**Sri Lanka Institute of Information Technology**

**Distributed System (SE3020)**

****

**Assignment 02**

**Alarm Monitor System**

**Assignment Report**

|  |  |
| --- | --- |
| **ID Number** | **Name** |
| **IT18045918** | **Akalanka P.K.G.C** |
| **IT18051612** | **Rajapaksha H.M.U.D** |
| **IT18078510** | **Vidhanaarachchi S.P** |
| **IT18051780** | **Gunasekara R.P.T.I** |

Contents

[Introduction 3](#_Toc39689266)

[High Level Architectural Diagram 4](#_Toc39689267)

[System Workflow 5](#_Toc39689268)

[Desktop Client Application 5](#_Toc39689269)

[Web Client Application 6](#_Toc39689270)

[System Workflow Scenario Execution 7](#_Toc39689271)

[Authentication and Security Mechanism 10](#_Toc39689272)

[Appendix 13](#_Toc39689273)

[Web Client Application 13](#_Toc39689274)

[Clientsensorapp 13](#_Toc39689275)

[Component 13](#_Toc39689276)

[Frontend (Component) 19](#_Toc39689277)

[Newbackend 25](#_Toc39689278)

[Controller 25](#_Toc39689279)

[Entity 30](#_Toc39689280)

[Repository 33](#_Toc39689281)

[Services 34](#_Toc39689282)

[resources 38](#_Toc39689283)

[Desktop Client Application 39](#_Toc39689284)

[Login (LoginForm.java) – Package: lk.amc.view 39](#_Toc39689285)

[Sign Up (Register.java) – Package: lk.amc.view 43](#_Toc39689286)

[Add Sensors (ManageSensor.java) – Package: lk.amc.view 49](#_Toc39689287)

[Sensor Details (SensorDetails.java) – Package: lk.amc.view 52](#_Toc39689288)

[Search Sensors (SearchSensor) – Package: lk.amc.view 56](#_Toc39689289)

[Sensor Service (SensorService.java) – Package: lk.amc.service 60](#_Toc39689290)

[Sensor Service (SensorServiceImpl.java) – Package: lk.amc.sensor.impl 61](#_Toc39689291)

[Sensor Controller (SensorController.java) – Package: lk.amc.controller 66](#_Toc39689292)

[Proxy Handler (ProxyHandler.java) – Package: lk.amc.ProxyHandler 67](#_Toc39689293)

[Model Class (Sensor.java) – Package: lk.amc.dto 68](#_Toc39689294)

[Model Class (User.java) – Package: lk.amc.dto 71](#_Toc39689295)

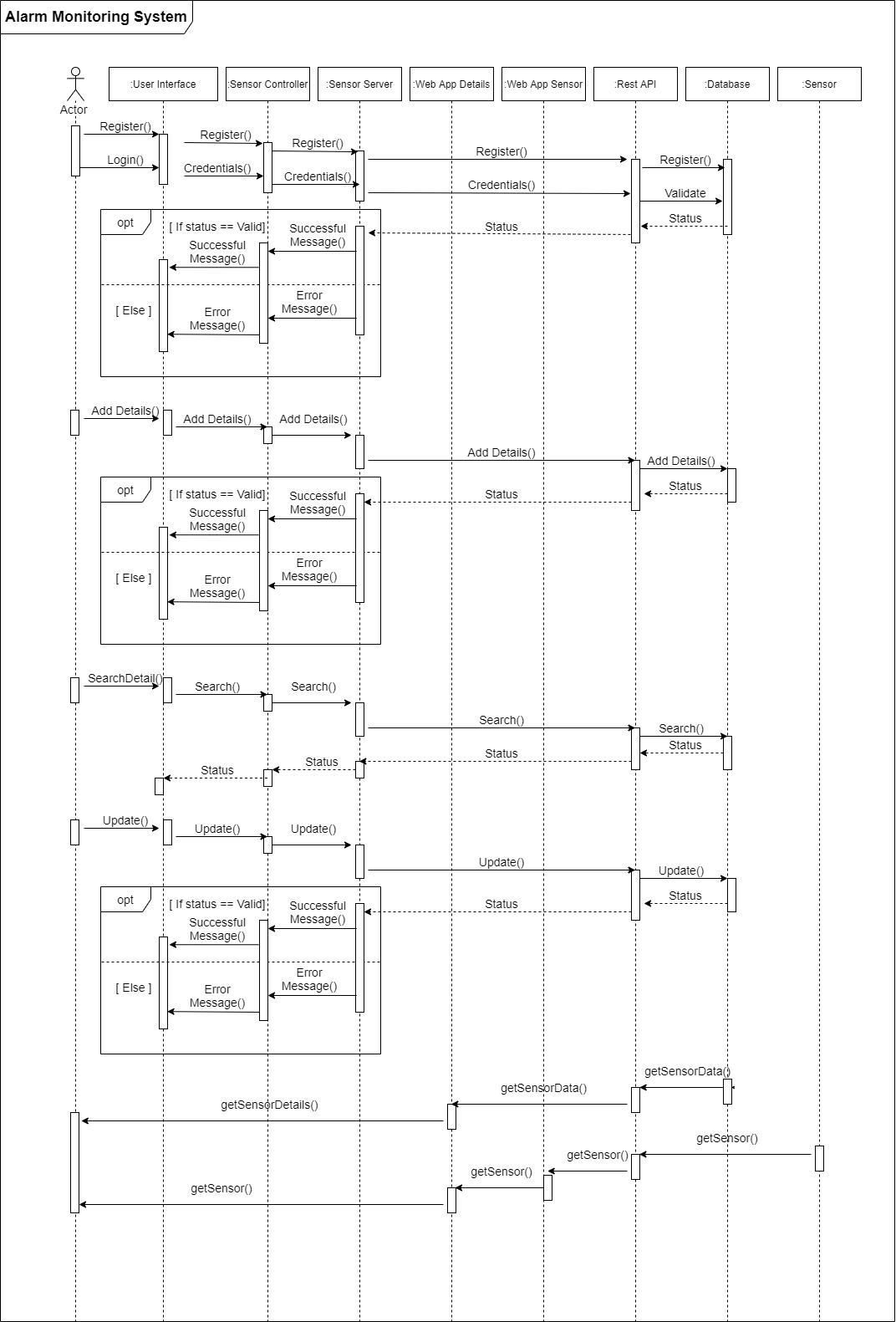
Introduction

Alarm Monitoring System is a web application where users are able to view all fire alarm sensor status. It is implemented by using technologies such as ReactJS for the frontend, spring boot for the backend. The system is a combination of a website and a REST API which is accompanied by an MYSQL database.

A User can view the status of all fire alarm sensors and for each sensor, the system displays whether the fire alarm sensor is active, the location, smoke level and the CO2 level. If the smoke level is above 5 then the color of the chart is changed. Every 40 seconds the sensor details will update.

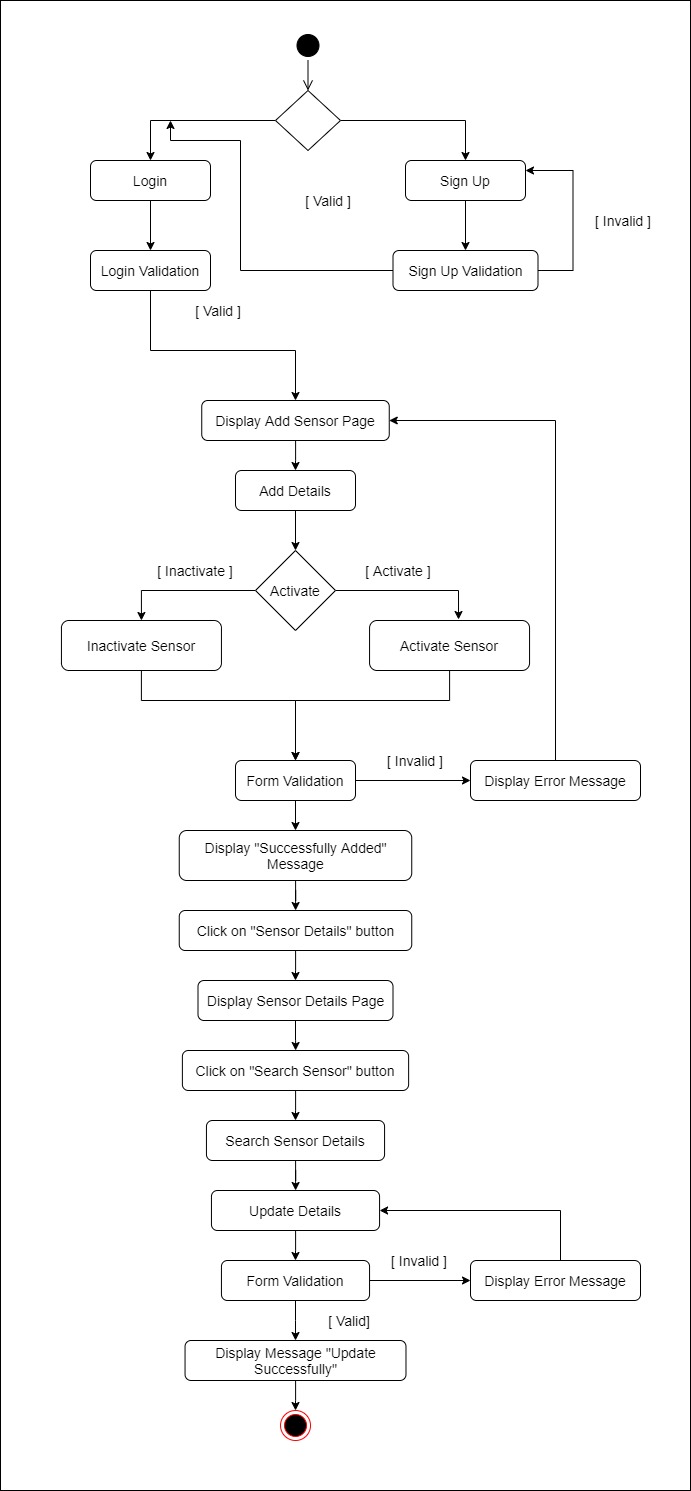
The system will automatically generate alerts (Desktop Application and Web Application), SMS and emails when CO2 and smoke levels are above 5. SMS and emails will be sent to the user. In the RMI server there are security mechanisms like email validations, password validations and numerical validations. Also, the username, password are encoded in RMI server and decoded in REST API.

# High Level Architectural Diagram

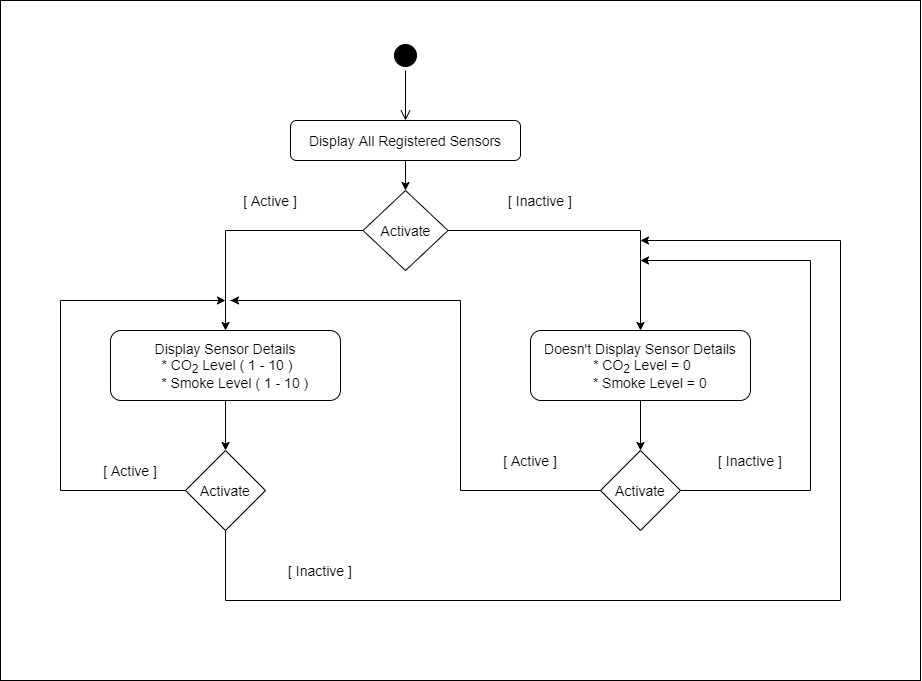
****

# System Workflow

## Desktop Client Application



## Web Client Application



# System Workflow Scenario Execution

* A screenshot of a cell phone

  Description automatically generatedThe User can register into the system by filling the details for the below form.
* A screenshot of a cell phone

  Description automatically generatedIf the user creates an account successfully, he/she can then login to the system by using valid credentials.
* After login, the system will display the below form to add floor number, room number, and status. The sensor ID will auto-generate. The status can be changed by ticking or unticking the checkbox. It will showwhether the status is active or not.

**A screenshot of a cell phone

Description automatically generated**

* **A screenshot of a social media post

  Description automatically generated**The user will redirect to a new page when clicking on the "Sensor Details" button. It will display a table with all the added details. Every 5 seconds, table details are updated and the page will be reloaded
* A screenshot of a cell phone

  Description automatically generatedUsers can search for details by entering sensor Id. As well as a user can update any detail. if he/she wants to.
* After updating the details. The table will display the updated values as below.

A screenshot of a social media post

Description automatically generated

* Finally, user can logout from the system by click on the “Logout” button.

# Authentication and Security Mechanism

* A screenshot of a cell phone

  Description automatically generatedIf the user enters an invalid email, the system will display an error message to re-enter the email address.
* A screenshot of a cell phone

  Description automatically generatedIf the user enters mismatching passwords, the system will display an error message.
* **A screenshot of a cell phone

  Description automatically generated**If a user tires to enter letters instead of numerical values, the system will display an error message.
* **Handle Exceptions.**

|  |
| --- |
| try {  sensor.setRoomNumber(Integer.*parseInt*(roomNumber));  sensor.setFloorNumber(Integer.*parseInt*(floorNumber));  sensor.setSensorId(sensorId);  sensor.setStatus(status);  boolean isAdded = SensorController.addSensor(sensor);  if (isAdded) {  JOptionPane.showMessageDialog(this, "Sensoor Details Added Successfully");  String lastId = SensorController.getLastId();  txtSensorId.setText(lastId);  txtFloorNumber.setText(" ");  txtRoomNumber.setText(" ");  chkStatus.setSelected(false);  } } catch (NumberFormatException ex) {  JOptionPane.showMessageDialog(this, "Please Enter Valid Values to the Field");} |

* The username, password are encoded in RMI server and decoded in REST API.
* **A screenshot of a cell phone

  Description automatically generated**The sensor details are updated every 10 seconds and simultaneously updates the database as well. If the sensor is in “Inactive” state the CO2 and smoke levels are indicated as 0.
* A screenshot of a cell phone

  Description automatically generatedAs this chart shows, if CO2 and Smoke's levels are above 5 then the blue color will change to red. As a bar chart only the sensors which are active is displayed.

# Appendix

## Web Client Application

## Clientsensorapp

### Component

#### Sensors.js

**import** React, {Component} **from "react"**;  
**import** {  
 MDBBtn,  
 MDBCard,  
 MDBCardBody,  
 MDBCardText,  
 MDBCardTitle,  
 MDBCol,  
 MDBContainer,  
 MDBJumbotron,  
 MDBRow  
} **from "mdbreact"**;  
**import './sensor.css'  
import** axios **from "axios"**;  
**import 'sweetalert2/src/sweetalert2.scss'**;  
**import** Swal **from 'sweetalert2/dist/sweetalert2.js'**;  
**import Loader from 'react-loader-spinner'**;  
  
**export default class** Sensors **extends** Component {  
 **\_isMounted** = **false**;  
 constructor(props) {  
 **super**(props);  
 **this**.getAllSensorDetails = **this**.getAllSensorDetails.bind(**this**);  
 **this**.activeSensor = **this**.activeSensor.bind(**this**);  
 **this**.InactiveSensor = **this**.InactiveSensor.bind(**this**);  
 **this**.sendStatus = **this**.sendStatus.bind(**this**);  
 **this**.**state** = {  
 **sensorDetails**: [],  
 **loaderStatus**: **true**,  
 }  
 }  
  
 componentDidMount() {  
 **this**.**\_isMounted** = **true**;  
 **this**.**interval** = *setInterval*(() => {  
 **this**.getAllSensorDetails();  
 **this**.sendStatus ();  
 }, 10000);  
  
 }  
  
 componentWillUnmount() {  
 **this**.**\_isMounted** = **false**;  
 }  
  
  
 activeSensor(sensor) {  
  
 **let** sensorId = sensor.**sensorId**;  
 **const** updatedSensor ={  
 **sensorId** :sensor.**sensorId**,  
 **floorNumber** :sensor.**floorNumber**,  
 **roomNumber** : sensor.**roomNumber**,  
 **smokeLevel** :sensor.**smokeLevel**,  
 **co2Level** : sensor.**co2Level**,  
 **status** : **'Active'** }  
 **this**.**state**.**sensorDetails**.map(sen=>{  
 **if**(sensor.**sensorId**===sen.**sensorId**){  
 sen.**status**=**'Active'**;  
 }  
 })  
 axios.post(**'http://localhost:8080/SensorController/updateSensor'**,updatedSensor).then(response => {  
 Swal.fire(  
 **''**,  
 sensorId+**' Is Activated .After Few Seconds you can notify it'**,  
 **'success'** )  
 }).catch(**function** (error) {  
 ***console***.log(error);  
 })  
  
 }  
  
 sendStatus(){  
 **this**.**state**.**sensorDetails**.map(sensor=>{  
 axios.post(**'http://localhost:8080/SensorController/updateSensor'**,sensor).then(response => {  
 *// Swal.fire(  
 // '',  
 // sensorId+' Is Activated .After Few Seconds you can notify it',  
 // 'success'  
 // )* }).catch(**function** (error) {  
 ***console***.log(error);  
 })  
 })  
 }  
  
 InactiveSensor(sensor) {  
 **let** sensorId = sensor.**sensorId**;  
 **const** updatedSensor ={  
 **sensorId** :sensor.**sensorId**,  
 **floorNumber** :sensor.**floorNumber**,  
 **roomNumber** : sensor.**roomNumber**,  
 **smokeLevel** :*parseInt*(**"0"**),  
 **co2Level** : *parseInt*(**"0"**),  
 **status** : **'Inactive'** }  
 **this**.**state**.**sensorDetails**.map(sen=>{  
 **if**(sensor.**sensorId**===sen.**sensorId**){  
 sen.**status**=**'Inactive'**;  
 }  
 })  
 axios.post(**'http://localhost:8080/SensorController/updateSensor'**,updatedSensor).then(response => {  
  
 Swal.fire(  
 **''**,  
 sensorId+**' Is Inactivated .After Few Seconds you can notify it'**,  
 **'success'** )  
  
 }).catch(**function** (error) {  
 ***console***.log(error);  
 })  
  
 }  
  
 getAllSensorDetails() {  
 axios.get(**'http://localhost:8080/SensorController/getAllSensorDetails'**).then(response => {  
 **if** (**this**.**\_isMounted**) {  
 **this**.setState({  
 **sensorDetails**: response.**data**,  
 **loaderStatus**: **false** });  
 }  
  
 }).catch(**function** (error) {  
 ***console***.log(error);  
 })  
 }  
  
  
 render() {  
 **return** (  
 <**div**>  
 <**MDBContainer**>  
 <**MDBRow**>  
 <**MDBCol**>  
 <**MDBJumbotron style=**{{**padding**: 0}}>  
 <**MDBCol className="text-white text-center px-2 my-2"  
 style=**{{**backgroundImage**: **`url(https://mdbootstrap.com/img/Photos/Others/gradient1.jpg)`**}}>  
 <**MDBCol className="py-5"**>  
 <**MDBCardTitle className="h1-responsive pt-3 m-5 font-bold"**>Sensor Management  
 </**MDBCardTitle**>  
  
 </**MDBCol**>  
 </**MDBCol**>  
 </**MDBJumbotron**>  
 </**MDBCol**>  
 </**MDBRow**>  
 </**MDBContainer**>  
  
 {  
 **this**.**state**.**loaderStatus** ?  
 <**Loader className="loaderClass"  
  
 type="Audio"  
 color="#00BFFF"  
 height=**{400}  
 **width=**{250}  
 **timeout=**{30000} *//3 secs* /> :  
  
 <**MDBContainer**>  
  
 <**MDBRow**>  
 {  
 **this**.**state**.**sensorDetails**.map(sensor => {  
 **return** (  
 <**MDBCol size="4" className="MarginClass" key=**{sensor.**sensorId**}>  
 <**MDBCard style=**{{**width**: **"22rem"**}}>  
 <**MDBCardBody**>  
 <**MDBCardTitle**>{sensor.**sensorId**}</**MDBCardTitle**>  
 <**MDBCardTitle**>Floor Number :{sensor.**floorNumber**}  
 </**MDBCardTitle**>  
 <**MDBCardTitle**>Room Number :{sensor.**roomNumber**} </**MDBCardTitle**>  
 <**MDBCardTitle**>Co2 Level :<**span  
 className="levelClass"**>{sensor.**co2Level**}</**span**>  
 </**MDBCardTitle**>  
 <**MDBCardTitle**>Smoke Level :<**span  
 className="levelClass"**>{sensor.**smokeLevel**}</**span**>  
 </**MDBCardTitle**>  
 <**MDBCardTitle**>Status : <**span  
 className="activeClass"**> {sensor.**status**}</**span**>  
 </**MDBCardTitle**>  
 <**MDBBtn onClick=**{() => **this**.activeSensor(sensor)}>Active  
 Status</**MDBBtn**>  
 <**MDBBtn onClick=**{() => **this**.InactiveSensor(sensor)}>Inactive  
 Status</**MDBBtn**>  
  
 </**MDBCardBody**>  
 </**MDBCard**>  
 </**MDBCol**>  
 )  
 })  
  
 }  
  
  
 </**MDBRow**>  
  
 </**MDBContainer**>  
 }  
 </**div**>  
 );  
 }  
}

#### Sensors.css

.**MarginClass**{  
 **margin-top**: 40**px**;  
}  
  
.**activeClass**{  
 **color**: **limegreen**;  
}  
  
.**levelClass**{  
 **color**: **red**;  
}

#### App.css

.**App** {  
 **text-align**: **center**;  
}  
  
.**App-logo** {  
 **height**: 40**vmin**;  
 **pointer-events**: **none**;  
}  
  
**@media** (**prefers-reduced-motion**: **no-preference**) {  
 .**App-logo** {  
 **animation**: **App-logo-spin infinite** 20**s linear**;  
 }  
}  
  
.**App-header** {  
 **background-color**: **#282c34**;  
 **min-height**: 100**vh**;  
 **display**: **flex**;  
 **flex-direction**: **column**;  
 **align-items**: **center**;  
 **justify-content**: **center**;  
 **font-size**: **calc**(10**px** + 2**vmin**);  
 **color**: **white**;  
}  
  
.**App-link** {  
 **color**: **#61dafb**;  
}  
  
**@keyframes App-logo-spin** {  
 **from** {  
 **transform**: **rotate**(0**deg**);  
 }  
 **to** {  
 **transform**: **rotate**(360**deg**);  
 }  
}

#### App.js

**import** React **from 'react'**;  
**import** logo **from './logo.svg'**;  
**import './App.css'**;  
**import** Sensors **from "./component/sensors"**;  
  
**function** *App*() {  
 **return** (  
 <**div className="App"**>  
 <**Sensors**/>  
 </**div**>  
 );  
}  
  
**export default** *App*;

#### index.css

**body** {  
 **margin**: 0;  
 **font-family**: **-apple-system**, **BlinkMacSystemFont**, **'Segoe UI'**, **'Roboto'**, **'Oxygen'**,  
 **'Ubuntu'**, **'Cantarell'**, **'Fira Sans'**, **'Droid Sans'**, **'Helvetica Neue'**,  
 **sans-serif**;  
 **-webkit-font-smoothing**: **antialiased**;  
 **-moz-osx-font-smoothing**: **grayscale**;  
}  
  
**code** {  
 **font-family**: **source-code-pro**, **Menlo**, **Monaco**, **Consolas**, **'Courier New'**,  
 **monospace**;  
}

#### index.js

**import** React **from 'react'**;  
**import** ReactDOM **from 'react-dom'**;  
**import './index.css'**;  
**import** *App* **from './App'**;  
**import** \* **as** serviceWorker **from './serviceWorker'**;  
**import '@fortawesome/fontawesome-free/css/all.min.css'**; **import  
 'bootstrap-css-only/css/bootstrap.min.css'**; **import  
 'mdbreact/dist/css/mdb.css'**;  
  
ReactDOM.*render*(  
 <**React.StrictMode**>  
 <**App** />  
 </**React.StrictMode**>,  
 ***document***.getElementById(**'root'**)  
);  
  
*// If you want your app to work offline and load faster, you can change  
// unregister() to register() below. Note this comes with some pitfalls.  
// Learn more about service workers: https://bit.ly/CRA-PWA*serviceWorker.*unregister*();

## Frontend (Component)

#### Sensor.css

.loaderClass{  
 margin-top: 50px;  
}

#### Sensorbody.js

import React, {Component} from "react";  
import axios from "axios";  
import {  
 MDBBtn,  
 MDBCard,  
 MDBCardBody,  
 MDBCardText,  
 MDBCardTitle,  
 MDBCol,  
 MDBContainer,  
 MDBJumbotron,  
 MDBRow  
} from "mdbreact";  
import SensorChart from "./sensorchart";  
import Loader from 'react-loader-spinner';  
import './sensor.css'  
import \* as Swal from "sweetalert2";  
  
export default class SensorBody extends Component {  
 \_isMounted = false;  
  
  
 constructor(props) {  
 super(props);  
  
 *// this.getSensorDetails = this.getSensorDetails.bind(this);* this.diplayDetails = this.diplayDetails.bind(this);  
  
 this.state = {  
 sensorDetails: [],  
 loaderStatus: true,  
 chartData: [],  
 Data: {},  
  
 }  
  
 }  
  
 componentDidMount() {  
 this.\_isMounted = true;  
  
 *//Set 10 seconds Time Interval* this.interval = *setInterval*(() => {  
 this.diplayDetails();  
 }, 10000);  
 }  
  
 componentWillUnmount() {  
 this.\_isMounted = false;  
 }  
  
  
 diplayDetails() {  
 axios.get('http://localhost:8080/SensorController/getActiveSensorDetails').then(response => {  
  
 if (this.\_isMounted) {  
  
 const Toast = Swal.mixin({  
 toast: true,  
 position: 'top-end',  
 showConfirmButton: false,  
 timer: 2000,  
 timerProgressBar: true,  
 onOpen: (toast) => {  
 toast.addEventListener('mouseenter', Swal.stopTimer)  
 toast.addEventListener('mouseleave', Swal.resumeTimer)  
 }  
 })  
  
 const sensorResponse = response.data;  
 const newSensors = [];  
 let smokeColor ;  
 let co2Color ;  
  
 *// set Sensor values to Bar Chart* for (let sensor in sensorResponse) {  
 const newValueData = [  
 0,  
 sensorResponse[sensor].co2Level,  
 sensorResponse[sensor].smokeLevel  
 ]  
  
 *// set Red color to chart while co2 level is above five* if(sensorResponse[sensor].co2Level >=5){  
 smokeColor = 'rgba(255,0,0,1.0)';  
 Toast.fire({  
 icon: 'warning',  
 title: sensorResponse[sensor].sensorId +' Co2 Level has increased than the normal level ',  
 })  
 }else{  
 smokeColor = 'rgba(75,192,192,1.0)';  
  
 }  
  
 *// set Red color to chart while smoke level is above five* if(sensorResponse[sensor].smokeLevel >=5){  
 co2Color = 'rgba(255,0,0,1.0)';  
 Toast.fire({  
 icon: 'warning',  
 title: sensorResponse[sensor].sensorId +' Smoke Level has increased than the normal level ',  
 })  
 }else{  
 co2Color = 'rgba(75,192,192,1.0)';  
 }  
 const newData = {  
 labels: ['', 'CO2', 'Smoke'],  
 datasets: [{  
 label: 'Level',  
 data: newValueData,  
 backgroundColor:[  
 '',  
 smokeColor,  
 co2Color  
 ]  
 }  
  
 ],  
  
 }  
 newSensors.push({  
 sensorId: sensorResponse[sensor].sensorId,  
 floorNumber: sensorResponse[sensor].floorNumber,  
 roomNumber: sensorResponse[sensor].roomNumber,  
 data: newData  
 });  
  
 } *//end for* this.setState({  
 sensorDetails: newSensors,  
 loaderStatus: false  
 });  
 }  
  
  
 }).catch(function (error) {  
 console.log(error);  
 })  
 }  
  
 render() {  
 return (  
  
 <div>  
 <MDBContainer>  
 <MDBRow>  
 <MDBCol>  
 <MDBJumbotron style={{padding: 0}}>  
 <MDBCol className="text-white text-center px-2 my-2"  
 style={{backgroundImage: `url(https://mdbootstrap.com/img/Photos/Others/gradient1.jpg)`}}>  
 <MDBCol className="py-5">  
 <MDBCardTitle className="h1-responsive pt-3 m-5 font-bold">Sensor  
 Details</MDBCardTitle>  
  
 </MDBCol>  
 </MDBCol>  
 </MDBJumbotron>  
 </MDBCol>  
 </MDBRow>  
 </MDBContainer>  
  
 {  
 this.state.loaderStatus ?  
 <Loader className="loaderClass"  
 type="Audio"  
 color="#00BFFF"  
 height={400}  
 width={250}  
 timeout={30000} *//3 secs* /> :  
  
 <MDBContainer>  
 <MDBRow>  
 {  
 this.state.sensorDetails.map(sensor => {  
 return (  
 <MDBCol size="6" className="marginCss" key={sensor.sensorId}>  
 <MDBCard style={{width: "33rem"}}>  
 <MDBCardBody>  
 <MDBCardTitle>{sensor.sensorId}</MDBCardTitle>  
 <MDBCardTitle>Floor Number  
 : {sensor.floorNumber} </MDBCardTitle>  
 <MDBCardTitle>Room Number : {sensor.roomNumber}</MDBCardTitle>  
  
 <SensorChart  
 chartData={sensor.data}  
 />  
  
 </MDBCardBody>  
 </MDBCard>  
 </MDBCol>  
 )  
 })  
 }  
  
  
 </MDBRow>  
 <br/>  
  
  
 </MDBContainer>  
 }  
  
 </div>  
 );  
 }  
}

#### Sensorcharts.js

import React, {Component} from "react";  
import {Bar, Line, Pie} from 'react-chartjs-2';  
  
  
export default class SensorChart extends Component {  
  
  
 render() {  
 const {chartData} = this.props;  
 console.log(chartData);  
 return (  
 <div className="chart">  
 <Bar  
 data={  
 chartData  
 }  
 options={{  
 title: {  
 display: true,  
 text: 'Active Sensor Details',  
 fontSize: 15  
 },  
 legend: {  
 display: true,  
 position: 'right'  
 },  
  
 }}  
  
 width={100}  
 height={50}  
 />  
 </div>  
 );  
 }  
}

#### App.css

.App {  
 text-align: center;  
}  
  
.marginCss {  
 margin-top: 20px;  
}

#### App.js

import React from 'react';  
import './App.css';  
import SensorBody from "./component/sensorbody";  
  
  
function *App*() {  
  
  
 return (  
 <div className="App">  
  
<SensorBody/>  
  
 </div>  
 );  
}  
  
export default *App*;

#### Index.css

body {  
 margin: 0;  
 font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',  
 'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',  
 sans-serif;  
 -webkit-font-smoothing: antialiased;  
 -moz-osx-font-smoothing: grayscale;  
}  
  
code {  
 font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',  
 monospace;  
}

#### Index.js

import React from 'react';  
import ReactDOM from 'react-dom';  
import './index.css';  
import *App* from './App';  
import \* as serviceWorker from './serviceWorker';  
import '@fortawesome/fontawesome-free/css/all.min.css'; import  
 'bootstrap-css-only/css/bootstrap.min.css'; import  
 'mdbreact/dist/css/mdb.css';  
  
ReactDOM.render(  
 <React.StrictMode>  
 <App />  
 </React.StrictMode>,  
 document.getElementById('root')  
);  
  
*// If you want your app to work offline and load faster, you can change  
// unregister() to register() below. Note this comes with some pitfalls.  
// Learn more about service workers: https://bit.ly/CRA-PWA*serviceWorker.*unregister*();

## Newbackend

### Controller

#### SensorController.java

package fas.controller;  
  
  
import fas.Entity.Sensor;  
import fas.Entity.User;  
import fas.services.EmailUtil;  
import fas.services.SensorService;  
import fas.services.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import javax.mail.Authenticator;  
import javax.mail.PasswordAuthentication;  
import javax.mail.Session;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.text.DateFormat;  
import java.text.NumberFormat;  
import java.text.SimpleDateFormat;  
import java.util.Date;  
import java.util.List;  
import java.util.Properties;  
  
@CrossOrigin  
@RestController  
@RequestMapping(value = "/SensorController")  
public class SensorController {  
  
  
 @Autowired  
 private SensorService sensorService;  
 @Autowired  
 private UserService userService;  
  
 *//Add Sensor Details to Database* @PostMapping(value = "addSensor")  
 public Sensor addSensor(@RequestBody Sensor sensor) {  
  
 return sensorService.addSensor(sensor);  
  
 }  
  
 *//Update Sesnor Detail* @PostMapping(value = "updateSensor")  
 public Sensor updateSensorStatus(@RequestBody Sensor sensor) {  
  
 return sensorService.addSensor(sensor);  
  
 }  
  
 *//Get Last Id Of Sensor* @GetMapping(value = "/getLastID")  
 public String getLastID() {  
 String lastId = sensorService.getResult();  
 String newID = "";  
 if (lastId != null) {  
 String subid = lastId.substring(5);  
 int id = Integer.parseInt(subid);  
 id++;  
 NumberFormat numberFormat = NumberFormat.getIntegerInstance();  
 numberFormat.setMinimumIntegerDigits(4);  
 numberFormat.setGroupingUsed(false);  
 newID = "S" + getCurrentYear() + numberFormat.format(id); *// Create last id using S , currentyear and last intger number* } else {  
 newID="S" + getCurrentYear() + "0001"; *// Create First id using S , currentyear and last intger number* }  
  
 return newID;  
  
 }  
  
 *//this method is used to get Sensor details according to its sensor Id* @GetMapping(value = "/getSensorDetailsAccordingToID/{sensorId}")  
 public Sensor getSensorDetailsAccordingToID(@PathVariable String sensorId){  
 return sensorService.getSensorDetailsAccordingToID(sensorId);  
 }  
  
 *// Get All Sensors Details* @GetMapping(value = "/getAllSensorDetails")  
 public List<Sensor> getAllSensorDetails(){  
 return sensorService.getAllSensorDetails();  
 }  
  
  
 *// Get All Active Sensor Details* @GetMapping(value = "/getActiveSensorDetails")  
 public List<Sensor> getActiveSensorDetails(){  
  
 updateSensorValues(sensorService.getActiveSensorDetails());  
  
 return sensorService.getActiveSensorDetails();  
 }  
  
 *//Get All Logged Users* @GetMapping(value = "/getLoggedUserDetails")  
 public List<User> getLoggedUserDetails(){  
 return userService.getLoggedUserDetails();  
 }  
  
 *// Update Sensor values( Co2 level and Smoke Level ) Using Random Number* public void updateSensorValues(List<Sensor> sensor){  
 for (Sensor s: sensor  
 ) {  
 Sensor s1 = new Sensor();  
 s1.setRoomNumber(s.getRoomNumber());  
 s1.setFloorNumber(s.getFloorNumber());  
 s1.setSensorId(s.getSensorId());  
 s1.setStatus("Active");  
 int smokeLevel = getRandomNumber(10, 1);  
 int co2Level = getRandomNumber(10, 1);  
 s1.setSmokeLevel(smokeLevel); *//get random number to update Smoke level* s1.setCo2Level(co2Level); *//get random number to update Co2 level* List<User> loggedUsers = getLoggedUserDetails(); *// Get Currently Active users* if(smokeLevel >=5){  
 sendMessage(loggedUsers, s.getSensorId(),smokeLevel,"Smoke");  
 System.out.println("Smoke Level above 5");  
 }  
 if(co2Level >=5){  
 sendMessage(loggedUsers, s.getSensorId(),co2Level,"Co2");  
 System.out.println("Co2 Level above 5");  
 }  
 sensorService.updateSensor(co2Level,smokeLevel,s.getSensorId());  
 }  
 }  
  
 private void sendMessage(List<User> loggedUsers, String sensorId, int level, String type) {  
 System.out.println("Mail Sending");  
  
 for (User user: loggedUsers ) {  
 System.out.println(user.getEmail());  
 if(type.equals("Smoke")){  
 System.err.println("Mail Co2");  
 *//sendTextMessage("Smoke",user,sensorId,level);* sendEmail("Smoke",user,sensorId,level);  
 }else{  
 *//sendTextMessage("Co2",user,sensorId,level);* sendEmail("Co2",user,sensorId,level);  
 }  
 }  
 }  
  
 public void sendEmail(String type,User user,String sensorId,int level){  
 try {  
 final String fromEmail = ""; *//requires valid gmail id* final String password = " ";*// correct password for gmail id* final String toEmail = user.getEmail(); *// can be any email id* System.out.println("TLSEmail Start");  
 Properties props = new Properties();  
 props.put("mail.smtp.host", "smtp.gmail.com"); *//SMTP Host* props.put("mail.smtp.port", "587"); *//TLS Port* props.put("mail.smtp.auth", "true"); *//enable authentication* props.put("mail.smtp.starttls.enable", "true"); *//enable STARTTLS  
  
 //create Authenticator object to pass in Session.getInstance argument* Authenticator auth = new Authenticator() {  
 *//override the getPasswordAuthentication method* protected PasswordAuthentication getPasswordAuthentication() {  
 return new PasswordAuthentication(fromEmail, password);  
 }  
 };  
 Session session = Session.getInstance(props, auth);  
 System.err.println("Mail Sending");  
  
 EmailUtil.sendEmail(session, toEmail,type+"Level Alert", type+" level has increased than usual.(5)Please Take Necessary Steps! ThankYou");  
  
 *//sendTextMessage("Smoke",user,sensorId,level);* }catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
  
 *//Send Text Message* public void sendTextMessage(String type,User user,String sensorId,int level){  
 String msg = type+"%20level%20has%20increased%20the%20limit!!" ; *//Text Message* String number = "94" + Integer.toString(user.getPhoneNo()); *//Receivers Phone Number* String accountName = "94772218111";  
 String password = "8694";  
 URL textit = null;  
 try {  
  
 *// Calling Message Api* textit = new URL("http://textit.biz/sendmsg/index.php?id=" + accountName + "&pw=" + password + "&to=" + number + "&text="+msg);  
  
 BufferedReader in = new BufferedReader(new InputStreamReader(textit.openStream()));  
  
 String inputLine;  
 while((inputLine = in.readLine()) != null)  
 System.out.println("GGG"+inputLine);  
 in.close();  
 }catch (MalformedURLException e) {  
 e.printStackTrace();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 *// This method is used to get random number* public static int getRandomNumber(int maxNumber, int minNumber){  
 return ((int) (Math.random()\*(maxNumber - minNumber))) + minNumber;  
 }  
  
 public void sendEmail(){  
  
 }  
  
  
 *// this method is used to get current year (to create Last id)* public static String getCurrentYear() {  
  
 DateFormat dateFormat = new SimpleDateFormat("yyyy");  
 Date date = new Date();  
 String newDate = dateFormat.format(date);  
 return newDate;  
 }  
  
  
  
}

#### UserController.java

package fas.controller;  
  
  
import fas.Entity.User;  
import fas.services.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.Base64;  
  
@CrossOrigin  
@RestController  
@RequestMapping(value = "/userController")  
public class UserController {  
  
 @Autowired  
 private UserService userService;  
  
  
 *// Add User Details To Database* @PostMapping(value = "addUser")  
 public User addUser(@RequestBody User user) {  
 return userService.addUser(user);  
 }  
  
 *// Check user credentials using username and password* @GetMapping(value = "loginUser/{username}/{password}")  
 public boolean loginUser(@PathVariable String username,@PathVariable String password) {  
  
  
 *//Decoded UserName* byte[] decodedBytesUserName = Base64.*getDecoder*().decode(username);  
 String decodedUserName = new String(decodedBytesUserName);  
  
 *//Decoded Password* byte[] decodedBytesPassword = Base64.*getDecoder*().decode(password);  
 String decodedPassword = new String(decodedBytesPassword);  
  
 return userService.loginUser(decodedUserName,decodedPassword);  
  
 }  
}

### Entity

#### Sensor.java

package fas.Entity;  
  
import javax.persistence.Entity;  
import javax.persistence.Id;  
  
  
*// Sensor Entity*@Entity  
public class Sensor {  
  
 @Id  
 private String sensorId;  
 private int floorNumber;  
 private int roomNumber;  
 private int smokeLevel;  
 private int co2Level;  
 private String status;  
  
 public int getCo2Level() {  
 return co2Level;  
 }  
  
 public void setCo2Level(int co2Level) {  
 this.co2Level = co2Level;  
 }  
  
 public String getSensorId() {  
 return sensorId;  
 }  
  
 public void setSensorId(String sensorId) {  
 this.sensorId = sensorId;  
 }  
  
 public int getFloorNumber() {  
 return floorNumber;  
 }  
  
 public void setFloorNumber(int floorNumber) {  
 this.floorNumber = floorNumber;  
 }  
  
 public int getRoomNumber() {  
 return roomNumber;  
 }  
  
 public void setRoomNumber(int roomNumber) {  
 this.roomNumber = roomNumber;  
 }  
  
 public int getSmokeLevel() {  
 return smokeLevel;  
 }  
  
 public void setSmokeLevel(int smokeLevel) {  
 this.smokeLevel = smokeLevel;  
 }  
  
 public String getStatus() {  
 return status;  
 }  
  
 public void setStatus(String status) {  
 this.status = status;  
 }  
}

#### User.java

package fas.Entity;  
  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;  
  
  
*// User Entity*@Entity  
public class User {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
 private String username;  
 private String email;  
 private int phoneNo;  
 private String password;  
 private String userStatus;  
  
 public String getUserStatus() {  
 return userStatus;  
 }  
  
 public void setUserStatus(String userStatus) {  
 this.userStatus = userStatus;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public int getPhoneNo() {  
 return phoneNo;  
 }  
  
 public void setPhoneNo(int phoneNo) {  
 this.phoneNo = phoneNo;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
}

### Repository

#### SensorRepository

package fas.repository;  
  
  
  
import fas.Entity.Sensor;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Modifying;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.query.Param;  
  
import java.util.List;  
  
public interface SensorRepository extends JpaRepository<Sensor,String> {  
  
 *//get Last Id* @Query(value = " select sensor\_id from sensor order by 1 desc limit 1",nativeQuery = true)  
 Object getLastId();  
  
 *// get sensor details according to its id* @Query(value = "from Sensor where sensorId =?1")  
 Sensor getSensorDetailsAccordingToID(String sensorId);  
  
 *// get only active sensor details* @Query(value = "from Sensor where status ='Active'")  
 List<Sensor> getActiveSensorDetails();  
  
 *//Update Sensor Details* @Modifying(clearAutomatically = true)  
 @Query(value = "update Sensor set smoke\_level=:smokeLevel,co2level=:co2Level where sensor\_id=:sensorId",nativeQuery = true)  
 void updateSensor(@Param("co2Level")int co2Level,@Param("smokeLevel") int smokeLevel,@Param("sensorId") String sensorId);  
  
  
}

#### UserRepository

package fas.repository;  
  
  
import fas.Entity.User;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
  
import java.util.List;  
  
public interface UserRepository extends JpaRepository<User,Integer> {  
  
 *//Check User Login Credentials* @Query(value = "from User where username =?1 and password=?2")  
 User loginUser(String username, String password);  
  
 *//Get Logged User Details* @Query(value = "from User where userStatus ='Logged'")  
 List<User> getLoggedUserDetails();  
}

### Services

#### SensorServiceImpl.java

package fas.services.impl;  
  
  
import fas.Entity.Sensor;  
import fas.repository.SensorRepository;  
import fas.services.SensorService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.List;  
  
@Service  
public class SensorServiceImpl implements SensorService {  
  
 @Autowired  
 private SensorRepository sensorRepository;  
  
 @Override  
 public Sensor addSensor(Sensor sensor) {  
  
 return sensorRepository.save(sensor);  
 }  
  
 @Override  
 public String getResult() {  
 Object lastId =sensorRepository.getLastId();  
 if(lastId!=null){  
 return lastId.toString();  
 }else{  
 return null;  
 }  
 }  
  
 @Override  
 public Sensor getSensorDetailsAccordingToID(String sensorId) {  
 return sensorRepository.getSensorDetailsAccordingToID(sensorId);  
 }  
  
 @Override  
 public List<Sensor> getActiveSensorDetails() {  
  
 return sensorRepository.getActiveSensorDetails();  
 }  
  
 @Override  
 public List<Sensor> getAllSensorDetails() {  
 return sensorRepository.findAll();  
 }  
  
 @Override  
 @Transactional  
 public void updateSensor(int co2Level, int smokeLevel, String sensorId) {  
 sensorRepository.updateSensor(co2Level,smokeLevel,sensorId);  
 }  
}

#### UserServiceImpl.java

package fas.services.impl;  
  
  
import fas.Entity.User;  
import fas.repository.UserRepository;  
import fas.services.UserService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class UserServiceImpl implements UserService {  
  
 @Autowired  
 private UserRepository userRepository;  
  
 @Override  
 public User addUser(User user) {  
 return userRepository.save(user);  
 }  
  
 @Override  
 public boolean loginUser(String username, String password) {  
 User user = userRepository.loginUser(username, password);  
  
 if (user != null) {  
 user.setUserStatus("Logged");  
 userRepository.save(user);  
 return true;  
  
  
 } else {  
 return false;  
 }  
 }  
  
 @Override  
 public List<User> getLoggedUserDetails() {  
 return userRepository.getLoggedUserDetails();  
 }  
  
}

#### EmailUtil.java

package fas.services;  
import java.io.UnsupportedEncodingException;  
import java.util.Date;  
  
import javax.activation.DataHandler;  
import javax.activation.DataSource;  
import javax.activation.FileDataSource;  
import javax.mail.BodyPart;  
import javax.mail.Message;  
import javax.mail.MessagingException;  
import javax.mail.Multipart;  
import javax.mail.Session;  
import javax.mail.Transport;  
import javax.mail.internet.InternetAddress;  
import javax.mail.internet.MimeBodyPart;  
import javax.mail.internet.MimeMessage;  
import javax.mail.internet.MimeMultipart;  
public class EmailUtil {  
 */\*\*  
 \* Utility method to send simple HTML email  
 \* @param session  
 \* @param toEmail  
 \* @param subject  
 \* @param body  
 \*/* public static void sendEmail(Session session, String toEmail, String subject, String body){  
 try  
 {  
 MimeMessage msg = new MimeMessage(session);  
 *//set message headers* msg.addHeader("Content-type", "text/HTML; charset=UTF-8");  
 msg.addHeader("format", "flowed");  
 msg.addHeader("Content-Transfer-Encoding", "8bit");  
  
 msg.setFrom(new InternetAddress("no\_reply@example.com", "NoReply"));  
  
 msg.setReplyTo(InternetAddress.*parse*("no\_reply@example.com", false));  
  
 msg.setSubject(subject, "UTF-8");  
  
 msg.setText(body, "UTF-8");  
  
 msg.setSentDate(new Date());  
  
 msg.setRecipients(Message.RecipientType.*TO*, InternetAddress.*parse*(toEmail, false));  
 System.*out*.println("Message is ready");  
 Transport.*send*(msg);  
  
 System.*out*.println("EMail Sent Successfully!!");  
 }  
 catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

#### SensorService

package fas.services;  
  
  
  
import fas.Entity.Sensor;  
  
import java.util.List;  
  
public interface SensorService {  
 Sensor addSensor(Sensor sensor);  
  
 String getResult();  
  
 Sensor getSensorDetailsAccordingToID(String sensorId);  
  
 List<Sensor> getActiveSensorDetails();  
  
 List<Sensor> getAllSensorDetails();  
  
 void updateSensor(int co2Level, int smokeLevel, String sensorId);  
}

#### UserService

package fas.services;  
  
  
  
  
import fas.Entity.User;  
  
import java.util.List;  
  
public interface UserService {  
 User addUser(User user);  
  
 boolean loginUser(String username, String password);  
  
 List<User> getLoggedUserDetails();  
}

### resources

#### Application.properties

spring.jpa.hibernate.ddl-auto=update  
spring.datasource.url=jdbc:mysql://localhost:3306/sensorDB?createDatabaseIfNotExist=true  
spring.datasource.username=root  
spring.datasource.password=mysql  
spring.jpa.show-sql=true  
spring.jpa.database-platform=org.hibernate.dialect.MySQL57Dialect

## Desktop Client Application

### Login (LoginForm.java) – Package: lk.amc.view

*/\*  
 \* 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 lk.amc.view;  
  
import java.awt.Color;  
import java.awt.Font;  
import javax.swing.JOptionPane;  
import lk.amc.controller.SensorController;  
  
*/\*\*  
 \*  
 \* @author Ruvini  
 \*/*public class LoginForm extends javax.swing.JFrame {  
  
 */\*\*  
 \* Creates new form LoginForm  
 \*/* public LoginForm() {  
 initComponents();  
 jPanel1.setBackground(new Color(0xE5FFCC));  
 Font font1 = new Font("SansSerif", Font.*BOLD*, 20);  
 jLabel1.setFont(font1);  
 }  
  
 */\*\*  
 \* This method is called from within the constructor to initialize the form.  
 \* WARNING: Do NOT modify this code. The content of this method is always  
 \* regenerated by the Form Editor.  
 \*/* @SuppressWarnings("unchecked")  
 *// <editor-fold defaultstate="collapsed" desc="Generated Code">* private void initComponents() {  
  
 jPanel1 = new javax.swing.JPanel();  
 jLabel2 = new javax.swing.JLabel();  
 jLabel1 = new javax.swing.JLabel();  
 jLabel4 = new javax.swing.JLabel();  
 usernameTxt1 = new javax.swing.JTextField();  
 PasswordField2 = new javax.swing.JPasswordField();  
 Register = new javax.swing.JButton();  
 loginbut = new javax.swing.JButton();  
  
 setDefaultCloseOperation(javax.swing.WindowConstants.*EXIT\_ON\_CLOSE*);  
  
 jLabel2.setText("Username");  
  
 jLabel1.setText("Login Form");  
  
 jLabel4.setText("Password");  
  
 Register.setText("Sign Up");  
 Register.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 RegisterActionPerformed(evt);  
 }  
 });  
  
 loginbut.setText("Login");  
 loginbut.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 loginbutActionPerformed(evt);  
 }  
 });  
  
 javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  
 jPanel1.setLayout(jPanel1Layout);  
 jPanel1Layout.setHorizontalGroup(  
 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addContainerGap()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addComponent(jLabel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 217, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(118, 118, 118))  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*TRAILING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGap(0, 181, Short.*MAX\_VALUE*)  
 .addComponent(Register, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 110, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, false)  
 .addComponent(jLabel4, javax.swing.GroupLayout.Alignment.*LEADING*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(jLabel2, javax.swing.GroupLayout.Alignment.*LEADING*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 97, Short.*MAX\_VALUE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(usernameTxt1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 148, Short.*MAX\_VALUE*)  
 .addComponent(PasswordField2))))  
 .addContainerGap(99, Short.*MAX\_VALUE*))))  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGap(56, 56, 56)  
 .addComponent(loginbut, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 110, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap(234, Short.*MAX\_VALUE*)))  
 );  
 jPanel1Layout.setVerticalGroup(  
 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGap(18, 18, 18)  
 .addComponent(jLabel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 44, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 44, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(usernameTxt1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(18, 18, 18)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel4, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 44, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(PasswordField2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*, 46, Short.*MAX\_VALUE*)  
 .addComponent(Register, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 38, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(42, 42, 42))  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, jPanel1Layout.createSequentialGroup()  
 .addContainerGap(220, Short.*MAX\_VALUE*)  
 .addComponent(loginbut, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 38, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(42, 42, 42)))  
 );  
  
 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
 getContentPane().setLayout(layout);  
 layout.setHorizontalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jPanel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 );  
 layout.setVerticalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jPanel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 );  
  
 pack();  
 }*// </editor-fold>* private void RegisterActionPerformed(java.awt.event.ActionEvent evt) {  
 Register r1=new Register();  
 r1.setVisible(true);  
  
 dispose();  
  
 }  
  
 private void loginbutActionPerformed(java.awt.event.ActionEvent evt) {  
  
 String username=usernameTxt1.getText();  
 String password=PasswordField2.getText();  
  
 try{  
  
 boolean istrue = SensorController.loginUser(username, password);  
  
 if (istrue) {  
 JOptionPane.*showMessageDialog*(this, "User Details Login Successfully");  
 usernameTxt1.setText(" ");  
 PasswordField2.setText(" ");  
 MainFrame mn=new MainFrame();  
 mn.setVisible(true);  
 dispose();  
  
 }else{  
 JOptionPane.*showMessageDialog*(this, "User Details are Incorrect!");  
 usernameTxt1.setText(null);  
 PasswordField2.setText(null);  
 }  
 }catch(Exception e){  
 JOptionPane.*showMessageDialog*(this, "Error Occured!");  
 }  
  
  
 }  
  
 */\*\*  
 \* @param args the command line arguments  
 \*/* public static void main(String args[]) {  
 */\* Set the Nimbus look and feel \*/  
 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  
 \*/* try {  
 for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.*getInstalledLookAndFeels*()) {  
 if ("Nimbus".equals(info.getName())) {  
 javax.swing.UIManager.*setLookAndFeel*(info.getClassName());  
 break;  
 }  
 }  
 } catch (ClassNotFoundException ex) {  
 java.util.logging.Logger.*getLogger*(LoginForm.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (InstantiationException ex) {  
 java.util.logging.Logger.*getLogger*(LoginForm.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (IllegalAccessException ex) {  
 java.util.logging.Logger.*getLogger*(LoginForm.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (javax.swing.UnsupportedLookAndFeelException ex) {  
 java.util.logging.Logger.*getLogger*(LoginForm.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 }  
 *//</editor-fold>  
  
 /\* Create and display the form \*/* java.awt.EventQueue.*invokeLater*(new Runnable() {  
 public void run() {  
 new LoginForm().setVisible(true);  
 }  
 });  
 }  
  
 *// Variables declaration - do not modify* private javax.swing.JPasswordField PasswordField2;  
 private javax.swing.JButton Register;  
 private javax.swing.JLabel jLabel1;  
 private javax.swing.JLabel jLabel2;  
 private javax.swing.JLabel jLabel4;  
 private javax.swing.JPanel jPanel1;  
 private javax.swing.JButton loginbut;  
 private javax.swing.JTextField usernameTxt1;  
 *// End of variables declaration*}

### Sign Up (Register.java) – Package: lk.amc.view

*/\*  
 \* 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 lk.amc.view;  
  
import java.awt.Color;  
import java.awt.Font;  
import javax.swing.JOptionPane;  
import javax.swing.JPasswordField;  
import lk.amc.controller.SensorController;  
import lk.amc.dto.User;  
  
*/\*\*  
 \*  
 \* @author Ruvini  
 \*/*public class Register extends javax.swing.JFrame {  
  
 */\*\*  
 \* Creates new form Register  
 \*/* public Register() {  
 initComponents();  
 jPanel1.setBackground(new Color(0xE5FFCC));  
 Font font1 = new Font("SansSerif", Font.*BOLD*, 20);  
 jLabel1.setFont(font1);  
  
 }  
  
 */\*\*  
 \* This method is called from within the constructor to initialize the form.  
 \* WARNING: Do NOT modify this code. The content of this method is always  
 \* regenerated by the Form Editor.  
 \*/* @SuppressWarnings("unchecked")  
 *// <editor-fold defaultstate="collapsed" desc="Generated Code">* private void initComponents() {  
  
 jPanel1 = new javax.swing.JPanel();  
 jLabel1 = new javax.swing.JLabel();  
 jLabel2 = new javax.swing.JLabel();  
 jLabel3 = new javax.swing.JLabel();  
 jLabel4 = new javax.swing.JLabel();  
 jLabel5 = new javax.swing.JLabel();  
 jLabel6 = new javax.swing.JLabel();  
 emailTxt = new javax.swing.JTextField();  
 usernameTxt1 = new javax.swing.JTextField();  
 phnlTxt2 = new javax.swing.JTextField();  
 ConPasswordField1 = new javax.swing.JPasswordField();  
 PasswordField2 = new javax.swing.JPasswordField();  
 savebut = new javax.swing.JButton();  
  
 setDefaultCloseOperation(javax.swing.WindowConstants.*EXIT\_ON\_CLOSE*);  
  
 jLabel1.setText("Register Form");  
  
 jLabel2.setText("Username");  
  
 jLabel3.setText("Phone Number");  
  
 jLabel4.setText("Password");  
  
 jLabel5.setText("Confirm Password");  
  
 jLabel6.setText("Email");  
  
 savebut.setText("Save");  
 savebut.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 savebutActionPerformed(evt);  
 }  
 });  
  
 javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);  
 jPanel1.setLayout(jPanel1Layout);  
 jPanel1Layout.setHorizontalGroup(  
 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGap(59, 59, 59)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jLabel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 217, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jLabel6, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 217, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 217, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel3, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 217, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel4, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 217, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel5, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 217, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(53, 53, 53)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(ConPasswordField1, javax.swing.GroupLayout.Alignment.*TRAILING*)  
 .addComponent(usernameTxt1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 186, Short.*MAX\_VALUE*)  
 .addComponent(emailTxt, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 186, Short.*MAX\_VALUE*)  
 .addComponent(phnlTxt2, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 186, Short.*MAX\_VALUE*)  
 .addComponent(PasswordField2, javax.swing.GroupLayout.Alignment.*TRAILING*)))))  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGap(260, 260, 260)  
 .addComponent(savebut, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 110, javax.swing.GroupLayout.*PREFERRED\_SIZE*)))  
 .addContainerGap(187, Short.*MAX\_VALUE*))  
 );  
 jPanel1Layout.setVerticalGroup(  
 jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGap(35, 35, 35)  
 .addComponent(jLabel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 44, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(18, 18, 18)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 44, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(usernameTxt1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel6, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 44, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(emailTxt, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(jLabel3, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 44, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(phnlTxt2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 34, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(jLabel4, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 44, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGroup(jPanel1Layout.createSequentialGroup()  
 .addGap(12, 12, 12)  
 .addComponent(PasswordField2)))  
 .addGap(18, 18, 18)  
 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(ConPasswordField1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel5, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 44, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(56, 56, 56)  
 .addComponent(savebut, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 38, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap(88, Short.*MAX\_VALUE*))  
 );  
  
 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
 getContentPane().setLayout(layout);  
 layout.setHorizontalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jPanel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 );  
 layout.setVerticalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, layout.createSequentialGroup()  
 .addComponent(jPanel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addGap(0, 0, Short.*MAX\_VALUE*))  
 );  
  
 pack();  
 }*// </editor-fold>* private void savebutActionPerformed(java.awt.event.ActionEvent evt) {  
 String username=usernameTxt1.getText();  
 String email=emailTxt.getText();  
 String phoneNo=phnlTxt2.getText();  
 String password=PasswordField2.getText();  
 String conpassword=ConPasswordField1.getText();  
  
  
  
 try{  
 if(*isValidEmail*(email)){  
  
 if(password.equalsIgnoreCase(conpassword)){  
  
 User user=new User(username,email,Integer.*parseInt*(phoneNo),password);  
 boolean isAdded = SensorController.addUser(user);  
 System.*out*.println(" Succesfully!");  
 if (isAdded) {  
 System.*out*.println("Added Succesfully!");  
 JOptionPane.*showMessageDialog*(this, "User Details Added Successfully");  
 usernameTxt1.setText(" ");  
 emailTxt.setText(" ");  
 phnlTxt2.setText(" ");  
 PasswordField2.setText(" ");  
 ConPasswordField1.setText(" ");  
  
  
 LoginForm mn=new LoginForm();  
 mn.setVisible(true);  
 dispose();  
 }  
  
  
 }else{  
 JOptionPane.*showMessageDialog*(this, "Password doesn't match!");  
 }  
  
 }else{  
 JOptionPane.*showMessageDialog*(this, "Enter a valid Email Address!");  
 }  
 }catch(Exception e){  
 JOptionPane.*showMessageDialog*(this, "Error!");  
 }  
  
  
  
 }  
  
 */\*\*  
 \* @param args the command line arguments  
 \*/* public static void main(String args[]) {  
 */\* Set the Nimbus look and feel \*/  
 //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">  
 /\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.  
 \* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html  
 \*/* try {  
 for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.*getInstalledLookAndFeels*()) {  
 if ("Nimbus".equals(info.getName())) {  
 javax.swing.UIManager.*setLookAndFeel*(info.getClassName());  
 break;  
 }  
 }  
 } catch (ClassNotFoundException ex) {  
 java.util.logging.Logger.*getLogger*(Register.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (InstantiationException ex) {  
 java.util.logging.Logger.*getLogger*(Register.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (IllegalAccessException ex) {  
 java.util.logging.Logger.*getLogger*(Register.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 } catch (javax.swing.UnsupportedLookAndFeelException ex) {  
 java.util.logging.Logger.*getLogger*(Register.class.getName()).log(java.util.logging.Level.*SEVERE*, null, ex);  
 }  
 *//</editor-fold>  
  
 /\* Create and display the form \*/* java.awt.EventQueue.*invokeLater*(new Runnable() {  
 public void run() {  
 new Register().setVisible(true);  
 }  
 });  
 }  
  
 *// Variables declaration - do not modify* private javax.swing.JPasswordField ConPasswordField1;  
 private javax.swing.JPasswordField PasswordField2;  
 private javax.swing.JTextField emailTxt;  
 private javax.swing.JLabel jLabel1;  
 private javax.swing.JLabel jLabel2;  
 private javax.swing.JLabel jLabel3;  
 private javax.swing.JLabel jLabel4;  
 private javax.swing.JLabel jLabel5;  
 private javax.swing.JLabel jLabel6;  
 private javax.swing.JPanel jPanel1;  
 private javax.swing.JTextField phnlTxt2;  
 private javax.swing.JButton savebut;  
 private javax.swing.JTextField usernameTxt1;  
 *// End of variables declaration* static boolean isValidEmail(String email) {  
 String regex = "^[\\w-\_\\.+]\*[\\w-\_\\.]\\@([\\w]+\\.)+[\\w]+[\\w]$";  
 return email.matches(regex);  
 }  
  
  
}

### Add Sensors (ManageSensor.java) – Package: lk.amc.view

*/\*  
 \* 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 lk.amc.view;  
  
import java.awt.Color;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.swing.JOptionPane;  
import lk.amc.controller.SensorController;  
import lk.amc.dto.Sensor;  
  
*/\*\*  
 \*  
 \* @author Dakshika  
 \*/*public class ManageSenor extends javax.swing.JPanel {  
  
 */\*\*  
 \* Creates new form ManageSenor  
 \*/* public ManageSenor() {  
 initComponents();  
 this.setBackground(new Color(0xE5FFCC));  
 try {  
 String lastId = SensorController.getLastId();  
 txtSensorId.setText(lastId);  
 } catch (Exception ex) {  
 Logger.*getLogger*(ManageSenor.class.getName()).log(Level.*SEVERE*, null, ex);  
 }  
 }  
  
 */\*\*  
 \* This method is called from within the constructor to initialize the form.  
 \* WARNING: Do NOT modify this code. The content of this method is always  
 \* regenerated by the Form Editor.  
 \*/* @SuppressWarnings("unchecked")  
 *// <editor-fold defaultstate="collapsed" desc="Generated Code">* private void initComponents() {  
  
 jButton1 = new javax.swing.JButton();  
 jLabel1 = new javax.swing.JLabel();  
 jLabel2 = new javax.swing.JLabel();  
 jLabel3 = new javax.swing.JLabel();  
 txtRoomNumber = new javax.swing.JTextField();  
 chkStatus = new javax.swing.JCheckBox();  
 txtFloorNumber = new javax.swing.JTextField();  
 txtSensorId = new javax.swing.JTextField();  
 jLabel4 = new javax.swing.JLabel();  
  
 jButton1.setText("Save");  
 jButton1.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 jButton1ActionPerformed(evt);  
 }  
 });  
  
 jLabel1.setText("Floor Number");  
  
 jLabel2.setText("Room Number");  
  
 jLabel3.setText("Status");  
  
 chkStatus.setText("Active");  
 chkStatus.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 chkStatusActionPerformed(evt);  
 }  
 });  
  
 jLabel4.setText("Sensor Id");  
  
 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);  
 this.setLayout(layout);  
 layout.setHorizontalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addGap(49, 49, 49)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(jLabel3, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 127, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 127, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 127, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel4, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 127, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(72, 72, 72)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addComponent(txtSensorId, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 233, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(chkStatus, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 79, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jButton1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 110, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(txtRoomNumber, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 233, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(txtFloorNumber, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 233, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addContainerGap(178, Short.*MAX\_VALUE*))  
 );  
 layout.setVerticalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addGap(30, 30, 30)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(txtSensorId, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel4, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(txtFloorNumber, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(18, 18, 18)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(txtRoomNumber, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(chkStatus)  
 .addComponent(jLabel3, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(26, 26, 26)  
 .addComponent(jButton1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 38, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap(117, Short.*MAX\_VALUE*))  
 );  
 }*// </editor-fold>* private void chkStatusActionPerformed(java.awt.event.ActionEvent evt) {  
 *// TODO add your handling code here:* }  
  
 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
  
 String sensorId = txtSensorId.getText();  
 String floorNumber = txtFloorNumber.getText();  
 String roomNumber = txtRoomNumber.getText();  
 String status = "";  
 if (chkStatus.isSelected()) {  
 status = "Active";  
 } else {  
 status = "inActive";  
 }  
 Sensor sensor = new Sensor();  
  
 try {  
 sensor.setRoomNumber(Integer.*parseInt*(roomNumber));  
 sensor.setFloorNumber(Integer.*parseInt*(floorNumber));  
 sensor.setSensorId(sensorId);  
 sensor.setStatus(status);  
 boolean isAdded = SensorController.addSensor(sensor);  
 if (isAdded) {  
 JOptionPane.*showMessageDialog*(this, "Sensoor Details Added Successfully");  
 String lastId = SensorController.getLastId();  
 txtSensorId.setText(lastId);  
 txtFloorNumber.setText(" ");  
 txtRoomNumber.setText(" ");  
 chkStatus.setSelected(false);  
 }  
 } catch (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(this, "Please Enter Valid Values to the Field");  
  
 } catch (Exception ex) {  
  
*// Logger.getLogger(ManageSenor.class.getName()).log(Level.SEVERE, null, ex);* }  
  
  
 }  
  
  
 *// Variables declaration - do not modify* private javax.swing.JCheckBox chkStatus;  
 private javax.swing.JButton jButton1;  
 private javax.swing.JLabel jLabel1;  
 private javax.swing.JLabel jLabel2;  
 private javax.swing.JLabel jLabel3;  
 private javax.swing.JLabel jLabel4;  
 private javax.swing.JTextField txtFloorNumber;  
 private javax.swing.JTextField txtRoomNumber;  
 private javax.swing.JTextField txtSensorId;  
 *// End of variables declaration*}

### Sensor Details (SensorDetails.java) – Package: lk.amc.view

*/\*  
 \* 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 lk.amc.view;  
  
import java.awt.Color;  
import java.awt.Component;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Vector;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.swing.JComponent;  
import javax.swing.JOptionPane;  
import javax.swing.table.DefaultTableModel;  
import javax.swing.table.TableCellRenderer;  
import lk.amc.controller.SensorController;  
import lk.amc.dto.Sensor;  
  
*/\*\*  
 \*  
 \* @author Dakshika  
 \*/*public class SensorDetails extends javax.swing.JPanel implements ActionListener {  
  
 *// javax.swing.Timer timer = new javax.swing.Timer(900, this);* javax.swing.Timer timer = new javax.swing.Timer(5000, this);  
  
 */\*\*  
 \* Creates new form SenosrDetails  
 \*/* public SensorDetails() {  
 initComponents();  
 timer.start();  
 loadSernsorDetails();  
 }  
  
 */\*\*  
 \* This method is called from within the constructor to initialize the form.  
 \* WARNING: Do NOT modify this code. The content of this method is always  
 \* regenerated by the Form Editor.  
 \*/* @SuppressWarnings("unchecked")  
 *// <editor-fold defaultstate="collapsed" desc="Generated Code">* private void initComponents() {  
  
 jScrollPane2 = new javax.swing.JScrollPane();  
 sensorTable = new javax.swing.JTable(){  
  
 @Override  
 public Component prepareRenderer(TableCellRenderer renderer, int row, int col) {  
 JComponent c =(JComponent) super.prepareRenderer(renderer, row, col);  
  
 String smokeLevel = (String)getModel().getValueAt(row, 4);  
 String co2Level = (String)getModel().getValueAt(row, 5);  
  
 if(smokeLevel!=null && co2Level !=null ){  
 if(col==4 && Integer.*parseInt*(smokeLevel)>=5){  
 c.setBackground(Color.*RED*);  
 c.setForeground(Color.*BLACK*);  
 }else if (col==5 && Integer.*parseInt*(co2Level)>=5) {  
 c.setBackground(Color.*RED*);  
 c.setForeground(Color.*BLACK*);  
 } else {  
 c.setBackground(Color.*WHITE*);  
 c.setForeground(Color.*BLACK*);  
 }  
 }  
  
 return c;  
 }  
 };  
  
 sensorTable.setModel(new javax.swing.table.DefaultTableModel(  
 new Object [][] {  
 {null, null, null, null},  
 {null, null, null, null},  
 {null, null, null, null},  
 {null, null, null, null}  
 },  
 new String [] {  
 "Title 1", "Title 2", "Title 3", "Title 4"  
 }  
 ));  
 jScrollPane2.setViewportView(sensorTable);  
  
 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);  
 this.setLayout(layout);  
 layout.setHorizontalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addGap(52, 52, 52)  
 .addComponent(jScrollPane2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 652, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap(22, Short.*MAX\_VALUE*))  
 );  
 layout.setVerticalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, layout.createSequentialGroup()  
 .addContainerGap(43, Short.*MAX\_VALUE*)  
 .addComponent(jScrollPane2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap())  
 );  
 }*// </editor-fold>   
  
  
 // Variables declaration - do not modify* private javax.swing.JScrollPane jScrollPane2;  
 private javax.swing.JTable sensorTable;  
 *// End of variables declaration* private void loadSernsorDetails() {  
  
 try {  
 List<Sensor> sensor = SensorController.getAllSensorDetails();  
 Vector<Vector<String>> vectors = new Vector<>();  
 Vector<String> header = new Vector<>();  
  
 for (Sensor s2 : sensor) {  
 Vector<String> vector = new Vector<>();  
  
 *//assign data to vector* vector.add(s2.getStatus());  
 vector.add(s2.getSensorId());  
 vector.add(Integer.*toString*(s2.getFloorNumber()));  
 vector.add(Integer.*toString*(s2.getRoomNumber()));  
 vector.add(Integer.*toString*(s2.getSmokeLevel()));  
 vector.add(Integer.*toString*(s2.getCo2Level()));  
 vectors.add(vector);  
  
 if(s2.getCo2Level()>5){  
 JOptionPane.*showMessageDialog*(this, "CO2 level of "+s2.getSensorId()+" has increased than the limit!!");  
 }  
 if(s2.getSmokeLevel()>5){  
 JOptionPane.*showMessageDialog*(this, "Smoke level of "+s2.getSensorId()+" has increased than the limit!!");  
 }  
  
 }  
  
 *// Add Headers to JTable* header.add("Status");  
 header.add("Sensor Id");  
 header.add("Floor Number");  
 header.add("Room Number");  
 header.add("Smoke Level");  
 header.add("Co2 Level");  
  
 *//Declare Default Table Model* DefaultTableModel model = new DefaultTableModel(vectors, header);  
 *//Add Model to sensor table* sensorTable.setModel(model); *//set table* } catch (Exception ex) {  
 Logger.*getLogger*(SensorDetails.class.getName()).log(Level.*SEVERE*, null, ex);  
 }  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 loadSernsorDetails();  
 }  
  
}

### Search Sensors (SearchSensor) – Package: lk.amc.view

*/\*  
 \* 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 lk.amc.view;  
  
import java.awt.Color;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import javax.swing.JOptionPane;  
import lk.amc.controller.SensorController;  
import lk.amc.dto.Sensor;  
  
*/\*\*  
 \*  
 \* @author Dakshika  
 \*/*public class SearchSensor extends javax.swing.JPanel {  
  
 */\*\*  
 \* Creates new form ManageSenor  
 \*/* private Sensor sensor;  
  
 public SearchSensor() {  
 initComponents();  
 this.setBackground(new Color(0xE5FFCC));  
  
 }  
  
 */\*\*  
 \* This method is called from within the constructor to initialize the form.  
 \* WARNING: Do NOT modify this code. The content of this method is always  
 \* regenerated by the Form Editor.  
 \*/* @SuppressWarnings("unchecked")  
 *// <editor-fold defaultstate="collapsed" desc="Generated Code">* private void initComponents() {  
  
 jButton1 = new javax.swing.JButton();  
 jLabel1 = new javax.swing.JLabel();  
 jLabel2 = new javax.swing.JLabel();  
 jLabel3 = new javax.swing.JLabel();  
 txtRoomNumber = new javax.swing.JTextField();  
 chkStatus = new javax.swing.JCheckBox();  
 txtFloorNumber = new javax.swing.JTextField();  
 txtSensorId = new javax.swing.JTextField();  
 jLabel4 = new javax.swing.JLabel();  
 btnSearch = new javax.swing.JButton();  
  
 jButton1.setText("Update");  
 jButton1.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 jButton1ActionPerformed(evt);  
 }  
 });  
  
 jLabel1.setText("Floor Number");  
  
 jLabel2.setText("Room Number");  
  
 jLabel3.setText("Status");  
  
 chkStatus.setText("Active");  
 chkStatus.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 chkStatusActionPerformed(evt);  
 }  
 });  
  
 jLabel4.setText("Sensor Id");  
  
 btnSearch.setText("Search");  
 btnSearch.addActionListener(new java.awt.event.ActionListener() {  
 public void actionPerformed(java.awt.event.ActionEvent evt) {  
 btnSearchActionPerformed(evt);  
 }  
 });  
  
 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);  
 this.setLayout(layout);  
 layout.setHorizontalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addGap(40, 40, 40)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*TRAILING*, false)  
 .addComponent(jLabel1, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*)  
 .addComponent(jLabel2, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 136, Short.*MAX\_VALUE*)  
 .addComponent(jLabel3, javax.swing.GroupLayout.*DEFAULT\_SIZE*, javax.swing.GroupLayout.*DEFAULT\_SIZE*, Short.*MAX\_VALUE*))  
 .addComponent(jLabel4, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 136, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(18, 18, 18)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*, false)  
 .addComponent(chkStatus, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 79, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jButton1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 110, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(txtFloorNumber, javax.swing.GroupLayout.*DEFAULT\_SIZE*, 210, Short.*MAX\_VALUE*)  
 .addComponent(txtSensorId)  
 .addComponent(txtRoomNumber))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*UNRELATED*)  
 .addComponent(btnSearch)  
 .addContainerGap(180, Short.*MAX\_VALUE*))  
 );  
 layout.setVerticalGroup(  
 layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*LEADING*)  
 .addGroup(layout.createSequentialGroup()  
 .addGap(29, 29, 29)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(txtSensorId, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel4, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(btnSearch, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(18, 18, 18)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(txtFloorNumber, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(18, 18, 18)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(txtRoomNumber, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addComponent(jLabel2, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.*RELATED*)  
 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.*BASELINE*)  
 .addComponent(chkStatus)  
 .addComponent(jLabel3, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 36, javax.swing.GroupLayout.*PREFERRED\_SIZE*))  
 .addGap(26, 26, 26)  
 .addComponent(jButton1, javax.swing.GroupLayout.*PREFERRED\_SIZE*, 38, javax.swing.GroupLayout.*PREFERRED\_SIZE*)  
 .addContainerGap(111, Short.*MAX\_VALUE*))  
 );  
 }*// </editor-fold>* private void chkStatusActionPerformed(java.awt.event.ActionEvent evt) {  
 *// TODO add your handling code here:* }  
  
 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
  
 String sensorId = txtSensorId.getText();  
 String floorNumber = txtFloorNumber.getText();  
 String roomNumber = txtRoomNumber.getText();  
 String status = "";  
 if (chkStatus.isSelected()) {  
 status = "Active";  
 } else {  
 status = "InActive";  
 }  
 Sensor sensor = new Sensor();  
  
 try {  
 sensor.setRoomNumber(Integer.*parseInt*(roomNumber));  
 sensor.setFloorNumber(Integer.*parseInt*(floorNumber));  
 sensor.setSensorId(sensorId);  
 sensor.setStatus(status);  
 boolean isAdded = SensorController.addSensor(sensor);  
 if (isAdded) {  
 JOptionPane.*showMessageDialog*(this, "Sensoor Details Updated Successfully");  
  
 txtFloorNumber.setText(" ");  
 txtRoomNumber.setText(" ");  
 chkStatus.setSelected(false);  
 }  
 } catch (NumberFormatException ex) {  
 JOptionPane.*showMessageDialog*(this, "Please Enter Valid Values to the Field");  
  
 } catch (Exception ex) {  
  
*// Logger.getLogger(ManageSenor.class.getName()).log(Level.SEVERE, null, ex);* }  
  
  
 }  
  
 private void btnSearchActionPerformed(java.awt.event.ActionEvent evt) {  
 try {  
 String sensorId = txtSensorId.getText();  
 sensor = SensorController.getSensorDetailsAccordingToID(sensorId);  
 if (sensor != null) {  
 txtRoomNumber.setText(""+sensor.getRoomNumber());  
 txtFloorNumber.setText(""+sensor.getFloorNumber());  
 if(sensor.getStatus().equalsIgnoreCase("Active")){  
 chkStatus.setSelected(true);  
 }else{  
 chkStatus.setSelected(false);  
 }  
  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Sensor Details Not Found");  
 }  
 } catch (Exception ex) {  
 Logger.*getLogger*(SearchSensor.class.getName()).log(Level.*SEVERE*, null, ex);  
 }  
  
 }  
  
  
 *// Variables declaration - do not modify* private javax.swing.JButton btnSearch;  
 private javax.swing.JCheckBox chkStatus;  
 private javax.swing.JButton jButton1;  
 private javax.swing.JLabel jLabel1;  
 private javax.swing.JLabel jLabel2;  
 private javax.swing.JLabel jLabel3;  
 private javax.swing.JLabel jLabel4;  
 private javax.swing.JTextField txtFloorNumber;  
 private javax.swing.JTextField txtRoomNumber;  
 private javax.swing.JTextField txtSensorId;  
 *// End of variables declaration*}

### Sensor Service (SensorService.java) – Package: lk.amc.service

*/\*  
 \* 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 lk.amc.service;  
  
import java.rmi.Remote;  
import java.util.List;  
import lk.amc.dto.Sensor;  
import lk.amc.dto.User;  
  
*/\*\*  
 \*  
 \* @author Dakshika  
 \*/*public interface SensorService extends Remote{  
  
 public boolean addSensor(Sensor sensor)throws Exception;  
  
  
 public int increment()throws Exception;  
  
 public String getLastId()throws Exception;  
  
 public List<Sensor> getAllSensorDetails()throws Exception;  
  
 public Sensor getSensorDetailsAccordingToID(String sensorId)throws Exception;  
  
 public boolean addUser(User user) throws Exception;  
  
 public boolean loginUser(String username,String password) throws Exception;  
}

### Sensor Service (SensorServiceImpl.java) – Package: lk.amc.sensor.impl

*/\*  
 \* 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 lk.amc.sensor.impl;  
  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
import java.io.OutputStream;  
import java.net.HttpURLConnection;  
import java.net.URL;  
import java.rmi.RemoteException;  
import java.rmi.server.UnicastRemoteObject;  
import java.util.ArrayList;  
import java.util.Base64;  
import java.util.List;  
import lk.amc.dto.Sensor;  
import lk.amc.dto.User;  
import lk.amc.service.SensorService;  
import org.json.JSONArray;  
import org.json.JSONObject;  
  
*/\*\*  
 \*  
 \* @author Dakshika  
 \*/*public class SensorServiceImpl extends UnicastRemoteObject implements SensorService {  
  
 private Boolean res = false;  
  
 public SensorServiceImpl() throws RemoteException {  
  
 }  
  
 private static int *count* = 0;  
  
 @Override  
 public boolean addSensor(Sensor s) throws Exception {  
  
 *//Add Sensor Details to sensor Object* JSONObject sensorDetails = new JSONObject();  
 sensorDetails.put("sensorId", s.getSensorId());  
 sensorDetails.put("floorNumber", s.getFloorNumber());  
 sensorDetails.put("roomNumber", s.getRoomNumber());  
 sensorDetails.put("smokeLevel", s.getSmokeLevel());  
 sensorDetails.put("status", s.getStatus());  
 JSONObject sensorObject = new JSONObject();  
 sensorObject.put("Sensor", sensorDetails);  
  
 *//Send This Sensor json object to springboot rest Api* URL url = new URL("http://localhost:8080/SensorController/addSensor");  
 HttpURLConnection con = (HttpURLConnection) url.openConnection();  
 con.setRequestMethod("POST");  
 con.setRequestProperty("Content-Type", "application/json; utf-8");  
 con.setRequestProperty("Accept", "application/json");  
 con.setDoOutput(true);  
 String jsonString = sensorDetails.toString();  
 try (OutputStream os = con.getOutputStream()) {  
 byte[] input = jsonString.getBytes("utf-8");  
 os.write(input, 0, input.length);  
 }  
 try (BufferedReader br = new BufferedReader(  
 new InputStreamReader(con.getInputStream(), "utf-8"))) {  
 StringBuilder response = new StringBuilder();  
 String responseLine = null;  
 while ((responseLine = br.readLine()) != null) {  
 response.append(responseLine.trim());  
 }  
 }  
 con.disconnect();  
 return true;  
 }  
  
 @Override  
 public synchronized int increment() throws Exception {  
 *count*++;  
 return *count*;  
  
 }  
  
 *//This method is use to get Last Sensor Id* @Override  
 public String getLastId() throws Exception {  
 HttpURLConnection connection = null;  
 StringBuffer content = null;  
 try {  
 URL url = new URL("http://localhost:8080/SensorController/getLastID");  
 connection = (HttpURLConnection) url.openConnection();  
 connection.setRequestMethod("GET");  
 connection.setUseCaches(false);  
 connection.setDoOutput(true);  
 connection.setRequestProperty("Content-Type", "application/json");  
 String contentType = connection.getHeaderField("Content-Type");  
  
 BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream()));  
 String inputLine;  
 content = new StringBuffer();  
 while ((inputLine = br.readLine()) != null) {  
 content.append(inputLine);  
 }  
 br.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 if (connection != null) {  
 connection.disconnect();  
 }  
 }  
  
 *//return the last sensor ID* return content.toString();  
 }  
  
 @Override  
 public List<Sensor> getAllSensorDetails() throws Exception {  
  
 URL url;  
 List<Sensor> sensorList = null;  
 *//Get All Sensor Details Using Spring Boot Rest Api* url = new URL("http://localhost:8080/SensorController/getAllSensorDetails");  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
 connection.setRequestMethod("GET");  
 connection.setRequestProperty("Content-Type", "application/json; utf-8");  
 connection.setRequestProperty("Accept", "application/json");  
 int responseCode = connection.getResponseCode();  
 *//System.out.println("GET Response Code :: " + responseCode);* if (responseCode == HttpURLConnection.*HTTP\_OK*) { *// success* BufferedReader br = new BufferedReader(new InputStreamReader(  
 connection.getInputStream()));  
 String inputLine;  
 StringBuffer response = new StringBuffer();  
 while ((inputLine = br.readLine()) != null) {  
 response.append(inputLine);  
 }  
 br.close();  
 String result = response.toString();  
 JSONArray jsonArray = new JSONArray(result);  
  
 sensorList = new ArrayList<>();  
  
 for (int i = 0; i < jsonArray.length(); i++) {  
 JSONObject jobj = jsonArray.getJSONObject(i);  
  
 Sensor s = new Sensor();  
 *//Convert json object data and Set it to sensor Object* s.setCo2Level(jobj.getInt("co2Level"));  
 s.setFloorNumber(jobj.getInt("floorNumber"));  
 s.setRoomNumber(jobj.getInt("roomNumber"));  
 s.setSensorId(String.*valueOf*(jobj.getString("sensorId")));  
 s.setSmokeLevel(jobj.getInt("smokeLevel"));  
 s.setStatus(String.*valueOf*(jobj.getString("status")));  
 sensorList.add(s);  
 }  
 } else {  
  
 }  
 return sensorList;  
 }  
  
 @Override  
 public Sensor getSensorDetailsAccordingToID(String sensorId) throws Exception {  
  
 Sensor sensor = new Sensor();  
 *//Send Sensor Id to the Spring boot Api to get Sensor Details* URL url = new URL("http://localhost:8080/SensorController/getSensorDetailsAccordingToID/" + sensorId);  
  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
 connection.setRequestMethod("GET");  
 connection.setRequestProperty("Content-Type", "application/json; utf-8");  
 connection.setRequestProperty("Accept", "application/json");  
 int responseCode = connection.getResponseCode();  
 if (responseCode == HttpURLConnection.*HTTP\_OK*) { *// success* BufferedReader br = new BufferedReader(new InputStreamReader(  
 connection.getInputStream()));  
 String inputLine;  
 StringBuffer response = new StringBuffer();  
  
 if ((inputLine = br.readLine()) != null) {  
 response.append(inputLine);  
 }  
 br.close();  
  
 String result = response.toString();  
  
 JSONObject jobj = new JSONObject(result);  
  
 *//Convert json object data and Set it to sensor Object* sensor.setCo2Level(jobj.getInt("co2Level"));  
 sensor.setFloorNumber(jobj.getInt("floorNumber"));  
 sensor.setRoomNumber(jobj.getInt("roomNumber"));  
 sensor.setSensorId(String.*valueOf*(jobj.getString("sensorId")));  
 sensor.setSmokeLevel(jobj.getInt("smokeLevel"));  
 sensor.setStatus(String.*valueOf*(jobj.getString("status")));  
  
 } else {  
 }  
 return sensor;  
 }  
  
 @Override  
 public boolean addUser(User user) throws Exception {  
  
 *// Add User Details To the Json Object* JSONObject userDetails = new JSONObject();  
 userDetails.put("username", user.getUsername());  
 userDetails.put("email", user.getEmail());  
 userDetails.put("phoneNo", user.getPhoneNo());  
 userDetails.put("password", user.getPassword());  
 JSONObject userJsonObject = new JSONObject();  
 userJsonObject.put("User", userDetails);  
 *//Send JSON User Object to Spring boot Server* URL url = new URL("http://localhost:8080/userController/addUser");  
 HttpURLConnection connection = (HttpURLConnection) url.openConnection();  
 connection.setRequestMethod("POST");  
 connection.setRequestProperty("Content-Type", "application/json; utf-8");  
 connection.setRequestProperty("Accept", "application/json");  
 connection.setDoOutput(true);  
 String jsonInputString = userDetails.toString();  
  
 try (OutputStream os = connection.getOutputStream()) {  
 byte[] input = jsonInputString.getBytes("utf-8");  
 os.write(input, 0, input.length);  
 }  
 try (BufferedReader br = new BufferedReader(  
 new InputStreamReader(connection.getInputStream(), "utf-8"))) {  
 StringBuilder response = new StringBuilder();  
 String responseLine = null;  
 while ((responseLine = br.readLine()) != null) {  
 response.append(responseLine.trim());  
 }  
 }  
 connection.disconnect();  
 return true;  
 }  
  
 @Override  
 public boolean loginUser(String username, String password) throws Exception {  
  
 *// Encoded UserName And Password Using Base64* String encodedUserName = Base64.*getEncoder*().encodeToString(username.getBytes());  
 String encodedPassword = Base64.*getEncoder*().encodeToString(password.getBytes());  
  
 User user = new User();  
  
 *// Send encoded password and Username to the Spring Boot Api Using* URL obj = new URL("http://localhost:8080/userController/loginUser/" + encodedUserName + "/" + encodedPassword);  
 HttpURLConnection connection = (HttpURLConnection) obj.openConnection();  
 connection.setRequestMethod("GET");  
 connection.setRequestProperty("Content-Type", "application/json; utf-8");  
 connection.setRequestProperty("Accept", "application/json");  
 int responseCode = connection.getResponseCode();  
  
 if (responseCode == HttpURLConnection.*HTTP\_OK*) { *// success* BufferedReader br = new BufferedReader(new InputStreamReader(  
 connection.getInputStream()));  
 String inputLine;  
 StringBuffer response = new StringBuffer();  
  
 if ((inputLine = br.readLine()) != null) {  
 response.append(inputLine);  
 }  
 br.close();  
 String result = response.toString();  
 res = Boolean.*parseBoolean*(result);  
  
 } else {  
 System.*out*.println("Error Occured");  
  
 }  
 return res;  
 }  
  
}

### Sensor Controller (SensorController.java) – Package: lk.amc.controller

*/\*  
 \* 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 lk.amc.controller;  
  
import java.util.List;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
import lk.amc.ProxyHandler.ProxyHandler;  
import lk.amc.dto.Sensor;  
import lk.amc.dto.User;  
import lk.amc.service.SensorService;  
  
*/\*\*  
 \*  
 \* @author Dakshika  
 \*/*public class SensorController {  
  
 public SensorController() {  
  
 }  
  
 public static int increment() throws Exception {  
 SensorService sensorService = ProxyHandler.getInstance().getService();  
 return sensorService.increment();  
 }  
  
 public static String getLastId() throws Exception {  
 SensorService sensorService = ProxyHandler.getInstance().getService();  
 return sensorService.getLastId();  
 }  
  
 public static boolean addSensor(Sensor sensor) throws Exception {  
 SensorService sensorService = ProxyHandler.getInstance().getService();  
 return sensorService.addSensor(sensor);  
 }  
  
 public static List<Sensor> getAllSensorDetails() throws Exception {  
 SensorService sensorService = ProxyHandler.getInstance().getService();  
 return sensorService.getAllSensorDetails();  
 }  
  
 public static Sensor getSensorDetailsAccordingToID(String sensorId) throws Exception {  
 SensorService sensorService = ProxyHandler.getInstance().getService();  
  
 return sensorService.getSensorDetailsAccordingToID(sensorId);  
 }  
  
 public static boolean addUser(User user) throws Exception{  
  
 SensorService sensorService = ProxyHandler.getInstance().getService();  
  
 System.*out*.println("divya");  
 return sensorService.addUser(user);  
  
 }  
  
 public static boolean loginUser(String username,String password) throws Exception{  
  
 SensorService sensorService = ProxyHandler.getInstance().getService();  
  
 return sensorService.loginUser(username,password);  
  
 }  
}

### Proxy Handler (ProxyHandler.java) – Package: lk.amc.ProxyHandler

*/\*  
 \* 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 lk.amc.ProxyHandler;  
  
import java.rmi.Naming;  
import java.rmi.RemoteException;  
import lk.amc.sensor.impl.SensorServiceImpl;  
import lk.amc.service.SensorService;  
  
*/\*\*  
 \*  
 \* @author Dakshika  
 \*/*public class ProxyHandler {  
  
 private static ProxyHandler *proxyHandler*;  
 private SensorService sensorService;  
  
  
 public static ProxyHandler getInstance()throws Exception{  
 if(*proxyHandler*==null){  
 *proxyHandler*= new ProxyHandler();  
 }  
 return *proxyHandler*;  
 }  
  
 private ProxyHandler() throws Exception{  
 sensorService =(SensorService) Naming.*lookup*("rmi://localhost:5050/sensor");  
 }  
  
 public SensorService getService() throws RemoteException{  
  
 if(sensorService==null){  
 sensorService = new SensorServiceImpl();  
 }  
 return sensorService;  
 }  
  
  
}

### Model Class (Sensor.java) – Package: lk.amc.dto

*/\*  
 \* 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 lk.amc.dto;  
  
import java.io.Serializable;  
  
*/\*\*  
 \*  
 \* @author Dakshika  
 \*/*public class Sensor implements Serializable {  
  
 private String sensorId;  
 private int floorNumber;  
 private int roomNumber;  
 private int smokeLevel;  
 private int co2Level;  
 private String status;  
  
 public Sensor() {  
 }  
  
 public Sensor(String sensorId, int floorNumber, int roomNumber, int smokeLevel, int co2Level, String status) {  
 this.sensorId = sensorId;  
 this.floorNumber = floorNumber;  
 this.roomNumber = roomNumber;  
 this.smokeLevel = smokeLevel;  
 this.co2Level = co2Level;  
 this.status = status;  
 }  
  
 */\*\*  
 \* @return the sensorId  
 \*/* public String getSensorId() {  
 return sensorId;  
 }  
  
 */\*\*  
 \* @param sensorId the sensorId to set  
 \*/* public void setSensorId(String sensorId) {  
 this.sensorId = sensorId;  
 }  
  
 */\*\*  
 \* @return the floorNumber  
 \*/* public int getFloorNumber() {  
 return floorNumber;  
 }  
  
 */\*\*  
 \* @param floorNumber the floorNumber to set  
 \*/* public void setFloorNumber(int floorNumber) {  
 this.floorNumber = floorNumber;  
 }  
  
 */\*\*  
 \* @return the roomNumber  
 \*/* public int getRoomNumber() {  
 return roomNumber;  
 }  
  
 */\*\*  
 \* @param roomNumber the roomNumber to set  
 \*/* public void setRoomNumber(int roomNumber) {  
 this.roomNumber = roomNumber;  
 }  
  
 */\*\*  
 \* @return the smokeLevel  
 \*/* public int getSmokeLevel() {  
 return smokeLevel;  
 }  
  
 */\*\*  
 \* @param smokeLevel the smokeLevel to set  
 \*/* public void setSmokeLevel(int smokeLevel) {  
 this.smokeLevel = smokeLevel;  
 }  
  
 */\*\*  
 \* @return the co2Level  
 \*/* public int getCo2Level() {  
 return co2Level;  
 }  
  
 */\*\*  
 \* @param co2Level the co2Level to set  
 \*/* public void setCo2Level(int co2Level) {  
 this.co2Level = co2Level;  
 }  
  
 */\*\*  
 \* @return the status  
 \*/* public String getStatus() {  
 return status;  
 }  
  
 */\*\*  
 \* @param status the status to set  
 \*/* public void setStatus(String status) {  
 this.status = status;  
 }  
  
  
  
}

### Model Class (User.java) – Package: lk.amc.dto

*/\*  
 \* 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 lk.amc.dto;  
  
import java.io.Serializable;  
  
*/\*\*  
 \*  
 \* @author Ruvini  
 \*/*public class User implements Serializable{  
 private String id;  
 private String username;  
 private String email;  
 private int phoneNo;  
 private String password;  
  
 public User(){  
 }  
  
 public User(String username,String email,int phoneNo,String password){  
 this.username=username;  
 this.email=email;  
 this.phoneNo=phoneNo;  
 this.password=password;  
 }  
  
 */\*\*  
 \* @return the id  
 \*/* public String getId() {  
 return id;  
 }  
  
 */\*\*  
 \* @param id the id to set  
 \*/* public void setId(String id) {  
 this.id = id;  
 }  
  
 */\*\*  
 \* @return the username  
 \*/* public String getUsername() {  
 return username;  
 }  
  
 */\*\*  
 \* @param username the username to set  
 \*/* public void setUsername(String username) {  
 this.username = username;  
 }  
  
 */\*\*  
 \* @return the email  
 \*/* public String getEmail() {  
 return email;  
 }  
  
 */\*\*  
 \* @param email the email to set  
 \*/* public void setEmail(String email) {  
 this.email = email;  
 }  
  
 */\*\*  
 \* @return the phoneNo  
 \*/* public int getPhoneNo() {  
 return phoneNo;  
 }  
  
 */\*\*  
 \* @param phoneNo the phoneNo to set  
 \*/* public void setPhoneNo(int phoneNo) {  
 this.phoneNo = phoneNo;  
 }  
  
 */\*\*  
 \* @return the password  
 \*/* public String getPassword() {  
 return password;  
 }  
  
 */\*\*  
 \* @param password the password to set  
 \*/* public void setPassword(String password) {  
 this.password = password;  
 }  
  
}