



Sri Lanka Institute of Information Technology

Distributed Systems Assignment 2

Fire Alarm Monitoring System

Project Report

Software Engineering 2020

Group:

Team CODE4

Date Submitted

06.05.2020

Declaration

We declare that this project report or part of it was not a copy of a document done by any organization, university any other institute or a previous student project group at SLIIT and was not copied from the Internet or other sources.

And all the references mentioned below used to study about the tools and technologies which are used to design and implement this project.

Group members.

Reg. No	Name	Signature
IT18038606	K.J. Gomez	<i>kjgomez</i>
IT18026962	V.D. Dantanarayana	<i>vidula</i>
IT18045840	S.D.S.L. Dissanayake	<i>sathira</i>
IT18134322	D.H Perera	<i>dilshan</i>

Abstract

This project is a solution for a real world problem on how a fire alarm monitoring system operates. This project is mainly focusing on the distributed architecture as the module name suggests. The project scope covers; where there is a RMI server, a desktop client, a rest API, asynchronous web client and a dummy sensor app. All these components are being interconnected to facilitate this distributed sensor system. The technologies methodologies are all being discussed in the report. The system architectures and how message passing happens the code snippets and some of their special functionalities will all be discussed in this report

Git-Hub Repo Link

https://github.com/VidulaDakshitha/Sensor_System_CODE4

Table of Contents

Abstract	ii
Table of Contents	iii
List of Figures	iv
1 Introduction and project scenario.....	1
2.The High Level Architecture	2
3.Component Diagram	3
4.Technologies Involved process and UI.	4
4.1Desktop Client	4
4.2RMI-server.....	6
4.3Rest-API.	7
4.4 Sensor App.....	8
4.5 Asynchronous web-Client	9
4.6Technology and port Summary.....	10
5. Sequence Diagram	11
6. Appendix	12
6.1 MYSQL Database Query code	12
6.2 Desktop Client Code	15
6.3 RMI- Server	50
6.4 Web-API.....	57

6.5 Sensor Application	66
------------------------------	----

List of Figures

figure 2: High-level Architecture.....	2
figure 3: Component Diagram	3
figure 4.1: The login UI.....	4
figure 4.3: Display sensor details.....	5
figure 4.2: The add sensor UI	5
figure 4.4: Email sent to admin.....	7
figure 4.5:email sent to client	8
figure 4.6: UI for web-client.....	9
figure 5: Sequence Diagram	11

1 Introduction and project scenario

The project stakeholders include system admin operating at the desktop client end and the clients/customers viewing the sensor system through the web client end. At the desktop client end the admin will be registering sensors to the system. The admin will be adding the sensor name, room name, and the floor number where the sensor is supposed to be installed. Once the sensor s being added the dummy sensor application will be generating random values to the smoke level and the carbon dioxide levels of the sensor every 10 seconds. Meanwhile the admin could edit the sensor details and also could make the sensor status inactive.

Once the sensor status is made in-active the dummy sensor application should stop generating random values for the sensor carbon dioxide and smoke levels. If in case the co2 level or the smoke level rises above 5 it should be notified to the admin and the clients in the rooms. The clients and the admin will be notified with an email which is sent mentioning the rooms that are in danger and telling them what should be done. The admin will also be notified with a notification in the desktop and the web clients are being notified at the web client end; indicating the sensors in red that are above 5. The system will check the sensor status every 15 seconds to notify the system users of the sensor status. This is how the project works to ensure that the fire alarm system ensures usability and provide immediate notifications to the users to stop any damage from happening.

The diagram below will show the system architecture on how the components in the system are being interconnected to each other to facilitate this distributed system.

2.The High Level Architecture

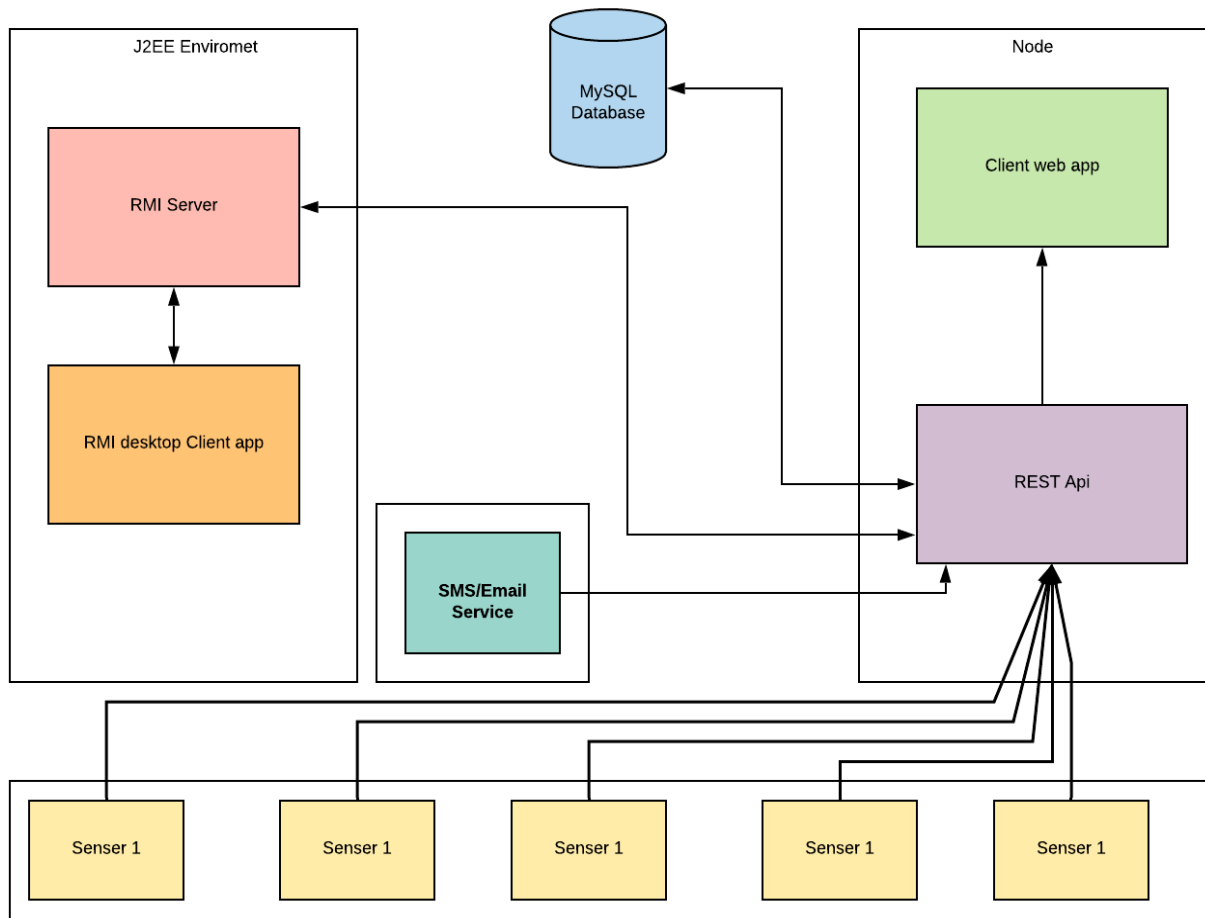


figure 2: High-level Architecture

As presented in the diagram above the distribution among the components can be seen. The functionalities involved and how the components engage with each other can be visualized well using a component diagram. The diagram below displays the component behavior.

3.Component Diagram

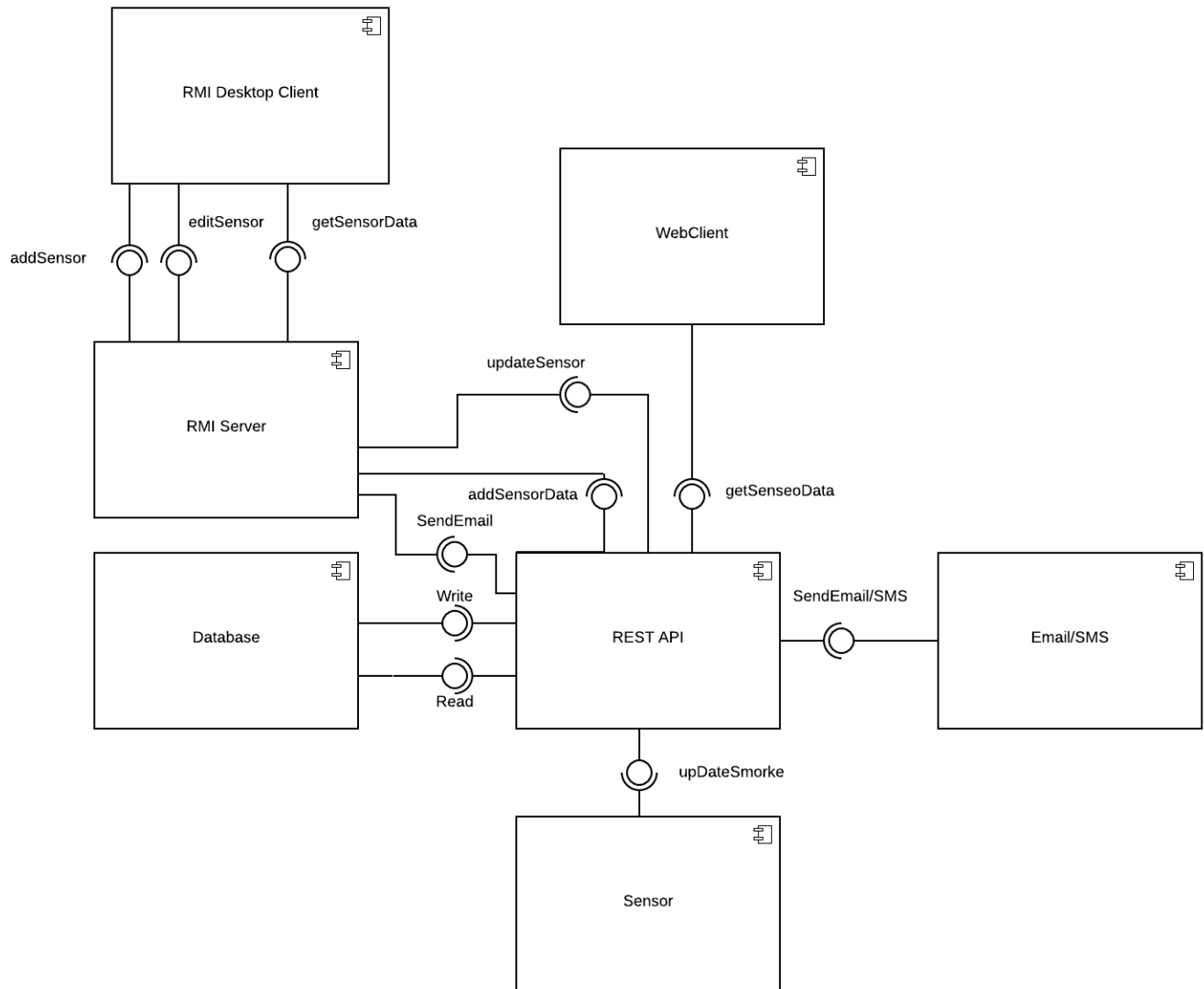


figure 3: Component Diagram

4. Technologies Involved process and UI.

As you can see in figure 2 there are different components and how they are being linked with each other are being clearly demonstrated. Each of these components uses different forms of technology.

4.1 Desktop Client

- Uses Java-fx for development.

At start-up first there is login UI. This validates login details from MYSQL DB. At first once the

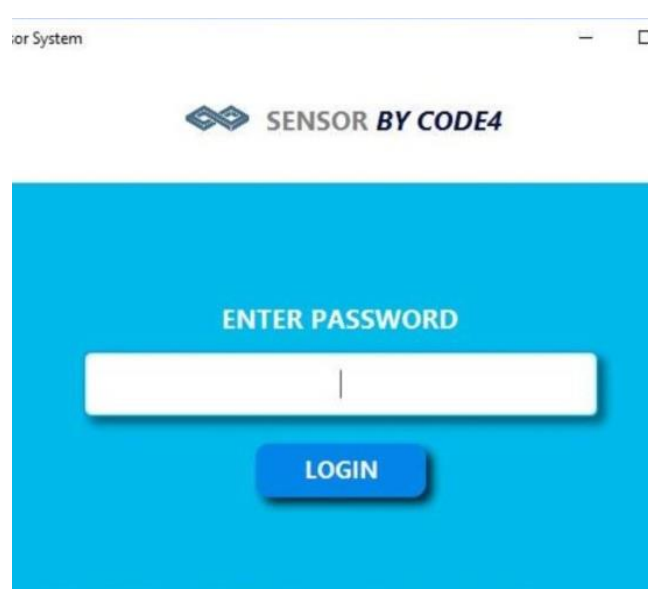


figure 4.1: The login UI

UI is loaded it requests for password. Once the password is entered the request is passed to the RMI server. The server requests for the login method in the Rest-API. This rest API then sends back the request for success or failure and upon that the admin is able to login in to the system.

After the admin logs in the admin is displayed with the add sensor UI as in the figure 4.2. The user is able to enter room no, floor number and also the sensor name. Once all of these are entered the request is passed

to the RMI server through the RMI registry connection that is being initialized already. The code snippet is displayed below.

```

@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
    System.setProperty("java.security.policy", "file:allowall.policy");
    try {
        Registry reg = LocateRegistry.getRegistry("127.0.0.1", 2000);
        sensorService = (SensorService) reg.lookup("sensorServer");

    } catch (Exception e) {

        System.err.println("error "+e);
    }
}

```

Once the connection is made the rmi registry sends the request to the RestAPi add sensor

figure 4.2: The add sensor UI

Sensor Name	Floor Name/ID	Room Name/ID	Co2 Level	Smoke Level	Status
sensor 1	floor 1	room 1	9.21	8.08	Active
sensor 2	floor 1	room 2	1.14	0.19	Active
sensor 3	floor 2	room 3	6.39	9.62	Active
sensor 4	floor 2	room 4	0.0	0.0	Inactive
sensor 5	floor 3	room 5	0.0	0.0	Inactive
sensor 6	floor 3	room 6	0.0	0.0	Inactive
sensor 7	floor 4	room 7	0.0	0.0	Inactive
sensor 8	floor 5	room 8	0.0	0.0	Inactive
Add a sensor	Floor 1 edited	Room 1	0.0	0.0	Inactive

figure 4.3: Display sensor details.

method. Then the rest-api will add the sensor to mysql db and the response will be returned to the desktop client as an alert message. Once the sensor is added admin can navigate to sensors UI. This UI will have all the added sensor details and this UI will be refreshed every 15

seconds. In order to update table every 15 seconds the desktop client will request for the get sensors method in RMI-server and RMI-server in return will request them from Rest-API. Figure 4.3 shows how the sensor table is displayed. If the smoke level or carbon dioxide level rises above 5. Then the

admin is displayed with a notification. The code snippet for the notification is shown below.

```
private void showNotification(String mess){
    Toolkit.getDefaultToolkit().beep(); // Notification sound
    Notifications.create()
        .title("CO2 Level is High ")
        .text(mess)
        .showWarning();
}
```

The admin can also update the sensor details. This is invoked using update sensor details method. And just as in other functions this also uses RMI-server and REST-api for update execution.

4.2RMI-server.

- RMI-server uses Java.

As mentioned in desktop client RMI-server is used for desktop client to interact with Rest-API. This RMI-server has some special functions.

For sending get/post/delete requests to REST-API and retrieving back responses

Also uses asynchronous threading to check whether there is any rise in co2 level or smoke level. The RMI-server checks this every 15 seconds. If there is any rise the RMI-server invokes the email method. The code snippet for this asynchronous communication and service initialization in RMI-server is as follows;

```
public static void main(String[] args) {
    // TODO code application logic here
    System.setProperty("java.security.policy", "file:allowall.policy");
    try {
        final JavaRMIServer jrmis=new JavaRMIServer();
        Registry reg =LocateRegistry.createRegistry(2000); // create rmi
registry
        reg.rebind("sensorServer", new JavaRMIServer()); // bind
JavaRMIServer class
        System.out.println ("Service started...");
    }
```

```

        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                try {
                    jirmi.SendMail();
                } catch (Exception ex) {

                    Logger.getLogger(JavaRMIServer.class.getName()).log(Level.SEVERE, null, ex);
                }
            }, 0, 15000);
    } catch (Exception e) {
        System.err.println ("err"+e);
    }

}


```

Once the email method is invoked the email method passes the request to the Rest-API. The REST-API uses nodemailer for mailing system. Nodemailer will send emails to all the clients.

4.3Rest-API.

The Rest-API is express running on node environment.

The rest-API processes all the GET/POST/DELETE requests made by the RMI-server. The REST-API has direct communication with the database. Apart from CRUD operations Rest-api does another important operation. That is to send emails to clients. Not only clients, but for admin as well. The figures below demonstrate how the different emails are sent as per the user type.

Subject: Admin Alert Important
From: Code4 Fire sensor system  <codefoursliit@gmail.com>
To: <codefoursliit@gmail.com>
Time: Today at 2:33
Message-ID: <b127a7ed-8db7-1f9d-19eb-762ffe0e9a83@gmail.com>

☒ HTML ☐ Plaintext

Hey Admin!!! Alert smoke levels co2 levels high in room 1,room 2,room 3. Take immediate action

figure 4.4: Email sent to admin

[Public URL of this message](#)

Subject: Alert Important

From: Code4 Fire sensor system 📧 <codefoursliit@gmail.com>

To: <sdsat756@gmail.com>

Time: Today at 2:32

Message-ID: <b5b8bed8-79a4-8e52-135f-3c553dc68a89@gmail.com>

☒ HTML ☐ Plaintext

Hey!!! Sathira Lamal Alert smoke levels co2 levels high in room 1,room 2,room 3. Run to the exit
Code4 fire alarming system. Thank you

figure 4.5:email sent to client

The emails address the clients uniquely with their name and the rooms where co2 and smoke levels high are informed to the clients and admin.

4.4 Sensor App

- Uses Java. A CLI application

This is a CLI application which generates random values for co2 levels and smoke levels. This sensor app will generate values for active sensors only. The code snippet for random value generation is as follows;

```
private void updateLavel() {
    DecimalFormat f = new DecimalFormat("##.00");
    DecimalFormat f2 = new DecimalFormat("##.00");
    try {
        for (Integer id : sensorId) {
            Random r= new Random();
            Random r2= new Random();
            int rangeMin=0,rangeMax=10;

            double level = Double.parseDouble( f.format(rangeMin + (rangeMax -
rangeMin) * r.nextDouble()));
            double smoke = Double.parseDouble( f2.format(rangeMin + (rangeMax -
rangeMin) * r2.nextDouble()));
            String data = updateSernsorLevel(id,level,smoke);
        }
    }
}
```

```

    }

    } catch (Exception e) {
        System.err.println("error "+e);
    }
}

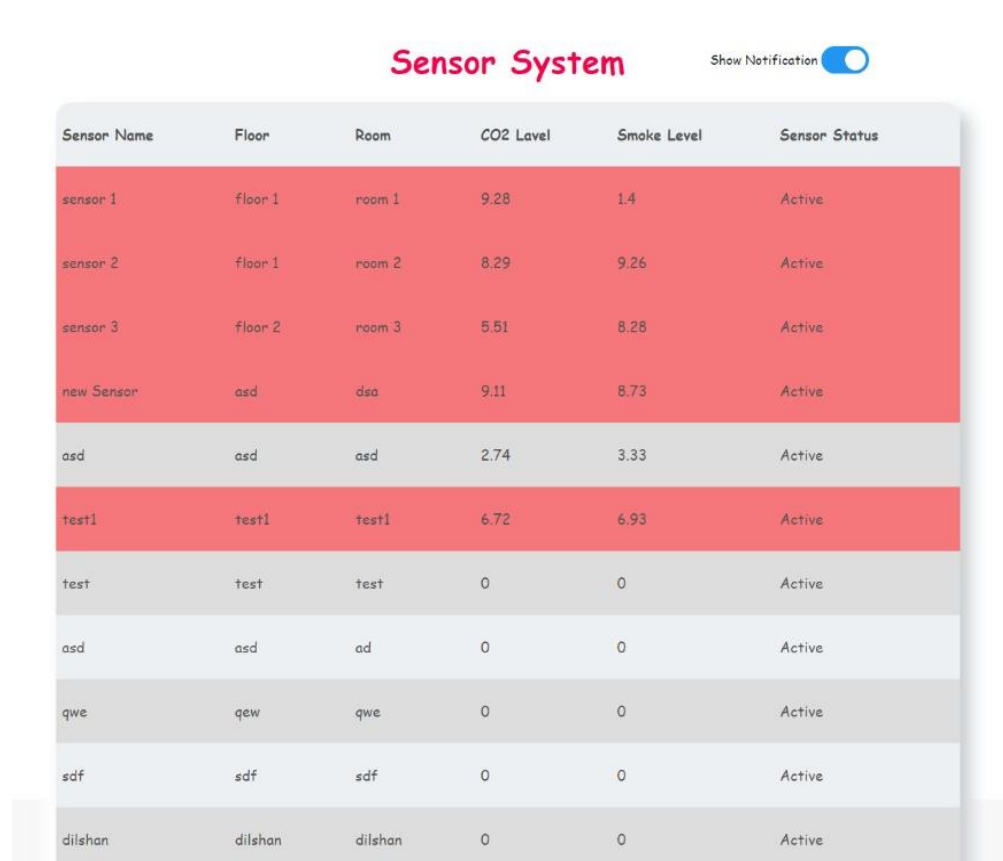
```

This sensor app will directly connect to the REST-API to update the sensor values.

4.5 Asynchronous web-Client

- Uses node for web-client.

The web-Client application displays the sensor details to the client. These sensor details should be updated every 5 second to the client. If incase the co2 levels or smoke levels are high the user



Sensor Name	Floor	Room	CO2 Level	Smoke Level	Sensor Status
sensor 1	floor 1	room 1	9.28	1.4	Active
sensor 2	floor 1	room 2	8.29	9.26	Active
sensor 3	floor 2	room 3	5.51	8.28	Active
new Sensor	asd	dsa	9.11	8.73	Active
asd	asd	asd	2.74	3.33	Active
test1	test1	test1	6.72	6.93	Active
test	test	test	0	0	Active
asd	asd	ad	0	0	Active
qwe	qew	qwe	0	0	Active
sdf	sdf	sdf	0	0	Active
dilshan	dilshan	dilshan	0	0	Active

figure 4.6: UI for web-client

update is shown below.

client is being notified with notification. The rows with co2 levels and smoke levels above 5 is displayed in red. The UI view is in figure 4.6. The node requests data from express server and displays them. The code snippet for the UI change and interval for table

- To set time interval for table updation

```
var div = document.getElementById("table");

fetch("http://localhost:3000/getSensorData")
  .then((res) => res.json())
  .then((data) => showEvents(data))
  .catch((err) => console.log(err));

setInterval(() => {
  fetch("http://localhost:3000/getSensorData")
    .then((res) => res.json())
    .then((data) => showEvents(data))
    .catch((err) => console.log(err));
}, 1000 * 5);
```

- To change UI based on co2 and smoke levels

```
if (event.colevel > 5 || event.smokelevel > 5) {
  tr.setAttribute("class", "danger");
} else if (event.status == "Inactive") {
  tr.setAttribute("class", "inactive");
} else {
  tr.setAttribute("class", "tr");
}
```

4.6 Technology and port Summary

Component Name	Technology	Ports used
Desktop client	Java-Fx	2000 registry-port
RMI-server	Java-RMI	2000-registry port
REST-API	Express server	3001 port
Web-Client	Java CLI	Not specified

5. Sequence Diagram

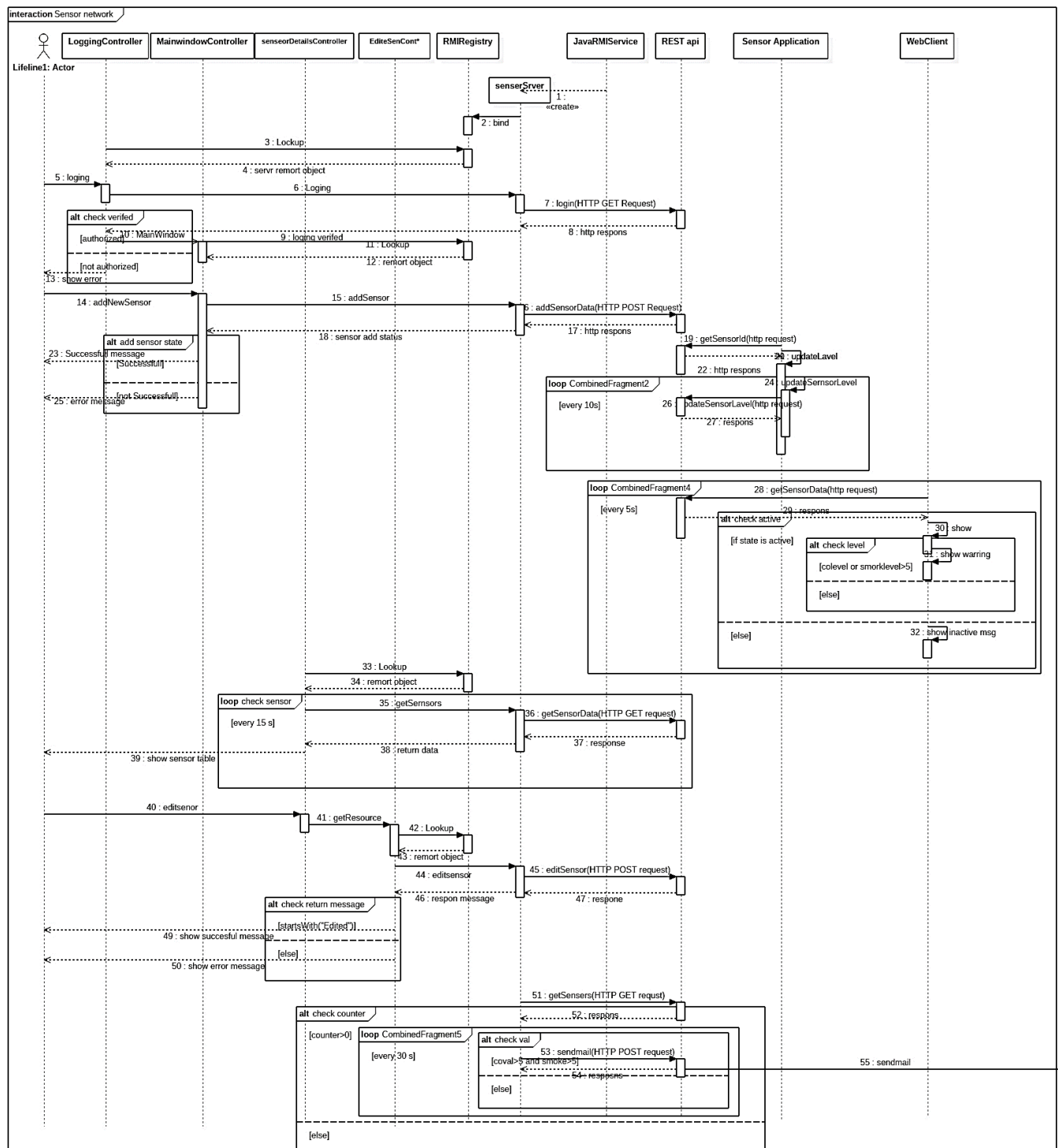


figure 5: Sequence Diagram

6. Appendix

6.1 MYSQL Database Query code

```
-- phpMyAdmin SQL Dump
-- version 4.8.5
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Apr 26, 2020 at 07:04 PM
-- Server version: 10.1.38-MariaDB
-- PHP Version: 7.3.4

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;

--
-- Database: `ds_sensor_system`
--

--
-- Table structure for table `admins`
--

CREATE TABLE `admins` (
  `id` int(11) NOT NULL,
  `name` varchar(150) NOT NULL,
  `email` varchar(150) NOT NULL,
  `username` varchar(150) NOT NULL,
  `password` varchar(150) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `admins`
--

INSERT INTO `admins` (`id`, `name`, `email`, `username`, `password`) VALUES
(1, 'admin', 'codefoursliit@gmail.com', 'admin1', '123');

--
-- Table structure for table `users`
--
```

```
--

CREATE TABLE `clients` (
  `id` int(11) NOT NULL,
  `name` varchar(150) NOT NULL,
  `email` varchar(150) NOT NULL,
  `room` int(5) NOT NULL,
  `phone` varchar(15) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--
-- Dumping data for table `clients`
--

INSERT INTO `clients` (`id`, `name`, `email`, `room`, `phone`) VALUES
(1, 'vidula', 'viduladakshitha@gmail.com', 1, '+94779819207'),
(2, 'kevin gomez', 'kevingomez890@gmail.com', 2, '+94771986561'),
(3, 'Dilshan Harendra', 'dilshanharendraperera123@gmail.com', 3,
'+94783253430'),
(4, 'Sathira Lamal', 'sdsat756@gmail.com', 4, '+94717732324');

-----

--
-- Table structure for table `sensors`
--

CREATE TABLE `sensors` (
  `id` int(11) NOT NULL,
  `name` varchar(100) DEFAULT NULL,
  `room` varchar(100) DEFAULT NULL,
  `floor` varchar(100) DEFAULT NULL,
  `status` varchar(100) DEFAULT NULL,
  `colevel` double DEFAULT NULL,
  `smokelevel` double DEFAULT NULL
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

--
-- Dumping data for table `sensors`
--

INSERT INTO `sensors` (`id`, `name`, `room`, `floor`, `status`, `colevel`,
`smokelevel`) VALUES
(1, 'sensor 1', 'room 1', 'floor 1', 'Active', 10, 10),
(2, 'sensor 2', 'room 2', 'floor 1', 'Active', 10, 10),
(3, 'sensor 3', 'room 3', 'floor 2', 'Active', 10, 10),
(4, 'sensor 4', 'room 4', 'floor 2', 'Inactive', 0, 0),
(5, 'sensor 5', 'room 5', 'floor 3', 'Inactive', 0, 0),
(6, 'sensor 6', 'room 6', 'floor 3', 'Inactive', 0, 0),
(7, 'sensor 7', 'room 7', 'floor 4', 'Inactive', 0, 0),
(8, 'sensor 8', 'room 8', 'floor 5', 'Inactive', 0, 0);

--
-- Indexes for dumped tables
--

--
```

```

-- Indexes for table `admins`
--
ALTER TABLE `admins`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `clients`
--
ALTER TABLE `clients`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `sensors`
--
ALTER TABLE `sensors`
  ADD PRIMARY KEY (`id`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `admins`
--
ALTER TABLE `admins`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table `clients`
--
ALTER TABLE `clients`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;

--
-- AUTO_INCREMENT for table `sensors`
--
ALTER TABLE `sensors`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=9;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

```

6.2 Desktop Client Code

Add sensor FXML

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<AnchorPane id="AnchorPane" prefHeight="400.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sensorsystem.AddNewSensorController">
    <children>
        <VBox fx:id="addNewBox" alignment="CENTER" fillWidth="false"
layoutX="10.0" layoutY="10.0" prefHeight="514.0" prefWidth="715.0" style="-
fx-background-color: #13477A;" AnchorPane.bottomAnchor="0.0"
AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">
            <children>
                <VBox fx:id="addNewVbox" alignment="CENTER"
onKeyPressed="#addNewKeyPress" prefHeight="403.0" prefWidth="382.0">
                    <children>
                        <Label alignment="CENTER" prefHeight="17.0"
prefWidth="332.0" text="ADD NEW SENSOR" textFill="WHITE">
                            <font>
                                <Font name="System Bold" size="37.0" />
                            </font>
                        </Label>
                        <Label alignment="CENTER" prefHeight="17.0"
prefWidth="722.0" text="SENSOR NAME" textFill="WHITE">
                            <font>
                                <Font name="System Bold" size="14.0" />
                            </font>
                            <VBox.margin>
                                <Insets top="10.0" />
                            </VBox.margin>
                        </Label>
                        <TextField fx:id="sName" prefHeight="79.0"
prefWidth="382.0" styleClass="textBox" stylesheets="@addNewSensor.css">
                            <VBox.margin>
                                <Insets top="10.0" />
                            </VBox.margin>
                            <font>
                                <Font size="14.0" />
                            </font>
                        </TextField>
                        <Label alignment="CENTER" prefHeight="17.0"
prefWidth="721.0" text="FLOOR NAME/NUMBER" textFill="WHITE">
                            <font>
```

```

        <Font name="System Bold" size="14.0" />
    </font>
    <VBox.margin>
        <Insets top="30.0" />
    </VBox.margin>
</Label>
<TextField fx:id="sFloor" prefHeight="80.0"
prefWidth="382.0" styleClass="textBox" stylesheets="@addNewSensor.css">
    <VBox.margin>
        <Insets top="10.0" />
    </VBox.margin>
    <font>
        <Font size="14.0" />
    </font>
</TextField>
<Label alignment="CENTER" prefHeight="17.0"
prefWidth="721.0" text="ROOM NAME/NUMBER" textFill="WHITE">
    <font>
        <Font name="System Bold" size="14.0" />
    </font>
    <VBox.margin>
        <Insets top="30.0" />
    </VBox.margin>
</Label>
<TextField fx:id="sRoom" prefHeight="77.0"
prefWidth="382.0" styleClass="textBox" stylesheets="@addNewSensor.css">
    <VBox.margin>
        <Insets top="10.0" />
    </VBox.margin>
    <font>
        <Font size="14.0" />
    </font>
</TextField>
<Button fx:id="addSensorBtn" alignment="CENTER"
mnemonicParsing="false" onMouseClicked="#addNewSensor" prefHeight="39.0"
prefWidth="129.0" stylesheets="@addNewSensor.css" text="Add"
textFill="WHITE">
    <VBox.margin>
        <Insets bottom="20.0" top="30.0" />
    </VBox.margin>
    <font>
        <Font name="System Bold" size="18.0" />
    </font>
</Button>
</children>
</VBox>
</children>
</VBox>
</children>
</AnchorPane>

```

Add New Sensor Controller. JAVA

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package sensorsystem;

import java.net.URL;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.VBox;

public class AddNewSensorController implements Initializable {

    @FXML
    private VBox addNewBox;
    @FXML
    private VBox addNewVbox;
    @FXML
    private TextField sName;
    @FXML
    private TextField sFloor;
    @FXML
    private TextField sRoom;
    @FXML
    private Button addSensorBtn;

    private SensorService sensorService = null;

    /**
     * Initializes the controller class.
     */
    @Override
    public void initialize(URL url, ResourceBundle rb) {
        // TODO
        /*
         * this method use to connect rmi server
         */
        System.setProperty("java.security.policy", "file:allowall.policy");
        try {
            Registry reg =LocateRegistry.getRegistry("127.0.0.1",2000);
```

```

        sensorService = (SensorService) reg.lookup("sensorServer");
    } catch (Exception e) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(e.toString());

        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }
}

@FXML
private void addNewSensor(MouseEvent event) {
    // check mouse is clicked
    addNew();
}

@FXML
private void addNewKeyPress(KeyEvent event) {
    // check enter key is pressed
    if (event.getCode() == KeyCode.ENTER) {
        addNew();
        System.out.println("addNewKeyPress(KeyEvent event)");
    }
}

private void addNew() {
    /*
    this method use tp add a new sensor
    */
    String name, floor, room;
    name = sName.getText().toString(); // get sensor name
    floor = sFloor.getText().toString(); // get sensor floor details
    room = sRoom.getText().toString(); // get sensor room details
    if (name.equals("") || floor.equals("") || room.equals("")) {
        // check whether the fields are empty or not
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Warning");
        alert.setHeaderText("All fields are mandatory!!!");
        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    } else {
        // if required fields are not empty then send all data to rmi
server
        sName.setText("");
        sFloor.setText("");
        sRoom.setText("");
        try {

            String newMess = sensorService.addSensor(name, floor, room, 0);
            // send data to rmi server and get response

```

```

        if (newMess.startsWith("Successfull")) {
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle("Successfull");
            alert.setHeaderText(name+" added successfully");
            alert.showAndWait().ifPresent(rs -> {
                if (rs == ButtonType.OK) {
                    // System.out.println("Pressed OK.");
                }
            });
        }
    }else{
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(newMess);

        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }

} catch (Exception e) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText(e.toString());

    alert.showAndWait().ifPresent(rs -> {
        if (rs == ButtonType.OK) {
            // System.out.println("Pressed OK.");
        }
    });
}

}

}

}

}

```



```

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package sensorsystem;

import java.net.URL;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.TextField;
import javafx.scene.control.ToggleGroup;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class EditSensorController implements Initializable {

    @FXML
    private VBox addNewBox;
    @FXML
    private VBox addNewVbox;
    @FXML
    private TextField sName;
    @FXML
    private TextField sFloor;
    @FXML
    private TextField sRoom;
    @FXML
    private Button addSensorBtn;

    private int editId;

    private SensorService sensorService = null;

    @FXML
    private RadioButton Active;
    @FXML

```

```

private RadioButton Inactive;
/**
 * Initializes the controller class.
 */

@FXML
ToggleGroup mainGroup;
@FXML
private ToggleGroup group1;

public void transferData(int id, String ob1, String ob2, String ob3,
String ob4) {
    /**
     * this method use to get sensor details from sensor details controller
     */
    this.editId = id;
    sName.setText(ob1);
    sFloor.setText(ob2);
    sRoom.setText(ob3);

    if(ob4.equals("Active"))
    {
        Active.setSelected(true);
    }
    else{
        Inactive.setSelected(true);
    }
}

@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
    /**
     * this method use to connect rmi server
     */
    System.setProperty("java.security.policy", "file:allowall.policy");
    try {
        Registry reg =LocateRegistry.getRegistry("127.0.0.1",2000);
        sensorService = (SensorService) reg.lookup("sensorServer");

        mainGroup = new ToggleGroup();
        Active.setToggleGroup(mainGroup);
        Inactive.setToggleGroup(mainGroup);

    } catch (Exception e) {

        System.err.println("error "+e);
    }

}

```

```

@FXML
private void editSensor(MouseEvent event) {
    // check mouse is clicked
    editSernsor();
}

@FXML
private void editSensorKeyPressed(KeyEvent event) {
    // check enter key is pressed
    if (event.getCode() == KeyCode.ENTER){
        editSernsor();
    }
}

private void editSernsor(){
    // get new values and update sensor details

    String name,floor,room, status;
    int id;
    id = editId;

    name=sName.getText().toString();
    floor=sFloor.getText().toString();
    room=sRoom.getText().toString();

    if(Active.isSelected())
        status = "Active";

    else
        status = "Inactive";

    //System.out.println(status);

    if (name.equals("") || floor.equals("") || room.equals("")) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Warning");
        alert.setHeaderText("All fields are mandatory!!!");
        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                System.out.println("Pressed OK.");
            }
        });
    }

    }else{
        try {
            // send new values to rmi server
            String newMess= sensorService.editSensor(id, name, floor,
room, status);
            if (newMess.startsWith("Edited")) {
                Alert alert = new Alert(Alert.AlertType.INFORMATION);
                alert.setTitle("Successfull");
            }
        }
    }
}

```

```

        alert.setHeaderText("Sensor: "+name+" Edited
successfully");
        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }else{
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(newMess);

        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }

    Stage stage=(Stage)addSensorBtn.getScene().getWindow();
    stage.close();

} catch (Exception e) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText(e.toString());

    alert.showAndWait().ifPresent(rs -> {
        if (rs == ButtonType.OK) {
            System.out.println("Pressed OK.");
        }
    });
}

}

}

}

```

```

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package sensorsystem;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.ArrayList;
import java.util.ResourceBundle;
import javafx.animation.Interpolator;
import javafx.animation.KeyFrame;
import javafx.animation.KeyValue;
import javafx.animation.Timeline;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.image.Image;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.util.Duration;

```

```

public class LoginController implements Initializable {

    /**
     * Initializes the controller class.
     */
    private SensorService sensorService = null;
    @FXML
    private TextField password;
    @FXML
    private Button loginBTN;

```

```

@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
    /*
    this method use to connect rmi server
    */
    System.setProperty("java.security.policy", "file:allowall.policy");
    try {
        Registry reg =LocateRegistry.getRegistry("127.0.0.1",2000);
        sensorService = (SensorService) reg.lookup("sensorServer");
    } catch (Exception e) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(e.toString());

        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }
}

private void login(){
    String pass;

    pass=password.getText().toString();// get password box value
(password)
    if (pass.equals("")) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText("Enter password");
        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }else{
        try{
            String logMsg = sensorService.login(pass);// send password
to rmi server

            //System.out.println(logMsg);
            if(logMsg.startsWith("Logged")){
                // if password is correct show main window
                FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("MainWindow.fxml"));
                Parent root = (Parent) fxmlLoader.load();
                Stage stage = new Stage();
                stage.setTitle("Sensor System");
                stage.getIcons().add(new
Image(this.getClass().getResourceAsStream("logoNew-removebg-preview.png")));
                stage.setScene(new Scene(root));
                stage.show();
                Stage stageClose=(Stage)loginBTN.getScene().getWindow();
                stageClose.close();
            }else if(logMsg.startsWith("Wrong")){

```

```

        // if password is wrong show error alert
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Unauthorized");
        alert.setHeaderText("Wrong password");
        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }else{
        // if rmi server return exception Show error message
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText("Oopzz!! Something went wrong!!");
        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }
}catch(Exception e){
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText(e.toString());
    alert.showAndWait().ifPresent(rs -> {
        if (rs == ButtonType.OK) {
            // System.out.println("Pressed OK.");
        }
    });
}

}

}

@FXML
private void loginMethod(MouseEvent event){
    login();
}

@FXML
private void keyPressedLogin(KeyEvent event) {
    if (event.getCode() == KeyCode.ENTER){
        login();
    }
}

}

```

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<AnchorPane id="AnchorPane" fx:id="mainAnchor_main" prefHeight="576.0"
prefWidth="967.0" xmlns="http://javafx.com/javafx/11.0.1"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sensorsystem.MainWindowController">
    <children>
        <BorderPane prefHeight="200.0" prefWidth="200.0"
AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
            <top>
                <HBox alignment="CENTER" opacity="0.79" prefHeight="62.0"
prefWidth="967.0" style="-fx-background-color: #ffffff;"
BorderPane.alignment="CENTER">
                    <children>
                        <ImageView fitHeight="69.0" fitWidth="66.0"
nodeOrientation="RIGHT_TO_LEFT" opacity="0.76" pickOnBounds="true"
preserveRatio="true">
                            <image>
                                <Image url="@logoNew-removebg-preview.png" />
                            </image>
                            <HBox.margin>
                                <Insets left="100.0" />
                            </HBox.margin>
                        </ImageView>
                        <Label alignment="CENTER" prefHeight="30.0"
prefWidth="71.0" text="SENSOR " textFill="#7e7e80">
                            <font>
                                <Font name="System Bold" size="17.0" />
                            </font>
                        </Label>
                        <Label alignment="CENTER" prefHeight="57.0"
prefWidth="228.0" text="APP FOR FIRE ALARMS" textAlignment="CENTER"
textFill="#050f42">
                            <font>
                                <Font name="System Bold Italic" size="20.0" />
                            </font>
                        </Label>
                    </children>
                </HBox>
            </top>
        </BorderPane>
    </children>
</AnchorPane>

```



```

        </top>
        <center>
            <VBox alignment="CENTER" prefHeight="576.0" prefWidth="917.0"
style="-fx-background-color: black;" BorderPane.alignment="CENTER">
                <children>
                    <HBox alignment="CENTER_LEFT" prefHeight="489.0"
prefWidth="967.0" style="-fx-background-color: white;" VBox.vgrow="ALWAYS">
                        <children>
                            <VBox fx:id="addNewBox" alignment="CENTER"
fillWidth="false" prefHeight="475.0" prefWidth="715.0" style="-fx-background-
color: #13477A;" HBox.hgrow="ALWAYS">
                                <children>
                                    <VBox fx:id="addNewVbox" alignment="CENTER"
onKeyPressed="#addNewKeyPress" prefHeight="470.0" prefWidth="378.0">
                                        <children>
                                            <Label alignment="CENTER"
prefHeight="17.0" prefWidth="332.0" text="ADD NEW SENSOR" textFill="WHITE">
                                                <font>
                                                    <Font name="System Bold"
size="37.0" />
                                                        </font>
                                                    </Label>
                                                    <Label alignment="CENTER"
prefHeight="17.0" prefWidth="722.0" text="SENSOR NAME" textFill="WHITE">
                                                        <font>
                                                            <Font name="System Bold"
size="14.0" />
                                                                </font>
                                                            <VBox.margin>
                                                                <Insets top="10.0" />
                                                            </VBox.margin>
                                                        </Label>
                                                        <TextField fx:id="sName"
prefHeight="54.0" prefWidth="378.0" promptText="Sensor Name"
styleClass="textBox" stylesheets="@mainWindow.css">
                                                            <VBox.margin>
                                                                <Insets top="10.0" />
                                                            </VBox.margin>
                                                            <font>
                                                                <Font size="14.0" />
                                                            </font>
                                                        </TextField>
                                                        <Label alignment="CENTER"
prefHeight="17.0" prefWidth="721.0" text="FLOOR NAME/NUMBER"
textFill="WHITE">
                                                            <font>
                                                                <Font name="System Bold"
size="14.0" />
                                                                    </font>
                                                                <VBox.margin>
                                                                    <Insets top="30.0" />
                                                                </VBox.margin>
                                                            </Label>
                                                            <TextField fx:id="sFloor"
prefHeight="56.0" prefWidth="378.0" promptText="Floor Name/Number"
styleClass="textBox" stylesheets="@mainWindow.css">
                                                                <VBox.margin>

```

```

        <Insets top="10.0" />
    </VBox.margin>
    <font>
        <Font size="14.0" />
    </font>
</TextField>
<Label alignment="CENTER"
prefHeight="17.0" prefWidth="721.0" text="ROOM NAME/NUMBER" textFill="WHITE">
    <font>
        <Font name="System Bold"
size="14.0" />
    </font>
    <VBox.margin>
        <Insets top="30.0" />
    </VBox.margin>
</Label>
<TextField fx:id="sRoom"
prefHeight="58.0" prefWidth="378.0" promptText="Room Name/Number"
styleClass="textBox" stylesheets="@mainWindow.css">
    <VBox.margin>
        <Insets top="10.0" />
    </VBox.margin>
    <font>
        <Font size="14.0" />
    </font>
</TextField>
<Button fx:id="addSensorBtn"
alignment="CENTER" mnemonicParsing="false" onMouseClicked="#addNewSensor"
prefHeight="39.0" prefWidth="129.0" stylesheets="@mainWindow.css" text="ADD"
textFill="WHITE">
    <VBox.margin>
        <Insets bottom="10.0" top="30.0" />
    </VBox.margin>
    <font>
        <Font name="System Bold"
size="18.0" />
    </font>
</Button>
</children>
</VBox>
</children>
</VBox>
</children>
</HBox>
</children>
</VBox>
</center>
<left>
    <VBox alignment="TOP_CENTER" prefHeight="520.0" prefWidth="251.0"
style="-fx-background-color: #dddd; BorderPane.alignment="CENTER">
        <children>
            <Label fx:id="addbtn" alignment="CENTER"
onMouseClicked="#showAddNew" prefHeight="63.0" prefWidth="228.0"
styleClass="active" stylesheets="@mainWindow.css" text="ADD NEW"
textFill="WHITE">
                <font>
                    <Font name="System Bold" size="24.0" />

```

```

        </font>
        <VBox.margin>
            <Insets top="80.0" />
        </VBox.margin>
    </Label>
    <Label fx:id="sensorbtn" alignment="CENTER"
onMouseClicked="#showSensors" prefHeight="60.0" prefWidth="234.0"
styleClass="box" stylesheets="@mainWindow.css" text="SENSORS"
textFill="#141414">
        <font>
            <Font name="System Bold" size="24.0" />
        </font>
        <VBox.margin>
            <Insets top="20.0" />
        </VBox.margin>
    </Label>
</children>
</VBox>
</left>
<bottom>
    <Label alignment="CENTER" contentDisplay="CENTER"
prefHeight="57.0" prefWidth="967.0" text="BY CODE4" textAlignment="CENTER"
textFill="#050f42" BorderPane.alignment="CENTER">
        <font>
            <Font name="System Bold Italic" size="18.0" />
        </font>
    </Label>
</bottom>
</BorderPane>
</children>
</AnchorPane>

```

```

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package sensorsystem;

```

```

import java.io.IOException;
import java.net.URL;
import java.rmi.Naming;
import java.rmi.Remote;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.ResourceBundle;
import javafx.animation.Interpolator;
import javafx.animation.KeyFrame;
import javafx.animation.KeyValue;

```

```

import javafx.animation.Timeline;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;
import static javafx.scene.layout.Region.USE_COMPUTED_SIZE;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
import javafx.util.Duration;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Locale;
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonType;
import javafx.scene.input.KeyCode;
import javafx.scene.input.KeyEvent;
import javafx.stage.Stage;

public class MainWindowController implements Initializable {

    private SensorService sensorService = null;
    private Boolean clicikShowAdd=false;

    @FXML
    private Label addbtn;
    @FXML
    private Label sensorbtn;
    @FXML
    private VBox addNewBox;
    @FXML
    private VBox addNewVbox;

    @FXML
    private Button addSensorBtn;
    @FXML
    private TextField sName;

    @FXML
    private TextField sRoom;
    @FXML
    private TextField sFloor;

    @FXML
    private AnchorPane mainAnchor_main;

```

```

/**
 * Initializes the controller class.
 */
@Override
public void initialize(URL url, ResourceBundle rb) {

    /*
    this method use to connect rmi server
    */
    System.setProperty("java.security.policy", "file:allowall.policy");
    try {
        Registry reg =LocateRegistry.getRegistry("127.0.0.1",2000);
        sensorService = (SensorService) reg.lookup("sensorServer");

    } catch (Exception e) {

        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(e.toString());

        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }

}

@FXML
private void showAddNew(MouseEvent event) throws IOException {
    /*
    This method use to show add new sensor window.
    when click on add new button this method will be triggered
    */
    if (clcikShowAdd) {
        Parent root
=FXMLLoader.load(getClass().getResource("AddNewSensor.fxml"));
        Scene scene = addbtn.getScene();
        root.translateXProperty().set(scene.getWidth());
        addNewBox.getChildren().add(root);

        Timeline timeline = new Timeline();
        KeyValue keyValue = new KeyValue(root.translateXProperty(), 0,
Interpolator.EASE_IN);
        KeyFrame keyFrame = new KeyFrame(Duration.seconds(1), keyValue);
        timeline.getKeyFrames().add(keyFrame);

        timeline.play();
        addNewBox.getChildren().remove(0);

        sensorbtn.getStyleClass().remove("active");
        sensorbtn.getStyleClass().add("box");
        addbtn.getStyleClass().remove("box");
        addbtn.getStyleClass().add("active");
        clcikShowAdd=false;
    }
}

```

```

    }

}

@FXML
private void showSensors(MouseEvent event) throws IOException {
    /*
     * This method use to show all sensor details table.
     * when click on sensor details button this method will be triggered
     */
    if (!clcikShowAdd) {
        Parent root
=FXMLLoader.load(getClass().getResource("SensorsDetails.fxml"));
        Scene scene = addbtn.getScene();
        root.translateYProperty().set(scene.getHeight());
        addNewBox.getChildren().add(root);

        Timeline timeline = new Timeline();
        KeyValue keyValue = new KeyValue(root.translateYProperty(), 0,
Interpolator.EASE_IN);
        KeyFrame keyFrame = new KeyFrame(Duration.seconds(1), keyValue);
        timeline.getKeyFrames().add(keyFrame);

        timeline.play();
        addNewBox.getChildren().remove(0);
        // addbtn.setTextFill(Color.GRAY);
        addbtn.getStyleClass().remove("active");
        addbtn.getStyleClass().add("box");
        sensorbtn.getStyleClass().remove("box");
        sensorbtn.getStyleClass().add("active");
        clcikShowAdd=true;
    }

}

@FXML
private void addNewSensor(MouseEvent event) {
    // check mouse is clicked
    addNew();
}

@FXML
private void addNewKeyPress(KeyEvent event) {
    // check enter key is pressed
    if (event.getCode() == KeyCode.ENTER) {
        addNew();
    }
}

private void addNew() {
    /*
     * this method use to add new sensor to system
     */
    String name, floor, room;
    name=sName.getText().toString();
    floor=sFloor.getText().toString();
}

```

```

room=sRoom.getText().toString();

if (name.equals("") || floor.equals("") || room.equals("")) {
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setTitle("Warning");
    alert.setHeaderText("All fields are mandatory!!!");
    alert.showAndWait().ifPresent(rs -> {
        if (rs == ButtonType.OK) {
            //System.out.println("Pressed OK.");
        }
    });
}else{
    sName.setText("");
    sFloor.setText("");
    sRoom.setText("");
    try {

        String newMess= sensorService.addSensor(name ,
floor,room,0);

        if (newMess.startsWith("Successfull")) {
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle("Successfull");
            alert.setHeaderText(name+" added successfully");
            alert.showAndWait().ifPresent(rs -> {
                if (rs == ButtonType.OK) {
                    // System.out.println("Pressed OK.");
                }
            });
        }else{
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Error");
            alert.setHeaderText(newMess);

            alert.showAndWait().ifPresent(rs -> {
                if (rs == ButtonType.OK) {
                    System.out.println("Pressed OK.");
                }
            });
        }

    } catch (Exception e) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(e.toString());

        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }

}

}

}

```

```

}

package sensorsystem;

public class Sensor {

    private int id;
    private String name,floor,room,colevel,status,smokelevel;
    private double level,smoke;

    public Sensor(String name, String floor, String room, double level) {
        this.name = name;
        this.floor = floor;
        this.room = room;
        this.level = level;
        this.colevel= new Double(level).toString();
    }
    public Sensor(String name, String floor, String room, String colevel) {
        this.name = name;
        this.floor = floor;
        this.room = room;
        this.level = level;
        this.level = Double.parseDouble(colevel);
    }

    public Sensor(int id, String name, String floor, String room, double
level, String status,double smoke) {
        this.id = id;
        this.name = name;
        this.floor = floor;
        this.room = room;
        this.level = level;
        this.smoke=smoke;
        this.colevel= new Double(level).toString();
        this.smokelevel=new Double(smoke).toString();
        this.status = status;
    }
    public Sensor(int id, String name, String floor, String room, String
colevel, String status,String smokelevel) {
        this.id = id;
        this.colevel = colevel;
        this.smokelevel=smokelevel;
        this.name = name;
        this.floor = floor;
        this.room = room;
        this.level = Double.parseDouble(colevel);
        this.smoke=Double.parseDouble(smokelevel);
        this.status = status;
    }

    public String getName() {
        return name;
    }
}

```



```

    public String getFloor() {
        return floor;
    }

    public String getRoom() {
        return room;
    }

    public String getColevel() {
        return colevel;
    }

    public String getSmokeLevel(){
        return smokelevel;
    }

    public double getLevel() {
        return level;
    }

    public double getSmoke(){
        return smoke;
    }

    public int getID()
    {
        return id;
    }

    public String getStatus()
    {
        return status;
    }

}

/*
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
*/
/*
    Created on : Apr 29, 2020, 12:37:22 AM
    Author      : Dilshan
*/
.table{

}
.tdth{
    -fx-border-color: #dddddd;
    -fx-padding: 10px;

}
.tdth:nth-child(even){
    -fx-background-color: #dddddd;

```

```

}
package sensorsystem;

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

public interface SensorService extends Remote{
    public String addSensor(String name, String floor, String room, double
level) throws RemoteException;
    public String getSensors() throws Exception;
    public String editSensor(int id, String name, String floor, String room,
String status) throws Exception;
    public String login(String password) throws Exception;

//    public String updateSensorLevel(int id,double level) throws Exception;
}

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package sensorsystem;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.stage.Stage;

public class SensorSystem_RMI extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));

```

```

        Scene scene = new Scene(root);
        stage.setTitle("Sensor System");
        stage.getIcons().add(new
Image(this.getClass().getResourceAsStream("logoNew-removebg-preview.png")));
        stage.setScene(scene);
        stage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        launch(args);

    }
}

```

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.VBox?>

<AnchorPane id="AnchorPane" xmlns="http://javafx.com/javafx/11.0.1"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sensorsystem.SensorsDetailsController">
    <children>
        <VBox alignment="CENTER" AnchorPane.bottomAnchor="0.0"
AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">
            <children>
                <TableView fx:id="sensorTable" prefHeight="569.0"
prefWidth="685.0" styleClass="table" stylesheets="@SensorDetails.css"
VBox.vgrow="ALWAYS">
                    <columns>

                        <TableColumn fx:id="name" prefWidth="115.0" styleClass="tdth"
text="Sensor Name" />
                        <TableColumn fx:id="floor" minWidth="0.0" prefWidth="123.0"
styleClass="tdth" text="Floor Name/ID" />
                        <TableColumn fx:id="room" minWidth="0.0" prefWidth="112.0"
styleClass="tdth" text="Room Name/ID" />
                        <TableColumn fx:id="colavel" minWidth="0.0"
prefWidth="104.0" styleClass="tdth" text="Co2 Level" />
                        <TableColumn fx:id="smokelevel" minWidth="0.0"
prefWidth="110.0" styleClass="tdth" text="Smoke Level" />
                        <TableColumn fx:id="status" minWidth="0.0"
prefWidth="107.0" styleClass="tdth" text="Status" />

```

```

        </columns>
        <columnResizePolicy>

            </columnResizePolicy>
        </TableView>
    </children>
</VBox>
</children>
</AnchorPane>

```

```

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

```

```

package sensorsystem;

```

```

import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import javafx.event.EventHandler;
import java.net.URL;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.Random;
import java.util.ResourceBundle;
import java.util.Timer;
import java.util.TimerTask;
import java.util.concurrent.TimeUnit;
import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.geometry.Insets;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.ButtonBar.ButtonData;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Dialog;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TablePosition;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;

```

```

import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.input.MouseButton;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
import javafx.util.Pair;
import javax.management.Notification;
import org.controlsfx.control.Notifications;
import org.controlsfx.control.PropertySheet.Item;
import org.json.JSONArray;
import org.json.JSONObject;

public class SensorsDetailsController implements Initializable{

    private int editID;
    private SensorService sensorService = null;
    @FXML
    private TableView<Sensor> sensorTable;
    @FXML
    private TableColumn<?, ?> name;
    @FXML
    private TableColumn<?, ?> floor;
    @FXML
    private TableColumn<?, ?> room;
    @FXML
    private TableColumn<?, ?> colavel;
    @FXML
    private TableColumn<?, ?> smokelevel;
    @FXML
    private TableColumn<?, ?> status;

    /**
     * Initializes the controller class.
     */
    ObservableList<Sensor> observableList =
FXCollections.observableArrayList();
    ArrayList<Integer> sensorId;

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        sensorId= new ArrayList<>();
        String mess="";
        /*
        this method use to connect rmi server
        */
        System.setProperty("java.security.policy", "file:allowall.policy");
        try {
            Registry reg =LocateRegistry.getRegistry("127.0.0.1",2000);
            sensorService = (SensorService) reg.lookup("sensorServer");

```

```

        // set Table View Columns to display details
        name.setCellValueFactory(new PropertyValueFactory<>("name"));
        floor.setCellValueFactory(new PropertyValueFactory<>("floor"));
        room.setCellValueFactory(new PropertyValueFactory<>("room"));
        colavel.setCellValueFactory(new
PropertyValueFactory<>("colavel"));

        status.setCellValueFactory(new PropertyValueFactory<>("status"));
        smokelevel.setCellValueFactory(new
PropertyValueFactory<>("smoke"));

        sensorTable.setItems(observableList);

        // handle sensor update
        // when user double click on the row edit details window will be
shown
        sensorTable.setOnMouseClicked(new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                if
(event.getButton() == MouseButton.PRIMARY && event.getClickCount() == 2) {
                    try{
                        System.out.println("Mouse clicked");

                        Sensor pos =
sensorTable.getSelectionModel().getSelectedItem();

                        editID = pos.getID();
                        String tempName = pos.getName();
                        String tempFloor = pos.getFloor();
                        String tempRoom = pos.getRoom();
                        String status = pos.getStatus();

                        // show edit details window
                        FXMLLoader fxmllLoader = new
FXMLLoader(getClass().getResource("editSensor.fxml"));
                        Parent root = (Parent) fxmllLoader.load();

                        EditSensorController controller =
fxmllLoader.getController();
                        // send data to EditSensorController class
                        controller.transferData(editID, tempName,
tempFloor, tempRoom, status);

                        Stage stage = new Stage();
                        stage.setTitle("Sensor System");
                        stage.getIcons().add(new
Image(this.getClass().getResourceAsStream("logoNew-removebg-preview.png")));
                        stage.setScene(new Scene(root));
                        stage.show();

                    }catch(Exception e){
                        Alert alert = new Alert(Alert.AlertType.ERROR);
                        alert.setTitle("Error");
                        alert.setHeaderText(e.toString());
                    }
                }
            }
        });

```

```

        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }
}

});

/*
The sensor details should be updated every 15 seconds
*/
Timer timer = new Timer();
timer.scheduleAtFixedRate(new TimerTask() {
    @Override
    public void run() {
        Platform.runLater()->{

            updatetable();
        }
    }
}, 0, 15000);

} catch (Exception e) {
    Alert alert = new Alert(Alert.AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText(e.toString());
    alert.showAndWait().ifPresent(rs -> {
        if (rs == ButtonType.OK) {
            // System.out.println("Pressed OK.");
        }
    });
}

}

private void showNotification(String mess){
    Toolkit.getDefaultToolkit().beep();// Notification sound
    Notifications.create()
        .title("CO2 Lovel is High ")
        .text(mess)
        .showWarning();
}

public void updatetable(){
    /*
    This methode use to display sensor details
    */
    String mess="";
    try {
        String data = sensorService.getSernsors();
        JSONArray jsonArray = new JSONArray(data);
    }
}

```

```

        System.out.println(jSONArray.length());
        sensorTable.getItems().clear();
        for (int i = 0; i < jSONArray.length(); i++) {
            JSONObject jsonObject = jSONArray.getJSONObject(i);
            // System.out.println(jsonObject);
            Sensor sensor = new
Sensor(Integer.parseInt(jsonObject.get("id").toString().trim()), jsonObject.get(
t("name").toString().trim(), jsonObject.get("floor").toString().trim(),
jSONObject.get("room").toString().trim(), Double.parseDouble(jsonObject.get("c
olevel").toString()),
jSONObject.get("status").toString().trim(), Double.parseDouble(jsonObject.get(
"smokelevel").toString()) );
            sensorTable.getItems().add(sensor);

            if (sensor.getLevel() >= 5) {
                mess += "Floor: " + sensor.getFloor() + " Room:
"+sensor.getRoom() + "\n";
            }
        }
        if (!mess.equals("")) {
            showNotification(mess);
        }
    } catch (Exception e) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error");
        alert.setHeaderText(e.toString());
        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                // System.out.println("Pressed OK.");
            }
        });
    }
}

}

.active{
    -fx-background-color: #0285E8;
    -fx-border-color: #5869DB;
    -fx-effect: dropshadow(three-pass-box, rgba(0,0,0,0.8), 10, 0, 5, 5);
}
.box{
    -fx-border-color: white;
    -fx-text-fill: black;
    -fx-border-width: 0.5 0.5 1 0.5;
    -fx-border-color: gray;
    -fx-effect: dropshadow(three-pass-box, rgba(0,0,0,0.8), 10, 0, 5, 5);
    -fx-background-color: #dddddd;
}

```



```

}
.box:hover{

    -fx-border-width: 1 1 2 1;
    -fx-border-color: black;

}
.button{
    -fx-background-color: #0285E8;
    -fx-effect: dropshadow(three-pass-box, rgba(0,0,0,0.8), 8, 0, 5, 5);
}
.button:hover{
    -fx-opacity: 0.9;
    -fx-text-fill: white;
}
.textBox{
    -fx-border-width: 0 0 3 0;
    -fx-border-color: white;
    -fx-border-radius: 5;
    -fx-background-color: transparent;
    -fx-text-fill: white;
}

grant{
    permission java.security.AllPermission;
};

.button{
    -fx-background-color: #0285E8;
    -fx-effect: dropshadow(three-pass-box, rgba(0,0,0,0.8), 8, 0, 5, 5);
}
.button:hover{
    -fx-opacity: 0.9;
    -fx-text-fill: white;
}
.textBox{
    -fx-border-width: 0 0 3 0;
    -fx-border-color: white;
    -fx-border-radius: 5;
    -fx-background-color: transparent;
    -fx-text-fill: white;
}

```

```

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.RadioButton?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.ToggleGroup?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

<AnchorPane id="AnchorPane" prefHeight="415.0" prefWidth="600.0" style="-fx-
background-color: #13477A;" xmlns="http://javafx.com/javafx/11.0.1"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sensorsystem.EditSensorController">
    <VBox fx:id="addNewBox" alignment="CENTER" fillWidth="false"
onKeyPressed="#editSensorKeyPressed" prefHeight="499.0" prefWidth="815.0"
style="-fx-background-color: #13477A;" AnchorPane.bottomAnchor="0.0"
AnchorPane.leftAnchor="0.0" AnchorPane.rightAnchor="0.0"
AnchorPane.topAnchor="0.0">
        <children>
            <VBox fx:id="addNewVbox" alignment="CENTER" prefHeight="481.0"
prefWidth="314.0">
                <children>
                    <Label alignment="CENTER" prefHeight="17.0" prefWidth="332.0"
text="EDIT SENSOR" textFill="WHITE">
                        <font>
                            <Font name="System Bold" size="35.0" />
                        </font>
                    </Label>
                    <Label alignment="CENTER" prefHeight="17.0" prefWidth="722.0"
text="SENSOR NAME" textFill="WHITE">
                        <font>
                            <Font name="System Bold" size="14.0" />
                        </font>
                        <VBox.margin>
                            <Insets top="10.0" />
                        </VBox.margin>
                    </Label>
                    <TextField fx:id="sName" focusTraversable="false"
prefHeight="46.0" prefWidth="314.0" styleClass="textBox"
stylesheets="@editDetails.css">
                        <VBox.margin>
                            <Insets top="10.0" />
                        </VBox.margin>
                        <font>
                            <Font size="14.0" />
                        </font>
                    </TextField>
                    <Label alignment="CENTER" prefHeight="17.0" prefWidth="721.0"
text="FLOOR NAME/NUMBER" textFill="WHITE">
                        <font>
                            <Font name="System Bold" size="14.0" />

```

```

        </font>
        <VBox.margin>
            <Insets top="30.0" />
        </VBox.margin>
    </Label>
    <TextField fx:id="sFloor" prefHeight="45.0" prefWidth="314.0"
styleClass="textBox" stylesheets="@editDetails.css">
        <VBox.margin>
            <Insets top="10.0" />
        </VBox.margin>
        <font>
            <Font size="14.0" />
        </font>
    </TextField>
    <Label alignment="CENTER" prefHeight="17.0" prefWidth="721.0"
text="ROOM NAME/NUMBER" textFill="WHITE">
        <font>
            <Font name="System Bold" size="14.0" />
        </font>
        <VBox.margin>
            <Insets top="30.0" />
        </VBox.margin>
    </Label>
    <TextField fx:id="sRoom" prefHeight="43.0" prefWidth="314.0"
styleClass="textBox" stylesheets="@editDetails.css">
        <VBox.margin>
            <Insets top="10.0" />
        </VBox.margin>
        <font>
            <Font size="14.0" />
        </font>
    </TextField>
    <HBox alignment="CENTER" prefHeight="100.0" prefWidth="200.0">
        <children>

            <RadioButton fx:id="Active" selected="true" style="-fx-
background-color: #13477A;" text="Active" textFill="#07ff07">
                <toggleGroup>
                    <ToggleGroup fx:id="mainGroup" />
                </toggleGroup>
                <font>
                    <Font name="System Bold Italic" size="14.0" />
                </font>
            </RadioButton>
            <RadioButton fx:id="Inactive" style="-fx-background-
color: #13477A;" text="Inactive" textFill="RED">
                <toggleGroup>
                    <ToggleGroup fx:id="group1" />
                </toggleGroup>
                <HBox.margin>
                    <Insets left="20.0" />
                </HBox.margin>
                <font>
                    <Font name="System Bold" size="14.0" />
                </font>
            </RadioButton>
        </children>

```

```

        <VBox.margin>
            <Insets top="20.0" />
        </VBox.margin>
    </HBox>

    <Button fx:id="addSensorBtn" alignment="CENTER"
mnemonicParsing="false" onMouseClicked="#editSensor" prefHeight="15.0"
prefWidth="70.0" stylesheets="@editDetails.css" text="Save" textFill="WHITE">
        <VBox.margin>
            <Insets bottom="20.0" top="30.0" />
        </VBox.margin>
        <font>
            <Font name="System Bold" size="15.0" />
        </font>
    </Button>
</children>
</VBox>
</children>
</VBox>
</AnchorPane>

```

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>

```

```

<AnchorPane id="login_main" onKeyPressed="#keyPressedLogin"
prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #00B9EB;"
xmlns="http://javafx.com/javafx/11.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="sensorsystem.LoginController">
    <children>
        <BorderPane prefHeight="200.0" prefWidth="200.0"
AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
            <top>
                <HBox alignment="CENTER" prefHeight="92.0" prefWidth="600.0"
style="-fx-background-color: white;" BorderPane.alignment="CENTER">
                    <children>
                        <ImageView fitHeight="63.0" fitWidth="66.0" opacity="0.76"
pickOnBounds="true" preserveRatio="true">
                            <image>
                                <Image url="@logoNew-removebg-preview.png" />
                            </image>
                        </ImageView>

```

```

        <Label alignment="CENTER" text="SENSOR "
textFill="#7e7e80">
            <font>
                <Font name="System Bold" size="20.0" />
            </font>
        </Label>
        <Label alignment="CENTER" contentDisplay="CENTER" text="BY
CODE4" textAlignment="CENTER" textFill="#050f42">
            <font>
                <Font name="System Bold Italic" size="20.0" />
            </font>
        </Label>

    </children>
</HBox>
</top>
<center>
    <VBox alignment="CENTER" fillWidth="false" prefHeight="292.0"
prefWidth="600.0" BorderPane.alignment="CENTER">
        <children>
            <Label alignment="CENTER" contentDisplay="CENTER"
prefHeight="49.0" prefWidth="600.0" text="ENTER PASSWORD"
textAlignment="CENTER" textFill="WHITE">
                <font>
                    <Font name="System Bold" size="20.0" />
                </font>
                <VBox.margin>
                    <Insets />
                </VBox.margin>
            </Label>
            <PasswordField fx:id="password" alignment="CENTER"
prefHeight="48.0" prefWidth="389.0" styleClass="passBox"
stylesheets="@login.css">
                <font>
                    <Font size="14.0" />
                </font>
            </PasswordField>
            <Button fx:id="loginBTN" alignment="CENTER"
contentDisplay="CENTER" mnemonicParsing="false" onMouseClicked="#loginMethod"
prefHeight="39.0" prefWidth="129.0" stylesheets="@login.css" text="LOGIN"
textFill="#fffdfd">
                <font>
                    <Font name="System Bold" size="18.0" />
                </font>
                <VBox.margin>
                    <Insets top="20.0" />
                </VBox.margin>
            </Button>
        </children>
    </VBox>
</center>
</BorderPane>
</children>
</AnchorPane>

```

```

.active{
  -fx-background-color: #0285E8;
  -fx-border-color: #5869DB;
  -fx-effect: dropshadow(three-pass-box, rgba(0,0,0,0.8), 10, 0, 5, 5);
  -fx-text-fill: white;
}
.box{
  -fx-border-color: white;
  -fx-text-fill: black;
  -fx-border-width: 0.5 0.5 1 0.5;
  -fx-border-color: gray;
  -fx-effect: dropshadow(three-pass-box, rgba(0,0,0,0.8), 10, 0, 5, 5);
  -fx-background-color: #dddddd;
}
.box:hover{

  -fx-border-width: 1 1 2 1;
  -fx-border-color: black;

}
.button{
  -fx-background-color: #0285E8;
  -fx-effect: dropshadow(three-pass-box, rgba(0,0,0,0.8), 8, 0, 5, 5);
  -fx-border-radius: 10;
}
.button:hover{
  -fx-opacity: 0.9;
  -fx-text-fill: white;
}
.textBox{
  -fx-border-width: 0 0 3 0;
  -fx-border-color: white;
  -fx-border-radius: 5;
  -fx-background-color: transparent;
  -fx-text-fill: white;
}

```

6.3 RMI- Server

```
package sensorsystem;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import jdk.nashorn.internal.parser.JSONParser;
import org.json.JSONArray;
import org.json.JSONObject;

import java.util.Timer;
import java.util.TimerTask;
import java.util.logging.Level;
import java.util.logging.Logger;

import sun.net.www.http.HttpClient;

public class JavaRMIServer extends UnicastRemoteObject implements
SensorService{

    public JavaRMIServer() throws RemoteException{
        super();
    }
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        System.setProperty("java.security.policy", "file:allowall.policy");
        try {
            final JavaRMIServer jrmi=new JavaRMIServer();
            Registry reg =LocateRegistry.createRegistry(2000); // create rmi
registry
            reg.rebind("sensorServer", new JavaRMIServer()); // bind
JavaRMIServer class

```

```

        System.out.println ("Service started....");

        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                try {
                    jirmi.SendMail();
                } catch (Exception ex) {

            }

        }, 0, 15000);
    } catch (Exception e) {
        System.err.println ("err"+e);
    }

}

@Override
public String addSensor(String name, String floor, String room, double d)
throws RemoteException {
    /*
     * This Method uses to add new sensor. After reciveing sensor details,
     * details pass as a json object
     * via api.
     * after adding new sensor details, if it is successfull then return
     * message called "Successfull"
     * otherwise return error
     */

    try {
        // creat a json object
        JSONObject jsonObject = new JSONObject();
        jsonObject.put("name",name );
        jsonObject.put("floorid",floor );
        jsonObject.put("room",room );
        jsonObject.put("colevel",d );

        byte[] postData
=jJSONObject.toString().getBytes(StandardCharsets.UTF_8);
        int length=postData.length;
        URL url = new URL("http://localhost:3000/addSensorData");
        HttpURLConnection conn;
        conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("POST");
        conn.setRequestProperty("userid", "123464");
        conn.setRequestProperty("Content-Type", "application/json");
    }
}

```



```

        conn.setDoOutput(true);
        DataOutputStream dataOutputStream = new
DataOutputStream(conn.getOutputStream());
        dataOutputStream.write(postData);
        dataOutputStream.flush();
        dataOutputStream.close();
        BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));

        String output = br.readLine();

        return "Successfull add";

    } catch (Exception e) {
        System.err.println("err "+e);
        return "error "+e;
    }

}

@Override
public String getSensors() throws Exception {

    /*
    This method use to get sensor details via api.
    after receveing sensor details return that all data as a string
    */
    try {

        URL url = new URL("http://localhost:3000/getSensorData");
        HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Accept", "application/json");
        BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));

        String output = br.readLine();

        conn.disconnect();
        return output;
    } catch (Exception e) {
        System.err.println("err "+e);
        return "err "+e;
    }

}

@Override
public String login(String password) throws Exception {
    /*

```

```

        This method use to check login credentials. when you try to login
        desktop application this method
        method will be triggered.
        */
        JSONArray jsonArray;
        JSONObject jsonObject;
        try {
            URL url = new URL("http://localhost:3000/login");
            HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
            conn.setRequestMethod("GET");
            conn.setRequestProperty("Accept", "application/json");
            BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
            String output = br.readLine();
            jsonArray = new JSONArray(output);
            jsonObject = jsonArray.getJSONObject(0);
            int gotPass =
Integer.parseInt(jsonObject.get("password").toString().trim());
            String passToString = String.valueOf(gotPass);

            if(passToString.equals(password)){
                return "Logged in";
            }
            else{
                return "Wrong pass";
            }

        } catch (Exception e) {
            System.err.println("err "+e);
            return "err "+e;
        }

    }

    @Override
    public String editSensor(int id, String name, String floor, String room,
String status) throws Exception {
        /*
        this method use to update existing sensor details.
        */

        try {
            JSONObject jsonObject = new JSONObject();

            jsonObject.put("name",name);
            jsonObject.put("room",room);
            jsonObject.put("floor",floor);
            jsonObject.put("status",status);
            jsonObject.put("id",id );

            byte[] postData
=jJSONObject.toString().getBytes(StandardCharsets.UTF_8);
            int length=postData.length;
            URL url = new URL("http://localhost:3000/editSensor");
            HttpURLConnection conn;

```

```

        conn = (URLConnection) url.openConnection();
        conn.setRequestMethod("POST");
        conn.setRequestProperty("userid", "123464");
        conn.setRequestProperty("Content-Type", "application/json");
        conn.setDoOutput(true);
        DataOutputStream dataOutputStream = new
DataOutputStream(conn.getOutputStream());
        dataOutputStream.write(postData);
        dataOutputStream.flush();
        dataOutputStream.close();
        BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));

        String output = br.readLine();
        System.out.println(output);
        return "Edited Successfull";

    } catch (Exception e) {
        System.err.println("err "+e);
        return "error "+e;
    }

}

@Override
public void SendMail() throws Exception {
    /*
    this method use to send send mails. this methode check co2 level and
    smoke level.
    if co2 level or somoke level is grater than 5 then it will send
    warning to user via email.
    this methode calls every 15 seconds
    */

    int counter=0;
    String room=null;
    ArrayList<Sensor> sensors = new ArrayList<>();
    ArrayList<Sensor> email=new ArrayList<Sensor>();
    ArrayList<String> rooms = new ArrayList<>();
    sensors.add( new Sensor("name", "floor", "room", 10));

    try {

        URL url = new URL("http://localhost:3000/getSensorData");
        HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Accept", "application/json");
        BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        System.out.println("abc"+sensors.get(0).getName());

        //System.out.println(Double.parseDouble(jsonObject.get("colevel").toString())
);

```

```

        System.out.println("Output from Server .... \n");
        String output = br.readLine();
        JSONArray jSONArray = new JSONArray(output);

        for (int i = 0; i < jSONArray.length(); i++) {
            JSONObject jSONObject = jSONArray.getJSONObject(i);
            double
            val=Double.parseDouble(jSONObject.get("colevel").toString());
            double
            val2=Double.parseDouble(jSONObject.get("smokelevel").toString());
            room=jSONObject.get("room").toString();
            if(val>5||val2>5)
            {
                rooms.add(room);
                counter=counter+1;
                System.out.println("sesnor details"+jSONObject);
            }
        }
        if(counter>0){

            JSONObject jSONObject = new JSONObject();
            jSONObject.put("room",rooms );

            byte[] postData
            =jSONObject.toString().getBytes(StandardCharsets.UTF_8);
            int length=postData.length;
            URL url1 = new URL("http://localhost:3000/sendmail");
            HttpURLConnection conn1;
            conn1 = (HttpURLConnection) url1.openConnection();
            conn1.setRequestMethod("POST");
            conn1.setRequestProperty("userid", "123464");
            conn1.setRequestProperty("Content-Type",
            "application/json");
            conn1.setDoOutput(true);
            DataOutputStream dataOutputStream1 = new
            DataOutputStream(conn1.getOutputStream());
            dataOutputStream1.write(postData);
            dataOutputStream1.flush();
            dataOutputStream1.close();
            BufferedReader br1 = new BufferedReader(new
            InputStreamReader(conn1.getInputStream()));

        }

        //System.out.println(jSONArray.getJSONObject(0));
        conn.disconnect();
        //System.out.println("abc"+sensors.get(0).getName());

    } catch (Exception e) {
        System.err.println("err "+e);
    }

}

```

```

}
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

public interface SensorService extends Remote{
    public String addSensor(String name, String floor, String room, double
level) throws RemoteException;
    public String getSensors() throws Exception;
    public String login(String password) throws Exception;
    public String editSensor(int id, String name, String room, String floor,
String status) throws Exception;
    // public String updateSensorLevel(int id, double level) throws
Exception;
    public void SendMail() throws Exception;
}

```

6.4 Web-API

```
const express = require("express");
const app = express();
const core = require("cors");
const bodyParser = require("body-parser");

const nodemailer = require("nodemailer");
app.use(core());
app.use(bodyParser());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static(__dirname));
var mysql = require("mysql");

try {
  /*
  create database connection
  */
  var con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "",
    database: "ds_sensor_system",
  });

  con.connect(function (err) {
    if (err) throw err;
    console.log("Connected!");
  });
} catch (e) {
  console.log("errot" + e);
}

try {
  /*
  configure mail functions
  */

  /*
  } catch (e) {
    console.log("error" + e);
  }

  /*set web web client
  url http://localhost:3000
```

```

*/

app.get("/", function (req, res) {

    res.send("index.html");
});


/*
create api for update sensor details
*/

app.post("/updateSensorLevel", async function (req, res) {
    try {
        var query =
            "UPDATE `sensors` SET `colevel`=" +
            req.body.lavel +
            ", smokelevel=" +
            req.body.smokeLevel +
            " WHERE id=" +
            req.body.id +
            " LIMIT 1 ";
        con.query(
            query,
            await function (err, result) {
                if (err) {
                    res.status(500).send("error");
                } else {
                    res.send("Sensor added");
                }
                // console.log("1 record inserted");
            }
        );
    } catch (e) {
        console.log("errot" + e);
    }
});


/*
create a api for get all sensor details
*/

app.get("/getSensorData", async function (req, res) {
    //console.log("get All data");

    try {
        await con.query("SELECT * FROM `sensors` order by status ASC ", function
(err, result, fields) {
            if (err) {
                res.status(500).send("error");
            } else {
                res.send(result);
            }
        });
    } catch (e) {

```

```

        console.log(e);
        res.status(200).send("error try ", e);
    }
});

/*
create api for add new sensor
*/

app.post("/addSensorData", async function (req, res) {
    try {
        var query =
            "INSERT INTO `sensors`(`id`, `name`, `room`, `floor`, `colevel`,
`smokelevel`, `status`) " +
            "VALUES (null, ' " +
            req.body.name +
            " ', ' " +
            req.body.room +
            " ', ' " +
            req.body.floorid +
            " ', 0, 0, 'Active')";
        con.query(
            query,
            await function (err, result) {
                if (err) {
                    res.status(500).send("error");
                } else {
                    res.send("Sensor added");
                }
                // console.log("1 record inserted");
            }
        );
    } catch (e) {
        console.log("errot" + e);
    }
});

/*
create api for login function
*/

app.get("/login", async function (req, res) {
    //console.log("get All data");

    try {
        await con.query("SELECT `password` FROM `admins`", function (
            err,
            result,
            fields
        ) {
            if (err) {
                res.status(500).send("error");
            } else {
                res.send(result);
            }
        });
    }
});

```



```

    } catch (e) {
      console.log(e);
      res.status(200).send("error try ", e);
    }
  });

  /*
  create api for update senesor details
  */

  app.post("/editSensor", async function (req, res) {
    try {
      if (req.body.status == "Active") {
        var query =
          "UPDATE `sensors` SET `name`='" +
          req.body.name +
          "', `floor`='" +
          req.body.floor +
          "', `room`='" +
          req.body.room +
          "', `status`='" +
          req.body.status +
          "' WHERE id=" +
          req.body.id +
          " LIMIT 1 ";
      } else {
        var query =
          "UPDATE `sensors` SET `name`= '" +
          req.body.name +
          "', `floor`='" +
          req.body.floor +
          "', `room`='" +
          req.body.room +
          "', `status`='" +
          req.body.status +
          "', `colevel`=0, `smokelevel`=0 WHERE id=" +
          req.body.id +
          " LIMIT 1 ";
      }
      // console.log(query);
      con.query(
        query,
        await function (err, result) {
          if (err) {
            console.log(err);
            res.status(500).send("error");
          } else {
            res.send("Sensor added");
          }
        }
      );
    } catch (e) {
      console.log("errot" + e);
    }
  });

```

```

/*
crate api for send mails
*/

app.post("/sendmail", async function (req, res) {
  try {
    await con.query("SELECT `email`,`name` FROM `clients`", async function (
      err,
      result,
      fields
    ) {
      if (err) {
        res.status(500).send("error");
      } else {
        res.send(result);

        if (result.length != 0) {
          for (var i = 0; i < result.length; i++) {

            // console.log("visited the function");
            let testAccount = await nodemailer.createTestAccount();

            // create reusable transporter object using the default SMTP
transport
            let transporter = nodemailer.createTransport({
              host: "smtp.ethereal.email",
              port: 587,
              secure: false, // true for 465, false for other ports
              auth: {
                user: "claude.lowe@ethereal.email", // generated ethereal
user
                pass: "HASKvX9Qu2WHA4CAC", // generated ethereal password
              },
            });

            // send mail with defined transport object
            let info = await transporter.sendMail({
              from: '"Code4 Fire sensor system" <codefoursliit@gmail.com>',
// sender address
              to: result[i]["email"].toString(), // list of receivers
              subject: "Alert Important", // Subject line
              text:
                "The co2 levels and smoke levels have risen above 5. Hurry!!
leave the room.", // plain text body
              html:
                "<b>Hey!!! " +
                result[i]["name"].toString() +
                " Alert smoke levels co2 levels high in " +
                req.body.room +
                ". Run to the exit</b> <p>Code4 fire alarming system. Thank
you</p>", // html body
            });

          }
        }
      }
    });
  }
});

```

```

    }

    await con.query("SELECT `email` FROM `admins`", async function (
      err1,
      result2,
      fields2
    ) {
      if (err1) {
      } else {
        // console.log("this is result 2" + result2["email"]);
        //console.log("this is result 2" + fields2);

        Object.keys(result2).forEach(async function (key) {
          var row = result2[key];

          let testAccount = await nodemailer.createTestAccount();

          // create reusable transporter object using the default SMTP
transport
          let transporter = nodemailer.createTransport({
            host: "smtp.ethereal.email",
            port: 587,
            secure: false, // true for 465, false for other ports
            auth: {
              user: "claude.lowe@ethereal.email", // generated ethereal
user
              pass: "HASKvX9Qu2WHA4CAC", // generated ethereal password
            },
          });

          // send mail with defined transport object
          let info = await transporter.sendMail({
            from: '"Code4 Fire sensor system 📧" <codefoursliit@gmail.com>',
// sender address
            to: row.email, // list of receivers
            subject: "Admin Alert Important", // Subject line
            text:
              "The co2 levels and smoke levels have risen above 5. Hurry!!
Take immediate action.", // plain text body
            html:
              "<b>Hey Admin!!! Alert smoke levels co2 levels high in " +
              req.body.room +
              ". Take immediate action</b>", // html body
          });

        });
      }
    });
  } catch (e) {
    console.log(e);
    res.status(200).send("error try ", e);
  }
});

```

```
app.listen(3000, console.log("server Start"));
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="logoNew-removebg-preview.png">
    <title>Sensor System</title>
    <link
      rel="stylesheet"

href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
    />
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></scri
pt>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.j
s"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></s
cript>
    <style>
      body {
      }
      table {
        font-family: arial, sans-serif;
        border-collapse: collapse;
        width: 100%;
      }

      td,
      th {
        border: 1px solid #dddddd;
        text-align: left;
        padding: 8px;
      }

      .tr:nth-child(even) {
        background-color: #dddddd;
      }

      h1 {
        text-align: center;
        padding: 20px;
        color: #f8004d;
      }
      .danger {
        color: white;
        background-color: red;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="row">
        <div class="col">
          <div class="card">
            <div class="card-header">
              <h1>Sensor System</h1>
            </div>
            <div class="card-body">
              <table>
                <tr>
                  <th>Sensor ID</th>
                  <th>Sensor Name</th>
                  <th>Sensor Value</th>
                </tr>
                <tr>
                  <td>1</td>
                  <td>Temperature</td>
                  <td>25.5</td>
                </tr>
                <tr>
                  <td>2</td>
                  <td>Humidity</td>
                  <td>65.2</td>
                </tr>
                <tr>
                  <td>3</td>
                  <td>Pressure</td>
                  <td>1013.25</td>
                </tr>
                <tr>
                  <td>4</td>
                  <td>Air Quality</td>
                  <td>150</td>
                </tr>
                <tr>
                  <td>5</td>
                  <td>Light Intensity</td>
                  <td>1200</td>
                </tr>
                <tr>
                  <td>6</td>
                  <td>Sound Level</td>
                  <td>75</td>
                </tr>
                <tr>
                  <td>7</td>
                  <td>Vibration</td>
                  <td>0.5</td>
                </tr>
                <tr>
                  <td>8</td>
                  <td>Proximity</td>
                  <td>10</td>
                </tr>
                <tr>
                  <td>9</td>
                  <td>Motion</td>
                  <td>1</td>
                </tr>
                <tr>
                  <td>10</td>
                  <td>Acceleration</td>
                  <td>9.8</td>
                </tr>
              </table>
            </div>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```

```

        .inactive {
            color: white;
            background-color: #000000da;
        }
    </style>
</head>

<body>
    <h1></h1>
    <div class="container">
        <div class="row" id="table"></div>
    </div>

    <script>
        var div = document.getElementById("table");
        fetch("http://localhost:3000/getSensorData")
            .then((res) => res.json())
            .then((data) => showEvents(data))
            .catch((err) => console.log(err));

        setInterval(() => {
            fetch("http://localhost:3000/getSensorData")
                .then((res) => res.json())
                .then((data) => showEvents(data))
                .catch((err) => console.log(err));
        }, 1000 * 5);

        function showEvents(events) {
            div.innerHTML = "";

            var table = document.createElement("table");
            var th1 = document.createElement("th");
            var th2 = document.createElement("th");
            var th3 = document.createElement("th");
            var th4 = document.createElement("th");
            var th5 = document.createElement("th");
            var th6 = document.createElement("th");
            var tr2 = document.createElement("tr");
            th1.innerHTML = "Sensor Name";
            th2.innerHTML = "Floor";
            th3.innerHTML = "Room";
            th4.innerHTML = "CO2 Level";
            th5.innerHTML = "Smoke Level";
            th6.innerHTML = "Sensor Status";

            tr2.appendChild(th1);
            tr2.appendChild(th2);
            tr2.appendChild(th3);
            tr2.appendChild(th4);
            tr2.appendChild(th5);
            tr2.appendChild(th6);

            table.appendChild(tr2);

            events.map((event) => {
                var tr = document.createElement("tr");

```

```

var td1 = document.createElement("td");
var td2 = document.createElement("td");
var td3 = document.createElement("td");
var td4 = document.createElement("td");
var td5 = document.createElement("td");
var td6 = document.createElement("td");

td1.innerHTML = event.name;
td2.innerHTML = event.floor;
td3.innerHTML = event.room;
td4.innerHTML = event.colevel;
td5.innerHTML = event.smokelevel;
td6.innerHTML = event.status;

if (event.colevel > 5 || event.smokelevel > 5) {
    tr.setAttribute("class", "danger");
} else if(event.status == "Inactive"){
    tr.setAttribute("class", "inactive");
}else{
    tr.setAttribute("class", "tr");
}

// if () {

// } else {

// }

tr.appendChild(td1);
tr.appendChild(td2);
tr.appendChild(td3);
tr.appendChild(td4);
tr.appendChild(td5);
tr.appendChild(td6);

table.appendChild(tr);
});
div.appendChild(table);
}
</script>
</body>
</html>

```

6.5 Sensor Application

```
package sensorapplication;

import com.sun.glass.ui.SystemClipboard;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;
import java.util.logging.Level;
import java.util.logging.Logger;
import org.json.JSONArray;
import org.json.JSONObject;

public class SensorApplication {

    ArrayList<Integer> sensorId;
    /**
     * @param args the command line arguments
     */

    public String updateSernsorLevel(int id, double lavel, double
smokeLevel) throws Exception {
        //To change body of generated methods, choose Tools | Templates.
        try {

            JSONObject jsonObject = new JSONObject();
            jsonObject.put("lavel",lavel );
            jsonObject.put("smokeLevel",smokeLevel);
            jsonObject.put("id",id );

            byte[] postData
=jJSONObject.toString().getBytes(StandardCharsets.UTF_8);
            int length=postData.length;
            URL url = new URL("http://localhost:3000/updateSensorLavel");
            HttpURLConnection conn;
            conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("POST");
            conn.setRequestProperty("userid", "123464");
            conn.setRequestProperty("Content-Type", "application/json");
```

```

        conn.setDoOutput(true);
        DataOutputStream dataOutputStream = new
DataOutputStream(conn.getOutputStream());
        dataOutputStream.write(postData);
        dataOutputStream.flush();
        dataOutputStream.close();
        BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));

        String output = br.readLine();
        System.out.println(output);
        return "Successfull Update";

    } catch (Exception e) {
        System.err.println("err "+e);
        return "error "+e;
    }

}

public void getSensorId()
{
    int count=0;

    sensorId= new ArrayList<>();

    try {

        URL url = new URL("http://localhost:3000/getSensorData");
        HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Accept", "application/json");
        BufferedReader br = new BufferedReader(new
InputStreamReader(conn.getInputStream()));

//System.out.println(Double.parseDouble(jsonObject.get("colevel").toString())
);

        System.out.println("Output from Server .... \n");
        String output = br.readLine();
        JSONArray jsonArray = new JSONArray(output);

        for (int i = 0; i < jsonArray.length(); i++) {
            JSONObject jsonObject = jsonArray.getJSONObject(i);
            int val=Integer.parseInt(jsonObject.get("id").toString());
            String Senstatus=jsonObject.get("status").toString();
            System.out.println("Senstatus "+Senstatus);
            if(Senstatus.equals("Active"))
            {
                count=count+1;
                sensorId.add(val);
            }
        }
    }
}

```



```

        }
        System.out.println("count"+count);
    }catch(Exception e)
    {
        System.err.println("err "+e);
    }
}

private void updateLavel(){
DecimalFormat f = new DecimalFormat("##.00");
DecimalFormat f2 = new DecimalFormat("##.00");
try {
    for (Integer id : sensorId) {
        Random r= new Random();
        Random r2= new Random();
        int rangeMin=0,rangeMax=10;

        double level = Double.parseDouble( f.format(rangeMin + (rangeMax -
rangeMin) * r.nextDouble()));
        double smoke = Double.parseDouble( f2.format(rangeMin + (rangeMax -
rangeMin) * r2.nextDouble()));
        String data = updateSernsorLevel(id,level,smoke);

    }

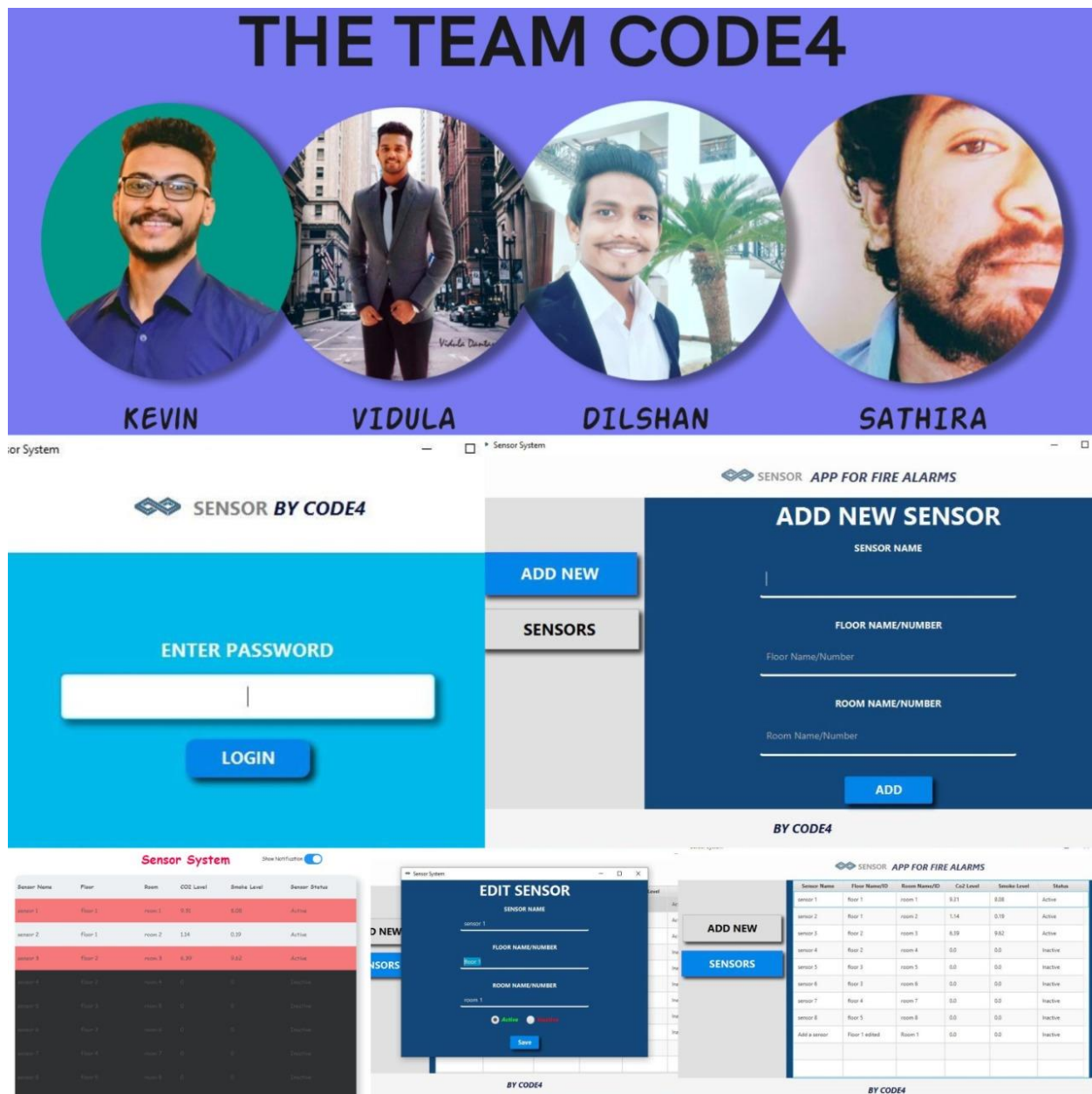
} catch (Exception e) {
    System.err.println("error "+e);
}
}

public static void main(String[] args) {
    // TODO code application logic here
    try{
        final SensorApplication sensorApp=new SensorApplication();
        sensorApp.getSensorId();

        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                try {
                    sensorApp.getSensorId();
                    sensorApp.updateLavel();
                } catch (Exception ex) {

                }
            }
        }, 0, 10000);
    }catch(Exception e)
    {

```



Done by Team CODE4
 Year 3, semester 1
 Sri Lanka Institute of Information Technology