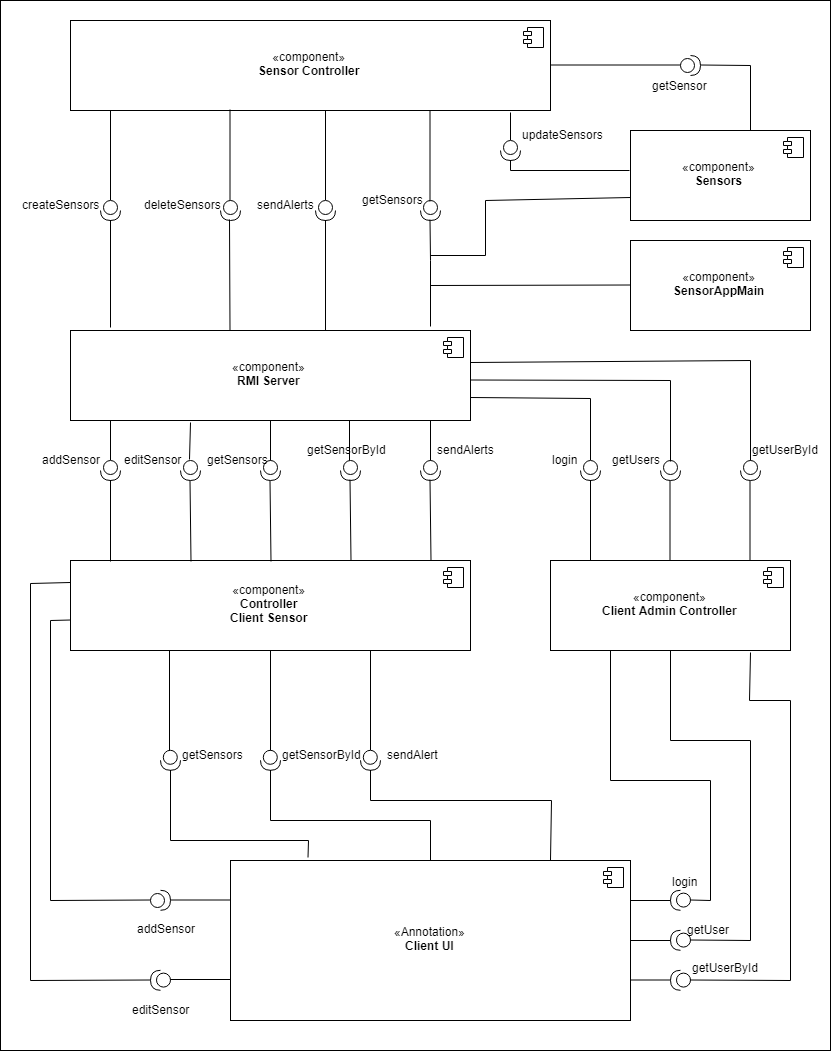
Alarm Monitoring System –

The following report is based on an Alarm Monitoring Client Application which was developed using Restful Web Services API. First of all, defining what the requirements of this application were, the main requirements were to develop a Web Client Application which showcases the status of the existing fire alarm sensors. The data that should be displayed were the functional status of the sensor which determines if it’s active or not, the location of the sensor which can be shown using the floor number and room number. Furthermore, an important detail that should be included was the CO2 smoke level which was needed to be ranked from levels 1-10 with 10 being the highest level of CO2. Also, another functionality that was required to be implemented was to identify the sensor data with CO2 levels higher than 5 and to highlight them. Moreover, another requirement was that sensor data should be updated to make sure the data is current and accurate to the present moment. Apart from the Web Client, a Desktop client was needed to be implemented as well which fulfilled the same above-mentioned requirements with the refresh of data as well. Additionally, features such as administrator logins, adding and editing sensor data, notification via email and text for high CO2 or smoke levels were also needed to be implemented.

Moving on to the implementation of the above requirements, for the implementation of the Asynchronous Web Client, the JavaScript framework React was used while using Java Swing for the Desktop Application and SpringBoot for the REST API.

Component Diagram for the system



Software System and Architecture used in the system –

Developed Alarm Monitoring system has four main components. The server ( The REST API – AlarmMonitoringSystem ) is the most important component of the system. All the other Components are communicating through this centralized server. The underlining system architecture of the developed system is the traditional Client-Server architecture. There are two web-clients ( sensor-app-client and the alarm-monitoring-client ) and one desktop client in the system. All these clients are referring to the interfaces provided by the REST API. Furthermore, all the processing of data resides in the client side and because of this the flat-client approach has been used to develop this system. Layered Architecture is the software architecture used in this system.

Controllers -

The main aspect of this system which is the connection to the REST API happens through the controllers in the Alarm Monitoring System. This has two controllers called SensorController and UserController. Describing the SensorController, this is used to get all sensors. Further explaining this, the SensorController has the requests that are sent in order to get the response. GetMapping is used to find all the sensors as well as using the ID to get one Sensor. PostMapping is used for creating a Sensor while PutMapping is used to Update a Sensor using its ID. For the deletion of a Sensor DeleteMapping is used while finally in the Sensor Controller PostMapping is used to send the Alert with the information on the Sensor such as Sensor Id, Sensor Floor Number, Sensor Room Number, the Responsible Person through a Text message or an Email.

When describing the REST functionalities further, in this program in order to get all the Sensors, the Web Clients and the Desktop application needs to connect to the REST API using the sensors() function. The URL that should be used for this is "/sensors" which is a GET request. When you use this URL and send a Get request, all the Sensors will be available. The response of this request would be an array filled with existing sensor objects which would be displayed in the interface of the Desktop or Web Client. When describing the Sensor object which is mentioned above, this includes the attributes that are defined in the Sensor Model class which are id, floorNo, roomNo, smokeLevel, co2Level and status. Briefly describing the attributes that are included in the Sensor Model, every sensor has a unique ID which helps identify the sensor. The Floor Number mentioned here, is used to point to the physical location of the sensor which could be in any floor of the building while the room number does the same with regards to locate the sensor. Both these attributes are needed in order to identify where the Fire is happening in case of a situation. The attributes Smoke Level and CO2 Level refer to the current levels of Smoke and CO2 surrounding the Sensor which is the most vital data as this is what determines whether a Fire is happening or not. Finally, the attribute status is defined in order to categorize whether a sensor is activated or not.

Furthermore, if only one sensor is needed, the function getSensor() is called which passes the ID of the Sensor through the URL ("/sensor/{id}") which is also a GET request. When this happens, REST would return the sensor that belongs to the specified ID in the request.

Moreover, to create a new Sensor, the createSensor() function is used which is a POST request. This PostMapping request expects a Sensor Object as the request. This could be called by using the URL "/sensor" which triggers the createSensor() function. The information of the sensor which should be included according to the Sensor model which was discussed above, should be sent with the POST request in order to Create a new sensor using the REST API. The return response would be the values of the newly created sensor which can be used to identify Smoke and CO2 levels.

Moreover, the functionality of Updating a sensor is enabled by the PutMapping. This request uses the sensor ID in order to manipulate the existing data on that sensor with the URL "/sensor/{id}"). Here a JSON Sensor Object is expected as the request. Additionally, the reason CrossOrigin is used is to make sure all origins are allowed which is important as both the Sensor-app-client and the alarm monitoring client are both used.

Also, the requirement of deleting a sensor is implemented by DeleteMapping which uses the deleteSensor() function. This uses the sensor ID in order to identify which sensor is required to be deleted and uses the sensorRepository.deleteById(id) method to complete the task.

Finally, on the aspect of the Sensor Controller functions, Sending Alerts functionality is implemented. PostMapping is used in order to achieve this with the URL ("/sensor/sendAlert"). This uses the sendAlert() function which takes in the sensor object and uses its values in order to send the alert. Briefly describing this alert system, there are two types of alerts enabled by this Alarm Monitoring System. These are SMS Alerts and Email Alerts. By having both these communication types in gives a higher probability of the user getting the real time critical information as soon as possible. Furthermore, describing the content of this alerts, the major details include the Sensor ID of the affected Sensor, Floor Number, Room Number and the contact details of the person responsible. For SMS alerts, the Phone number of the responsible person is required while for Email alerts, the Email address of the responsible person is required. This concludes the description on how the Sensor Controller manages and interacts with the REST API and handles all the functions.

Moving on to the User Controller, this is very similar to the implementation of the Sensor Controller. Further explaining this, the UserController has the requests that are sent in order to get the response. GetMapping is used to find all the Users as well as using the User ID to get one User response. PostMapping is used for creating a User while PutMapping is used to Update User details using the user ID. In order to delete a User, DeleteMapping is used.

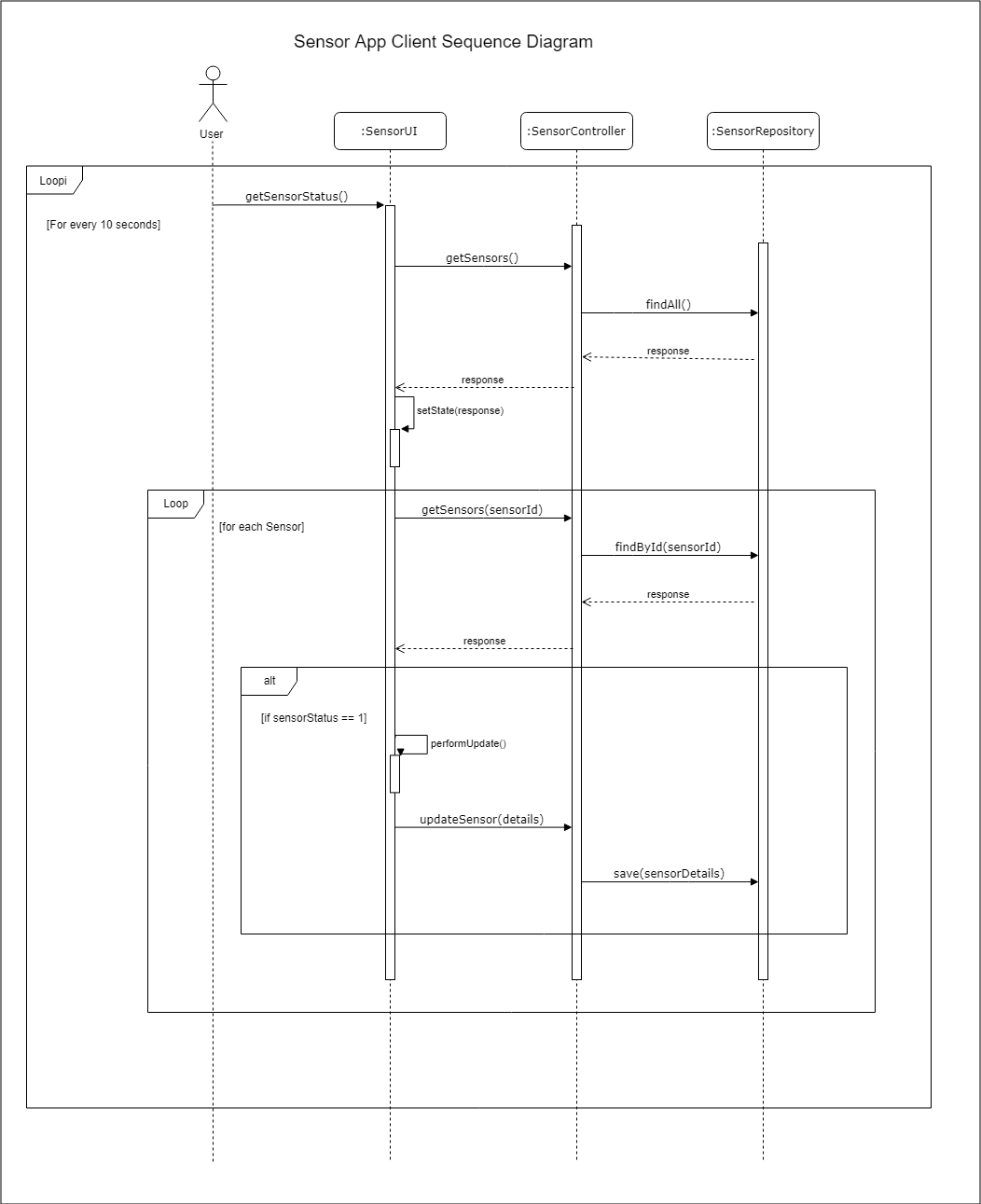
When describing the REST functionalities of the User Controller further, in this program in order to get all the Sensors, the Web Clients and the Desktop application needs to connect to the REST API using the users() function. The URL that should be used for this is "/ users" which is a GET request. When you use this URL and send a Get request, all the Users of this system will be available. The response of this request would be an array filled with existing user objects which would be displayed in the interface of the Desktop or Web Client. When describing the User object which is mentioned above, this includes the attributes that are defined in the User Model class which are id, name, userName, password, email and mobile. Briefly describing the attributes that are included in the User Model, every User has a unique ID which helps identify the individual User. The Name attribute is used to store the Name of the user which comes in handy when visualizing the data while Username is required in order for the User to login to the system. The password is used in order to verify the user’s authenticity and legitimacy while the Email address and Mobile Number are stored as they are needed in order to contact the user in case of an emergency and send alerts. These are the information that are stored in the User Model that was mentioned above.

Furthermore, if only one User is needed the function getUser () is called which passes the ID of the User through the URL ("/user/{id}") which is also a GET request. When this happens, REST would return the User that belongs to the specified ID in the request.

Furthermore, to create a new User, createUser() method is used which is a POST request. This PostMapping request expects a User Object as the request. This could be called by using the URL "/user" which triggers the createUser() function. The information of the User which should be included according to the User model which was discussed above, should be sent with the POST request in order to Create a new User using the REST API. The return response would be the values of the newly created User which are the Name, Username, Password, email and mobile.

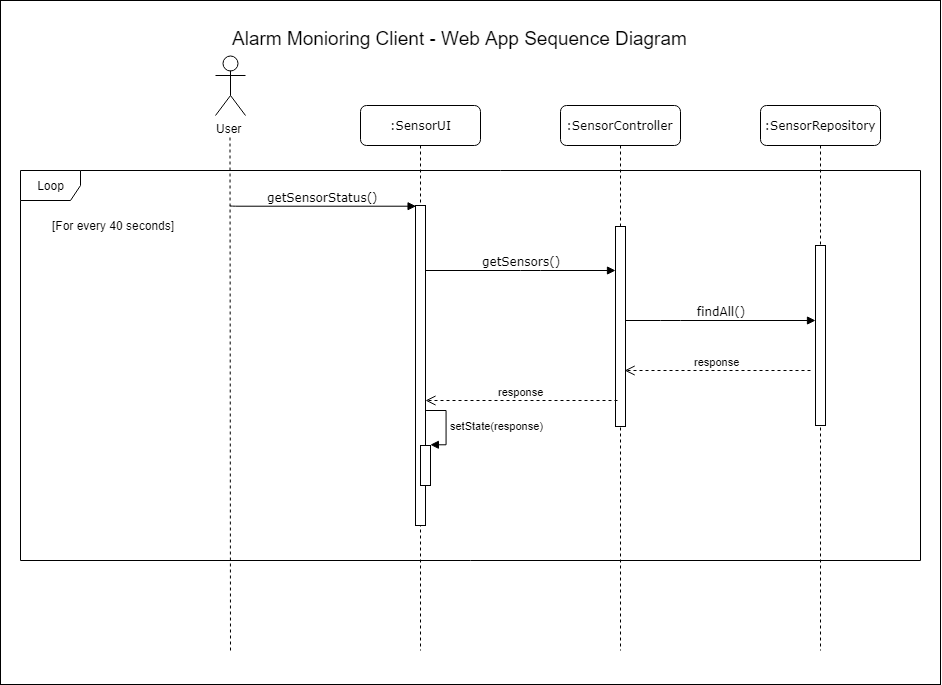
Moreover, the functionality of Updating a User is enabled by PutMapping. This request uses the User ID in order to manipulate the existing data on that User triggering the URL ("/user/{id}"). Here a JSON User Object is expected as the request. Finally, the requirement of deleting a User is implemented by DeleteMapping which uses the deleteUser() function. This uses the user ID in order to identify which user is required to be deleted and completes the task. This concludes the description on how the User Controller manages and interacts with the REST API and handles all the functions.

Sensor-App-Client



The sensor-app client is used to update the state of each sensor periodically. This application acts as a dummy in the implemented system. Fetch API has used to access and manipulate the REST Server in this sensor-app-client. By using the setInterval() method in JavaScript, the current state of all the sensors are received periodically. This is done by calling the API method ‘’getSensors’’ using the URL – <http://localhost:8080/api/sensors>. This makes the server return an array of the Sensor objects which include all the data that is available for a sensor according to the sensor model. Once the response from the server is received, the map() function is used to update the CO2 and Smoke levels randomly. In order to achieve this Math.randon() function is used. After each modification, the API method updateSensor using the URL - “[http://localhost:8080/api/{id}](http://localhost:8080/api/%7bid%7d)” is called while the ID specified in this can be derived from the Sensor object data. Furthermore, in this application, different functions are used in order to implement and manage the requirements of this application and have a clear functional separation. In order to get the current state of a sensor the function getSensorState() is called and to perform random modifications the function performDummyUpdate is used.

Alarm-Monitoring-Client



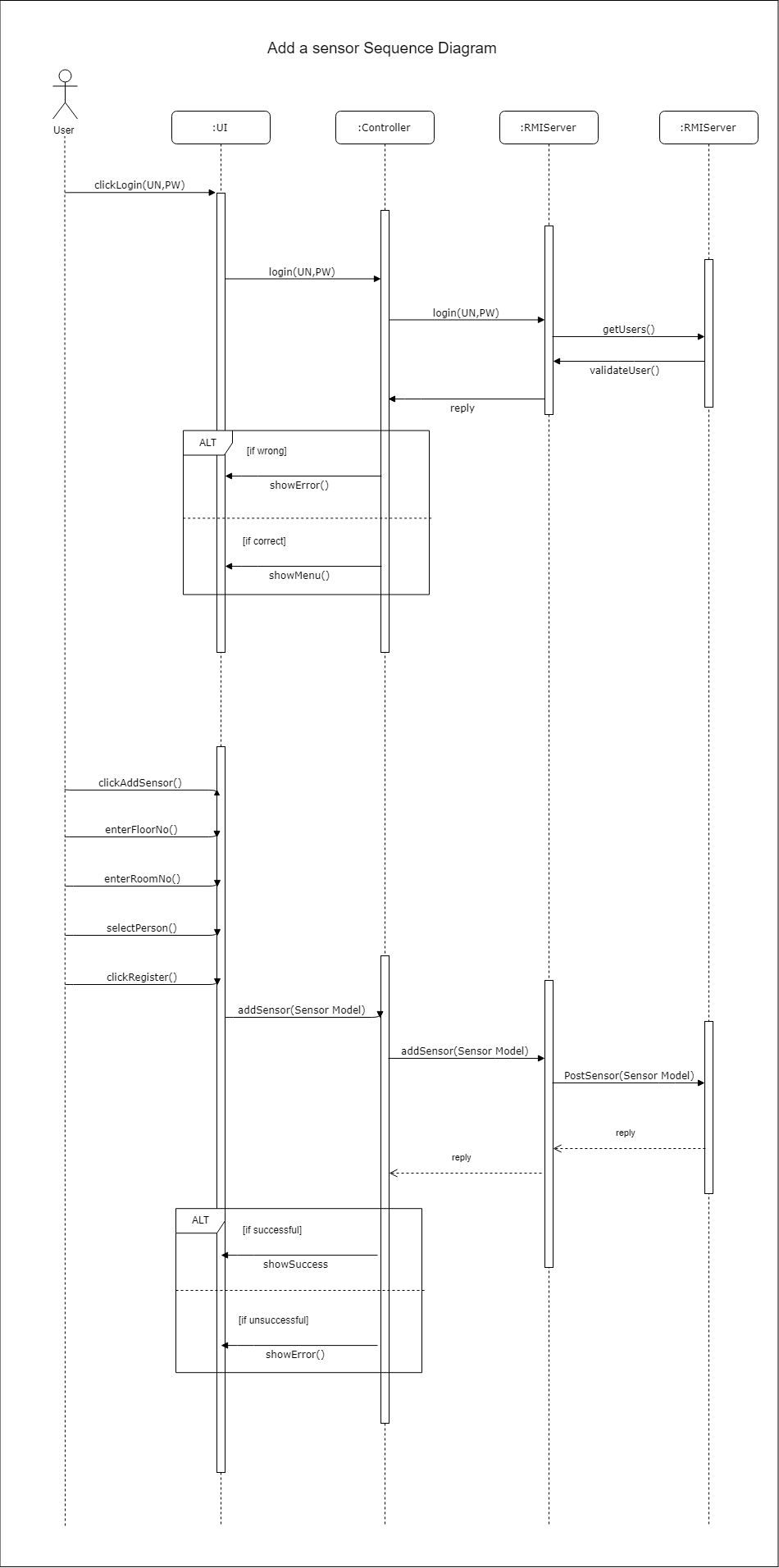
The Alarm-monitoring client is used to monitor the state of each sensor periodically. In this the SensorAppMain has a method called getSensorStare which calls the getAll() method and fetches the data using the URL <http://localhost:8080/api/sensors>. Fetch API has used to access and manipulate the REST Server in this as well while by using the setInterval() method in JavaScript, the current state of all the sensors are received periodically.This makes the server return an array of the Sensor objects which include all the data that is available for a sensor according to the sensor model. Once the response from the server is received, the Alarm Monitoring client can display the received data in the interface. This is done using the rendor() method which checks whether the sensor data is still loading and if not displays the data. Sensors.map is used to map the sensor data and return the sensor id, status, co2 level, floor number, room number, smoke level and the key which is also the sensor id.

Desktop client

Desktop client is used to add sensors, save the sensors and all its data, edit sensors and delete sensors as well as view them. The desktop client goes through the RMI server and uses the RESI API in order to save the data in the database. Furthermore, the Desktop client fetches the Sensor data from the database which includes the CO2 and Smoke levels at the present moment. If the CO2 or Smoke levels of the fetched data is higher than 5, the desktop client is the one who sends the request to send the alerts to the Alarm Monitoring System. The desktop clients also accurately shows the fluctuating CO2 and Smoke levels from the sensor which is vital as the user has a visual representation of all the sensor data at the present moment. The desktop client also has the feature login which is important when it comes to the security aspect of this system. Login is interconnected with the user model which has user data such has ID, Name, Username, Password, Email and Mobile.

First describing the views that are implemented by the Desktop client, namely these are Add Sensors view, Login view, Main view and View Sensors view. Explaining very briefly these views, the ‘Add Sensors’ view is used in order to provide the user access to add a sensor to the system. ‘Login’ view is used to enable the user to login to the system and access the sensor data while the ‘Main’ view showcases the sensor data and gives the user the proper interface to interact with the system. Finally, ‘View Sensors’ enables the user to view all the sensors that are in the system at the present moment.

Another main component of the desktop client is the Sensor Server. This is used to add sensors, edit sensors, get all sensors, get sensor by id, login, get all users, get user by id and send alerts.



Overall, when discussing about the authentication and security aspect of this application, the major security feature is that a Login function has been implemented in the Desktop client and is properly functional. This makes sure only authenticated users can access this system and view its data and also helps prevent misuse of this data. Furthermore, when discussing the importance of having proper security and authentication, this is extremely vital as the data should be secured and not accessible to unknown users.

In conclusion, this report describes the main components of the Alarm Monitoring System and individually explains and describes the functionality of each component.

Appendix A

Alarm Monitoring Client

Sensor.js

import React, { Component } from "react";

import {Bar} from "react-chartjs-2";

class Sensor extends Component {

constructor(props) {

super(props);

let co2RGBColor = "rgba(98, 182, 239,0.4)";

let smokeRGBColor = "rgba(98, 182, 239,0.4)";

if (props.smokeLevel > 5){

smokeRGBColor = "rgba(255, 134,159,0.4)";

}

if(props.co2Level > 5){

co2RGBColor = "rgba(255, 134,159,0.4)";

}

this.state = {

id: props.id,

roomNo: props.roomNo,

floorNo: props.floorNo,

status: props.status,

chartData: {

labels: ["Smoke Level", "CO2 Level"],

datasets: [{

data: [props.smokeLevel, props.co2Level],

backgroundColor: [

smokeRGBColor,

co2RGBColor,

]

}]

}

}

}

componentWillReceiveProps(nextProps) {

let co2RGBColor = "rgba(98, 182, 239,0.4)";

let smokeRGBColor = "rgba(98, 182, 239,0.4)";

if (nextProps.smokeLevel > 5){

smokeRGBColor = "rgba(255, 134,159,0.4)";

}

if(nextProps.co2Level > 5){

co2RGBColor = "rgba(255, 134,159,0.4)";

}

this.setState({

id: nextProps.id,

roomNo: nextProps.roomNo,

floorNo: nextProps.floorNo,

status: nextProps.status,

chartData: {

labels: ["Smoke Level", "CO2 Level"],

datasets: [{

data: [nextProps.smokeLevel, nextProps.co2Level],

backgroundColor: [

smokeRGBColor,

co2RGBColor,

]

}]

}

});

}

render() {

return (

<div>

<h5 className="mt-5" style={{fontWeight: "bold"}}>SENSOR ID : {this.props.id}</h5>

<span className="mt-1 mr-3" style={{fontWeight: "bold"}}>FLOOR NO : {this.state.floorNo}</span>

<span className="mt-1 mr-3" style={{fontWeight: "bold"}}>ROOM NO : {this.state.roomNo}</span>

<span className="mt-1 mb-5" style={{fontWeight: "bold"}}>SENSOR STATUS : <span style={this.props.status === 1 ? {color: 'green', fontWeight: "bold"} : {color: 'red', fontWeight: "bold"}}>{this.props.status === 1 ? 'ACTIVE' : 'INACTIVE'}</span></span>

<br/>

<br/>

<div className="chart">

<Bar

data={this.state.chartData}

options={{

legend: {

display: false

},

scales: {

xAxes: [

{

barPercentage: 1,

gridLines: {

display: true,

color: "rgba(0, 0, 0, 0.1)"

}

}

],

yAxes: [

{

gridLines: {

display: true,

color: "rgba(0, 0, 0, 0.1)"

},

ticks: {

beginAtZero: true

}

}

]

}

}}

/>

</div>

</div>

);

}

}

export default Sensor;

SensorAppMain.js

import React, {Component} from "react";

import Sensor from './Sensor';

import { MDBContainer, MDBRow, MDBCol } from "mdbreact";

class SensorAppMain extends Component {

constructor(props) {

super(props);

this.state = {

isLoading: true,

sensors: []

}

}

getSensorState() {

fetch('http://localhost:8080/api/sensors')

.then(response => {

return response.json();

})

.then(result => {

this.setState({isLoading: false, sensors: result});

})

.catch(error => {

console.log(error)

});

}

componentDidMount() {

this.getSensorState();

this.interval = setInterval(() => {

this.getSensorState();

}, 5000);

}

render() {

const {isLoading, sensors} = this.state;

if (isLoading)

return <div>Loading ...</div>

else {

return (

<MDBContainer>

<h2>Sensors</h2>

<MDBRow>

<MDBCol sm="12" md="12" lg="6" xl="6">

{

sensors.map((sensor, i) => {

if(i%2 === 0){

return (

<Sensor id={sensor.id} status={sensor.status} co2Level={sensor.co2Level}

floorNo={sensor.floorNo} roomNo={sensor.roomNo} smokeLevel={sensor.smokeLevel} key={sensor.id}/>

);

}

})

}

</MDBCol>

<MDBCol sm="12" md="12" lg="6" xl="6">

{

sensors.map((sensor, i) => {

if(i%2 === 1){

return (

<Sensor id={sensor.id} status={sensor.status} co2Level={sensor.co2Level}

floorNo={sensor.floorNo} roomNo={sensor.roomNo} smokeLevel={sensor.smokeLevel} key={sensor.id}/>

);

}

})

}

</MDBCol>

</MDBRow>

</MDBContainer>

);

}

}

}

export default SensorAppMain;

Appendix B

Sensor App Client

Sensor.js

import React, { Component } from "react";

import { MDBBtn, MDBCard, MDBCardBody, MDBCardTitle, MDBCardText} from 'mdbreact';

class Sensor extends Component {

constructor(props) {

super(props);

this.state = {

co2Level : props.co2Level,

smokeLevel: props.smokeLevel

}

}

componentWillReceiveProps(nextProps) {

this.setState({

co2Level : nextProps.co2Level,

smokeLevel: nextProps.smokeLevel

})

}

render() {

return (

<div className="mt-5">

<MDBCard>

<MDBCardBody>

<MDBCardTitle>SENSOR ID : {this.props.id}</MDBCardTitle>

<MDBCardText>

CO2 LEVEL : {this.state.co2Level} {' || '}

SMOKE LEVEL : {this.state.smokeLevel}

</MDBCardText>

</MDBCardBody>

</MDBCard>

</div>

);

}

}

export default Sensor;

Sensors.js

import React, {Component} from "react";

import Sensor from './Sensor';

import { MDBContainer, MDBRow, MDBCol } from "mdbreact";

class Sensors extends Component {

constructor(props) {

super(props);

this.state = {

isLoading: true,

sensors: []

}

}

getSensorState() {

fetch('http://localhost:8080/api/sensors')

.then(response => {

return response.json();

})

.then(result => {

this.setState({isLoading: false, sensors: result});

})

.catch(error => {

console.log(error)

});

}

updateSensorState(id, updatedState){

fetch(`http://localhost:8080/api/sensor/${id}`, {

method: 'PUT',

headers : {

'Accept' : 'application/json',

'Content-Type' : 'application/json'

},

body: JSON.stringify(updatedState)

});

}

preformDummyUpdate(){

this.getSensorState();

this.state.sensors.map(sensor => {

let updation = sensor;

if(updation.status === 1){

updation.co2Level = Math.ceil(Math.random() \* Math.floor(10));

updation.smokeLevel = Math.ceil(Math.random() \* Math.floor(10));

this.updateSensorState(sensor.id, updation);

}

});

}

componentDidMount() {

this.interval = setInterval(() => {

this.preformDummyUpdate();

}, 10000);

}

render() {

const {isLoading, sensors} = this.state;

if (isLoading)

return <div>Loading ...</div>

else {

return (

<MDBContainer>

<h2>Sensors</h2>

<MDBRow>

<MDBCol sm="12" md="12" lg="6" xl="6">

{

sensors.map((sensor, i) => {

if(i%2 === 0){

return (

<Sensor id={sensor.id} status={sensor.status} co2Level={sensor.co2Level}

floorNo={sensor.floorNo} roomNo={sensor.roomNo} smokeLevel={sensor.smokeLevel} key={sensor.id}/>

);

}

})

}

</MDBCol>

<MDBCol sm="12" md="12" lg="6" xl="6">

{

sensors.map((sensor, i) => {

if(i%2 === 1){

return (

<Sensor id={sensor.id} status={sensor.status} co2Level={sensor.co2Level}

floorNo={sensor.floorNo} roomNo={sensor.roomNo} smokeLevel={sensor.smokeLevel} key={sensor.id}/>

);

}

})

}

</MDBCol>

</MDBRow>

</MDBContainer>

);

}

}

}

export default Sensors;

Appendix C

Sensor App Client

AlarmMonitoringSystemApplication.java

package com.project.AlarmMonitoringSystem;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import org.springframework.remoting.rmi.RmiServiceExporter;

@SpringBootApplication

public class AlarmMonitoringSystemApplication {

public static void main(String[] args) {

SpringApplication.run(AlarmMonitoringSystemApplication.class, args);

}

}

AlarmMonitoringSystemApplication.java

package com.project.AlarmMonitoringSystem.controllers;

import java.net.URI;

import java.net.URISyntaxException;

import java.util.Collection;

import java.util.Optional;

import javax.validation.Valid;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.project.AlarmMonitoringSystem.models.Sensor;

import com.project.AlarmMonitoringSystem.repositories.SensorRepository;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/api")

public class SensorController {

private SensorRepository sensorRepository;

public SensorController(SensorRepository sensorRepository) {

super();

this.sensorRepository = sensorRepository;

}

@GetMapping("/sensors")

@CrossOrigin(origins = {"http://localhost:3000", "http://localhost:3001"})

Collection<Sensor> sensors(){

return sensorRepository.findAll();

}

@GetMapping("/sensor/{id}")

ResponseEntity<?> getSensor(@PathVariable Long id){

Optional<Sensor> sensor = sensorRepository.findById(id);

return sensor.map(response -> ResponseEntity.ok().body(response))

.orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

@PostMapping("/sensor")

ResponseEntity<Sensor> createSensor(@Valid @RequestBody Sensor sensor) throws URISyntaxException{

Sensor result = sensorRepository.save(sensor);

return ResponseEntity.created(new URI("/api/sensor" + result.getId())).body(result);

}

@PutMapping("/sensor/{id}")

@CrossOrigin(origins = {"http://localhost:3000", "http://localhost:3001"})

ResponseEntity<Sensor> updateSensor(@Valid @RequestBody Sensor sensor){

Sensor result = sensorRepository.save(sensor);

return ResponseEntity.ok().body(result);

}

@DeleteMapping("/sensor/{id}")

ResponseEntity<?> deleteSensor(@PathVariable Long id){

sensorRepository.deleteById(id);

return ResponseEntity.ok().build();

}

@PostMapping("/sensor/sendAlert")

void sendAlert(@Valid @RequestBody Sensor sensor) throws URISyntaxException{

System.out.println("=============SENDING SMS=============");

System.out.println("ALERT!\nSensor : " + sensor.getId()

+ "\nFloor No : "+ sensor.getFloorNo()

+ "\nRoom No " + sensor.getRoomNo()

+ "\nMessage to : " + sensor.getResponsiblePerson().getMobile());

System.out.println("=============SENDING EMAIL=============");

System.out.println("ALERT!\nSensor : " + sensor.getId()

+ "\nFloor No : "+ sensor.getFloorNo()

+ "\nRoom No " + sensor.getRoomNo()

+ "\nMessage to : " + sensor.getResponsiblePerson().getEmail());

}

}

UserController.java

package com.project.AlarmMonitoringSystem.controllers;

import java.net.URI;

import java.net.URISyntaxException;

import java.util.Collection;

import java.util.Optional;

import javax.validation.Valid;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.project.AlarmMonitoringSystem.models.Sensor;

import com.project.AlarmMonitoringSystem.models.User;

import com.project.AlarmMonitoringSystem.repositories.SensorRepository;

import com.project.AlarmMonitoringSystem.repositories.UserRepository;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/api")

public class UserController {

private UserRepository userRepository;

public UserController(UserRepository userRepository) {

super();

this.userRepository = userRepository;

}

@GetMapping("/users")

Collection<User> users(){

return userRepository.findAll();

}

@GetMapping("/user/{id}")

ResponseEntity<?> getUser(@PathVariable Long id){

Optional<User> user = userRepository.findById(id);

return user.map(response -> ResponseEntity.ok().body(response))

.orElse(new ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

@PostMapping("/user")

ResponseEntity<User> createUser(@Valid @RequestBody User user) throws URISyntaxException{

User result = userRepository.save(user);

return ResponseEntity.created(new URI("/api/user" + result.getId())).body(result);

}

@PutMapping("/user/{id}")

ResponseEntity<User> updateUser(@Valid @RequestBody User user){

User result = userRepository.save(user);

return ResponseEntity.ok().body(result);

}

@DeleteMapping("/user/{id}")

ResponseEntity<?> deleteUser(@PathVariable Long id){

userRepository.deleteById(id);

return ResponseEntity.ok().build();

}

}

Sensor.java

package com.project.AlarmMonitoringSystem.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.ManyToOne;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@NoArgsConstructor

@AllArgsConstructor

@Entity

@Data

@Table(name = "sensor")

public class Sensor {

@Id @GeneratedValue(strategy=GenerationType.IDENTITY)

private Long id;

private int floorNo;

private int roomNo;

private int smokeLevel;

private int co2Level;

private int status;

@ManyToOne

User addedBy;

@ManyToOne

User responsiblePerson;

}

User.java

package com.project.AlarmMonitoringSystem.models;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.AllArgsConstructor;

import lombok.Data;

import lombok.NoArgsConstructor;

@NoArgsConstructor

@AllArgsConstructor

@Entity

@Data

@Table(name = "user")

public class User {

@Id @GeneratedValue(strategy=GenerationType.IDENTITY)

private Long id;

private String name;

private String userName;

private String password;

private String email;

private String mobile;

}

SensorRepository.java

package com.project.AlarmMonitoringSystem.repositories;

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

import com.project.AlarmMonitoringSystem.models.Sensor;

public interface SensorRepository extends JpaRepository<Sensor, Long> {

}

UserRepositoru.java

package com.project.AlarmMonitoringSystem.repositories;

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

import com.project.AlarmMonitoringSystem.models.User;

public interface UserRepository extends JpaRepository<User, Long> {

}

Appendix D

Desktop Client

AdminController.java

package com.dc.controller;

import com.dc.model.SensorModel;

import com.dc.model.UserModel;

import com.dc.server.ProxyHandler;

import com.dc.server.SensorService;

import java.util.ArrayList;

public class AdminController {

//method to invoke login method in server side

public static boolean login(UserModel um) throws Exception {

SensorService service = ProxyHandler.getInstance().getService();

return service.login(um);

}

//method to invoke get all users method

public static ArrayList<UserModel> getUsers() throws Exception {

SensorService service = ProxyHandler.getInstance().getService();

return service.getUsers();

}

//method to invoke get user by id method

public static UserModel userById(int id) throws Exception {

SensorService service = ProxyHandler.getInstance().getService();

return service.getUserById(id);

}

}

SensorController.java

package com.dc.controller;

import com.dc.model.SensorModel;

import com.dc.model.UserModel;

import com.dc.server.ProxyHandler;

import com.dc.server.SensorService;

import java.util.ArrayList;

import java.util.Timer;

import java.util.TimerTask;

public class SensorController {

//method to invoke server side add sensor method

public static boolean add(SensorModel se, UserModel um) throws Exception {

SensorService service = ProxyHandler.getInstance().getService();

return service.addSensor(se, um);

}

//method to invoke server side edit sensor method

public static boolean edit(SensorModel se, UserModel um) throws Exception {

SensorService service = ProxyHandler.getInstance().getService();

return service.editSensor(se, um);

}

//method to invoke server side get all sensors method

public static ArrayList<SensorModel> get() throws Exception {

SensorService service = ProxyHandler.getInstance().getService();

return service.getSensors();

}

//method to invoke server side get sensor by id method

public static SensorModel getById(String id) throws Exception {

SensorService service = ProxyHandler.getInstance().getService();

return service.getSensorById(id);

}

//method to invoke server side send alert method

public static boolean sendAlert(SensorModel sm, UserModel um) throws Exception {

SensorService service = ProxyHandler.getInstance().getService();

return service.sendAlert(sm, um);

}

}

SensorModel.java

package com.dc.model;

import java.io.Serializable;

public class SensorModel implements Serializable {

private int id;

private String floorNo;

private String roomNo;

private int smokeLevel;

private int coLevel;

private int status;

private int resId;

private String resName;

private UserModel um;

public SensorModel() {

}

public SensorModel(String floorNo, String roomNo, int status) {

this.floorNo = floorNo;

this.roomNo = roomNo;

this.status = status;

}

public String getFloorNo() {

return floorNo;

}

public void setFloorNo(String floorNo) {

this.floorNo = floorNo;

}

public String getRoomNo() {

return roomNo;

}

public void setRoomNo(String roomNo) {

this.roomNo = roomNo;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getSmokeLevel() {

return smokeLevel;

}

public void setSmokeLevel(int smokeLevel) {

this.smokeLevel = smokeLevel;

}

public int getCoLevel() {

return coLevel;

}

public void setCoLevel(int coLevel) {

this.coLevel = coLevel;

}

public int getStatus() {

return status;

}

public void setStatus(int status) {

this.status = status;

}

public int getResId() {

return resId;

}

public void setResId(int resId) {

this.resId = resId;

}

public String getResName() {

return resName;

}

public void setResName(String resName) {

this.resName = resName;

}

public UserModel getUm() {

return um;

}

public void setUm(UserModel um) {

this.um = um;

}

}

UserModel.java

package com.dc.model;

import java.io.Serializable;

public class UserModel implements Serializable{

private int id;

private String name;

private String username;

private String password;

private String email;

private String mobile;

public UserModel() {

}

public UserModel(String username, String password) {

this.username = username;

this.password = password;

}

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 getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getMobile() {

return mobile;

}

public void setMobile(String mobile) {

this.mobile = mobile;

}

}

ProxyHandler.java

package com.dc.server;

import java.net.MalformedURLException;

import java.rmi.Naming;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class ProxyHandler {

private static ProxyHandler handler;

private SensorService service;

//make sure only create one object from proxy handler using singleton

public static ProxyHandler getInstance() throws Exception {

if (handler == null) {

handler = new ProxyHandler();

}

return handler;

}

private ProxyHandler() throws Exception {

//get binded registry and lookup for sensor

Registry reg = LocateRegistry.getRegistry("localhost", 5052);

service = (SensorService) reg.lookup("sensor");

}

public SensorService getService() throws RemoteException {

if (service == null) {

service = new SensorServer();

}

return service;

}

}

SensorServer.java

package com.dc.server;

import java.rmi.Naming;

import java.rmi.RemoteException;

import com.dc.model.SensorModel;

import com.dc.model.UserModel;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.net.HttpURLConnection;

import java.net.MalformedURLException;

import java.net.ProtocolException;

import java.net.URL;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

public class SensorServer extends UnicastRemoteObject implements SensorService {

public SensorServer() throws RemoteException {

}

//method to add sensors

@Override

public boolean addSensor(SensorModel se, UserModel um) throws RemoteException {

try {

//put given user details from the client side

JSONObject userDetails = new JSONObject();

userDetails.put("id", um.getId());

userDetails.put("name", um.getName());

userDetails.put("userName", um.getUsername());

userDetails.put("password", um.getPassword());

userDetails.put("email", um.getEmail());

userDetails.put("mobile", um.getMobile());

//add all user and sensor details to one object

JSONObject sensorDetails = new JSONObject();

sensorDetails.put("id", se.getId());

sensorDetails.put("roomNo", se.getRoomNo());

sensorDetails.put("floorNo", se.getFloorNo());

sensorDetails.put("status", se.getStatus());

sensorDetails.put("responsiblePerson", userDetails);

JSONObject sensorObject = new JSONObject();

sensorObject.put("Sensor", sensorDetails);

URL url;

try {

url = new URL("http://localhost:8080/api/sensor"); //api url to save sensor details

HttpURLConnection con = (HttpURLConnection) url.openConnection(); //set connection

con.setRequestMethod("POST"); //set rest method as post

con.setRequestProperty("Content-Type", "application/json; utf-8");

con.setRequestProperty("Accept", "application/json"); //set headers

con.setDoOutput(true);

//convert input data to string

String jsonInputString = sensorDetails.toString();

//save input data

try (OutputStream os = con.getOutputStream()) {

byte[] input = jsonInputString.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());

}

System.out.println(response.toString());

}

con.disconnect(); //disconnect the connection

} catch (MalformedURLException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

} catch (JSONException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

return true;

}

//method to add sensors

@Override

public boolean editSensor(SensorModel se, UserModel um) throws RemoteException {

try {

//put given user details from the client side

JSONObject userDetails = new JSONObject();

userDetails.put("id", um.getId());

userDetails.put("name", um.getName());

userDetails.put("userName", um.getUsername());

userDetails.put("password", um.getPassword());

userDetails.put("email", um.getEmail());

userDetails.put("mobile", um.getMobile());

//add all user and sensor details to one object

JSONObject sensorDetails = new JSONObject();

sensorDetails.put("id", se.getId());

sensorDetails.put("roomNo", se.getRoomNo());

sensorDetails.put("floorNo", se.getFloorNo());

sensorDetails.put("status", se.getStatus());

sensorDetails.put("responsiblePerson", userDetails);

JSONObject sensorObject = new JSONObject();

sensorObject.put("Sensor", sensorDetails);

URL url;

try {

url = new URL("http://localhost:8080/api/sensor/" + se.getId()); //api url to update sensor details

HttpURLConnection con = (HttpURLConnection) url.openConnection(); //set connection

con.setRequestMethod("PUT"); //set rest method as put

con.setRequestProperty("Content-Type", "application/json; utf-8");

con.setRequestProperty("Accept", "application/json"); //set headers

con.setDoOutput(true);

//convert input data to string

String jsonInputString = sensorDetails.toString();

//save input data

try (OutputStream os = con.getOutputStream()) {

byte[] input = jsonInputString.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());

}

System.out.println(response.toString());

}

con.disconnect(); //disconnect the connection

} catch (MalformedURLException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

} catch (JSONException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

return true;

}

//method to get all sensors

@Override

public ArrayList<SensorModel> getSensors() throws RemoteException {

URL obj;

ArrayList<SensorModel> list = null; //array to save data

try {

obj = new URL("http://localhost:8080/api/sensors"); //rest url to get sensor details

HttpURLConnection con = (HttpURLConnection) obj.openConnection(); //set connection

con.setRequestMethod("GET"); //set rest method as get

con.setRequestProperty("Content-Type", "application/json; utf-8");

con.setRequestProperty("Accept", "application/json"); //set headers

//get conncetion response code

int responseCode = con.getResponseCode();

System.out.println("GET Response Code :: " + responseCode);

//if conncetion success, retrieve data

if (responseCode == HttpURLConnection.HTTP\_OK) { // success

BufferedReader in = new BufferedReader(new InputStreamReader(

con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

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

response.append(inputLine);

}

in.close();

//convert output data to string and make JSON array

String result = response.toString();

JSONArray array = new JSONArray(result);

list = new ArrayList<>();

//loop through json array

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

JSONObject jsonObj = array.getJSONObject(i);

//set data to sensor object

SensorModel sm = new SensorModel();

sm.setId(jsonObj.getInt("id"));

sm.setFloorNo(String.valueOf(jsonObj.getInt("floorNo")));

sm.setRoomNo(String.valueOf(jsonObj.getInt("roomNo")));

sm.setSmokeLevel(jsonObj.getInt("smokeLevel"));

sm.setCoLevel(jsonObj.getInt("co2Level"));

sm.setStatus(jsonObj.getInt("status"));

JSONObject jsonObj1 = jsonObj.getJSONObject("responsiblePerson");

//set user data to user object

UserModel um = new UserModel();

um.setId(jsonObj1.getInt("id"));

um.setName(jsonObj1.getString("name"));

um.setUsername(jsonObj1.getString("userName"));

um.setPassword(jsonObj1.getString("password"));

um.setEmail(jsonObj1.getString("email"));

um.setMobile(jsonObj1.getString("mobile"));

sm.setUm(um);

list.add(sm);

}

} else {

System.out.println("Error Occured");

}

} catch (MalformedURLException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (ProtocolException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (JSONException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

return list; //return sensor data to client side

}

//method to get sensor by id

@Override

public SensorModel getSensorById(String id) throws RemoteException {

URL obj;

SensorModel sm = new SensorModel(); //create an sensor object

try {

obj = new URL("http://localhost:8080/api/sensor/" + id); //rest url to get sensor by id details

HttpURLConnection con = (HttpURLConnection) obj.openConnection(); //set connection

con.setRequestMethod("GET"); //set rest method as get

con.setRequestProperty("Content-Type", "application/json; utf-8");

con.setRequestProperty("Accept", "application/json"); //set headers

//get conncetion response code

int responseCode = con.getResponseCode();

System.out.println("GET Response Code :: " + responseCode);

//if conncetion success, retrieve data

if (responseCode == HttpURLConnection.HTTP\_OK) { // success

BufferedReader in = new BufferedReader(new InputStreamReader(

con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

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

response.append(inputLine);

}

in.close();

//convert output data to string

String result = response.toString();

//set output sensor and user details to the sensor object

JSONObject jsonObj = new JSONObject(result);

sm.setId(jsonObj.getInt("id"));

sm.setFloorNo(String.valueOf(jsonObj.getInt("floorNo")));

sm.setRoomNo(String.valueOf(jsonObj.getInt("roomNo")));

sm.setStatus(jsonObj.getInt("status"));

JSONObject jsonUser = jsonObj.getJSONObject("responsiblePerson");

sm.setResId(jsonUser.getInt("id"));

sm.setResName(jsonUser.getString("name"));

} else {

System.out.println("Error Occured");

}

} catch (MalformedURLException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (ProtocolException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (JSONException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

return sm; //return data to client side

}

//method to login users

@Override

public boolean login(UserModel um) throws RemoteException {

URL obj;

try {

obj = new URL("http://localhost:8080/api/users"); //rest url to login users

HttpURLConnection con = (HttpURLConnection) obj.openConnection(); //set conncetion

con.setRequestMethod("GET"); //set rest method as get

con.setRequestProperty("Content-Type", "application/json; utf-8");

con.setRequestProperty("Accept", "application/json"); //set headers

//get conncetion response code

int responseCode = con.getResponseCode();

System.out.println("GET Response Code :: " + responseCode);

//if conncetion success, retrieve data

if (responseCode == HttpURLConnection.HTTP\_OK) { // success

BufferedReader in = new BufferedReader(new InputStreamReader(

con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

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

response.append(inputLine);

}

in.close();

//convert output data to string and make JSON array

String result = response.toString();

JSONArray array = new JSONArray(result);

//loop through json array while given username and password matches

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

JSONObject jsonObj = array.getJSONObject(i);

//if matches return true

if (jsonObj.getString("userName").equals(um.getUsername())

&& jsonObj.getString("password").equals(um.getPassword())) {

return true;

}

}

} else {

System.out.println("Error Occured");

}

} catch (MalformedURLException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (ProtocolException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (JSONException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

return false; //if not matches return false

}

//method to get all users

@Override

public ArrayList<UserModel> getUsers() throws RemoteException {

URL obj;

ArrayList<UserModel> list = null;

try {

obj = new URL("http://localhost:8080/api/users"); //rest url to get all users

HttpURLConnection con = (HttpURLConnection) obj.openConnection(); //set connection

con.setRequestMethod("GET"); //set rest method as get

con.setRequestProperty("Content-Type", "application/json; utf-8");

con.setRequestProperty("Accept", "application/json"); //set headers

//get connection response code

int responseCode = con.getResponseCode();

System.out.println("GET Response Code :: " + responseCode);

//if conncetion success, retrieve data

if (responseCode == HttpURLConnection.HTTP\_OK) { // success

BufferedReader in = new BufferedReader(new InputStreamReader(

con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

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

response.append(inputLine);

}

in.close();

//convert output data to string and make JSON array

String result = response.toString();

JSONArray array = new JSONArray(result);

list = new ArrayList<>();

//loop through json array while given username and password matches

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

JSONObject jsonObj = array.getJSONObject(i);

UserModel um = new UserModel();

um.setId(jsonObj.getInt("id"));

um.setName(jsonObj.getString("name"));

um.setUsername(jsonObj.getString("userName"));

um.setPassword(jsonObj.getString("password"));

um.setEmail(jsonObj.getString("email"));

um.setMobile(jsonObj.getString("mobile"));

list.add(um);

}

} else {

System.out.println("Error Occured");

}

} catch (MalformedURLException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (ProtocolException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (JSONException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

return list;

}

//method to get users by id

@Override

public UserModel getUserById(int id) throws RemoteException {

URL obj;

try {

obj = new URL("http://localhost:8080/api/users"); //rest url to get users by id

HttpURLConnection con = (HttpURLConnection) obj.openConnection(); //set connection

con.setRequestMethod("GET"); //set rest method as get

con.setRequestProperty("Content-Type", "application/json; utf-8");

con.setRequestProperty("Accept", "application/json"); //set headers

//get connection response code

int responseCode = con.getResponseCode();

System.out.println("GET Response Code :: " + responseCode);

//if conncetion success, retrieve data

if (responseCode == HttpURLConnection.HTTP\_OK) { // success

BufferedReader in = new BufferedReader(new InputStreamReader(

con.getInputStream()));

String inputLine;

StringBuffer response = new StringBuffer();

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

response.append(inputLine);

}

in.close();

//convert output data to string and make JSON array

String result = response.toString();

JSONArray array = new JSONArray(result);

//loop through json array

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

JSONObject jsonObj = array.getJSONObject(i);

//if given id is equals retrieve only those data

if (jsonObj.getInt("id") == id) {

UserModel um = new UserModel();

um.setId(jsonObj.getInt("id"));

um.setName(jsonObj.getString("name"));

um.setUsername(jsonObj.getString("userName"));

um.setPassword(jsonObj.getString("password"));

um.setEmail(jsonObj.getString("email"));

um.setMobile(jsonObj.getString("mobile"));

return um; //return object

}

}

} else {

System.out.println("Error Occured");

}

} catch (MalformedURLException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (ProtocolException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (JSONException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

return null;

}

//method to send alerts

@Override

public boolean sendAlert(SensorModel se, UserModel um) throws RemoteException {

try {

//put given user details from the client side

JSONObject userDetails = new JSONObject();

userDetails.put("id", um.getId());

userDetails.put("name", um.getName());

userDetails.put("userName", um.getUsername());

userDetails.put("password", um.getPassword());

userDetails.put("email", um.getEmail());

userDetails.put("mobile", um.getMobile());

//add all user and sensor details to one object

JSONObject sensorDetails = new JSONObject();

sensorDetails.put("id", se.getId());

sensorDetails.put("roomNo", se.getRoomNo());

sensorDetails.put("floorNo", se.getFloorNo());

sensorDetails.put("status", se.getStatus());

sensorDetails.put("smokeLevel", se.getSmokeLevel());

sensorDetails.put("co2Level", se.getCoLevel());

sensorDetails.put("responsiblePerson", userDetails);

JSONObject sensorObject = new JSONObject();

sensorObject.put("Sensor", sensorDetails);

URL url;

try {

url = new URL("http://localhost:8080/api/sensor/sendAlert"); //rest url to send alerts

HttpURLConnection con = (HttpURLConnection) url.openConnection(); //set connection

con.setRequestMethod("POST"); //set rest method as post

con.setRequestProperty("Content-Type", "application/json; utf-8");

con.setRequestProperty("Accept", "application/json"); //set headers

con.setDoOutput(true);

//convert input data to string

String jsonInputString = sensorDetails.toString();

//save input data

try (OutputStream os = con.getOutputStream()) {

byte[] input = jsonInputString.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());

}

System.out.println(response.toString());

}

con.disconnect();

} catch (MalformedURLException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

} catch (JSONException ex) {

Logger.getLogger(SensorServer.class.getName()).log(Level.SEVERE, null, ex);

}

return true;

}

}

SensorService.java

package com.dc.server;

import java.rmi.Remote;

import java.rmi.RemoteException;

import com.dc.model.SensorModel;

import com.dc.model.UserModel;

import java.util.ArrayList;

public interface SensorService extends Remote {

//server side method to add sensors

public boolean addSensor(SensorModel se, UserModel um) throws RemoteException;

//method to edit sensors

public boolean editSensor(SensorModel se, UserModel um) throws RemoteException;

//method to get all sensors

public ArrayList<SensorModel> getSensors() throws RemoteException;

//method to get a sensor by id

public SensorModel getSensorById(String id) throws RemoteException;

//method to login

public boolean login(UserModel um) throws RemoteException;

//method to get all users

public ArrayList<UserModel> getUsers() throws RemoteException;

//method to get a user by id

public UserModel getUserById(int id) throws RemoteException;

//method to send alerts

public boolean sendAlert(SensorModel sm, UserModel um) throws RemoteException;

}

ServerInit.java

package com.dc.server;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.logging.Level;

import java.util.logging.Logger;

public class ServerInit {

public static void main(String[] args) {

try {

//create rmi registry on port 5052

Registry registry = LocateRegistry.createRegistry(5052);

SensorService stub = new SensorServer(); //create server object

registry.rebind("sensor", stub); //bind server to registry

System.out.println("Server Has Been Started");

} catch (RemoteException ex) {

Logger.getLogger(ServerInit.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

AddSensors.java

package com.dc.view;

import com.dc.controller.AdminController;

import com.dc.controller.SensorController;

import com.dc.model.SensorModel;

import com.dc.model.UserModel;

import com.dc.server.ProxyHandler;

import java.util.ArrayList;

import java.util.Vector;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

public class AddSensors extends javax.swing.JFrame {

UserModel umSelected;

/\*\*

\* Creates new form AddSensors

\*/

public AddSensors() {

initComponents();

view\_data(); //retrieve all sensor data

fill\_persons(); //get users

}

public void view\_data() {

//assign vectors for show in the table

ArrayList<SensorModel> list = null;

Vector<Vector<String>> vectors = new Vector<>();

Vector<String> header = new Vector<>();

try {

list = SensorController.get(); //get sensor data from server side

} catch (Exception ex) {

Logger.getLogger(AddSensors.class.getName()).log(Level.SEVERE, null, ex);

}

//loop through sensors

for (SensorModel se : list) {

//assign data to vectors

Vector<String> vector = new Vector<>();

vector.add(String.valueOf(se.getId()));

vector.add(se.getFloorNo());

vector.add(se.getRoomNo());

//check the status of the sensor

if (se.getStatus() == 1) {

vector.add("Active");

} else {

vector.add("Inactive");

}

vectors.add(vector);

}

//add table headers

header.add("ID");

header.add("Floor No");

header.add("Room No");

header.add("Status");

DefaultTableModel model = new DefaultTableModel(vectors, header);

table\_sensors.setModel(model); //set table

}

//method to add users to dropdown

public void fill\_persons() {

ArrayList<UserModel> list = null; //array to save user data

try {

list = AdminController.getUsers(); //get users from server side

} catch (Exception ex) {

Logger.getLogger(AddSensors.class.getName()).log(Level.SEVERE, null, ex);

}

//add users to drop down

for (UserModel um : list) {

person.addItem(String.valueOf(um.getId()) + ". " + um.getName());

}

}

/\*\*

\* 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">//GEN-BEGIN:initComponents

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jLabel8 = new javax.swing.JLabel();

floor\_no = new javax.swing.JTextField();

jLabel9 = new javax.swing.JLabel();

room\_no = new javax.swing.JTextField();

register\_btn = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

table\_sensors = new javax.swing.JTable();

jPanel5 = new javax.swing.JPanel();

jLabel3 = new javax.swing.JLabel();

jLabel10 = new javax.swing.JLabel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

status = new javax.swing.JComboBox<>();

jLabel15 = new javax.swing.JLabel();

jLabel16 = new javax.swing.JLabel();

person = new javax.swing.JComboBox<>();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setUndecorated(true);

jPanel1.setBackground(new java.awt.Color(255, 255, 255));

jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(255, 255, 255)));

jLabel8.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

jLabel8.setText("Floor No:");

floor\_no.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

floor\_no.setForeground(new java.awt.Color(0, 0, 153));

floor\_no.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(153, 153, 153)));

floor\_no.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

floor\_noMouseClicked(evt);

}

});

floor\_no.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

floor\_noActionPerformed(evt);

}

});

jLabel9.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

jLabel9.setText("Room No:");

room\_no.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

room\_no.setForeground(new java.awt.Color(0, 0, 153));

room\_no.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(153, 153, 153)));

room\_no.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

room\_noMouseClicked(evt);

}

});

room\_no.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

room\_noActionPerformed(evt);

}

});

register\_btn.setBackground(new java.awt.Color(102, 102, 255));

register\_btn.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N

register\_btn.setForeground(new java.awt.Color(255, 255, 255));

register\_btn.setText("Register");

register\_btn.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

register\_btnActionPerformed(evt);

}

});

table\_sensors.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null}

},

new String [] {

"ID", "Floor No", "Room No"

}

));

table\_sensors.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

table\_sensorsMouseClicked(evt);

}

});

jScrollPane1.setViewportView(table\_sensors);

jPanel5.setBackground(new java.awt.Color(0, 0, 204));

jLabel3.setFont(new java.awt.Font("Arial", 1, 30)); // NOI18N

jLabel3.setForeground(new java.awt.Color(255, 255, 255));

jLabel3.setText("Add Sensors");

jLabel10.setFont(new java.awt.Font("Calibri Light", 1, 14)); // NOI18N

jLabel10.setForeground(new java.awt.Color(255, 255, 255));

jLabel10.setText("Logout");

jLabel10.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

jLabel10logoutMouseClicked(evt);

}

public void mouseEntered(java.awt.event.MouseEvent evt) {

jLabel10MouseEntered(evt);

}

public void mouseExited(java.awt.event.MouseEvent evt) {

jLabel10MouseExited(evt);

}

});

jLabel1.setFont(new java.awt.Font("Gill Sans MT", 0, 24)); // NOI18N

jLabel1.setForeground(new java.awt.Color(255, 255, 255));

jLabel1.setText("X");

jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(255, 255, 255)));

jLabel1.addMouseListener(new java.awt.event.MouseAdapter() {

public void mousePressed(java.awt.event.MouseEvent evt) {

jLabel1MousePressed(evt);

}

});

jLabel2.setFont(new java.awt.Font("Gill Sans MT", 0, 24)); // NOI18N

jLabel2.setForeground(new java.awt.Color(255, 255, 255));

jLabel2.setText("Back");

jLabel2.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(255, 255, 255)));

jLabel2.addMouseListener(new java.awt.event.MouseAdapter() {

public void mousePressed(java.awt.event.MouseEvent evt) {

jLabel2MousePressed(evt);

}

});

javax.swing.GroupLayout jPanel5Layout = new javax.swing.GroupLayout(jPanel5);

jPanel5.setLayout(jPanel5Layout);

jPanel5Layout.setHorizontalGroup(

jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel5Layout.createSequentialGroup()

.addGap(55, 55, 55)

.addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 679, Short.MAX\_VALUE)

.addComponent(jLabel10)

.addGap(32, 32, 32))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel5Layout.createSequentialGroup()

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel1))

);

jPanel5Layout.setVerticalGroup(

jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel5Layout.createSequentialGroup()

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 20, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 20, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 32, Short.MAX\_VALUE)

.addGroup(jPanel5Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel10, javax.swing.GroupLayout.PREFERRED\_SIZE, 47, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel3))

.addContainerGap())

);

status.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

status.setForeground(new java.awt.Color(0, 0, 153));

status.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "Active", "Inactive" }));

status.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

statusActionPerformed(evt);

}

});

jLabel15.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

jLabel15.setText("Status");

jLabel16.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

jLabel16.setText("Responsible Person");

person.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

person.setForeground(new java.awt.Color(0, 0, 153));

person.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

personActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addGap(41, 41, 41)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(room\_no, javax.swing.GroupLayout.DEFAULT\_SIZE, 200, Short.MAX\_VALUE)

.addComponent(jLabel9)

.addComponent(floor\_no)

.addComponent(jLabel8)

.addComponent(register\_btn, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel15)

.addComponent(status, 0, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel16)

.addComponent(person, 0, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 693, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(24, 24, 24))

.addComponent(jPanel5, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jPanel5, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(80, 80, 80)

.addComponent(jLabel8)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(floor\_no, javax.swing.GroupLayout.PREFERRED\_SIZE, 33, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(12, 12, 12)

.addComponent(jLabel9)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(room\_no, javax.swing.GroupLayout.PREFERRED\_SIZE, 33, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(13, 13, 13)

.addComponent(jLabel15)

.addGap(6, 6, 6)

.addComponent(status, javax.swing.GroupLayout.PREFERRED\_SIZE, 33, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jLabel16)

.addGap(11, 11, 11)

.addComponent(person, javax.swing.GroupLayout.PREFERRED\_SIZE, 33, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(register\_btn, javax.swing.GroupLayout.DEFAULT\_SIZE, 42, Short.MAX\_VALUE)

.addGap(48, 48, 48))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 365, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(28, 28, 28))))

);

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(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();

setLocationRelativeTo(null);

}// </editor-fold>//GEN-END:initComponents

private void floor\_noMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_floor\_noMouseClicked

}//GEN-LAST:event\_floor\_noMouseClicked

private void floor\_noActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_floor\_noActionPerformed

// TODO add your handling code here:

}//GEN-LAST:event\_floor\_noActionPerformed

private void room\_noMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_room\_noMouseClicked

// TODO add your handling code here:

}//GEN-LAST:event\_room\_noMouseClicked

private void room\_noActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_room\_noActionPerformed

// TODO add your handling code here:

}//GEN-LAST:event\_room\_noActionPerformed

private void register\_btnActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_register\_btnActionPerformed

//get form data

String floor = floor\_no.getText();

String room = room\_no.getText();

int getStatus;

//get provided sensor status

if (status.getSelectedItem().equals("Active")) {

getStatus = 1;

} else {

getStatus = 0;

}

//assign value to the object

SensorModel sm = new SensorModel(floor, room, getStatus);

//if user clicked register, save sensor data in db

if (evt.getActionCommand().equalsIgnoreCase("register")) {

try {

boolean value = SensorController.add(sm, umSelected); //server side method to add users

if (value) {

JOptionPane.showMessageDialog(null, "Location Added Successfully"); //success message

//empty the form values

floor\_no.setText("");

room\_no.setText("");

view\_data(); //refresh table data

}

} catch (Exception ex) {

Logger.getLogger(AddSensors.class.getName()).log(Level.SEVERE, null, ex);

}

} else { //if user clicked edit, update sensor data

try {

int row = table\_sensors.getSelectedRow(); //get selected sensor

String click = (table\_sensors.getModel().getValueAt(row, 0).toString());

sm.setId(Integer.parseInt(click));

boolean value = SensorController.edit(sm, umSelected); //server side function to update sensor data

if (value) {

JOptionPane.showMessageDialog(null, "Location Edited Successfully"); //success message

floor\_no.setText("");

room\_no.setText("");

register\_btn.setText("Register"); //set button back to register

view\_data();

}

} catch (Exception ex) {

Logger.getLogger(AddSensors.class.getName()).log(Level.SEVERE, null, ex);

}

}

}//GEN-LAST:event\_register\_btnActionPerformed

private void table\_sensorsMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_table\_sensorsMouseClicked

//get sensor clicked by the user

int row = table\_sensors.getSelectedRow();

String click = (table\_sensors.getModel().getValueAt(row, 0).toString());

try {

SensorModel sm = SensorController.getById(click); //get details of the clicked sensor from server side

//set data to the form

floor\_no.setText(sm.getFloorNo());

room\_no.setText(sm.getRoomNo());

//set status

String getStatus;

if (sm.getStatus() == 1) {

getStatus = "Active";

} else {

getStatus = "Inactive";

}

status.setSelectedItem(getStatus);

person.setSelectedItem(sm.getResId() + ". " + sm.getResName()); //set username

register\_btn.setText("Edit"); //set button to edit

} catch (Exception ex) {

Logger.getLogger(AddSensors.class.getName()).log(Level.SEVERE, null, ex);

}

}//GEN-LAST:event\_table\_sensorsMouseClicked

private void jLabel10logoutMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel10logoutMouseClicked

}//GEN-LAST:event\_jLabel10logoutMouseClicked

private void jLabel10MouseEntered(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel10MouseEntered

}//GEN-LAST:event\_jLabel10MouseEntered

private void jLabel10MouseExited(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel10MouseExited

}//GEN-LAST:event\_jLabel10MouseExited

private void jLabel1MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel1MousePressed

System.exit(0);

}//GEN-LAST:event\_jLabel1MousePressed

Main main;

private void jLabel2MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel2MousePressed

if (main == null) {

main = new Main();

main.setVisible(true);

} else {

main.setVisible(true);

}

dispose();

}//GEN-LAST:event\_jLabel2MousePressed

private void statusActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_statusActionPerformed

}//GEN-LAST:event\_statusActionPerformed

private void personActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_personActionPerformed

String item = (String) person.getSelectedItem();

Pattern p = Pattern.compile("^[^.]+");

Matcher m = p.matcher(item);

if (m.find()) {

try {

//get user details by id and assign to a global variable

umSelected = AdminController.userById(Integer.parseInt(m.group(0)));

} catch (Exception ex) {

Logger.getLogger(AddSensors.class.getName()).log(Level.SEVERE, null, ex);

}

}

}//GEN-LAST:event\_personActionPerformed

/\*\*

\* @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(AddSensors.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(AddSensors.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(AddSensors.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(AddSensors.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 AddSensors().setVisible(true);

}

});

}

// Variables declaration - do not modify//GEN-BEGIN:variables

private javax.swing.JTextField floor\_no;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel10;

private javax.swing.JLabel jLabel15;

private javax.swing.JLabel jLabel16;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel8;

private javax.swing.JLabel jLabel9;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel5;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JComboBox<String> person;

private javax.swing.JButton register\_btn;

private javax.swing.JTextField room\_no;

private javax.swing.JComboBox<String> status;

private javax.swing.JTable table\_sensors;

// End of variables declaration//GEN-END:variables

}

Login.java

package com.dc.view;

import com.dc.controller.AdminController;

import com.dc.controller.SensorController;

import com.dc.model.UserModel;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

public class Login extends javax.swing.JFrame {

private boolean unClicked = false;

private boolean pwClicked = false;

/\*\*

\* Creates new form Login

\*/

public Login() {

initComponents();

}

/\*\*

\* 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">//GEN-BEGIN:initComponents

private void initComponents() {

jPanel3 = new javax.swing.JPanel();

jPanel1 = new javax.swing.JPanel();

id\_text = new javax.swing.JTextField();

pw\_text = new javax.swing.JPasswordField();

login = new javax.swing.JButton();

jSeparator1 = new javax.swing.JSeparator();

jSeparator2 = new javax.swing.JSeparator();

jLabel2 = new javax.swing.JLabel();

jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setUndecorated(true);

jPanel3.setBackground(new java.awt.Color(8, 26, 56));

jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 102, 255), 3));

jPanel1.setBackground(new java.awt.Color(12, 106, 231));

jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(255, 255, 255)));

jPanel1.setForeground(new java.awt.Color(12, 75, 178));

jPanel1.setToolTipText("");

id\_text.setBackground(new java.awt.Color(12, 106, 231));

id\_text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N

id\_text.setForeground(new java.awt.Color(255, 255, 255));

id\_text.setText("Username");

id\_text.setToolTipText("");

id\_text.setBorder(null);

id\_text.setCaretColor(new java.awt.Color(255, 255, 255));

id\_text.setDisabledTextColor(new java.awt.Color(255, 255, 255));

id\_text.setName(""); // NOI18N

id\_text.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

id\_textMouseClicked(evt);

}

});

id\_text.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

id\_textActionPerformed(evt);

}

});

pw\_text.setBackground(new java.awt.Color(12, 106, 231));

pw\_text.setFont(new java.awt.Font("Arial", 0, 14)); // NOI18N

pw\_text.setForeground(new java.awt.Color(255, 255, 255));

pw\_text.setText("Password");

pw\_text.setBorder(null);

pw\_text.addFocusListener(new java.awt.event.FocusAdapter() {

public void focusGained(java.awt.event.FocusEvent evt) {

pw\_textFocusGained(evt);

}

});

pw\_text.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

pw\_textMouseClicked(evt);

}

});

login.setBackground(new java.awt.Color(54, 168, 240));

login.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

login.setText("Login");

login.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

login.setBorderPainted(false);

login.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

loginActionPerformed(evt);

}

});

jSeparator1.setForeground(new java.awt.Color(255, 255, 255));

jSeparator2.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT\_CURSOR));

jSeparator2.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N

jLabel2.setFont(new java.awt.Font("Dialog", 0, 24)); // NOI18N

jLabel2.setForeground(new java.awt.Color(255, 255, 255));

jLabel2.setText("Log in");

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(37, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED\_SIZE, 271, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED\_SIZE, 271, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(login, javax.swing.GroupLayout.PREFERRED\_SIZE, 271, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(pw\_text, javax.swing.GroupLayout.PREFERRED\_SIZE, 271, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(id\_text, javax.swing.GroupLayout.PREFERRED\_SIZE, 271, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(27, 27, 27))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addComponent(jLabel2)

.addGap(131, 131, 131))))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(61, 61, 61)

.addComponent(jLabel2)

.addGap(18, 18, 18)

.addComponent(id\_text, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(45, 45, 45)

.addComponent(pw\_text, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(37, 37, 37)

.addComponent(login, javax.swing.GroupLayout.PREFERRED\_SIZE, 39, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(63, Short.MAX\_VALUE))

);

jLabel1.setFont(new java.awt.Font("Gill Sans MT", 0, 24)); // NOI18N

jLabel1.setForeground(new java.awt.Color(255, 255, 255));

jLabel1.setText("X");

jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(255, 255, 255)));

jLabel1.addMouseListener(new java.awt.event.MouseAdapter() {

public void mousePressed(java.awt.event.MouseEvent evt) {

jLabel1MousePressed(evt);

}

});

javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);

jPanel3.setLayout(jPanel3Layout);

jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel3Layout.createSequentialGroup()

.addContainerGap(273, Short.MAX\_VALUE)

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel1, javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel3Layout.createSequentialGroup()

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(249, 249, 249))))

);

jPanel3Layout.setVerticalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel3Layout.createSequentialGroup()

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 20, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(41, Short.MAX\_VALUE))

);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

setLocationRelativeTo(null);

}// </editor-fold>//GEN-END:initComponents

private void id\_textActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_id\_textActionPerformed

}//GEN-LAST:event\_id\_textActionPerformed

private void pw\_textFocusGained(java.awt.event.FocusEvent evt) {//GEN-FIRST:event\_pw\_textFocusGained

}//GEN-LAST:event\_pw\_textFocusGained

Main main;

private void loginActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_loginActionPerformed

//get username and password form text fields

String username = id\_text.getText();

String password = new String(pw\_text.getPassword());

//set details to the object

UserModel um = new UserModel(username, password);

boolean value = false;

try {

//check whehter provided info are correct by calling server side method

value = AdminController.login(um);

} catch (Exception ex) {

Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);

}

//if correct show main menu

if (value) {

if (main == null) {

main = new Main();

main.setVisible(true);

} else {

main.setVisible(true);

}

dispose();

} else {

JOptionPane.showMessageDialog(null, "Username or Password incorrect");

}

}//GEN-LAST:event\_loginActionPerformed

private void jLabel1MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel1MousePressed

System.exit(0);

}//GEN-LAST:event\_jLabel1MousePressed

private void id\_textMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_id\_textMouseClicked

if (!unClicked) {

id\_text.setText("");

unClicked = true;

}

}//GEN-LAST:event\_id\_textMouseClicked

private void pw\_textMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_pw\_textMouseClicked

if (!pwClicked) {

pw\_text.setText("");

pwClicked = true;

}

}//GEN-LAST:event\_pw\_textMouseClicked

/\*\*

\* @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(Login.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Login.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Login.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 Login().setVisible(true);

}

});

}

// Variables declaration - do not modify//GEN-BEGIN:variables

private javax.swing.JTextField id\_text;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel3;

private javax.swing.JSeparator jSeparator1;

private javax.swing.JSeparator jSeparator2;

private javax.swing.JButton login;

private javax.swing.JPasswordField pw\_text;

// End of variables declaration//GEN-END:variables

}

Main.java

package com.dc.view;

import java.awt.Color;

import javax.swing.JOptionPane;

public class Main extends javax.swing.JFrame {

/\*\*

\* Creates new form Main

\*/

public Main() {

initComponents();

}

/\*\*

\* 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">//GEN-BEGIN:initComponents

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

panel\_add = new javax.swing.JPanel();

jLabel4 = new javax.swing.JLabel();

panel\_addSensor = new javax.swing.JPanel();

jLabel5 = new javax.swing.JLabel();

header = new javax.swing.JPanel();

jLabel3 = new javax.swing.JLabel();

logout\_icon = new javax.swing.JLabel();

logout = new javax.swing.JLabel();

jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setUndecorated(true);

jPanel1.setBackground(new java.awt.Color(0, 0, 102));

jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 51, 255), 5));

jPanel1.setForeground(new java.awt.Color(0, 0, 153));

panel\_add.setBackground(new java.awt.Color(255, 255, 255));

panel\_add.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102, 255, 102), 3));

panel\_add.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

panel\_addMouseClicked(evt);

}

public void mouseEntered(java.awt.event.MouseEvent evt) {

panel\_addMouseEntered(evt);

}

public void mouseExited(java.awt.event.MouseEvent evt) {

panel\_addMouseExited(evt);

}

public void mouseReleased(java.awt.event.MouseEvent evt) {

panel\_addMouseReleased(evt);

}

});

jLabel4.setFont(new java.awt.Font("Calibri", 1, 24)); // NOI18N

jLabel4.setForeground(new java.awt.Color(0, 0, 204));

jLabel4.setText("View Sensor Status");

javax.swing.GroupLayout panel\_addLayout = new javax.swing.GroupLayout(panel\_add);

panel\_add.setLayout(panel\_addLayout);

panel\_addLayout.setHorizontalGroup(

panel\_addLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel\_addLayout.createSequentialGroup()

.addGap(22, 22, 22)

.addComponent(jLabel4)

.addContainerGap(23, Short.MAX\_VALUE))

);

panel\_addLayout.setVerticalGroup(

panel\_addLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, panel\_addLayout.createSequentialGroup()

.addContainerGap(93, Short.MAX\_VALUE)

.addComponent(jLabel4)

.addGap(91, 91, 91))

);

panel\_addSensor.setBackground(new java.awt.Color(255, 255, 255));

panel\_addSensor.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(102, 255, 102), 3));

panel\_addSensor.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

panel\_addSensorMouseClicked(evt);

}

public void mouseEntered(java.awt.event.MouseEvent evt) {

panel\_addSensorMouseEntered(evt);

}

public void mouseExited(java.awt.event.MouseEvent evt) {

panel\_addSensorMouseExited(evt);

}

});

jLabel5.setFont(new java.awt.Font("Calibri", 1, 24)); // NOI18N

jLabel5.setForeground(new java.awt.Color(0, 0, 204));

jLabel5.setText("Add Sensors");

javax.swing.GroupLayout panel\_addSensorLayout = new javax.swing.GroupLayout(panel\_addSensor);

panel\_addSensor.setLayout(panel\_addSensorLayout);

panel\_addSensorLayout.setHorizontalGroup(

panel\_addSensorLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, panel\_addSensorLayout.createSequentialGroup()

.addContainerGap(57, Short.MAX\_VALUE)

.addComponent(jLabel5)

.addGap(54, 54, 54))

);

panel\_addSensorLayout.setVerticalGroup(

panel\_addSensorLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(panel\_addSensorLayout.createSequentialGroup()

.addGap(89, 89, 89)

.addComponent(jLabel5)

.addContainerGap(95, Short.MAX\_VALUE))

);

header.setBackground(new java.awt.Color(0, 0, 51));

jLabel3.setFont(new java.awt.Font("Arial", 1, 30)); // NOI18N

jLabel3.setForeground(new java.awt.Color(255, 255, 255));

jLabel3.setText("Dashboard");

logout\_icon.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

logout\_iconMouseClicked(evt);

}

});

logout.setFont(new java.awt.Font("Calibri Light", 1, 14)); // NOI18N

logout.setForeground(new java.awt.Color(255, 255, 255));

logout.setText("Logout");

logout.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

logoutMouseClicked(evt);

}

public void mouseEntered(java.awt.event.MouseEvent evt) {

logoutMouseEntered(evt);

}

public void mouseExited(java.awt.event.MouseEvent evt) {

logoutMouseExited(evt);

}

});

jLabel1.setFont(new java.awt.Font("Gill Sans MT", 0, 24)); // NOI18N

jLabel1.setForeground(new java.awt.Color(255, 255, 255));

jLabel1.setText("X");

jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(255, 255, 255)));

jLabel1.addMouseListener(new java.awt.event.MouseAdapter() {

public void mousePressed(java.awt.event.MouseEvent evt) {

jLabel1MousePressed(evt);

}

});

javax.swing.GroupLayout headerLayout = new javax.swing.GroupLayout(header);

header.setLayout(headerLayout);

headerLayout.setHorizontalGroup(

headerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(headerLayout.createSequentialGroup()

.addGap(53, 53, 53)

.addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(logout\_icon, javax.swing.GroupLayout.PREFERRED\_SIZE, 27, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(logout)

.addGap(32, 32, 32))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, headerLayout.createSequentialGroup()

.addGap(0, 0, Short.MAX\_VALUE)

.addComponent(jLabel1))

);

headerLayout.setVerticalGroup(

headerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(headerLayout.createSequentialGroup()

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 20, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 33, Short.MAX\_VALUE)

.addGroup(headerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(headerLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, headerLayout.createSequentialGroup()

.addComponent(logout\_icon, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(9, 9, 9))

.addComponent(logout, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 47, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 29, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap())

);

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(183, 183, 183)

.addComponent(panel\_add, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(134, 134, 134)

.addComponent(panel\_addSensor, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(163, Short.MAX\_VALUE))

.addComponent(header, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addComponent(header, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(78, 78, 78)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(panel\_add, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(panel\_addSensor, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, 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)

.addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

pack();

setLocationRelativeTo(null);

}// </editor-fold>//GEN-END:initComponents

ViewSensors view;

private void panel\_addMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_panel\_addMouseClicked

//singleton object creation for View frame

if (view == null) {

view = new ViewSensors();

view.setVisible(true);

}

else {

view.setVisible(true);

}

this.setVisible(false);

}//GEN-LAST:event\_panel\_addMouseClicked

private void panel\_addMouseEntered(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_panel\_addMouseEntered

panel\_add.setBackground(new Color(148,226,255));

}//GEN-LAST:event\_panel\_addMouseEntered

private void panel\_addMouseExited(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_panel\_addMouseExited

panel\_add.setBackground(new Color(255,255,255));

}//GEN-LAST:event\_panel\_addMouseExited

private void panel\_addMouseReleased(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_panel\_addMouseReleased

}//GEN-LAST:event\_panel\_addMouseReleased

AddSensors add;

private void panel\_addSensorMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_panel\_addSensorMouseClicked

//singleton object creation for Add frame

if (add == null) {

add = new AddSensors();

add.setVisible(true);

}

else {

add.setVisible(true);

}

this.setVisible(false);

}//GEN-LAST:event\_panel\_addSensorMouseClicked

private void panel\_addSensorMouseEntered(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_panel\_addSensorMouseEntered

panel\_addSensor.setBackground(new Color(148,226,255));

}//GEN-LAST:event\_panel\_addSensorMouseEntered

private void panel\_addSensorMouseExited(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_panel\_addSensorMouseExited

panel\_addSensor.setBackground(new Color(255,255,255));

}//GEN-LAST:event\_panel\_addSensorMouseExited

private void jLabel1MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel1MousePressed

System.exit(0);

}//GEN-LAST:event\_jLabel1MousePressed

Login login;

private void logout\_iconMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_logout\_iconMouseClicked

//on icon press

//yes or no message box for logout

int log = JOptionPane.showConfirmDialog(null, "Are you sure?", "Logout", JOptionPane.YES\_NO\_OPTION);

//if answer is yes redirect to the login frame

if (log == 0) {

//singleton object creation for Availability frame

if (login == null) {

login = new Login();

login.setVisible(true);

}

else {

login.setVisible(true);

}

dispose();

}

}//GEN-LAST:event\_logout\_iconMouseClicked

private void logoutMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_logoutMouseClicked

//on text click

//yes or no message box for logout

int log = JOptionPane.showConfirmDialog(null, "Are you sure?", "Logout", JOptionPane.YES\_NO\_OPTION);

//if answer is yes redirect to the login frame

if (log == 0) {

//singleton object creation for Availability frame

if (login == null) {

login = new Login();

login.setVisible(true);

}

else {

login.setVisible(true);

}

dispose();

}

}//GEN-LAST:event\_logoutMouseClicked

private void logoutMouseEntered(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_logoutMouseEntered

//change logout to red colour on mouse over

logout.setForeground(Color.red);

}//GEN-LAST:event\_logoutMouseEntered

private void logoutMouseExited(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_logoutMouseExited

//change logout to normal colour on mouse exit

logout.setForeground(Color.white);

}//GEN-LAST:event\_logoutMouseExited

/\*\*

\* @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(Main.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Main.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 Main().setVisible(true);

}

});

}

// Variables declaration - do not modify//GEN-BEGIN:variables

private javax.swing.JPanel header;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JPanel jPanel1;

private javax.swing.JLabel logout;

private javax.swing.JLabel logout\_icon;

private javax.swing.JPanel panel\_add;

private javax.swing.JPanel panel\_addSensor;

// End of variables declaration//GEN-END:variables

}

ViewSensors.java

package com.dc.view;

import com.dc.controller.SensorController;

import com.dc.model.SensorModel;

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.Vector;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JComponent;

import javax.swing.SwingUtilities;

import javax.swing.Timer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableCellRenderer;

public class ViewSensors extends javax.swing.JFrame {

/\*\*

\* Creates new form ViewSensors

\*/

public ViewSensors() {

initComponents();

set\_timer(); //method to set timer to update sensor data

}

public void set\_timer() {

set\_data(); //refresh table

//set timer

Timer timer = new Timer(0, new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

set\_data(); //after every time interval, refresh table

}

});

timer.setDelay(5000); // set delay time

timer.start(); //start timer

}

public void set\_data() {

//assign vectors to set table data

ArrayList<SensorModel> list = null;

Vector<Vector<String>> vectors = new Vector<>();

Vector<String> header = new Vector<>();

try {

list = SensorController.get(); //get sensor data from server side

} catch (Exception ex) {

Logger.getLogger(AddSensors.class.getName()).log(Level.SEVERE, null, ex);

}

//loop through sensors

for (SensorModel se : list) {

Vector<String> vector = new Vector<>();

//assign data to vectors

vector.add(se.getFloorNo());

vector.add(se.getRoomNo());

vector.add(String.valueOf(se.getSmokeLevel()));

vector.add(String.valueOf(se.getCoLevel()));

vector.add(String.valueOf(se.getStatus()));

vectors.add(vector);

//if smoke level or co2 level high, send alert

if (se.getSmokeLevel() > 5 || se.getCoLevel() > 5) {

try {

//serverside method to send alerts

SensorController.sendAlert(se, se.getUm());

} catch (Exception ex) {

Logger.getLogger(ViewSensors.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

//set table headers

header.add("Floor No");

header.add("Room No");

header.add("Smoke Level");

header.add("CO2 Level");

header.add("Status");

DefaultTableModel model = new DefaultTableModel(vectors, header);

table\_sensors.setModel(model); //set table

}

/\*\*

\* 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">//GEN-BEGIN:initComponents

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jScrollPane1 = new javax.swing.JScrollPane();

table\_sensors = new javax.swing.JTable() {

@Override

public Component prepareRenderer (TableCellRenderer renderer, int rowIndex, int columnIndex){

JComponent component = (JComponent)super.prepareRenderer(renderer, rowIndex, columnIndex);

String value = (String)getModel().getValueAt(rowIndex, 2);

String value1 = (String)getModel().getValueAt(rowIndex, 3);

table\_sensors.setRowHeight(30);

table\_sensors.setShowGrid(true);

if (columnIndex == 2 && Integer.parseInt(value) > 5){

component.setBackground(Color.RED);

component.setForeground(Color.BLACK);

} else if (columnIndex == 3 && Integer.parseInt(value1) > 5){

component.setBackground(Color.RED);

component.setForeground(Color.BLACK);

} else {

component.setBackground(Color.WHITE);

component.setForeground(Color.BLACK);

}

return component;

}

};

jLabel2 = new javax.swing.JLabel();

jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setUndecorated(true);

jPanel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 255)));

table\_sensors.setBackground(new java.awt.Color(204, 204, 204));

table\_sensors.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

table\_sensors.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

table\_sensors.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

{null, null, null},

{null, null, null},

{null, null, null},

{null, null, null}

},

new String [] {

"ID", "Floor No", "Room No"

}

));

table\_sensors.addMouseListener(new java.awt.event.MouseAdapter() {

public void mouseClicked(java.awt.event.MouseEvent evt) {

table\_sensorsMouseClicked(evt);

}

});

jScrollPane1.setViewportView(table\_sensors);

jLabel2.setBackground(new java.awt.Color(51, 255, 255));

jLabel2.setFont(new java.awt.Font("Gill Sans MT", 0, 24)); // NOI18N

jLabel2.setText("Back");

jLabel2.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 255)));

jLabel2.addMouseListener(new java.awt.event.MouseAdapter() {

public void mousePressed(java.awt.event.MouseEvent evt) {

jLabel2MousePressed(evt);

}

});

jLabel1.setFont(new java.awt.Font("Gill Sans MT", 0, 36)); // NOI18N

jLabel1.setText("X");

jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(51, 51, 255)));

jLabel1.addMouseListener(new java.awt.event.MouseAdapter() {

public void mousePressed(java.awt.event.MouseEvent evt) {

jLabel1MousePressed(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()

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel1))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(21, 21, 21)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 1063, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(21, Short.MAX\_VALUE))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 0, Short.MAX\_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 604, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, 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(layout.createSequentialGroup()

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 7, Short.MAX\_VALUE))

);

pack();

setLocationRelativeTo(null);

}// </editor-fold>//GEN-END:initComponents

private void table\_sensorsMouseClicked(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_table\_sensorsMouseClicked

}//GEN-LAST:event\_table\_sensorsMouseClicked

Main main;

private void jLabel2MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel2MousePressed

if (main == null) {

main = new Main();

main.setVisible(true);

} else {

main.setVisible(true);

}

dispose();

}//GEN-LAST:event\_jLabel2MousePressed

private void jLabel1MousePressed(java.awt.event.MouseEvent evt) {//GEN-FIRST:event\_jLabel1MousePressed

System.exit(0);

}//GEN-LAST:event\_jLabel1MousePressed

/\*\*

\* @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(ViewSensors.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ViewSensors.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ViewSensors.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(ViewSensors.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 ViewSensors().setVisible(true);

}

});

}

// Variables declaration - do not modify//GEN-BEGIN:variables

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JPanel jPanel1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JTable table\_sensors;

// End of variables declaration//GEN-END:variables

}