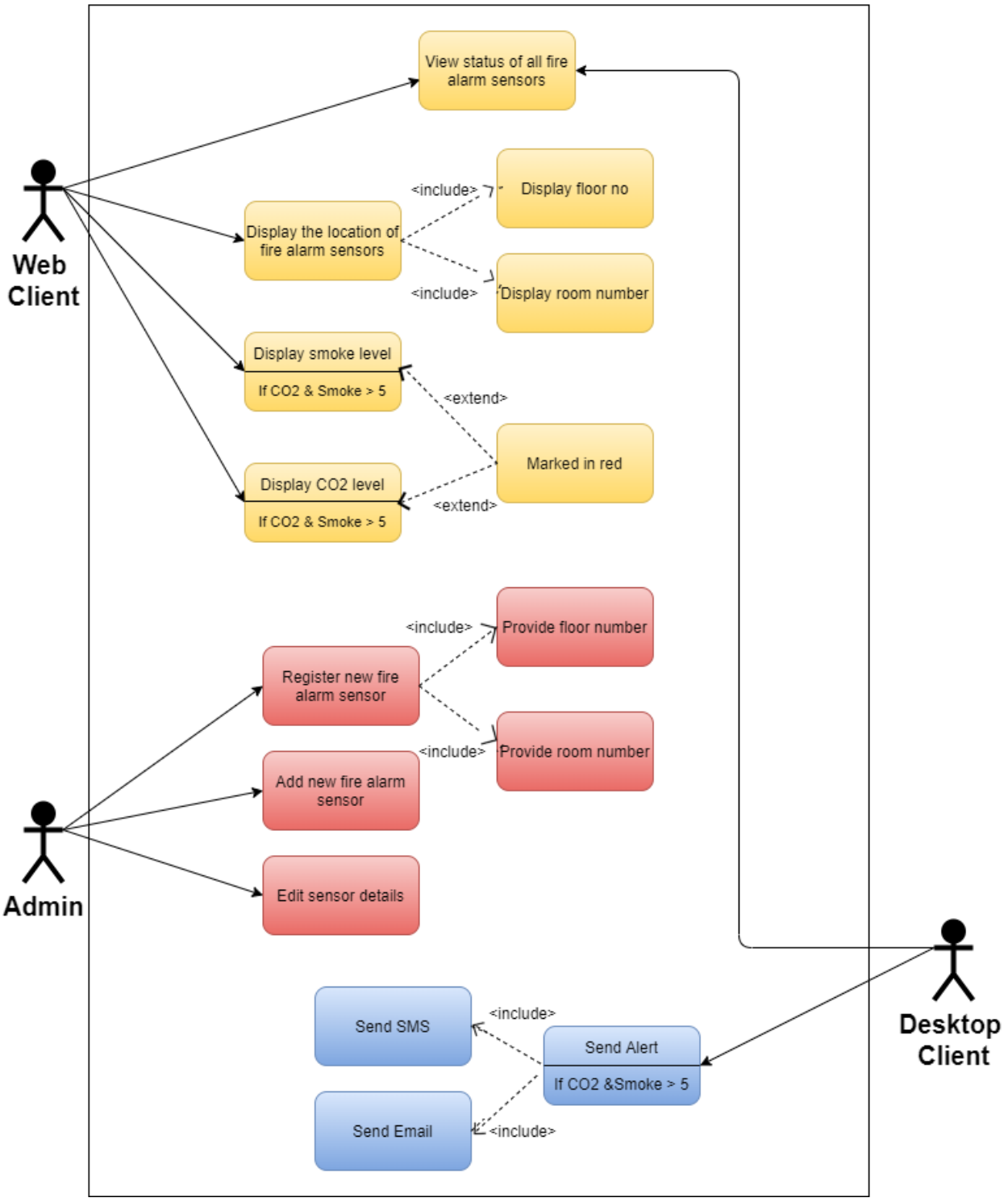


Sequence Diagram




Web Client
RMI Server
Desktop Client
REST API
DB
Email Service
Fire Alarm Sensor Demo App
SMS Service

Use Case Diagram : Fire Alarm Monitoring System



Appendix

RMI

FireAlarmService Interface

```
import java.rmi.RemoteException;

public interface FireAlarmService {

    public String addSensor(String sensorid, int floor_no , String room_no , boolean
i) throws RemoteException;
    public String updateSensor(String sensorid, int floor_no , String room_no ,
boolean is_active) throws RemoteException;
    public String deleteSensor(String sensorid) throws RemoteException;
    public String getSensorDetails(String sensorid) throws RemoteException;
    public String adminLogin(String username, String password) throws
RemoteException;
    public String getAllSensorDetails() throws RemoteException;


}
```

FireAlarmServer Calss

```
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import java.net.MalformedURLException;
import java.rmi.AlreadyBoundException;
import java.rmi.Naming;
import java.io.Console;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.lang.SecurityManager;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
```

```
public class FireAlarmServer extends UnicastRemoteObject implements FireAlarmService{

     protected FireAlarmServer() throws RemoteException {
           super();
       }

       @Override
       public String addSensor(String sensorid, int floor_no, String room_no, boolean
is_active) throws RemoteException {

          String result = "";
         HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/load");

         post.addHeader("content-type", "application/json");

         StringBuilder json = new StringBuilder();
         json.append("{");
         json.append("\"sensorid\":"+sensorid+",");
         json.append("\"floor_no\":\""+floor_no+"\"");
         json.append("\"room_no\":\""+room_no+"\"");
         json.append("\"is_active\":\""+is_active+"\"");
         json.append("}");

         System.out.println(json);

         // send a JSON data
         try {
            post.setEntity(new StringEntity(json.toString()));
         } catch (UnsupportedEncodingException e1) {
            e1.printStackTrace();
         }

         try (CloseableHttpClient httpClient = HttpClients.createDefault();
             CloseableHttpResponse response = httpClient.execute(post)) {

            result = EntityUtils.toString(response.getEntity());
         }catch (Exception e) {

         }

         System.out.println(result);

         return result;


       }

       @Override
       public String updateSensor(String sensorid, int floor_no, String room_no,
boolean is_active) throws RemoteException {

          String result = "";
         HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/load");

         post.addHeader("content-type", "application/json");
```

```
          StringBuilder json = new StringBuilder();
          json.append("{");
          json.append("\"sensorid\":"+sensorid+",");
          json.append("\"floor_no\":\""+floor_no+"\"");
          json.append("\"room_no\":\""+room_no+"\"");
          json.append("\"is_active\":\""+is_active+"\"");
          json.append("}");

          System.out.println(json);

          // send a JSON data
          try {
             post.setEntity(new StringEntity(json.toString()));
          } catch (UnsupportedEncodingException e1) {
             e1.printStackTrace();
          }

          try (CloseableHttpClient httpClient = HttpClients.createDefault();
              CloseableHttpResponse response = httpClient.execute(post)) {

             result = EntityUtils.toString(response.getEntity());
          }catch (Exception e) {

          }

          System.out.println(result);

          return result;

       }

       @Override
       public String deleteSensor(String sensorid) throws RemoteException {
          String result = "";
         HttpPost post = new
HttpPost("http://localhost:8080/rest/sensor/delete/"+sensorid+"");


          try (CloseableHttpClient httpClient = HttpClients.createDefault();
              CloseableHttpResponse response = httpClient.execute(post)) {

             result = EntityUtils.toString(response.getEntity());
          } catch (IOException e) {
             e.printStackTrace();
          }

          System.out.println(result);

          return result;



       }
```

```
        @Override
        public String getSensorDetails(String sensorid) throws RemoteException {



          String result = "";
          HttpGet get = new
HttpGet("http://localhost:8080/rest/sensor/getsensor/"+sensorid+"");


          try (CloseableHttpClient httpClient = HttpClients.createDefault();
              CloseableHttpResponse response = httpClient.execute(get)) {

             result = EntityUtils.toString(response.getEntity());
          } catch (IOException e) {
             e.printStackTrace();
          }

          System.out.println(result);

          return result;

        }

        public  String adminLogin(String username, String password) throws
RemoteException {
           String res="";
           try {
             res =sendPOSTAdminLogin(username,password);
          } catch (IOException e) {
             e.printStackTrace();
          }

            return res;
        }

        public static void main(String[] args) {
            System.setProperty("java.security.policy", "file:allowall.policy");

            try{
                FireAlarmServer svr = new FireAlarmServer();
                // Bind the remote object's stub in the registry
                Registry registry = LocateRegistry.getRegistry();
                registry.bind("FireAlarmService", svr);

                System.out.println ("Service started....");
            }
            catch(RemoteException re){
                System.err.println(re.getMessage());
            }
            catch(AlreadyBoundException abe){
                System.err.println(abe.getMessage());
            }
        }
```

```
       private static String sendPOSTAdminLogin(String username, String password)
throws IOException {

         String result = "";
         HttpPost post = new HttpPost("http://localhost:8080/rest/users/auth");

         post.addHeader("content-type", "application/json");

         StringBuilder json = new StringBuilder();
         json.append("{");
         json.append("\"username\":"+username+",");
         json.append("\"password\":\""+password+"\"");
         json.append("}");

         System.out.println(json);

         // send a JSON data
         post.setEntity(new StringEntity(json.toString()));

         try (CloseableHttpClient httpClient = HttpClients.createDefault();
             CloseableHttpResponse response = httpClient.execute(post)) {

            result = EntityUtils.toString(response.getEntity());
         }

         System.out.println(result);

         return result;

      }

      @Override
      public String getAllSensorDetails() throws RemoteException {

         String result = "";
         HttpPost post = new HttpPost("http://localhost:8080/rest/sensor/all");

         post.addHeader("content-type", "application/json");

         StringBuilder json = new StringBuilder();
         json.append("{");
         json.append("}");

         System.out.println(json);

         // send a JSON data
         try {
            post.setEntity(new StringEntity(json.toString()));
         } catch (UnsupportedEncodingException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
```

```
        try (CloseableHttpClient httpClient = HttpClients.createDefault();
            CloseableHttpResponse response = httpClient.execute(post)) {

           result = EntityUtils.toString(response.getEntity());
        } catch (IOException e) {
           e.printStackTrace();
        }

        System.out.println(result);

        return result;

    }
}
```

ClientApp (GUI)

```
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.MalformedURLException;
import java.util.ArrayList;
import java.util.List;

import com.google.gson.*;
import org.apache.http.HttpEntity;
import org.apache.http.ParseException;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;


import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.stage.Stage;

public class ClientApp extends Application  {
```

```
    Stage window;

      private TableView<SensorData> table = new TableView<SensorData>();
        private ObservableList<SensorData> data =
            FXCollections.observableArrayList();


    public static void main(String[] args) {
//      start();
        launch(args);

    }

    @Override
    public void start(Stage primaryStage) throws Exception {
        window = primaryStage;
        window.setTitle("Fire-Sensor Desktop Client");
        getsensorData();
        BorderPane borderpane = new BorderPane();
        borderpane.setTop(addHBox());
        borderpane.setCenter(addVBox());

        Scene scene = new Scene(borderpane, 800, 400);
        window.setScene(scene);
        window.show();

    }

    public HBox addHBox() {
        HBox hbox = new HBox();
        hbox.setPadding(new Insets(15, 12, 15, 12));
        hbox.setSpacing(10);
        hbox.setStyle("-fx-background-color: #336699;");

        Button buttonCurrent = new Button("Admin Login");
        buttonCurrent.setPrefSize(150, 20);

        buttonCurrent.setOnAction(e->{
           Login.adminLoginView();
        });

        hbox.getChildren().addAll(buttonCurrent);

        return hbox;
    }

    public VBox addVBox() {

        final Label label = new Label("Fire-Sensor Monitor");
         label.setFont(new Font("Arial", 20));

//       TableView table = new TableView();

         table.setEditable(false);
```

```
        TableColumn sensoridCol = new TableColumn("Sensor ID");
          sensoridCol.setMinWidth(100);
          sensoridCol.setCellValueFactory( new PropertyValueFactory<SensorData,
String>("sensorid"));

           TableColumn room_noCol = new TableColumn("Room No");
           room_noCol.setMinWidth(100);
           room_noCol.setCellValueFactory( new PropertyValueFactory<SensorData,
String>("room_no"));

           TableColumn floor_noCol = new TableColumn("Floor No");
           floor_noCol.setMinWidth(100);
           floor_noCol.setCellValueFactory( new PropertyValueFactory<SensorData,
String>("floor_no"));

           TableColumn is_activeCol = new TableColumn("Sensor Status");
           is_activeCol.setMinWidth(100);
           is_activeCol.setCellValueFactory( new PropertyValueFactory<SensorData,
String>("is_active"));

           TableColumn co2levelCol = new TableColumn("CO2 Level");
           co2levelCol.setMinWidth(100);
           co2levelCol.setCellValueFactory( new PropertyValueFactory<SensorData,
String>("co2level"));

           TableColumn smoke_levelCol = new TableColumn("Smoke Level");
           smoke_levelCol.setMinWidth(100);
           smoke_levelCol.setCellValueFactory(new PropertyValueFactory<SensorData,
String>("smoke_level"));

           TableColumn dateCol = new TableColumn("Date");
           dateCol.setMinWidth(100);
           dateCol.setCellValueFactory( new PropertyValueFactory<SensorData,
String>("date"));

           TableColumn timeCol = new TableColumn("Time");
           timeCol.setMinWidth(100);
           timeCol.setCellValueFactory( new PropertyValueFactory<SensorData,
String>("time"));

           table.setItems(data);

           table.getColumns().addAll(sensoridCol, room_noCol,
floor_noCol,is_activeCol, co2levelCol, smoke_levelCol, dateCol, timeCol);

           final VBox vbox1 = new VBox();
           vbox1.setSpacing(5);
           vbox1.setPadding(new Insets(10, 0, 0, 10));
           vbox1.getChildren().addAll(label, table);


       return vbox1;
   }
```

```
   public void getsensorData() {

      String result = "";
        HttpGet request = new HttpGet("http://localhost:8080/rest/sensordata/all");



       try (CloseableHttpClient httpClient = HttpClients.createDefault();
                CloseableHttpResponse response = httpClient.execute(request)) {

               // Get HttpResponse Status
               System.out.println(response.getProtocolVersion());              //
HTTP/1.1
               System.out.println(response.getStatusLine().getStatusCode());   // 200
               System.out.println(response.getStatusLine().getReasonPhrase()); // OK
               System.out.println(response.getStatusLine().toString());        //
HTTP/1.1 200 OK

               HttpEntity entity = response.getEntity();
               if (entity != null) {
                   // return it as a String
                   result = EntityUtils.toString(entity);
//                 System.out.println(result);
               }

           } catch (ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      data.removeAll(data);
//    System.out.println(result);
      JsonArray arr = new Gson().fromJson(result, JsonArray.class);
      for(JsonElement e : arr){
         System.out.println(e.getAsJsonObject());
         JsonObject o = e.getAsJsonObject();

         data.add(new SensorData(
               o.get("sensor").getAsJsonObject().get("sensorid").toString(),
               o.get("sensor").getAsJsonObject().get("roomNo").toString(),
               o.get("sensor").getAsJsonObject().get("floorNo").toString(),
               o.get("sensor").getAsJsonObject().get("active").toString(),
               o.get("co2Level").toString(),
               o.get("smokeLevel").toString(),
               o.get("date").toString(),
               o.get("time").toString()));
      }
```

Login Form

```
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.stage.Modality;
import javafx.stage.Stage;
import utils.Utils;

import java.sql.*;

import org.json.JSONException;
import org.json.JSONObject;

import org.json.JSONException;
import org.json.JSONObject;

import java.io.Console;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.lang.SecurityManager;
import java.rmi.RemoteException;


public class Login {

   public static void adminLoginView() {
      Stage window = new Stage();
      window.initModality(Modality.APPLICATION_MODAL);
      window.setTitle("Admin Login");
      window.setWidth(400);
      window.setHeight(600);

      GridPane grid = new GridPane();
      grid.setAlignment(Pos.CENTER);
      grid.setHgap(10);
      grid.setVgap(10);
      grid.setPadding(new Insets(25, 25, 25, 25));

      Text scenetitle = new Text("Admin Login");
      scenetitle.setFont(Font.font("Tahoma", FontWeight.BOLD, 20));
```

```
        grid.add(scenetitle, 0, 0, 2, 1);

        Label userName = new Label("User Name:");
        grid.add(userName, 0, 1);

        TextField userTextField = new TextField();
        grid.add(userTextField, 1, 1);

        Label pw = new Label("Password:");
        grid.add(pw, 0, 2);

        PasswordField pwBox = new PasswordField();
        grid.add(pwBox, 1, 2);

        Button btn = new Button("Sign in");
        HBox hbBtn = new HBox(10);
        hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
        hbBtn.getChildren().add(btn);
        grid.add(hbBtn, 1, 4);

        btn.setOnAction(e->{

           if(userTextField.getText().equals("")  || pwBox.getText().equals("") ) {
              Alert a = new Alert(AlertType.NONE);
                a.setAlertType(AlertType.ERROR);
                a.setContentText("Please Enter Username and Password");
                 a.show();
           }else {
              String username = userTextField.getText();
              String password = pwBox.getText();


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

              FireAlarmService service = null;
              try {
                 service = (FireAlarmService)
Naming.lookup("//localhost/FireAlarmService");
                 String result = service.adminLogin(username,password);

                 try {
                       JSONObject jsonObject = new JSONObject(result);

                       if(!jsonObject.isEmpty()) {
                         //Successully Login
                         String loggedinName ="";// jsonObject.name;
                         window.close();
                         AdminDashboard.displayAdminDashboard(loggedinName);
                       }else {
                         //Invalid Username or password
                        Alert a = new Alert(AlertType.NONE);
                        a.setAlertType(AlertType.ERROR);
                        a.setContentText("Incorrect username or password");
                        a.show();
                       }
```

```
                }catch (JSONException err){
                     System.out.println(err);
                }

            } catch (NotBoundException ex) {
                ex.printStackTrace();
            } catch (MalformedURLException ex) {
                System.err.println(ex.getMessage());
            } catch (RemoteException ex) {
                System.err.println(ex.getMessage());
            }




        }
    });

    Scene scene = new Scene(grid, 300, 275);
    window.setScene(scene);

    window.showAndWait();

  }
}
```

Admin Dashboard

```
import com.google.gson.Gson;
import com.google.gson.JsonObject;
import javafx.collections.FXCollections;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.stage.Modality;
import javafx.stage.Stage;
import utils.Utils;

import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
//import javax.swing.*;
import java.sql.*;

import org.json.JSONException;
import org.json.JSONObject;

public class AdminDashboard {
   public static void displayAdminDashboard(String name) {

      Stage window = new Stage();
      window.initModality(Modality.APPLICATION_MODAL);
      window.setTitle("Admin Dashboard");
      window.setWidth(1200);
      window.setHeight(600);


      GridPane grid = new GridPane();
      grid.setAlignment(Pos.BASELINE_LEFT);
      grid.setHgap(10);
      grid.setVgap(10);
      grid.setPadding(new Insets(25, 25, 25, 25));

      Label lblTopic = new Label("Admin Dashboard");
      lblTopic.setFont(Font.font("Verdana", FontWeight.BOLD, 35));
      grid.add(lblTopic, 3, 0);


      Label lblSensorId = new Label("Sensor ID:");
      grid.add(lblSensorId, 0, 1);
```

```
        TextField sensorIdTextField = new TextField();
        grid.add(sensorIdTextField, 1, 1);



        Label lblFloor_no = new Label("Floor No:");
        grid.add(lblFloor_no, 0, 2);
        TextField floor_noTextField = new TextField();
        grid.add(floor_noTextField, 1, 2);



        Label lblRoom_no = new Label("Room No:");
        grid.add(lblRoom_no, 0, 3);
        TextField room_noTextField = new TextField();
        grid.add(room_noTextField, 1, 3);


        Label lblIs_active = new Label("Active Status:");
        grid.add(lblIs_active, 0, 4);
        String activeStatus[] ={ "Active", "Inactive" };
        ComboBox combo_box = new
ComboBox(FXCollections.observableArrayList(activeStatus));
        combo_box.getSelectionModel().select(0);
        grid.add(combo_box, 1, 4);



        Button btnAddUpdate = new Button("Add New Sensor");
        HBox hbBtn = new HBox(10);
        hbBtn.setAlignment(Pos.CENTER);
        hbBtn.getChildren().add(btnAddUpdate);
        grid.add(hbBtn, 0, 5);


        Label lblSensorIdForUpdate = new Label("Sensor ID :");
        grid.add(lblSensorIdForUpdate, 4, 1);
        TextField sensorIdForUpdateTextField = new TextField();
        grid.add(sensorIdForUpdateTextField, 5, 1);



        Button btnGetDetailsForUpdate = new Button("Get Details For Update");
        HBox hbBtn2 = new HBox(10);
        hbBtn2.setAlignment(Pos.CENTER);
        grid.add(btnGetDetailsForUpdate, 4, 2);

        Button btnDelete = new Button("Delete");
        HBox hbBtn3 = new HBox(10);
        hbBtn3.setAlignment(Pos.CENTER);
        grid.add(btnDelete, 5, 2);
```

```
      btnAddUpdate.setOnAction(e ->{


         if(sensorIdTextField.getText().equals("")  ||
floor_noTextField.getText().equals("") || room_noTextField.getText().equals("")){
            Alert a = new Alert(Alert.AlertType.NONE);
            a.setAlertType(Alert.AlertType.ERROR);
            a.setContentText("Please fill all fields");
            a.show();
         }else if ( ! AdminDashboard.isNumeric(floor_noTextField.getText()) ){
            Alert a = new Alert(Alert.AlertType.NONE);
            a.setAlertType(Alert.AlertType.ERROR);
            a.setContentText("Please Enter a number as a Floor number");
            a.show();
         }else{
            if(btnAddUpdate.getText().equals("Add New Sensor")){
               System.setProperty("java.security.policy", "file:allowall.policy");

               FireAlarmService service = null;
               try {

                  String sensorId = sensorIdTextField.getText();
                  int floorNo= Integer.parseInt(floor_noTextField.getText());
                  String roomNo = room_noTextField.getText();
                  boolean status;
                  if(combo_box.getSelectionModel().getSelectedIndex() == 0) {
                     status= true;
                  }else {
                     status = false;
                  }

                  service = (FireAlarmService)
Naming.lookup("//localhost/FireAlarmService");
                  String result = service.addSensor(sensorId,floorNo, roomNo, status
);


                  try {
                     JSONObject jsonObject = new JSONObject(result);
                     if(!jsonObject.isEmpty()) {
                        Alert a1 = new Alert(AlertType.NONE);
                        a1.setAlertType(AlertType.CONFIRMATION);
                        a1.setContentText("Added Successfully");
                        a1.show();
                     }else {
                        Alert a2 = new Alert(AlertType.NONE);
                        a2.setAlertType(AlertType.ERROR);
                        a2.setContentText("Error occurred while adding");
                        a2.show();
                     }

                  }catch (JSONException err){
                     System.out.println(err);
                  }
```

```
                } catch (NotBoundException ex) {
                    ex.printStackTrace();
                } catch (MalformedURLException ex) {
                    System.err.println(ex.getMessage());
                } catch (RemoteException ex) {
                    System.err.println(ex.getMessage());
                }
            }else if(btnAddUpdate.getText().equals("Update Sensor")){
                System.setProperty("java.security.policy", "file:allowall.policy");

                FireAlarmService service = null;
                try {

                    String sensorId = sensorIdTextField.getText();
                    int floorNo= Integer.parseInt(floor_noTextField.getText());
                    String roomNo = room_noTextField.getText();
                    boolean status;
                    if(combo_box.getSelectionModel().getSelectedIndex() == 0) {
                        status= true;
                    }else {
                        status = false;
                    }

                    service = (FireAlarmService)
Naming.lookup("//localhost/FireAlarmService");
                    String result = service.addSensor(sensorId,floorNo, roomNo, status
);


                    try {
                        JSONObject jsonObject = new JSONObject(result);
                        if(!jsonObject.isEmpty()) {
                            Alert a1 = new Alert(AlertType.NONE);
                            a1.setAlertType(AlertType.CONFIRMATION);
                            a1.setContentText("Updated Successfully");
                            a1.show();
                        }else {
                            Alert a2 = new Alert(AlertType.NONE);
                            a2.setAlertType(AlertType.ERROR);
                            a2.setContentText("Error occurred while updating");
                            a2.show();
                        }

                    }catch (JSONException err){
                        System.out.println(err);
                    }

                } catch (NotBoundException ex) {
                    ex.printStackTrace();
                } catch (MalformedURLException ex) {
                    System.err.println(ex.getMessage());
                } catch (RemoteException ex) {
                    System.err.println(ex.getMessage());
                }
```

```
            }

         }

      });
      btnGetDetailsForUpdate.setOnAction(e ->{
         System.out.println("Update Clicked");

         if(sensorIdForUpdateTextField.getText().equals("")){
            Alert a = new Alert(Alert.AlertType.NONE);
            a.setAlertType(Alert.AlertType.ERROR);
            a.setContentText("Please enter sensor ID for update");
            a.show();
         }else{
            //Button- Get Details for update - start
            System.setProperty("java.security.policy", "file:allowall.policy");

            FireAlarmService service = null;
            try {

               String sensorId = sensorIdForUpdateTextField.getText();
//             int floorNo= Integer.parseInt(floor_noTextField.getText());
//             String roomNo = room_noTextField.getText();
//             boolean status;
//             if(combo_box.getSelectionModel().getSelectedIndex() == 0) {
//                status= true;
//             }else {
//                status = false;
//             }

               service = (FireAlarmService)
Naming.lookup("//localhost/FireAlarmService");
               String result =
service.getSensorDetails(sensorIdForUpdateTextField.getText());

               try {
                  JSONObject jsonObject = new JSONObject(result);

                  try{
                     JsonObject jsonObj = new Gson().fromJson(result,
JsonObject.class);
                     sensorIdTextField.setText( jsonObj.get("sensorid").getAsString()
);
                     floor_noTextField.setText( jsonObj.get("floorNo").getAsString()
);
                     room_noTextField.setText( jsonObj.get("roomNo").getAsString()
);


                     if(jsonObj.get("active").getAsBoolean() == false){
                        combo_box.getSelectionModel().select(1);
                     }else{
                        combo_box.getSelectionModel().select(0);
                     }
                     btnAddUpdate.setText("Update Sensor");
```

```
                    sensorIdForUpdateTextField.setText("");
                    sensorIdTextField.setEditable(false);
                }catch (Exception ex){
                    Alert a1 = new Alert(AlertType.NONE);
                    a1.setAlertType(AlertType.ERROR);
                    a1.setContentText("Sensor Not Found");
                    a1.show();
                }

             }catch (JSONException err){
                System.out.println(err);
             }

          } catch (NotBoundException ex) {
             ex.printStackTrace();
          } catch (MalformedURLException ex) {
             System.err.println(ex.getMessage());
          } catch (RemoteException ex) {
             System.err.println(ex.getMessage());
          }

       }


    });
    btnDelete.setOnAction(e ->{
       System.out.println("Delete Clicked");

       if(sensorIdForUpdateTextField.getText().equals("")) {
          Alert a = new Alert(Alert.AlertType.NONE);
          a.setAlertType(Alert.AlertType.ERROR);
          a.setContentText("Please add sensor ID for delete");
          a.show();
       }else {

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

             FireAlarmService service = null;
             try {

                String sensorId = sensorIdForUpdateTextField.getText();

                service = (FireAlarmService)
Naming.lookup("//localhost/FireAlarmService");
                String result = service.deleteSensor(sensorId);

                try {
                    JSONObject jsonObject = new JSONObject(result);
                    if(!jsonObject.isEmpty()) {
                         Alert a1 = new Alert(AlertType.NONE);
                         a1.setAlertType(AlertType.CONFIRMATION);
                         a1.setContentText("Deleted Successfully");
                         a1.show();
                    }else {
                         Alert a2 = new Alert(AlertType.NONE);
```

```
                        a2.setAlertType(AlertType.ERROR);
                        a2.setContentText("Error occurred");
                        a2.show();
                    }

                }catch (JSONException err){
                    System.out.println(err);
                }

            } catch (NotBoundException ex) {
                ex.printStackTrace();
            } catch (MalformedURLException ex) {
                System.err.println(ex.getMessage());
            } catch (RemoteException ex) {
                System.err.println(ex.getMessage());
            }
        }

    });


    Scene scene = new Scene(grid, 300, 275);
    window.setScene(scene);

    window.showAndWait();

}

public static boolean isNumeric(String strNum) {
    if (strNum == null) {
        return false;
    }
    try {
        double d = Double.parseDouble(strNum);
    } catch (NumberFormatException nfe) {
        return false;
    }
    return true;
}
}
```

SensorData classs

```
import javafx.beans.property.SimpleStringProperty;

public class SensorData {
    private final SimpleStringProperty sensorid;
    private final SimpleStringProperty room_no ;
    private final SimpleStringProperty floor_no ;
    private final SimpleStringProperty is_active ;
    private final SimpleStringProperty co2level;
    private final SimpleStringProperty smoke_level;
    private final SimpleStringProperty date;
    private final SimpleStringProperty time;


    SensorData(String sensorid, String room_no, String floor_no, String is_active,
String co2level, String smoke_level,
                                                                    String date, String
time) {

        this.sensorid = new SimpleStringProperty(sensorid);
        this.room_no = new SimpleStringProperty(room_no);
        this.floor_no = new SimpleStringProperty(floor_no);
        this.is_active = new SimpleStringProperty(is_active);
        this.co2level = new SimpleStringProperty(co2level);
        this.smoke_level = new SimpleStringProperty(smoke_level);
        this.date = new SimpleStringProperty(date);
        this.time = new SimpleStringProperty(time);
    }


    public String getSensorid() {
        return sensorid.get();
    }
    public String getRoom_no() {
        return room_no.get();
    }
    public String getFloor_no() {
        return floor_no.get();
    }
    public String getIs_active() {
        return is_active.get();
    }
    public String getCo2level() {
        return co2level.get();
    }
    public String getSmoke_level() {
        return smoke_level.get();
    }
    public String getDate() {
        return date.get();
    }
```

```
    public String getTime() {
        return time.get();
    }


    public void setSensorid(String sensorid) {
       this.sensorid.set(sensorid);
    }
    public void setRoom_no(String room_no) {
       this.room_no.set(room_no);
    }
    public void setFloor_no(String floor_no) {
       this.floor_no.set(floor_no);
    }
    public void setIs_active(String is_active) {
       this.is_active.set(is_active);
    }
    public void setco2level(String co2level) {
       this.co2level.set(co2level);
    }
    public void setSmoke_level(String smoke_level) {
       this.smoke_level.set(smoke_level);
    }
    public void setDate(String date) {
       this.date.set(date);
    }
    public void setTime(String time) {
       this.time.set(time);
    }


}
```

Backend

Sensor Model Class

```
package com.techprimers.db.springbootmysql.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Sensor {

    @Id
    @Column(name = "sensorid")
    private String sensorid;

    @Column(name = "floorNo")
    private int floorNo ;

    @Column(name = "roomNo")
    private String roomNo;

    @Column(name = "isActive")
    private boolean isActive;



    public String getSensorid() {
        return sensorid;
    }

    public void setSensorid(String sensorid) {
        this.sensorid = sensorid;
    }


    public int getFloorNo() {
        return floorNo;
    }

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

    public String getRoomNo() {
        return roomNo;
    }

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

```
    public boolean isActive() {
        return isActive;
    }

    public void setActive(boolean active) {
        isActive = active;
    }

}
```

SensorData Model Class

```
package com.techprimers.db.springbootmysql.model;

import javax.persistence.*;

@Entity
public class SensorData {

    @Id
    @Column(name = "sensordataid")
    private String sensordataid;

    @ManyToOne
    private Sensor sensor;

    @Column(name = "smokeLevel")
    private int smokeLevel;

    @Column(name = "co2Level")
    private int co2Level;

    @Column(name = "date")
    private String date;

    @Column(name = "time")
    private String time;

    public SensorData() {
    }

    public Sensor getSensor() {
        return sensor;
    }

    public void setSensor(Sensor sensor) {
        this.sensor = sensor;
    }

    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 String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public String getTime() {
        return time;
    }

    public void setTime(String time) {
        this.time = time;
    }

    public String getSensordataid() {
        return sensordataid;
    }

    public void setSensordataid(String sensordataid) {
        this.sensordataid = sensordataid;
    }
}
```

Users Model Class

```
package com.techprimers.db.springbootmysql.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Users {

    @Id
    @GeneratedValue
    @Column(name = "id")
    private Integer id;
    @Column(name = "name")
    private String name;

    @Column(name = "type")
    private String type;

    @Column(name = "password")
    private String password;


    public Users() {
    }

    public Integer getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }
```

```
    public String getPassword() {
        return password;
    }

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

Sending Email

```
package com.techprimers.db.springbootmysql.NotifyUser;
import java.util.Properties;

import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class SendEmail {

//    public static void main(String[] args) {
//        sendEmailtoUser();
//    }

    public static void sendEmailtoUser(){
        // Recipient's email ID needs to be mentioned.
        String to =  "receiver@gmail.com";

        // Sender's email ID needs to be mentioned
        String from = "sender@gmail.com";

        // Assuming you are sending email from through gmails smtp
        String host = "smtp.gmail.com";

        // Get system properties
        Properties properties = System.getProperties();

        // Setup mail server
        properties.put("mail.smtp.host", host);
        properties.put("mail.smtp.port", "465");
        properties.put("mail.smtp.ssl.enable", "true");
        properties.put("mail.smtp.auth", "true");

        // Get the Session object.// and pass username and password
        Session session = Session.getInstance(properties, new
javax.mail.Authenticator() {
```

```
            protected PasswordAuthentication getPasswordAuthentication() {

                return new PasswordAuthentication("sender@gmail.com", "xxxxxxxxxxx");

            }

        });

        // Used to debug SMTP issues
        session.setDebug(true);

        try {
            // Create a default MimeMessage object.
            MimeMessage message = new MimeMessage(session);

            // Set From: header field of the header.
            message.setFrom(new InternetAddress(from));

            // Set To: header field of the header.
            message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));

            // Set Subject: header field
            message.setSubject("Warning!");

            // Now set the actual message
            message.setText("Co2 level or Smoke level is above 5");

//            System.out.println("sending...");
            // Send message
            Transport.send(message);
            System.out.println("Sent message successfully....");
        } catch (MessagingException mex) {
            mex.printStackTrace();
        }
    }

}
```

Sending SMS

```
package com.techprimers.db.springbootmysql.NotifyUser;

import com.techprimers.db.springbootmysql.utils.Properties;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class SendSMS {

    public static String sendSmsMessage() {
        try {
            // Construct data
            String apiKey = "apikey=" + Properties.smsAPIKey;
            String message = "&message=" + "This is your message";
            String sender = "&sender=" + "TXTLCL";
            String numbers = "&numbers=" + Properties.smsToNumber;

            // Send data
            HttpURLConnection conn = (HttpURLConnection) new
URL("https://api.textlocal.in/send/?").openConnection();
            String data = apiKey + numbers + message + sender;
            conn.setDoOutput(true);
            conn.setRequestMethod("POST");
            conn.setRequestProperty("Content-Length",
Integer.toString(data.length()));
            conn.getOutputStream().write(data.getBytes("UTF-8"));
            final BufferedReader rd = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
            final StringBuffer stringBuffer = new StringBuffer();
            String line;
            while ((line = rd.readLine()) != null) {
                stringBuffer.append(line);
            }
            rd.close();
            System.out.println("Sent Successfully");
            return stringBuffer.toString();
        } catch (Exception e) {
            System.out.println("Error SMS "+e);
            return "Error "+e;
        }
    }


}
```

SensorRepository

```
package com.techprimers.db.springbootmysql.repository;

import com.techprimers.db.springbootmysql.model.Sensor;
import org.springframework.data.jpa.repository.JpaRepository;

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

SensorDataRepository

```
package com.techprimers.db.springbootmysql.repository;

import com.techprimers.db.springbootmysql.model.SensorData;
import org.springframework.data.jpa.repository.JpaRepository;

public interface SensorDataRepository extends JpaRepository<SensorData, Integer> {
}
```

UserRepository

```
package com.techprimers.db.springbootmysql.repository;

import com.techprimers.db.springbootmysql.model.Users;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UsersRepository extends JpaRepository<Users, Integer> {
}
```

SensorDataResource

```
package com.techprimers.db.springbootmysql.resource;

import com.techprimers.db.springbootmysql.NotifyUser.SendEmail;
import com.techprimers.db.springbootmysql.model.SensorData;
import com.techprimers.db.springbootmysql.model.Users;
import com.techprimers.db.springbootmysql.repository.SensorDataRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@CrossOrigin
@RequestMapping(value = "/rest/sensordata")
public class SensorDataResource {

    @Autowired
    SensorDataRepository sensorDataRepository;

    @GetMapping(value = "/all")
    public List<SensorData> getAll(){
        return sensorDataRepository.findAll();
    }

    @PostMapping(value = "/load")
    public List<SensorData> persist(@RequestBody final SensorData sensorData){


        if(sensorData.getCo2Level() >5 ){
            SendEmail.sendEmailtoUser();
        }
        if(sensorData.getSmokeLevel() > 5 ){
            SendEmail.sendEmailtoUser();
        }


        sensorDataRepository.save(sensorData);
        return sensorDataRepository.findAll();
    }

}
```

SensorResource

```
package com.techprimers.db.springbootmysql.resource;

import com.techprimers.db.springbootmysql.model.Sensor;
import com.techprimers.db.springbootmysql.model.SensorData;
import com.techprimers.db.springbootmysql.repository.SensorDataRepository;
```

```java
import com.techprimers.db.springbootmysql.repository.SensorRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@CrossOrigin
@RequestMapping(value = "/rest/sensor")
public class SensorResource {

    @Autowired
    SensorRepository sensorRepository;

    @GetMapping(value = "/all")
    public List<Sensor> getAll(){
        return sensorRepository.findAll();
    }

    @PostMapping(value = "/load")
    public List<Sensor> persist(@RequestBody final Sensor sensor){

        System.out.println(sensor.getFloorNo());
        System.out.println(sensor.isActive());

        sensorRepository.save(sensor);
        return sensorRepository.findAll();
    }

    @PostMapping(value = "/all")
    public List<Sensor> getAll(@RequestBody final Sensor sensor){
        return sensorRepository.findAll();
    }


    @DeleteMapping(value ="/delete/{sensorid}")
    void deleteEmployee(@PathVariable String sensorid) {
        sensorRepository.deleteById(sensorid);
    }

    @GetMapping(value ="/getsensor/{id}")
    Sensor one(@PathVariable String id) throws Exception {
        return sensorRepository.findById(id)
                .orElseThrow(() -> new Exception(String.valueOf(id)));
    }

}
```

UsersResource

```
package com.techprimers.db.springbootmysql.resource;

import com.techprimers.db.springbootmysql.model.Users;
import com.techprimers.db.springbootmysql.repository.UsersRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping(value = "/rest/users")
public class UsersResource {

    @Autowired
    UsersRepository usersRepository;

    @GetMapping(value = "/all")
    public List<Users> getAll(){
        return usersRepository.findAll();
    }

    @PostMapping(value = "/load")
    public List<Users> persist(@RequestBody final Users user){
        usersRepository.save(user);
        return usersRepository.findAll();
    }

    @PostMapping(value = "/auth")
    public Users authUSer(@RequestBody final Users user) throws  Exception{
        List<Users> us =usersRepository.findAll();
        for(Users u:us){
            if(u.getName().equals(user.getName()) &&
u.getPassword().equals(user.getPassword())){
                return u;
            }
        }
        throw new Exception("User not found");
    }

}
```

React Web App

App.js

```
// src/App.js

import React, {Component} from 'react';
import Sensor from './components/sensor';



class App extends Component {

  state = {
    sensordata: []
  }

  getItems() {
    fetch('http://localhost:8080/rest/sensordata/all')
    .then(res => res.json())
    .then((data) => {
      this.setState({ sensordata: data })
      console.log(data);
    })
    .catch(console.log)
  }

  componentDidMount() {
    this.getItems();
    this.timer = setInterval(()=> this.getItems(), 40000);
  }
  render () {
    return (<Sensor sensordata={this.state.sensordata} />);
  }
}

export default App;
```

components/sensor.js

```
import React from 'react'
import Table from 'react-bootstrap/Table'

const Sensor = ({ sensordata }) => {
    const badcolor = {
        color: "red"
      };
      const goodcolor = {
        color: "green"
      };

    return (
    <div>
        <Table striped bordered hover>

        <thead>
            <tr>
                <th>Sensor ID</th>
                <th>Floor No.</th>
                <th>Room No.</th>
                <th>Smoke Level</th>
                <th>CO2 Level</th>
                <th>Date</th>
                <th>Time</th>
            </tr>
        </thead>
        <tbody>
        {sensordata.map((sensord) => (
            <tr>
                <td>{sensord.sensor.sensorid}</td>
                <td>{sensord.sensor.floorNo}</td>
                <td>{sensord.sensor.roomNo}</td>
                <td>{sensord.smokeLevel}</td>
                <td style={sensord.co2Level>5?badcolor:goodcolor}>{sensord.co2Lev
el}</td>
                <td>{sensord.date}</td>
                <td>{sensord.time}</td>
            </tr>
        ))}
        </tbody>
        </Table>
    </div>
    )
```

```
};

export default Sensor
```