

**SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY.**

Distributed System – SE3020.

Assignment 2.

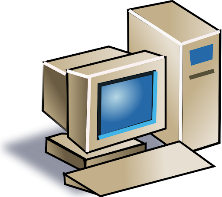
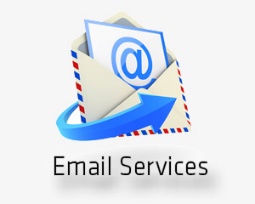
**FIRE ALARM MONITORING SYSTEM.**

Project Report.

|  |  |
| --- | --- |
| Registration Number | Name with Initials |
| IT18129090 | Vidanage B.I |
| IT18135930 | Mihiranga K.A |
| IT18139136 | Darvin S.B.M |
| IT18139204 | De Silva A.H.A.U |

Group Details.

**High Level Architecture Diagram.**



**Email Service**

**Sensor App**

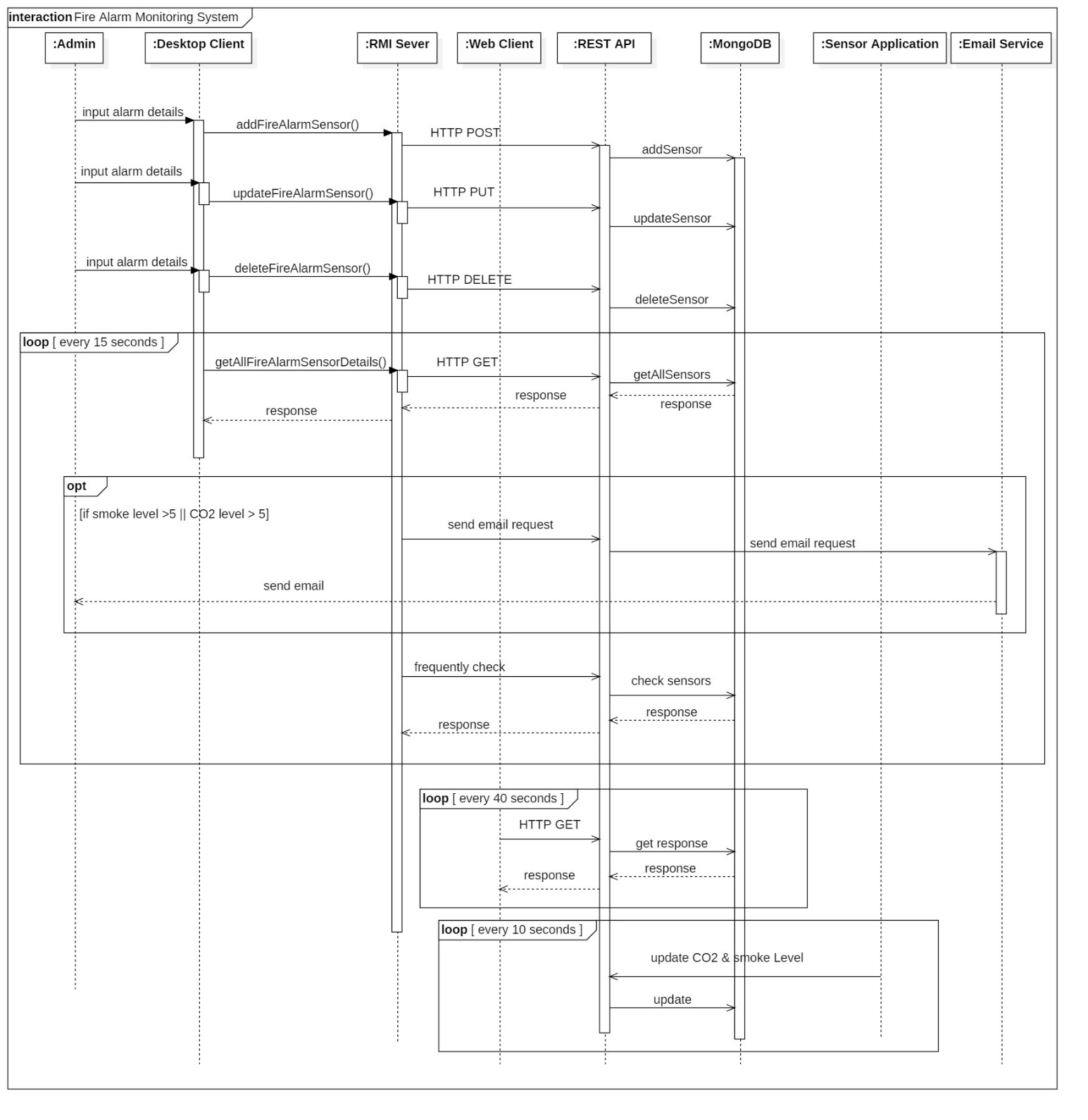
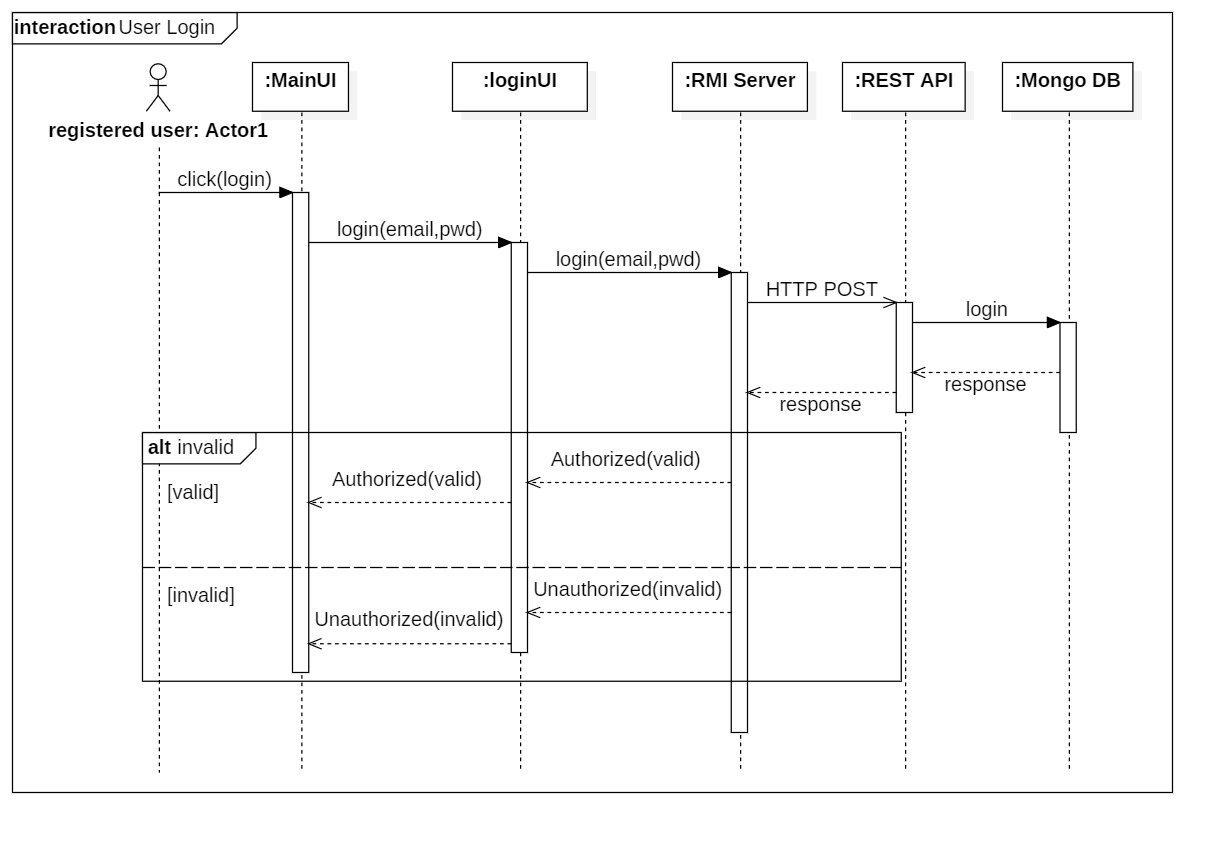
**RMI Server**

**Desktop Client**

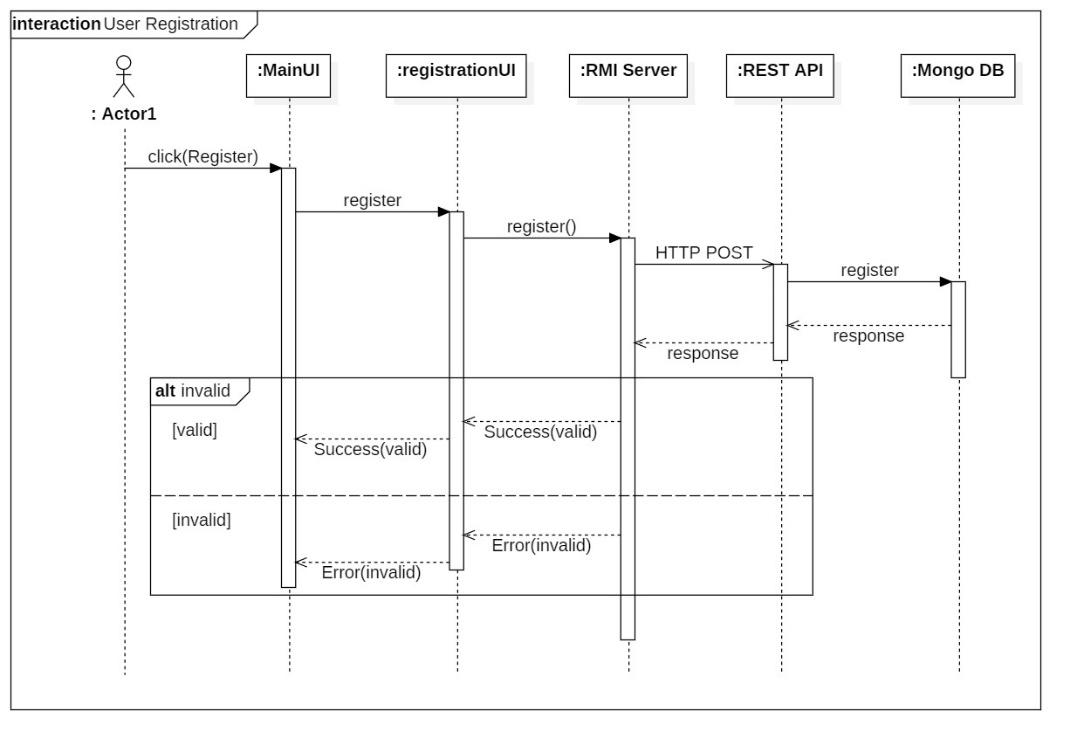
**Database**

**Web Client**

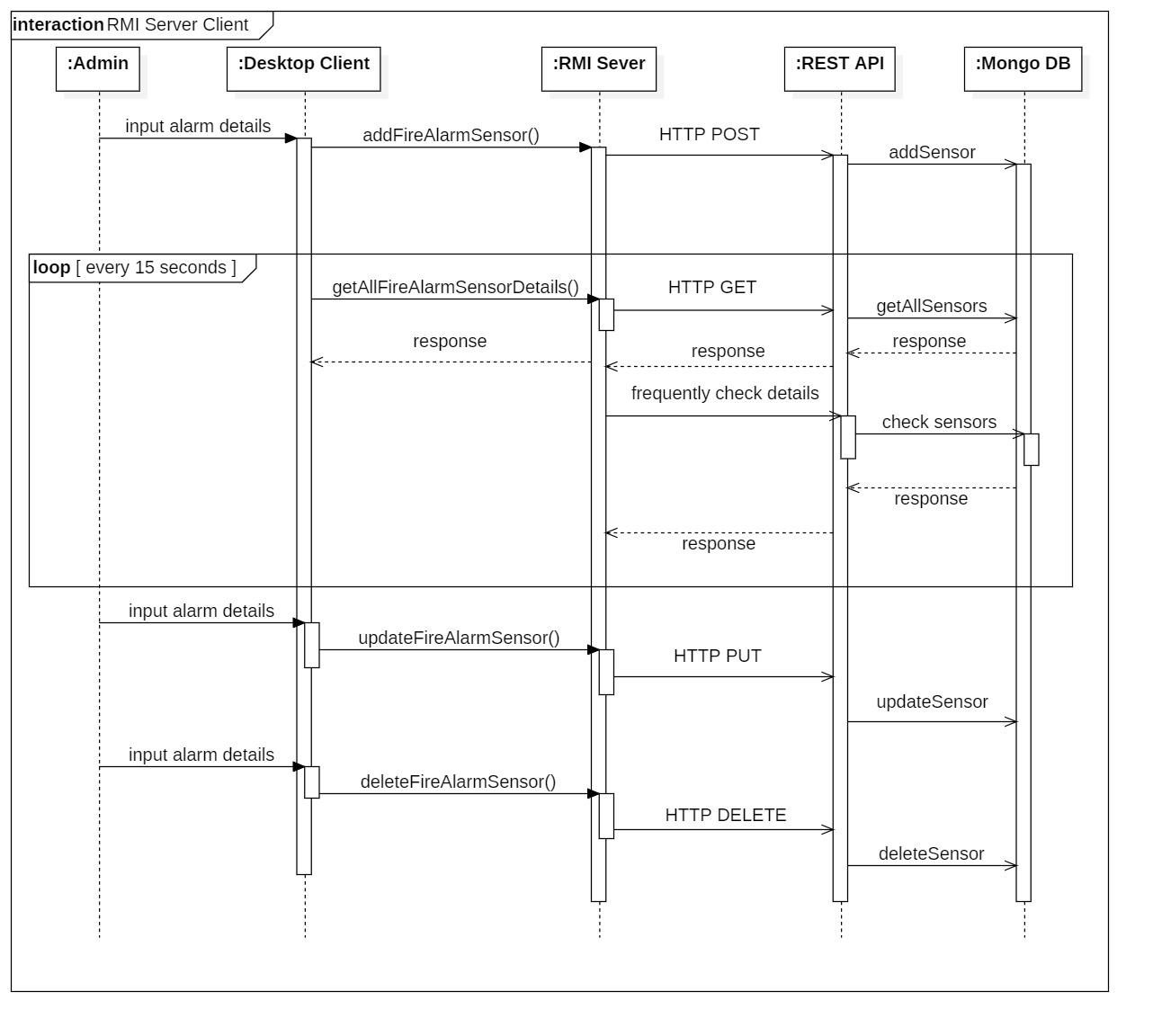
**Sequence Diagrams.**

1. **Fire alarm monitoring System.**
2. **Login.**

User can login to the system entering his or her valid email address and password. Then user email address and password are send to the sever interface using login() method. Then the HTTP POST request will call to REST API route called <http://localhost:5000/users/login> . After verifying the email address and password, the user is logged in to the system successfully and send a successful message to the user. If email address and password are invalid, then system send an error message to the user.

**3. Register.**

The user must register to the system before he/she login to the system. User can register to the system after fill the registration form and submit it. Then user details are send to the sever interface using register() method. Then the HTTP POST request will call to REST API route called <http://localhost:5000/users/register>. After verifying the user details, all details are added to the database successfully and send a successful message to the user. If details are invalid, then system send an error message to the user.

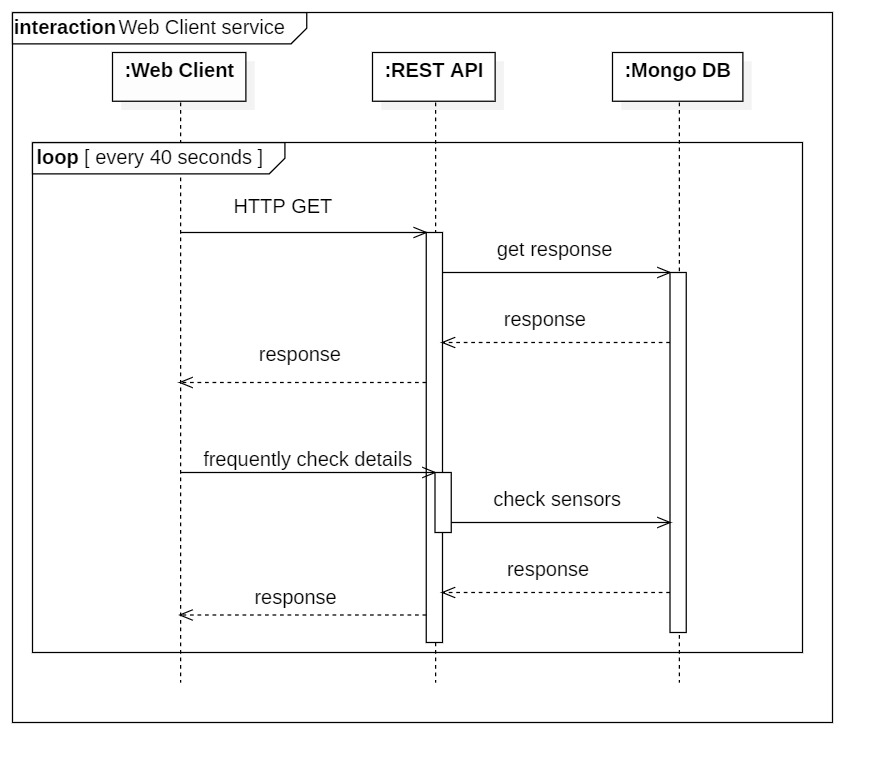
1. **RMI Server / Client.**

Admin can add a new sensor by clicking the ADD button in the main interface. Then admin has to fill the adding form and submit details by clicking submit button. After that submitted data are passed to the server using addFireAlarmSensor() method. Then the HTTP POST request will call to REST API route called <http://localhost:5000/alarmSensors/add> . Then the new sensor details are passed in to the database through the REST API according to that route.

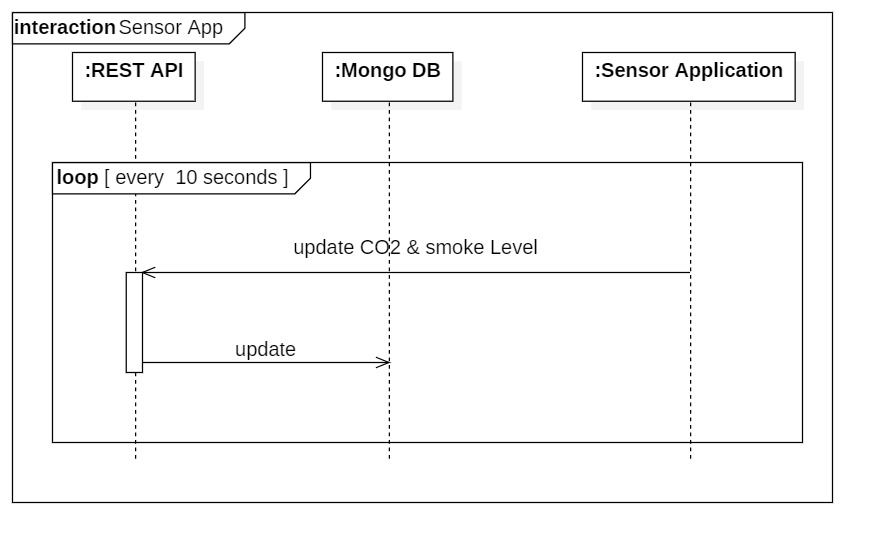
getAllFireAlarmSensorDetails() is the method to use get the details of all the sensors in the database. After calling to the getAllFireAlarmSensorDetails() method, the HTTP GET request will call to REST API route called <http://localhost:5000/alarmSensors> and display all Sensor details.

Admin can update alarm details using the UPDATE button. When the admin change the alarm details and clicks the UPADTE button, The updated data are passed to the server using the updateFireAlarmSensors() method. Then the HTTP PUT request will call to REST API route called <http://localhost:5000/alarmSensors/update/:id> and update alarm details successfully using the alarm id.

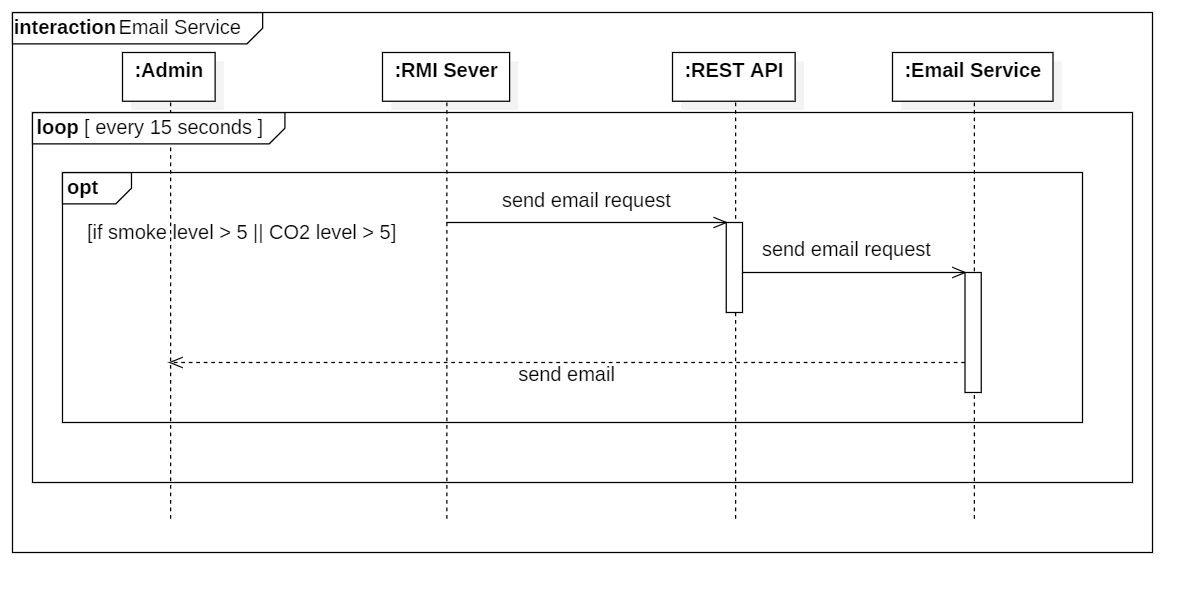
Admin can delete an alarm using the DELETE button. When the admin select an alarm and clicks the DELETE button, The deleted alarm id is passed to the server using the deleteFireAlarm() method. Then the HTTP DELETE request will call to REST API route called <http://localhost:5000/alarmSensors/:id> and delete alarm details successfully using its id.

1. **Web Client Application.**

After running the web client app, the HTTP GET request will call to REST API route called <http://localhost:5000/alarmSensors> . After that fetch the data from database according to that route and display data in a table using id as the key. This happens every 40 seconds. System get data from database through the REST API and display updated data to the user in every 40 seconds.

1. **Simulate Application (Sensor App)**

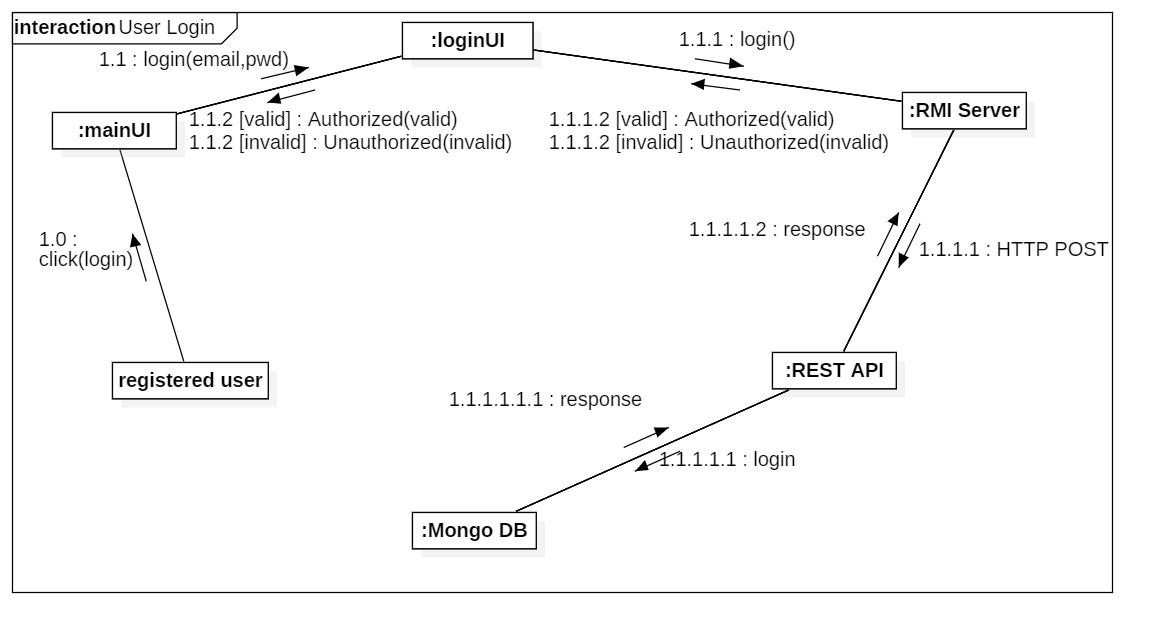
After running the Sensor App, all the data in the database is fetched to the system. After that randomly generate two values (between 1 and 10) for Co2 level and Smoke level instead of the values of CO2 level and Smoke level in that fetched data from database. Then the HTTP PUT request will call to REST API route called <http://localhost:5000/alarmSensors/update/:id> and update the database data again.

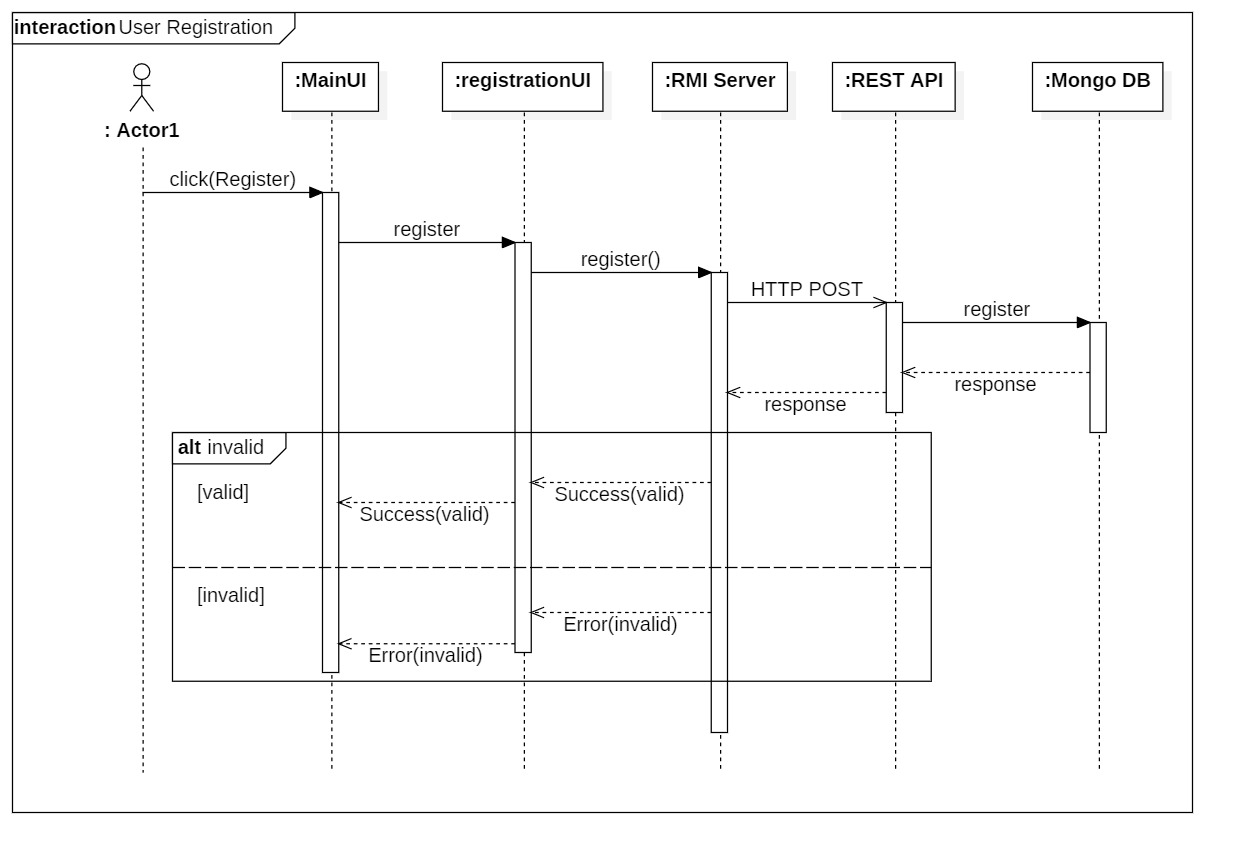
1. **Email Service.**

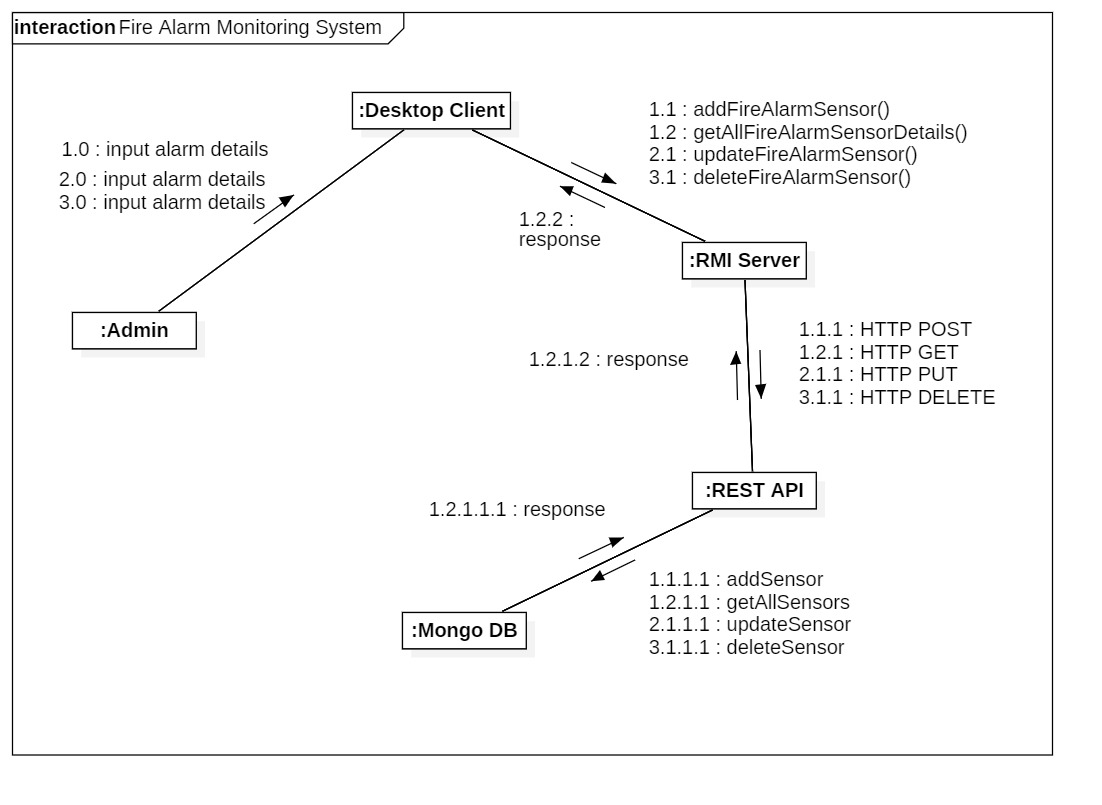
If the CO2 level or the smoke level is greater than 5, then send a request including user email address, floor number and room number from the RMI Server. Then it send to the admin.

The RMI Server refreshes every 15 seconds. If the CO2 level or Smoke level is greater than 5, then it will be sent email until the Co2 level or Smoke level is smaller than 5.

**Communication Diagrams.**

1. **Login.**
2. **Register.**

****

**3. RMI Server / Client.**

**Appendix.**

**1. RESTFul web Service API.**

**1.1: app.js**

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

require('dotenv/config');

const app = express();

//set port number

const port = process.env.PORT || 5000;

//user cors for the app

app.use(cors());

app.use(express.json());

//Import User route and Fire Alarm route.

const userRoute = require('./routes/user');

const alarmSensorRoute = require('./routes/alarmsensor');

//then use the imported routes.

app.use('/users', userRoute);

app.use('/alarmSensors', alarmSensorRoute);

//Connect DB

mongoose.connect(process.env.MongoDB\_CONNECTION,{ useNewUrlParser: true, useUnifiedTopology: true }, () =>

console.log('Connected to DB.!')

);

//Start server

app.listen(port, () => {

console.log(`Server is running on port: ${port}`);

});

**1.2: models/AlarmSensor.js**

const mongoose = require('mongoose');

//Create Fire Alarm model

//then this model for the database creation.

const SensorSchema = new mongoose.Schema({

//set sensorCode as String type value and is is required

sensorCode: {

type: String,

required: true

},

//set floorNumber as Number type value and is is required

floorNumber: {

type: Number,

required: true

},

//set roomNumber as Number type value and is is required

roomNumber: {

type: Number,

required: true

},

//set smokeLevel as Number type value and is is required.

//and set minimum value as 1 and maximum value as 10.

smokeLevel: {

type: Number,

min: 1,

max: 10,

required: true

},

//set co2Level as Number type value and is is required.

//and set minimum value as 1 and maximum value as 10.

co2Level: {

type: Number,

min: 1,

max: 10,

required: true

},

//set status as Boolean type value and is is required

status: {

type: Boolean,

required: true

}

},{ //set timestamps to get created and updated date and time.

timestamps: true,

}); //create database table name called AlarmSensor.

const AlarmSensor = mongoose.model('AlarmSensor', SensorSchema);

module.exports = AlarmSensor;

**1.3: models/User.js**

const mongoose = require('mongoose');

//Create User model

//then this model for the database creation.

const UserSchema = new mongoose.Schema({

//set name as String type value and is is required

name: {

type: String,

required: true

},

//set email as String type value and is is required

email: {

type: String,

required: true

},

//set mobile as String type value and is is required

mobile: {

type:String,

required: true

},

//set password as String type value and is is required

password: {

type: String,

required: true

}

},{

//set timestamps to get created and updated date and time.

timestamps: true,

});

//create database table name called User.

const User = mongoose.model('User', UserSchema);

module.exports = User;

**1.4: routes/alarmsensor.js**

const express = require('express');

const router = express.Router();

const AlarmSensor = require('../models/AlarmSensor');

//implement route for get all fire alarm sensor details.

//if request was success then pass all data and success status.

//if in an error,then pass error status.

router.route('/').get((req, res) => {

AlarmSensor.find()

.then(alarms => res.status(200).json({ status: "Success", data: alarms }))

.catch(() => res.status(400).json({ status: "Error", data:{} }));

});

//implement route for add new alarm for the database.

router.route('/add').post((req, res) => {

//get each values for the new variables

const sensorCode = req.body.sensorCode;

const floorNumber = req.body.floorNumber;

const roomNumber = req.body.roomNumber;

const smokeLevel = '1';//set default smoke level as 1.

const co2Level = '1';//set default cO2 level as 1.

const status = 'false';//set default status as false

//Then crate new AlarmSensor object and assign values

const newAlarmSensor = new AlarmSensor({

sensorCode,

floorNumber,

roomNumber,

smokeLevel,

co2Level,

status

});

//if request was success, then crate new alarm sensor record and pass it with successful status.

//if in an error,then pass error status.

newAlarmSensor.save()

.then((alarm) => res.status(200).json({ status: "Success", data: alarm }))

.catch(() => res.status(400).json({ status: "Error", data: {} }));

});

//implement route for get one fire alarm sensor details according to the id.

//if request was success then pass the successful status with relevant alarm details .

//if in an error,then pass error status with empty array.

router.route('/:id').get((req, res) => {

AlarmSensor.findById(req.params.id)

.then(alarm =>res.status(200).json({ status: "Success", data: alarm }))

.catch(() => res.status(400).json({ status: "Error", data:{} }));

});

//implement route for get one fire alarm sensor based on id and delete alarm sensor details

//if request was success then pass then delete the record and pass success status with deleted details.

//if in an error,then record delete process is unsuccessful and pass error status.

router.route('/:id').delete((req, res) => {

AlarmSensor.findByIdAndDelete(req.params.id)

.then((alarm) => res.status(200).json({ status: "Success", data: alarm }))

.catch(() => res.status(400).json({ status: "Error", data:{} }));

});

//implement route for update fire alarm sensor details.

//so that search the sensor details based on the id and update it.

//if request was success then pass then delete the record and pass success status with deleted details.

//if in an error,then record update get error and send error as the status.

router.route('/update/:id').put((req, res) => {

AlarmSensor.findById(req.params.id)

.then(updateAlarm => {

//assign all new details into variables

updateAlarm.sensorCode = req.body.sensorCode;

updateAlarm.floorNumber = req.body.floorNumber;

updateAlarm.roomNumber = req.body.roomNumber;

updateAlarm.smokeLevel = req.body.smokeLevel;

updateAlarm.co2Level = req.body.co2Level;

updateAlarm.status = req.body.status;

updateAlarm.save()

.then((alarm) => res.status(200).json({ status: "Success", data: alarm }))

.catch(() => res.status(400).json({ status: "Error", data:{} }));

})

.catch(() => res.status(400).json({ status: "Error", data:{} }));

});

module.exports = router;

**1.5: routes/user.js**

const express = require('express');

const router = express.Router();

const User = require('../models/User');

const bcrypt = require('bcryptjs');

//implement route for get all user details.

//if request was success then pass the success message and relevant all user details

//if in an error,then pass the error status.

router.route('/').get((req,res) => {

User.find()

.then(users => res.status(200).json({ status: "Success", data: users }))

.catch(() => res.status(400).json({ status: "Error", data:{} }));

});

//Register Handle

//implement route for new users register to the system.

router.route('/register').post((req, res) => {

const { name,email, mobile, password } = req.body;

//if already registered email using to register, then show error status.

User.findOne({email:email})

.then(user => {

if(user){

res.status(400).json({ status: "Error", data:{} });

} else {

//if not then new user details added to new User object.

const newUser = new User({

name,

email,

mobile,

password

});

bcrypt.genSalt(10, (err, salt) => //secure the password, then it hash using bcrypt

bcrypt.hash(newUser.password, salt, (err, hash) => {

if(err) throw err;

//Set password to hashed

newUser.password = hash;

//Save new user details.

//if process was success,then pass the new user details as data and success as the status.

//Or registration goes wrong,then pass error status.

newUser.save()

.then((user) => res.status(200).json({ status: "Success", data: user }))

.catch(() => res.status(400).json({ status: "Error", data: {} }));

}));

}

});

});

//Login Handle

//implement route for user login

router.route('/login').post((req,res) => {

//find specific user according to the email

User.findOne({

email: req.body.email

})

//then check whether email and password are matched.

.then(user => {

if(user){

if(bcrypt.compareSync(req.body.password, user.password)){

const payload = {

\_id : user.\_id,

name: user.name,

mobile: user.mobile,

email: user.email

}

//then password and email matched then pass success status and Authorized as the message.

res.status(200).json({ status: "Success", msg: "Authorized" });

}else {

//then password and email not matched then pass error status and Unauthorized as the message.

res.status(400).json({ status: "Error", msg: "Unauthorized" });

}

}else {

//if email not found, then send error status and message.

res.status(400).json({ status: "Error", msg: "Unauthorized" });

}

})

.catch(err => {

//if user not found, then send error status.

res.status(400).json({ status: "Error", msg: "Unauthorized" });

})

});

module.exports = router;

**2. Web Client Application.**

**2.1: src/App.js**

import React, { Component} from 'react';

import Table from './component/Table';

import './App.css';

class App extends Component {

constructor(props){

super(props);

this.state = {

alarms: [], //create array to store alarm details.

isLoaded: true //create variable for show data loaded or not.

}

}

componentDidMount = async () => {

try {

setInterval(async () => { //set interval for fetch data.

const url = "http://localhost:5000/alarmSensors"; //set the url as get all alarm details route in the rest api.

const response = await fetch(url); //fetch the data and assign response in variable

const result = await response.json(); //and set fetching data into json.

this.setState({alarms:result.data, isLoaded:false}) // then assign that values into alarms variable and isLoaded variable.

},40000);//every 40 seconds refreshed.

} catch (e) {

console.log(e); //if get any error, print error message.

}

};

//render table component,set simple design,if isLoaded variable false, that means values are loading,then show simple message

//if alarms array length is 0, then show message as didn't get a alarm,then all things are fine,then render the table components.

render(){

if(this.state.isLoaded){

return <div className="container text-center mt-lg-5">

<i className="fas fa-spinner fa-5x"></i><br/>

<h1 className="mt-2">Loading...</h1>

</div>

}

if(!this.state.alarms.length){

return <div className="container text-center mt-lg-5">

<h1>Didn't get a Alarm</h1>

</div>

}

return (<Table alarms={this.state.alarms} /> );

}

}

export default App;

**2.2: src/component/Table.js**

import React from "react";

const Table = ({alarms}) => {

//set red color code

const red = {

background: "rgba(237,67,66,0.76)"

};

//set green color code

const green = {

color: "green"

};

//render the Table.

//design simple table design an set all alarm details.

//fetching data set to table and give specific key as alarm id.

//then only we cal identify each record correctly and get details according to the id.

//then co2 level grater than 5, then set column as red and red icon.

//if co2 level less tha 5,then set green color text.

//then smoke level grater than 5, then set column as red and red icon.

//if smoke level less tha 5,then set green color text.

//if co2 level or smoke level grater than 5, then set status as Active with red badge.

//if co2 level or smoke level less than 5, then set status as Inactive with green badge.

return(

<div className="row mt-5">

<div className="col-md-10 m-auto">

<div className="card card-body text-center">

<i className="fab fa-gripfire fa-6x red"></i>

<h1>Fire Alarm Monitoring System</h1>

<div className="row mt-1">

<div className="col-md-12 m-auto">

<div className="card border-primary card-body text-center">

<table className="table table-hover">

<thead>

<tr className="table-primary">

<th scope="col">SENSOR CODE</th>

<th scope="col">FLOOR NUMBER</th>

<th scope="col">ROOM NUMBER</th>

<th scope="col">SMOKE LEVEL</th>

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

<th scope="col">ALARM STATUS</th>

</tr>

</thead>

<tbody>

{alarms.map(fireAlarm => (

<tr key={fireAlarm.\_id}>

<th>{fireAlarm.sensorCode}</th>

<th>{fireAlarm.floorNumber}</th>

<th>{fireAlarm.roomNumber}</th>

<th style={fireAlarm.smokeLevel > 5 ? red : green}>{fireAlarm.smokeLevel} {fireAlarm.smokeLevel > 5 ? <i className="fas fa-exclamation-triangle dark-red"></i>:<i></i>}</th>

<th style={fireAlarm.co2Level > 5 ? red : green}>{fireAlarm.co2Level} {fireAlarm.co2Level > 5 ? <i className="fas fa-exclamation-triangle dark-red"></i>:<i></i>}</th>

<td>{fireAlarm.status?<h4><span className="badge badge-danger">ACTIVE</span> <i class="fas fa-bell red"></i></h4>:<h4><span className="badge badge-success">INACTIVE</span></h4>}</td>

</tr>

))}

</tbody>

</table>

</div>

</div>

</div>

</div>

</div>

</div>

);

} export default Table;

**3. RMI Server Client Service.**

**3.1: User.js**

package com.models;

import java.io.Serializable;

public class User implements Serializable {

private static final long serialVersionUID = 1L;

//model for fire alarms.

private String \_id;

private String name;

private String email;

private String mobile;

private String password;

public String get\_id() {

return \_id;

}

public void set\_id(String \_id) {

this.\_id = \_id;

}

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;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**3.2: FireAlarmSensor.js**

package com.models;

import java.io.Serializable;

public class FireAlarmSensor implements Serializable {

private static final long serialVersionUID = 1L;

//model for fire alarms.

private String \_id;

private String sensorCode;

private int floorNumber;

private int roomNumber;

private int smokeLevel;

private int co2Level;

private boolean status;

public String get\_id() {

return \_id;

}

public void set\_id(String \_id) {

this.\_id = \_id;

}

public String getSensorCode() {

return sensorCode;

}

public void setSensorCode(String sensorCode) {

this.sensorCode = sensorCode;

}

public int getFloorNumber() {

return floorNumber;

}

public void setFloorNumber(int floorNumber) {

this.floorNumber = floorNumber;

}

**3.3: RegisteredUser.js**

package com.models;

import java.io.Serializable;

public class RegisteredUser implements Serializable {

private static final long serialVersionUID = 1L;

//model for store registered user details responses.

private String status;

private User data;

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public User getData() {

return data;

}

public void setData(User data) {

this.data = data;

}

}

**3.4: LoggedUser.js**

package com.models;

import java.io.Serializable;

public class LoggedUser implements Serializable {

private static final long serialVersionUID = 1L;

//model for store logged user details responses.

private String status;

private String msg;

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public String getMsg() {

return msg;

}

public void setMsg(String msg) {

this.msg = msg;

}

}

**3.5: CreatedFireAlarmSensorDetails.js**

package com.models;

import java.io.Serializable;

public class CreatedFireAlarmSensorDetails implements Serializable{

private static final long serialVersionUID = 1L;

//model for store one inserted fire alarm details responses.

private String status;

private FireAlarmSensor data;

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public FireAlarmSensor getData() {

return data;

}

public void setData(FireAlarmSensor data) {

this.data = data;

}

}

**3.6: AllFireAlarmSensorsDetails.js**

package com.models;

import java.io.Serializable;

import java.util.List;

public class AllFireAlarmSensorsDetails implements Serializable{

private static final long serialVersionUID = 1L;

//model for store all inserted fire alarm details responses.

private String status;

private List<FireAlarmSensor> data;

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public List<FireAlarmSensor> getData() {

return data;

}

public void setData(List<FireAlarmSensor> data) {

this.data = data;

}

}

**3.7: FireAlarmServiceInterface.java**

package com.rmi.service.interfaces;

import java.rmi.Remote;

import java.rmi.RemoteException;

import com.models.AllFireAlarmSensorsDetails;

import com.models.CreatedFireAlarmSensorDetails;

import com.models.FireAlarmSensor;

import com.models.LoggedUser;

import com.models.RegisteredUser;

import com.models.User;

public interface FireAlarmServiceInterface extends Remote {

//method for user login.

public LoggedUser login(User loginUser) throws RemoteException;

//method for user registration.

public RegisteredUser register(User registerUser) throws RemoteException;

//method for add new alarm sensor.

public CreatedFireAlarmSensorDetails addFireAlarmSensor(FireAlarmSensor newFireAlarmSensor) throws RemoteException;

////method for update fire alarm sensor details.

public CreatedFireAlarmSensorDetails updateFireAlarmSensor(FireAlarmSensor updadateFireAlarmSensor, String id) throws RemoteException;

//method for get fire alarm sensor details by id.

public CreatedFireAlarmSensorDetails getAlarmSensorDetailsById(String id) throws RemoteException;

//method for delete fire alarm sensor details.

public CreatedFireAlarmSensorDetails deleteFireAlarmSensor(String id) throws RemoteException;

//method for get all fire alarm sensor details.

public AllFireAlarmSensorsDetails getAllFireAlarmSensorDetails() throws RemoteException;

}

**3.8: FireAlarmServiceInterfaceImpl.java**

package com.rmi.service.interfacesImpl;

import java.io.Serializable;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import javax.ws.rs.client.Client;

import javax.ws.rs.client.ClientBuilder;

import javax.ws.rs.client.Entity;

import javax.ws.rs.core.Response;

import com.models.AllFireAlarmSensorsDetails;

import com.models.CreatedFireAlarmSensorDetails;

import com.models.FireAlarmSensor;

import com.models.LoggedUser;

import com.models.RegisteredUser;

import com.models.User;

import com.rmi.service.interfaces.FireAlarmServiceInterface;

public class FireAlarmServiceInterfaceImpl extends UnicastRemoteObject implements FireAlarmServiceInterface, Serializable{

private static final long serialVersionUID = 1L;

//create new Client object.

public Client client = ClientBuilder.newClient();

public FireAlarmServiceInterfaceImpl() throws RemoteException {

super();

}

//implementation of user login method.

public LoggedUser login(User loginUser) throws RemoteException {

//get user response.

Response response = client.target("http://localhost:5000/users/login").request().post(Entity.json(loginUser));

//set Logged user response.

LoggedUser loggedUser = response.readEntity(LoggedUser.class);

//Logged user response.

return loggedUser;

}

//implementation of user register method.

public RegisteredUser register(User registerUser) throws RemoteException {

//get user response.

Response response = client.target("http://localhost:5000/users/register").request().post(Entity.json(registerUser));

//set Registered User response.

RegisteredUser registeredUser = response.readEntity(RegisteredUser.class);

//return Registered User response.

return registeredUser;

}

//implementation of add new fire alarm sensor method.

public CreatedFireAlarmSensorDetails addFireAlarmSensor(FireAlarmSensor newFireAlarmSensor) throws RemoteException {

//get add fire alarm response.

Response response = client.target("http://localhost:5000/alarmSensors/add").request().post(Entity.json(newFireAlarmSensor));

//set CreatedFireAlarmSensorDetails response.

CreatedFireAlarmSensorDetails createdFireAlarmSensorDetails = response.readEntity(CreatedFireAlarmSensorDetails.class);

//return CreatedFireAlarmSensorDetails response.

return createdFireAlarmSensorDetails;

}

//implementation of update alarm sensor details method.

public CreatedFireAlarmSensorDetails updateFireAlarmSensor(FireAlarmSensor updadateFireAlarmSensor, String id) throws RemoteException {

//get update fire alarm response.

Response response = client.target("http://localhost:5000/alarmSensors/update/" + id).request().put(Entity.json(updadateFireAlarmSensor));

//set CreatedFireAlarmSensorDetails response.

CreatedFireAlarmSensorDetails createdFireAlarmSensorDetails = response.readEntity(CreatedFireAlarmSensorDetails.class);

//set CreatedFireAlarmSensorDetails response.

return createdFireAlarmSensorDetails;

}

//implementation of get alarm details by id method.

public CreatedFireAlarmSensorDetails getAlarmSensorDetailsById(String id) throws RemoteException {

//get fire alarm response.

Response response = client.target("http://localhost:5000/alarmSensors/" + id).request().get();

//set CreatedFireAlarmSensorDetails response.

CreatedFireAlarmSensorDetails createdFireAlarmSensorDetails = response.readEntity(CreatedFireAlarmSensorDetails.class);

//set CreatedFireAlarmSensorDetails response.

return createdFireAlarmSensorDetails;

}

//implementation of delete fire alarm sensor method.

public CreatedFireAlarmSensorDetails deleteFireAlarmSensor(String id) throws RemoteException {

//get delete fire alarm response.

Response response = client.target("http://localhost:5000/alarmSensors/" + id).request().delete();

//set CreatedFireAlarmSensorDetails response.

CreatedFireAlarmSensorDetails createdFireAlarmSensorDetails = response.readEntity(CreatedFireAlarmSensorDetails.class);

//set CreatedFireAlarmSensorDetails response.

return createdFireAlarmSensorDetails;

}

//implementation of get all fire alarm sensor details method.

public AllFireAlarmSensorsDetails getAllFireAlarmSensorDetails() throws RemoteException {

//get all fire alarm details response.

Response response = client.target("http://localhost:5000/alarmSensors").request().get();

//set AllFireAlarmSensorsDetails response.

AllFireAlarmSensorsDetails createdFireAlarmSensorDetails = response.readEntity(AllFireAlarmSensorsDetails.class);

//set AllFireAlarmSensorsDetails response.

return createdFireAlarmSensorDetails;

}

}

**3.9: Server.java**

package com.rmi.server;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import com.rmi.service.interfacesImpl.FireAlarmServiceInterfaceImpl;

public class Server extends FireAlarmServiceInterfaceImpl{

private static final long serialVersionUID = 1L;

public Server() throws RemoteException {

super();

}

public static void main(String[] args) throws RemoteException{

try {

//create new FireAlarmServiceInterfaceImpl object.

FireAlarmServiceInterfaceImpl fireAlarmServiceInterfaceImpl = new FireAlarmServiceInterfaceImpl();

//create new Registry object.

//assign port number for create new registry.

Registry registry = LocateRegistry.createRegistry(2099);

//bind registry.

registry.rebind("FireAlarmRMIService", fireAlarmServiceInterfaceImpl);

//print server started message.

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

} catch (Exception e) {

System.err.println("Server exception: " + e.toString());

e.printStackTrace();

}

}

}

**3.10: Client.java**

package com.rmi.client;

import java.io.Serializable;

import java.net.MalformedURLException;

import java.rmi.Naming;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import com.rmi.service.interfaces.FireAlarmServiceInterface;

public class Client implements Serializable{

private static final long serialVersionUID = 1L;

private FireAlarmServiceInterface fireAlarmServiceInterface = null;

//create constructor.

public Client() {

this.setFireAlarmServiceInterface();

}

//get FireAlarmServiceInterface and return value

public FireAlarmServiceInterface getFireAlarmServiceInterface() {

return fireAlarmServiceInterface;

}

//set FireAlarmServiceInterface

public void setFireAlarmServiceInterface() {

try {

//get access for remote object in rmi.

this.fireAlarmServiceInterface = (FireAlarmServiceInterface) Naming.lookup("rmi://localhost:2099/FireAlarmRMIService");

//Handle Exception.

} catch (MalformedURLException e) {

e.printStackTrace();

} catch (RemoteException e) {

e.printStackTrace();

} catch (NotBoundException e) {

e.printStackTrace();

}

}

}

**4. Desktop Application.**

**4.1: userLoginFrame.java**

package com.client.interfaces;

import javax.swing.JOptionPane;

import com.rmi.service.interfaces.FireAlarmServiceInterface;

import com.models.LoggedUser;

import com.models.User;

import com.rmi.client.Client;

public class userLoginFrame extends javax.swing.JFrame {

private static final long serialVersionUID = 1L;

//create constructor.

//call getFireAlarmServiceInterface client object to get rmi server connection.

private FireAlarmServiceInterface fireAlarmServiceInterface = null;

private Client client = new Client();

public userLoginFrame() {

initComponents();

this.setLocationRelativeTo(null);

this.fireAlarmServiceInterface = client.getFireAlarmServiceInterface();

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jPanel1 = new javax.swing.JPanel();

jPanel2 = new javax.swing.JPanel();

jLabel3 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

txtFieldEmail = new javax.swing.JTextField();

txtFieldPassword = new javax.swing.JPasswordField();

userLoginBtn = new javax.swing.JButton();

userRegisterBtn = new javax.swing.JButton();

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

jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/com/client/img/alarm.jpg"))); // NOI18N

jPanel1.setBackground(new java.awt.Color(204, 204, 255));

jPanel2.setBackground(new java.awt.Color(153, 204, 255));

jLabel3.setFont(new java.awt.Font("Algerian", 1, 48)); // NOI18N

jLabel3.setText("LOGIN");

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(177, 177, 177)

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 156, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

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

jLabel2.setText("Email");

jLabel4.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

jLabel4.setText("Password");

txtFieldEmail.setFont(new java.awt.Font("Open Sans Semibold", 0, 18)); // NOI18N

txtFieldPassword.setFont(new java.awt.Font("Open Sans Semibold", 0, 18)); // NOI18N

userLoginBtn.setBackground(new java.awt.Color(51, 102, 255));

userLoginBtn.setFont(new java.awt.Font("Open Sans Semibold", 1, 24)); // NOI18N

userLoginBtn.setText("LOGIN");

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

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

userLoginBtnActionPerformed(evt);

}

});

userRegisterBtn.setBackground(new java.awt.Color(51, 102, 255));

userRegisterBtn.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

userRegisterBtn.setText("REGISTER");

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

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

userRegisterBtnActionPerformed(evt);

}

});

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

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

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

.addGroup(jPanel1Layout.createSequentialGroup()

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

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(79, 79, 79)

.addComponent(jLabel2)

.addGap(17, 17, 17)

.addComponent(txtFieldEmail, javax.swing.GroupLayout.PREFERRED\_SIZE, 330, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(30, 30, 30)

.addComponent(jLabel4)

.addGap(17, 17, 17)

.addComponent(txtFieldPassword, javax.swing.GroupLayout.PREFERRED\_SIZE, 330, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(110, 110, 110)

.addComponent(userLoginBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 332, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(110, 110, 110)

.addComponent(userRegisterBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 332, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(0, 25, Short.MAX\_VALUE))

.addComponent(jPanel2, 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(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(66, 66, 66)

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

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(2, 2, 2)

.addComponent(jLabel2))

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

.addGap(45, 45, 45)

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

.addComponent(jLabel4)

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

.addGap(77, 77, 77)

.addComponent(userLoginBtn)

.addGap(19, 19, 19)

.addComponent(userRegisterBtn)

.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)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel1)

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

.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()

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

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

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

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

);

pack();

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

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

//User click register button,then dispose the this login interface and show register form user interface.

userRegisterFrame userReg = new userRegisterFrame();

userReg.setVisible(true);

userReg.setLocationRelativeTo(null);

this.dispose();

}//GEN-LAST:event\_userRegisterBtnActionPerformed

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

//get email input field value and assign into variable.

String email = txtFieldEmail.getText();

//get password input field value and assign into variable.

String password = String.valueOf(txtFieldPassword.getPassword());

//if email input field is empty,show message.

if(email.trim().equals("")){

JOptionPane.showMessageDialog(rootPane, "Enter Your Email to Login", "Empty Email", 2);

}

//if password input field is empty,show message.

else if(password.trim().equals("")){

JOptionPane.showMessageDialog(rootPane, "Enter Your Password to Login", "Empty Password", 2);

}

else {

//create new User object

User user = new User();

//set email

user.setEmail(email);

//set password

user.setPassword(password);

try {

//create LoggedUser for get responses.

//call the login method in the remote interface and pass User object.

LoggedUser loggedUser = this.fireAlarmServiceInterface.login(user);

//then get the status and assign in to variable.

String checkMsg = loggedUser.getMsg();

//if status "Authorized",that means User Logged in successful.

//redirected to main dashboard.

if(checkMsg.equalsIgnoreCase("Authorized")) {

this.setVisible(false);

String[] logUser = new String[1];

logUser[0] = user.getEmail(); //get logged user email address.

mainDashboardFrame.main(logUser); // open main dashboard.

} else {

//if user giving wrong email or password,show error message.

JOptionPane.showMessageDialog(rootPane, "Wrong Username or Password", "Login Error", 2);

}

} catch (Exception e) {

System.out.println(e);

}

}

}//GEN-LAST:event\_userLoginBtnActionPerformed

public static void main(String args[]) {

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(userLoginFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new userLoginFrame().setVisible(true);

}

});

}

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

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JTextField txtFieldEmail;

private javax.swing.JPasswordField txtFieldPassword;

private javax.swing.JButton userLoginBtn;

private javax.swing.JButton userRegisterBtn;

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

}

**4.2: userRegisterFrame.java**

package com.client.interfaces;

import java.util.regex.Pattern;

import javax.swing.JOptionPane;

import com.models.LoggedUser;

import com.models.RegisteredUser;

import com.models.User;

import com.rmi.client.Client;

import com.rmi.service.interfaces.FireAlarmServiceInterface;

public class userRegisterFrame extends javax.swing.JFrame {

private static final long serialVersionUID = 1L;

//create constructor.

//call getFireAlarmServiceInterface client object to get rmi server connection.

private FireAlarmServiceInterface fireAlarmServiceInterface = null;

private Client client = new Client();

public userRegisterFrame() {

initComponents();

this.fireAlarmServiceInterface = client.getFireAlarmServiceInterface();

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jPanel1 = new javax.swing.JPanel();

jPanel2 = new javax.swing.JPanel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jLabel5 = new javax.swing.JLabel();

jLabel6 = new javax.swing.JLabel();

txtRegName = new javax.swing.JTextField();

txtRegEmail = new javax.swing.JTextField();

txtRegMobile = new javax.swing.JTextField();

txtRegPassword = new javax.swing.JPasswordField();

registerUserBtn = new javax.swing.JButton();

loginUserBtn = new javax.swing.JButton();

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

jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/com/client/img/alarm.jpg"))); // NOI18N

jPanel2.setBackground(new java.awt.Color(153, 204, 255));

jLabel2.setFont(new java.awt.Font("Algerian", 1, 46)); // NOI18N

jLabel2.setText("REGISTer");

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(130, 130, 130)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 387, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel2)

);

jLabel3.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

jLabel3.setText("Name");

jLabel4.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

jLabel4.setText("Email");

jLabel5.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

jLabel5.setText("Mobile");

jLabel6.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

jLabel6.setText("Password");

txtRegName.setFont(new java.awt.Font("Open Sans Semibold", 0, 18)); // NOI18N

txtRegEmail.setFont(new java.awt.Font("Open Sans Semibold", 0, 18)); // NOI18N

txtRegMobile.setFont(new java.awt.Font("Open Sans Semibold", 0, 18)); // NOI18N

txtRegPassword.setFont(new java.awt.Font("Open Sans Semibold", 0, 18)); // NOI18N

registerUserBtn.setBackground(new java.awt.Color(51, 102, 255));

registerUserBtn.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

registerUserBtn.setText("REGISTER");

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

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

registerUserBtnActionPerformed(evt);

}

});

loginUserBtn.setBackground(new java.awt.Color(51, 102, 255));

loginUserBtn.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

loginUserBtn.setText("LOGIN");

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

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

loginUserBtnActionPerformed(evt);

}

});

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

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

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

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(70, 70, 70)

.addComponent(jLabel3)

.addGap(20, 20, 20)

.addComponent(txtRegName, javax.swing.GroupLayout.PREFERRED\_SIZE, 327, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(80, 80, 80)

.addComponent(jLabel4)

.addGap(15, 15, 15)

.addComponent(txtRegEmail, javax.swing.GroupLayout.PREFERRED\_SIZE, 327, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(64, 64, 64)

.addComponent(jLabel5)

.addGap(15, 15, 15)

.addComponent(txtRegMobile, javax.swing.GroupLayout.PREFERRED\_SIZE, 327, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(31, 31, 31)

.addComponent(jLabel6)

.addGap(15, 15, 15)

.addComponent(txtRegPassword, javax.swing.GroupLayout.PREFERRED\_SIZE, 327, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(110, 110, 110)

.addComponent(registerUserBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 332, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(110, 110, 110)

.addComponent(loginUserBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 332, javax.swing.GroupLayout.PREFERRED\_SIZE))

);

jPanel1Layout.setVerticalGroup(

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

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(33, 33, 33)

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

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(1, 1, 1)

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(txtRegName, javax.swing.GroupLayout.PREFERRED\_SIZE, 40, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(25, 25, 25)

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

.addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(txtRegEmail, javax.swing.GroupLayout.PREFERRED\_SIZE, 40, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(31, 31, 31)

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

.addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(txtRegMobile, javax.swing.GroupLayout.PREFERRED\_SIZE, 40, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(32, 32, 32)

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

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(3, 3, 3)

.addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED\_SIZE, 30, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(txtRegPassword, javax.swing.GroupLayout.PREFERRED\_SIZE, 40, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(27, 27, 27)

.addComponent(registerUserBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 44, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(16, 16, 16)

.addComponent(loginUserBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 42, 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)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel1)

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

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 517, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, 0))

);

layout.setVerticalGroup(

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

.addComponent(jLabel1)

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

);

pack();

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

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

//User click login button,then dispose the this register interface and show login form user interface.

userLoginFrame userLog = new userLoginFrame();

userLog.setVisible(true);

userLog.setLocationRelativeTo(null);

this.dispose();

}//GEN-LAST:event\_loginUserBtnActionPerformed

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

//get name input field value and assign into variable.

String name = txtRegName.getText();

//get email input field value and assign into variable.

String email = txtRegEmail.getText();

//get mobile number input field value and assign into variable.

String mobile = txtRegMobile.getText();

//get password input field value and assign into variable.

String password = String.valueOf(txtRegPassword.getPassword());

//if any input field is empty,show message error message.

if(name.trim().equals("") || email.trim().equals("") || mobile.trim().equals("") || password.trim().equals("")){

JOptionPane.showMessageDialog(rootPane, "One Or More Fields Are Empty", "Fill Details Error", JOptionPane.ERROR\_MESSAGE);

}

//if email pattern is wrong ,show error message.

else if(!(Pattern.matches("^[a-zA-Z0-9]+[@]{1}+[a-zA-Z0-9]+[.]{1}+[a-zA-Z0-9]+$", email))){

JOptionPane.showMessageDialog(rootPane, "Please enter valid Email", "Fill Details Error", JOptionPane.ERROR\_MESSAGE);

}

//if mobile number has more than 10 number or string values,show error message.

else if(!(Pattern.matches("^[0-9]{10}+$", mobile))){

JOptionPane.showMessageDialog(rootPane, "Please enter valid phone Number", "Fill Details Error", JOptionPane.ERROR\_MESSAGE);

}

else {

//create new User object

User user = new User();

//set name

user.setName(name);

//set email

user.setEmail(email);

//set mobile number

user.setMobile(mobile);

//set password

user.setPassword(password);

try {

//create RegisteredUser for get responses.

//call the register method in the remote interface and pass User object.

RegisteredUser registeredUser = this.fireAlarmServiceInterface.register(user);

//then get the status and assign in to variable.

String checkStatus = registeredUser.getStatus();

//if status "Success",that means User registered successful.

//redirected to main login interface.

if(checkStatus.equalsIgnoreCase("Success")) {

this.setVisible(false);

String[] regUser = new String[4];

regUser[0] = user.getName(); //get user Name

regUser[1] = user.getEmail(); //get user email

regUser[2] = user.getMobile(); //get user mobile

regUser[3] = user.getPassword(); //get user password

userLoginFrame.main(regUser); //redirect to login interface.

} else {

//if status was "Error",show error message.

JOptionPane.showMessageDialog(rootPane, "Error Occurred While Registration", "Registration Error", 2);

}

} catch (Exception e) {

System.out.println(e);

}

}

}//GEN-LAST:event\_registerUserBtnActionPerformed

public static void main(String args[]) {

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(userRegisterFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new userRegisterFrame().setVisible(true);

}

});

}

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

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JButton loginUserBtn;

private javax.swing.JButton registerUserBtn;

private javax.swing.JTextField txtRegEmail;

private javax.swing.JTextField txtRegMobile;

private javax.swing.JTextField txtRegName;

private javax.swing.JPasswordField txtRegPassword;

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

}

**4.3: addAlarmSensorFrame.java**

package com.client.interfaces;

import com.models.CreatedFireAlarmSensorDetails;

import com.models.FireAlarmSensor;

import javax.swing.JOptionPane;

import com.rmi.client.Client;

import com.rmi.service.interfaces.FireAlarmServiceInterface;

public class addAlarmSensorFrame extends javax.swing.JFrame {

private static final long serialVersionUID = 1L;

//create new FireAlarmServiceInterface object and rmi client object

private FireAlarmServiceInterface fireAlarmServiceInterface = null;

private Client client = new Client();

//create constructor.

//call getFireAlarmServiceInterface client object to get rmi server connection.

public addAlarmSensorFrame() {

initComponents();

this.setLocationRelativeTo(null);

this.fireAlarmServiceInterface = client.getFireAlarmServiceInterface();

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jPanel1 = new javax.swing.JPanel();

jPanel2 = new javax.swing.JPanel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jLabel5 = new javax.swing.JLabel();

textSensorCode = new javax.swing.JTextField();

textRoomNo = new javax.swing.JTextField();

textFloorNo = new javax.swing.JTextField();

cancelBtn = new javax.swing.JButton();

addBtn = new javax.swing.JButton();

jLabel1.setText("jLabel1");

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

jPanel1.setBackground(new java.awt.Color(204, 204, 255));

jPanel2.setBackground(new java.awt.Color(0, 0, 51));

jLabel2.setFont(new java.awt.Font("Trajan Pro 3", 1, 30)); // NOI18N

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

jLabel2.setText("ADD NEW ALARM SENSOR");

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(74, 74, 74)

.addComponent(jLabel2)

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

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, 46, Short.MAX\_VALUE)

.addContainerGap())

);

jLabel3.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

jLabel3.setText("Sensor Code");

jLabel4.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

jLabel4.setText("Floor Number");

jLabel5.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

jLabel5.setText("Room Number");

textSensorCode.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

textRoomNo.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

textFloorNo.setFont(new java.awt.Font("Open Sans Semibold", 0, 24)); // NOI18N

cancelBtn.setBackground(new java.awt.Color(51, 102, 255));

cancelBtn.setFont(new java.awt.Font("Open Sans Semibold", 1, 24)); // NOI18N

cancelBtn.setText("CANCEL");

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

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

cancelBtnActionPerformed(evt);

}

});

addBtn.setBackground(new java.awt.Color(51, 102, 255));

addBtn.setFont(new java.awt.Font("Open Sans Semibold", 1, 24)); // NOI18N

addBtn.setText("ADD");

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

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

addBtnActionPerformed(evt);

}

});

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

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

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

.addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(56, 56, 56)

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

.addGroup(jPanel1Layout.createSequentialGroup()

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

.addComponent(jLabel4)

.addComponent(jLabel3)

.addComponent(jLabel5))

.addGap(45, 45, 45)

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

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

.addComponent(textFloorNo, javax.swing.GroupLayout.PREFERRED\_SIZE, 277, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(textSensorCode, javax.swing.GroupLayout.PREFERRED\_SIZE, 277, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(textRoomNo, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 277, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap())

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(0, 15, Short.MAX\_VALUE)

.addComponent(cancelBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 196, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(60, 60, 60)

.addComponent(addBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 196, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(72, 72, 72))))

);

jPanel1Layout.setVerticalGroup(

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

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(26, 26, 26)

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

.addComponent(jLabel3)

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

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 37, Short.MAX\_VALUE)

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

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

.addComponent(jLabel4))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 45, Short.MAX\_VALUE)

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

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

.addComponent(jLabel5))

.addGap(55, 55, 55)

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

.addComponent(cancelBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 45, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(addBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 45, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(53, 53, 53))

);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

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

);

layout.setVerticalGroup(

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

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

);

pack();

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

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

//if user click cancel button.then dispose the add sensor user interface.

this.dispose();

}//GEN-LAST:event\_cancelBtnActionPerformed

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

//then get user entered data into variables.

String sensorCode = textSensorCode.getText();

String stFloorNo = textFloorNo.getText();

String stRoomNo = textRoomNo.getText();

//if one of these input fields are empty,then show error message.

if(sensorCode.trim().equals("") || stFloorNo.trim().equals("") || stRoomNo.trim().equals("")){

JOptionPane.showMessageDialog(rootPane, "One Or More Fields Are Empty", "Fill Details Error", JOptionPane.ERROR\_MESSAGE);

}

//if all input fields filled and click ADD,then create new FireAlarmSensor object.

// then assign user entered values into new object.

else {

int floorNo = Integer.parseInt(stFloorNo);

int roomNo = Integer.parseInt(stRoomNo);

FireAlarmSensor newAlarm = new FireAlarmSensor();

newAlarm.setSensorCode(sensorCode);

newAlarm.setFloorNumber(floorNo);

newAlarm.setRoomNumber(roomNo);

newAlarm.setSmokeLevel(1); //set default smoke level as 1.

newAlarm.setCo2Level(1); // set default co2 level as 1.

newAlarm.setStatus(false); // set default status as false.

try {

//create CreatedFireAlarmSensorDetails for get server response.

//call the addFireAlarmSensor method in the remote interface and pass FireAlarmSensor object.

CreatedFireAlarmSensorDetails addFireAlarmSensor = this.fireAlarmServiceInterface.addFireAlarmSensor(newAlarm);

//then get the status and assign in to variable.

String checkStatus = addFireAlarmSensor.getStatus();

//if status "Success",that means new record successfully stored.

//show Inserted Successfully message.

if(checkStatus.equalsIgnoreCase("Success")) {

JOptionPane.showMessageDialog(rootPane, "Fire Alarm Inserted Successfully", "Add Fire Alarm", JOptionPane.INFORMATION\_MESSAGE);

//if status "Error",that means new record not inserted

//then show Data Not Inserted message.

} else {

JOptionPane.showMessageDialog(rootPane, "Data Not Inserted", "Add Fire Alarm", JOptionPane.WARNING\_MESSAGE);;

}

//handle exceptions

} catch (Exception e) {

System.out.println(e);

}

}

}//GEN-LAST:event\_addBtnActionPerformed

/\*\*

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

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

});

}

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

private javax.swing.JButton addBtn;

private javax.swing.JButton cancelBtn;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JTextField textFloorNo;

private javax.swing.JTextField textRoomNo;

private javax.swing.JTextField textSensorCode;

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

}

**4.4: mainDashboardFrame.java**

package com.client.interfaces;

import java.util.List;

import java.util.Timer;

import java.util.TimerTask;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

import com.rmi.service.interfaces.FireAlarmServiceInterface;

import com.client.java.SendEmail;

import com.client.java.TableRowColor;

import com.models.AllFireAlarmSensorsDetails;

import com.models.CreatedFireAlarmSensorDetails;

import com.models.FireAlarmSensor;

import com.rmi.client.Client;

public class mainDashboardFrame extends javax.swing.JFrame {

private static final long serialVersionUID = 1L;

//create constructor.

//call getFireAlarmServiceInterface client object to get rmi server connection.

//create new List.

private Client client = new Client();

private FireAlarmServiceInterface fireAlarmServiceInterface = null;

private List<FireAlarmSensor> AlarmList;

public mainDashboardFrame() {

initComponents();

this.fireAlarmServiceInterface = client.getFireAlarmServiceInterface();

this.getAllFireAlarmListAndSetTable(); //call table load method.

this.refreshTable(); //call table refreshed method.

this.setLocationRelativeTo(null);

}

//implement table load method.

public void getAllFireAlarmListAndSetTable() {

try {

//create AllFireAlarmSensorsDetails for get responses.

//call the getAllFireAlarmSensorDetails method in the remote interface.

AllFireAlarmSensorsDetails allFireAlarmSensorsDetails = this.fireAlarmServiceInterface.getAllFireAlarmSensorDetails();

//then get the status and assign in to list.

this.AlarmList = allFireAlarmSensorsDetails.getData();

//print the status.

System.out.println(allFireAlarmSensorsDetails.getStatus());

//set table column names.

String[] column = {"ID", "Sensor Code", "Floor Number", "Room Number", "Smoke Level", "CO2 Level", "Alarm Status"};

//set that column names in default table model.

final DefaultTableModel model = new DefaultTableModel(column, 0);

//run for loop to add Alarm List details into row by row.

for (FireAlarmSensor sensor : this.AlarmList) {

//get values in yhe list one by one and assign to row variable.

Object[] row = {sensor.get\_id(), sensor.getSensorCode(), sensor.getFloorNumber(), sensor.getRoomNumber(), sensor.getSmokeLevel(), sensor.getCo2Level(), sensor.isStatus()};

//set row in the table model.

model.addRow(row);

//if smoke level or CO2 level grater than 5, send email to admin.

if(sensor.getSmokeLevel() > 5 || sensor.getCo2Level() > 5) {

//pass email address and floor number and room number.

SendEmail.sendMail("akilamihiranga484@gmail.com", sensor.getFloorNumber(), sensor.getRoomNumber());

}

//if smoke level or CO2 level less than 5, print fine message.

else {

System.out.println("Sensors OK!");

}

}

//set model to table

AlarmListTable.setModel(model);

//create TableRowColor new object.

TableRowColor renderer = new TableRowColor();

//render the table values design.

AlarmListTable.setDefaultRenderer(Object.class, renderer);

//handle exceptions

} catch (Exception e) {

System.out.println(e);

}

}

//implement table refreshed method.

public void refreshTable() {

Timer time = new Timer();

TimerTask timerTask = new TimerTask() {

@Override

public void run() {

getAllFireAlarmListAndSetTable(); // call table details load method.

System.out.println("Table Refreshed.!"); //print message.

};

};

time.scheduleAtFixedRate(timerTask, 15000, 15000); //in every 15 seconds, table refreshed.

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents

private void initComponents() {

jPanel5 = new javax.swing.JPanel();

jPanel1 = new javax.swing.JPanel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jPanel2 = new javax.swing.JPanel();

addDetailsBtn = new javax.swing.JButton();

updateDetailsBtn = new javax.swing.JButton();

deleteDetailBtn = new javax.swing.JButton();

jPanel3 = new javax.swing.JPanel();

jScrollPane1 = new javax.swing.JScrollPane();

AlarmListTable = new javax.swing.JTable();

tableRefreshBtn = new javax.swing.JButton();

jPanel4 = new javax.swing.JPanel();

jLabel4 = new javax.swing.JLabel();

txtID = new javax.swing.JLabel();

txtSensorCode = new javax.swing.JTextField();

txtFloorNo = new javax.swing.JTextField();

txtRoomNo = new javax.swing.JTextField();

txtSmokeLevel = new javax.swing.JSpinner();

txtCO2Level = new javax.swing.JSpinner();

jPanel7 = new javax.swing.JPanel();

jLabel12 = new javax.swing.JLabel();

jLabel6 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();

jLabel8 = new javax.swing.JLabel();

jLabel5 = new javax.swing.JLabel();

jLabel9 = new javax.swing.JLabel();

jLabel10 = new javax.swing.JLabel();

clearFieldsBtn = new javax.swing.JButton();

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

jPanel6 = new javax.swing.JPanel();

logoutBtn = new javax.swing.JButton();

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

jPanel5.setBackground(new java.awt.Color(0, 0, 51));

jPanel1.setBackground(new java.awt.Color(204, 0, 0));

jLabel1.setFont(new java.awt.Font("Algerian", 1, 48)); // NOI18N

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

jLabel1.setText("Fire Alarm Monitoring System");

jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/com/client/img/alarm2.jpg"))); // NOI18N

jLabel3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/com/client/img/alarm2.jpg"))); // NOI18N

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, javax.swing.GroupLayout.PREFERRED\_SIZE, 130, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(42, 42, 42)

.addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 63, Short.MAX\_VALUE)

.addComponent(jLabel3))

);

jPanel1Layout.setVerticalGroup(

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

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 80, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(13, 13, 13)

.addComponent(jLabel1))

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 80, javax.swing.GroupLayout.PREFERRED\_SIZE)

);

jPanel2.setBackground(new java.awt.Color(204, 204, 255));

addDetailsBtn.setBackground(new java.awt.Color(102, 153, 255));

addDetailsBtn.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N

addDetailsBtn.setText("ADD");

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

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

addDetailsBtnActionPerformed(evt);

}

});

updateDetailsBtn.setBackground(new java.awt.Color(102, 153, 255));

updateDetailsBtn.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N

updateDetailsBtn.setText("UPDATE");

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

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

updateDetailsBtnActionPerformed(evt);

}

});

deleteDetailBtn.setBackground(new java.awt.Color(102, 153, 255));

deleteDetailBtn.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N

deleteDetailBtn.setText("DELETE");

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

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

deleteDetailBtnActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

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

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

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

.addContainerGap())

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(116, 116, 116)

.addComponent(addDetailsBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 65, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(71, 71, 71)

.addComponent(updateDetailsBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 65, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(75, 75, 75)

.addComponent(deleteDetailBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 65, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap())

);

jPanel3.setBackground(new java.awt.Color(204, 204, 255));

AlarmListTable.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

{null, null, null, null, null, null, null},

{null, null, null, null, null, null, null},

{null, null, null, null, null, null, null},

{null, null, null, null, null, null, null},

{null, null, null, null, null, null, null}

},

new String [] {

"ID", "Sensor Code", "Floor Number", "Room Number", "Smoke Level", "CO2 Level", "Alarm Status"

}

));

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

AlarmListTable.setRowHeight(36);

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

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

AlarmListTableMouseClicked(evt);

}

});

jScrollPane1.setViewportView(AlarmListTable);

tableRefreshBtn.setBackground(new java.awt.Color(102, 153, 255));

tableRefreshBtn.setFont(new java.awt.Font("Sitka Text", 1, 24)); // NOI18N

tableRefreshBtn.setText("REFRESH TABLE");

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

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

tableRefreshBtnActionPerformed(evt);

}

});

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

jPanel3.setLayout(jPanel3Layout);

jPanel3Layout.setHorizontalGroup(

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

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

.addContainerGap()

.addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT\_SIZE, 619, Short.MAX\_VALUE)

.addContainerGap())

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

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

.addComponent(tableRefreshBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 389, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(109, 109, 109))

);

jPanel3Layout.setVerticalGroup(

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

.addGroup(jPanel3Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 509, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(tableRefreshBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 41, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

);

jPanel4.setBackground(new java.awt.Color(204, 204, 255));

jLabel4.setFont(new java.awt.Font("Times New Roman", 1, 21)); // NOI18N

jLabel4.setText("ID");

txtID.setFont(new java.awt.Font("Times New Roman", 0, 14)); // NOI18N

txtID.setText("00000000000000000");

txtSensorCode.setFont(new java.awt.Font("Times New Roman", 0, 22)); // NOI18N

txtFloorNo.setFont(new java.awt.Font("Times New Roman", 0, 22)); // NOI18N

txtRoomNo.setFont(new java.awt.Font("Times New Roman", 0, 22)); // NOI18N

txtSmokeLevel.setFont(new java.awt.Font("Times New Roman", 0, 22)); // NOI18N

txtSmokeLevel.setModel(new javax.swing.SpinnerNumberModel(1, 1, 10, 1));

txtCO2Level.setFont(new java.awt.Font("Times New Roman", 0, 22)); // NOI18N

txtCO2Level.setModel(new javax.swing.SpinnerNumberModel(1, 1, 10, 1));

jPanel7.setBackground(new java.awt.Color(0, 0, 51));

jLabel12.setFont(new java.awt.Font("Trajan Pro 3", 1, 24)); // NOI18N

jLabel12.setForeground(new java.awt.Color(255, 255, 255));

jLabel12.setText("Update Details");

javax.swing.GroupLayout jPanel7Layout = new javax.swing.GroupLayout(jPanel7);

jPanel7.setLayout(jPanel7Layout);

jPanel7Layout.setHorizontalGroup(

jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel7Layout.createSequentialGroup()

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

.addComponent(jLabel12, javax.swing.GroupLayout.PREFERRED\_SIZE, 257, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap())

);

jPanel7Layout.setVerticalGroup(

jPanel7Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel7Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel12, javax.swing.GroupLayout.DEFAULT\_SIZE, 29, Short.MAX\_VALUE)

.addContainerGap())

);

jLabel6.setFont(new java.awt.Font("Times New Roman", 1, 21)); // NOI18N

jLabel6.setText("Floor Number");

jLabel7.setFont(new java.awt.Font("Times New Roman", 1, 21)); // NOI18N

jLabel7.setText("Room Number");

jLabel8.setFont(new java.awt.Font("Times New Roman", 1, 21)); // NOI18N

jLabel8.setText("Smoke Level");

jLabel5.setFont(new java.awt.Font("Times New Roman", 1, 21)); // NOI18N

jLabel5.setText("Sensor Code");

jLabel9.setFont(new java.awt.Font("Times New Roman", 1, 21)); // NOI18N

jLabel9.setText("CO2 Level");

jLabel10.setFont(new java.awt.Font("Times New Roman", 1, 21)); // NOI18N

jLabel10.setText("Alarm Status");

clearFieldsBtn.setBackground(new java.awt.Color(102, 153, 255));

clearFieldsBtn.setFont(new java.awt.Font("Tahoma", 1, 16)); // NOI18N

clearFieldsBtn.setText("CLEAR FIELDS");

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

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

clearFieldsBtnActionPerformed(evt);

}

});

txtStatus.setFont(new java.awt.Font("Times New Roman", 0, 22)); // NOI18N

txtStatus.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "false", "true" }));

javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);

jPanel4.setLayout(jPanel4Layout);

jPanel4Layout.setHorizontalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createSequentialGroup()

.addContainerGap()

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel7, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(jPanel4Layout.createSequentialGroup()

.addGap(35, 35, 35)

.addComponent(jLabel4)

.addGap(18, 18, 18)

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

.addGroup(jPanel4Layout.createSequentialGroup()

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createSequentialGroup()

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addComponent(jLabel6)

.addComponent(jLabel7)

.addComponent(jLabel8)

.addComponent(jLabel5)

.addComponent(jLabel9)

.addComponent(jLabel10))

.addGap(18, 18, 18)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(txtSensorCode, javax.swing.GroupLayout.PREFERRED\_SIZE, 140, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(txtRoomNo)

.addComponent(txtSmokeLevel, javax.swing.GroupLayout.DEFAULT\_SIZE, 143, Short.MAX\_VALUE)

.addComponent(txtCO2Level)

.addComponent(txtFloorNo)

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

.addComponent(clearFieldsBtn))

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

.addContainerGap())

);

jPanel4Layout.setVerticalGroup(

jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel4Layout.createSequentialGroup()

.addGap(7, 7, 7)

.addComponent(jPanel7, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(35, 35, 35)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel4)

.addComponent(txtID))

.addGap(18, 18, 18)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

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

.addComponent(jLabel5))

.addGap(18, 18, 18)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

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

.addComponent(jLabel6))

.addGap(27, 27, 27)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

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

.addComponent(jLabel7))

.addGap(30, 30, 30)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(txtSmokeLevel, javax.swing.GroupLayout.PREFERRED\_SIZE, 36, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel8))

.addGap(32, 32, 32)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(txtCO2Level, javax.swing.GroupLayout.PREFERRED\_SIZE, 36, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel9))

.addGap(40, 40, 40)

.addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel10)

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

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 66, Short.MAX\_VALUE)

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

.addGap(22, 22, 22))

);

jPanel6.setBackground(new java.awt.Color(204, 204, 255));

logoutBtn.setBackground(new java.awt.Color(102, 153, 255));

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

logoutBtn.setText("LOGOUT");

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

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

logoutBtnActionPerformed(evt);

}

});

javax.swing.GroupLayout jPanel6Layout = new javax.swing.GroupLayout(jPanel6);

jPanel6.setLayout(jPanel6Layout);

jPanel6Layout.setHorizontalGroup(

jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel6Layout.createSequentialGroup()

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

.addComponent(logoutBtn, javax.swing.GroupLayout.PREFERRED\_SIZE, 130, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap())

);

jPanel6Layout.setVerticalGroup(

jPanel6Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(logoutBtn, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT\_SIZE, 38, Short.MAX\_VALUE)

);

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

jPanel5.setLayout(jPanel5Layout);

jPanel5Layout.setHorizontalGroup(

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

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

.addGroup(jPanel5Layout.createSequentialGroup()

.addContainerGap()

.addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

.addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

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

.addContainerGap())

.addComponent(jPanel6, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

);

jPanel5Layout.setVerticalGroup(

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

.addGroup(jPanel5Layout.createSequentialGroup()

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

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

.addComponent(jPanel6, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

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

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

.addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addContainerGap())

);

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

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

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

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

);

layout.setVerticalGroup(

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

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

);

pack();

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

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

//if any field is empty,show message error message.

if (txtID.getText().equals("00000000000000000") || txtSensorCode.getText().equals("") || txtFloorNo.getText().equals("") || txtRoomNo.getText().equals("") ||

txtSmokeLevel.getValue().toString().equals("") || txtCO2Level.getValue().toString().equals("")) {

JOptionPane.showMessageDialog(rootPane, "Required Fields are Empty OR Invalid Data", "Empty Fields", JOptionPane.WARNING\_MESSAGE);

}

else {

//Asking confirmation for delete the record.

int x = JOptionPane.showConfirmDialog(null, "Do You Really Want to Delete this Record.?");

//if click yes, s value is 0, click no, s value is 1.

if (x == 0) {

//get id field value and assign into variable.

String id = txtID.getText();

try {

//create CreatedFireAlarmSensorDetails for get responses.

//call the deleteFireAlarmSensor method in the remote interface and pass id.

CreatedFireAlarmSensorDetails deleteFireAlarmSensor = this.fireAlarmServiceInterface.deleteFireAlarmSensor(id);

//then get the status and assign in to variable.

String checkStatus = deleteFireAlarmSensor.getStatus();

//if status "Success",that means Alarm Deleted successful.

if(checkStatus.equalsIgnoreCase("Success")) {

this.getAllFireAlarmListAndSetTable(); //call table load method to refreshed the table.

this.clearFields(); //clear fields.

//show delete successful message

JOptionPane.showMessageDialog(rootPane, "Fire Alarm Deleted Successfully", "Delete Fire Alarm", JOptionPane.INFORMATION\_MESSAGE);

} else {

//if status was "Error",show error message.

JOptionPane.showMessageDialog(rootPane, "Data Not Deleted", "Delete Fire Alarm", JOptionPane.ERROR\_MESSAGE);;

}

} catch (Exception e) {

System.out.println(e);

}

}

else{

//if user click no or cancel,show error message.

JOptionPane.showMessageDialog(rootPane, "Data Not Deleted", "Delete Fire Alarm Details", JOptionPane.ERROR\_MESSAGE);

}

}

}//GEN-LAST:event\_deleteDetailBtnActionPerformed

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

//User click ADD button,then show add new alarm sensor form user interface.

addAlarmSensorFrame addAlarm = new addAlarmSensorFrame();

addAlarm.setVisible(true);

addAlarm.setLocationRelativeTo(null);

addAlarm.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

}//GEN-LAST:event\_addDetailsBtnActionPerformed

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

//if any field is empty,show message error message.

if (txtID.getText().equals("00000000000000000") || txtSensorCode.getText().equals("") || txtFloorNo.getText().equals("") || txtRoomNo.getText().equals("") ||

txtSmokeLevel.getValue().toString().equals("") || txtCO2Level.getValue().toString().equals("")) {

JOptionPane.showMessageDialog(rootPane, "Required Fields are Empty OR Invalid Data", "Empty Fields", JOptionPane.WARNING\_MESSAGE);

}

else {

//if click yes, s value is 0, click no, s value is 1.

int x = JOptionPane.showConfirmDialog(null, "Do You Really Want to Update.?");

//if click yes, s value is 0, click no, s value is 1.

if (x == 0) {

//get input fields value and assign into variables.

String id = txtID.getText();

String uSensorCode = txtSensorCode.getText();

int uFloorNo = Integer.parseInt(txtFloorNo.getText());

int uRoomNo = Integer.parseInt(txtRoomNo.getText());

int uSmokeLevel = (int) txtSmokeLevel.getValue();

int uCO2Level = (int) txtCO2Level.getValue();

boolean uStatus;

//if smoke level or CO2 level grater than 5,set status as "true".

if(uSmokeLevel > 5 || uCO2Level > 5) {

uStatus = Boolean.parseBoolean("true");

}

//if smoke level or CO2 level less than 5,set status as "false".

else {

uStatus = Boolean.parseBoolean("false");

}

//create new FireAlarmSensor object and set values

FireAlarmSensor updateAlarm = new FireAlarmSensor();

updateAlarm.setSensorCode(uSensorCode); //set sensor code.

updateAlarm.setFloorNumber(uFloorNo); //set floor number.

updateAlarm.setRoomNumber(uRoomNo); //set room number.

updateAlarm.setSmokeLevel(uSmokeLevel); // set smoke level.

updateAlarm.setCo2Level(uCO2Level); //set CO2 level.

updateAlarm.setStatus(uStatus); //set status.

try {

//create CreatedFireAlarmSensorDetails for get responses.

//call the updateFireAlarmSensor method in the remote interface and pass updateAlarm object and id.

CreatedFireAlarmSensorDetails updateFireAlarmSensor = this.fireAlarmServiceInterface.updateFireAlarmSensor(updateAlarm, id);

//then get the status and assign in to variable.

String checkStatus = updateFireAlarmSensor.getStatus();

//if status "Success",that means Alarm Updated successful.

if(checkStatus.equalsIgnoreCase("Success")) {

this.getAllFireAlarmListAndSetTable(); //call table load method to refreshed the table.

//show update successful message

JOptionPane.showMessageDialog(rootPane, "Fire Alarm Updated Successfully", "Update Fire Alarm", JOptionPane.INFORMATION\_MESSAGE);

} else {

//if status was "Error",show error message.

JOptionPane.showMessageDialog(rootPane, "Data Not Updated", "Update Fire Alarm", JOptionPane.ERROR\_MESSAGE);;

}

} catch (Exception e) {

System.out.println(e);

}

}

else{

//if user click no or cancel,show error message.

JOptionPane.showMessageDialog(rootPane, "Data Not Updated", "Update Fire Alarm Details", JOptionPane.ERROR\_MESSAGE);

}

}

}//GEN-LAST:event\_updateDetailsBtnActionPerformed

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

try {

//get user clicked table row number and assign to variable.

int row = AlarmListTable.getSelectedRow();

//get row values and assign to variables

String tId = AlarmListTable.getValueAt(row, 0).toString();

String tSensorCode = AlarmListTable.getValueAt(row, 1).toString();

String tFloorNo = AlarmListTable.getValueAt(row, 2).toString();

String tRoomNo = AlarmListTable.getValueAt(row, 3).toString();

int tSmokeLevel = (int) AlarmListTable.getValueAt(row, 4);

int tCO2Level = (int) AlarmListTable.getValueAt(row, 5);

String tStatus = AlarmListTable.getValueAt(row, 6).toString();

//set data into input fields

txtID.setText(tId);

txtSensorCode.setText(tSensorCode);

txtFloorNo.setText(tFloorNo);

txtRoomNo.setText(tRoomNo);

txtSmokeLevel.setValue(tSmokeLevel);

txtCO2Level.setValue(tCO2Level);

txtStatus.setSelectedItem(tStatus);

} catch (Exception e) {

System.out.println(e);

}

}//GEN-LAST:event\_AlarmListTableMouseClicked

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

//if user clicked clear fields button,then clear fields.

this.clearFields();

}//GEN-LAST:event\_clearFieldsBtnActionPerformed

public void clearFields() {

//if any input field is empty,show message error message.

if (txtID.getText().equals("00000000000000000") && txtSensorCode.getText().equals("") && txtFloorNo.getText().equals("") && txtRoomNo.getText().equals("") &&

txtSmokeLevel.getValue().toString().equals("") && txtCO2Level.getValue().toString().equals("")) {

JOptionPane.showMessageDialog(rootPane, "Already Fields are Empty", "Empty Fields", JOptionPane.INFORMATION\_MESSAGE);

}

else{

//set defaults values to input fields

txtID.setText("00000000000000000");

txtSensorCode.setText("");

txtFloorNo.setText("");

txtRoomNo.setText("");

txtSmokeLevel.setValue(1);

txtCO2Level.setValue(1);

txtStatus.setSelectedItem("true");

}

}

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

//User click logout button,then dispose the this main dashboard interface and show login form user interface.

userLoginFrame logout = new userLoginFrame();

logout.setVisible(true);

logout.setLocationRelativeTo(null);

this.dispose();

}//GEN-LAST:event\_logoutBtnActionPerformed

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

//call table load method

this.getAllFireAlarmListAndSetTable();

}//GEN-LAST:event\_tableRefreshBtnActionPerformed

public static void main(String args[]) {

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(mainDashboardFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new mainDashboardFrame().setVisible(true);

}

});

}

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

private javax.swing.JTable AlarmListTable;

private javax.swing.JButton addDetailsBtn;

private javax.swing.JButton clearFieldsBtn;

private javax.swing.JButton deleteDetailBtn;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel10;

private javax.swing.JLabel jLabel12;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JLabel jLabel8;

private javax.swing.JLabel jLabel9;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel3;

private javax.swing.JPanel jPanel4;

private javax.swing.JPanel jPanel5;

private javax.swing.JPanel jPanel6;

private javax.swing.JPanel jPanel7;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JButton logoutBtn;

private javax.swing.JButton tableRefreshBtn;

private javax.swing.JSpinner txtCO2Level;

private javax.swing.JTextField txtFloorNo;

private javax.swing.JLabel txtID;

private javax.swing.JTextField txtRoomNo;

private javax.swing.JTextField txtSensorCode;

private javax.swing.JSpinner txtSmokeLevel;

private javax.swing.JComboBox<String> txtStatus;

private javax.swing.JButton updateDetailsBtn;

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

}

**4.5: SendEmail.java**

package com.client.java;

import java.util.Properties;

import javax.mail.Authenticator;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

public class SendEmail {

public static void sendMail(String recepient,int floor,int room) throws MessagingException {

System.out.println("Preparing to Send Emergency Alert Email.!");

Properties properties = new Properties();

//set properties

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

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

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

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

//set senders email address and password

String adminEmail = "akilamihiranganew484@gmail.com";

String adminPassword = "DSassignment@8990";

Session session = Session.getInstance(properties, new Authenticator() {

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(adminEmail, adminPassword);

}

});

//prepare email

Message emergencyMessage = prepareMessage(session,adminEmail,recepient, floor, room);

//send email

Transport.send(emergencyMessage);

//show successful message.

System.out.println("Emergency Message Send Successfully.!");

}

private static Message prepareMessage(Session session, String adminEmail, String recepient,int floor,int room) {

try {

//create new Message object.

Message setMessage = new MimeMessage(session);

//set email form.

setMessage.setFrom(new InternetAddress(adminEmail));

//set Recipient.

setMessage.setRecipient(Message.RecipientType.TO, new InternetAddress(recepient));

//set email subject.

setMessage.setSubject("Emergency.!Fire Alarm Sensor Active.");

//set email text.

setMessage.setText("!!!!! EMERGENCY !!!!! \nFloor Number : " + floor + " , Room Number : " + room + "\nTake an Immediate Action.!");

//return emai message

return setMessage;

} catch (Exception e) {

System.out.println("Error" + e);

}

return null;

}

}

**4.6: TableRowColor.java**

package com.client.java;

import java.awt.Component;

import javax.swing.JTable;

import javax.swing.table.DefaultTableCellRenderer;

import javax.swing.table.TableCellRenderer;

public class TableRowColor implements TableCellRenderer {

private static final TableCellRenderer RENDERER = new DefaultTableCellRenderer();

@Override

public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus,

int row, int column) {

Component c = RENDERER.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);

if (column == 4) {

//get column value

Object result = table.getModel().getValueAt(row, column);

int smokeLevel = Integer.parseInt(result.toString());

//if smoke level grater than 5,then show it in red color.

if (smokeLevel > 5 && smokeLevel <= 10) {

c.setForeground(java.awt.Color.RED);

//if smoke level less than 5,then show it in black color.

} else if (smokeLevel < 5) {

c.setForeground(java.awt.Color.BLACK);

}

} else if (column == 5) {

//get column value

Object result = table.getModel().getValueAt(row, column);

int co2Level = Integer.parseInt(result.toString());

//if CO2 level grater than 5,then show it in red color.

if (co2Level > 5 && co2Level <= 10) {

c.setForeground(java.awt.Color.RED);

//if smoke level less than 5,then show it in black color.

} else if (co2Level < 5) {

c.setForeground(java.awt.Color.BLACK);

}

} else if (column == 6) {

Object result = table.getModel().getValueAt(row, column);

String status = result.toString();

//if status equal "true",then show it in red color.

if (status.equals("true")) {

c.setForeground(java.awt.Color.RED);

//if status equal "false",then show it in green color.

} else if (status.equals("false")) {

c.setForeground(java.awt.Color.GREEN);

}

} else {

//show in black color.

c.setForeground(java.awt.Color.BLACK);

}

return c;

}

}

**5. Simulate Application. (Sensor App)**

**5.1: App.js**

import React from 'react';

import './App.css';

//import all sensor components

//then only that components.

import SensorOne from './component/SensorOneSimulate';

import SensorTwo from './component/SensorTwoSimulate';

import SensorThree from './component/SensorThreeSimulate';

import SensorFour from './component/SensorFourSimulate';

import SensorFive from './component/SensorFiveSimulate';

//render simple user interface

//design simple design

//and then user imported sensors for render

function App() {

return (

<div className="row mt-5">

<div className="col-md-10 m-auto">

<div className="card card-body text-center">

<i className="fab fa-gripfire fa-6x red"></i>

<h1>Fire Alarm Monitoring System</h1>

<SensorOne></SensorOne>

<SensorTwo></SensorTwo>

<SensorThree></SensorThree>

<SensorFour></SensorFour>

<SensorFive></SensorFive>

</div>

</div>

</div>

); } export default App;

**5.2: SensorOneSimulate.js**

import React, {Component} from 'react';

const axios = require('axios');

export class SensorOneSimulate extends Component {

async AlarmSensorOneSendData() {

try {

//Simply set the values for the sensor

const id = "5eadc0cd189bdd4098045a72";

const sensorCode = "A1";

const floorNumber = 1;

const roomNumber = 5;

const min = 1;

const max = 10;

//Create random value for smoke level and co2 level.

//that auto generated value in between 1 and 10

const smokeLevel = Math.floor(Math.random() \* max) + min;

const co2Level = Math.floor(Math.random() \* max) + min;

//if generated smoke level or co2 level grater than 5,then status set as true.

if(smokeLevel > 5 || co2Level > 5){

const status = "true";

await axios.request({

method: "PUT", //use put method for get access with api route

headers: {

"Content-Type": "application/json" // set header values

},

url: `http://localhost:5000/alarmSensors/update/${id}`, //then call the rest api update fire alarm route.

//then pass the new data.

data: JSON.stringify({ sensorCode, floorNumber, roomNumber, smokeLevel, co2Level, status }),

});

}

//if generated smoke level or co2 level less than 5,then status set as false.

else {

const status = "false";

await axios.request({

method: "PUT", //use put method for get access with api route

headers: {

"Content-Type": "application/json" // set header values

},

url: `http://localhost:5000/alarmSensors/update/${id}`, //then call the rest api update fire alarm route.

//then pass the new data.

data: JSON.stringify({ sensorCode, floorNumber, roomNumber, smokeLevel, co2Level, status }),

});

}

}

catch (error) { //if get any error then show error message.

console.log(error);

}

}

async componentDidMount() {

try {

this.AlarmSensorOneSendData();

setInterval(async () => { //set interval for the process

this.AlarmSensorOneSendData();

}, 10000); // in every 10 seconds pass the values.

}

catch (error) { // if get any error then print error message.

console.log(error);

}

}

//render simple user interface

//set simple design

render() {

return (

<div className="col-md-12 m-auto">

<div className="card border-primary card-body">

<h2>Fire Alarm Sensor One Sending Data... <i class="fas fa-arrow-alt-circle-right"></i></h2>

</div>

</div>

);

}

}

export default SensorOneSimulate;

**5.3: SensorTwoSimulate.js**

import React, {Component} from 'react';

const axios = require('axios');

export class SensorTwoSimulate extends Component {

async AlarmSensorTwoSendData() {

try {

const id = "5eadc0ec189bdd4098045a73";

const sensorCode = "C3";

const floorNumber = 3;

const roomNumber = 32;

const min = 1;

const max = 10;

const smokeLevel = Math.floor(Math.random() \* max) + min;

const co2Level = Math.floor(Math.random() \* max) + min;

if(smokeLevel > 5 || co2Level > 5){

const status = "true";

await axios.request({

method: "PUT",

headers: {

"Content-Type": "application/json"

},

url: `http://localhost:5000/alarmSensors/update/${id}`,

data: JSON.stringify({ sensorCode, floorNumber, roomNumber, smokeLevel, co2Level, status }),

});

}else {

const status = "false";

await axios.request({

method: "PUT",

headers: {

"Content-Type": "application/json" },

url: `http://localhost:5000/alarmSensors/update/${id}`,

data: JSON.stringify({ sensorCode, floorNumber, roomNumber, smokeLevel, co2Level, status }),

});

}

}

catch (error) {

console.log(error);

}

}

async componentDidMount() {

try {

this.AlarmSensorTwoSendData();

setInterval(async () => {

this.AlarmSensorTwoSendData();

}, 10000);

}

catch (error) {

console.log(error);

}

}

render() {

return (

<div className="col-md-12 m-auto">

<div className="card border-primary card-body">

<h2>Fire Alarm Sensor Two Sending Data... <i class="fas fa-arrow-alt-circle-right"></i></h2>

</div>

</div>

);

}

} export default SensorTwoSimulate;

**5.4: SensorThreeSimulate.js**

import React, {Component} from 'react';

const axios = require('axios');

export class SensorThreeSimulate extends Component {

async AlarmSensorThreeSendData() {

try {

const id = "5eadc127189bdd4098045a74";

const sensorCode = "B1";

const floorNumber = 2;

const roomNumber = 25;

const min = 1;

const max = 10;

const smokeLevel = Math.floor(Math.random() \* max) + min;

const co2Level = Math.floor(Math.random() \* max) + min;

if(smokeLevel > 5 || co2Level > 5){

const status = "true";

await axios.request({

method: "PUT",

headers: {

"Content-Type": "application/json"

},

url: `http://localhost:5000/alarmSensors/update/${id}`,

data: JSON.stringify({ sensorCode, floorNumber, roomNumber, smokeLevel, co2Level, status }),

});

}else {

const status = "false";

await axios.request({

method: "PUT",

headers: {

"Content-Type": "application/json" },

url: `http://localhost:5000/alarmSensors/update/${id}`,

data: JSON.stringify({ sensorCode, floorNumber, roomNumber, smokeLevel, co2Level, status }),

});

}

} catch (error) {

console.log(error);

}

}

async componentDidMount() {

try {

this.AlarmSensorThreeSendData();

setInterval(async () => {

this.AlarmSensorThreeSendData();

}, 10000);

}

catch (error) {

console.log(error);

}

}

render() {

return (

<div className="col-md-12 m-auto">

<div className="card border-primary card-body">

<h2>Fire Alarm Sensor Three Sending Data... <i class="fas fa-arrow-alt-circle-right"></i></h2>

</div>

</div>

);

}

} export default SensorThreeSimulate;

**5.5: SensorFourSimulate.js**

import React, {Component} from 'react';

const axios = require('axios');

export class SensorFourSimulate extends Component {

async AlarmSensorFourSendData() {

try {

const id = "5eadc132189bdd4098045a75";

const sensorCode = "A2";

const floorNumber = 1;

const roomNumber = 10;

const min = 1;

const max = 10;

const smokeLevel = Math.floor(Math.random() \* max) + min;

const co2Level = Math.floor(Math.random() \* max) + min;

if(smokeLevel > 5 || co2Level > 5){

const status = "true";

await axios.request({

method: "PUT",

headers: {

"Content-Type": "application/json"

},

url: `http://localhost:5000/alarmSensors/update/${id}`,

data: JSON.stringify({ sensorCode, floorNumber, roomNumber, smokeLevel, co2Level, status }),

});

}

else {

const status = "false";

await axios.request({

method: "PUT",

headers: {

"Content-Type": "application/json"

},

url: `http://localhost:5000/alarmSensors/update/${id}`,

data: JSON.stringify({ sensorCode, floorNumber, roomNumber, smokeLevel, co2Level, status }),

});

}

} catch (error) {

console.log(error);

}

}

async componentDidMount() {

try {

this.AlarmSensorFourSendData();

setInterval(async () => {

this.AlarmSensorFourSendData();

}, 10000);

}catch (error) {

console.log(error);

}

}

render() {

return (

<div className="col-md-12 m-auto">

<div className="card border-primary card-body">

<h2>Fire Alarm Sensor Four Sending Data... <i class="fas fa-arrow-alt-circle-right"></i></h2>

</div>

</div>

);

}

} export default SensorFourSimulate;

**5.6: SensorFiveSimulate.js**

import React, {Component} from 'react';

const axios = require('axios');

export class SensorFiveSimulate extends Component {

async AlarmSensorFiveSendData() {

try {

//Simply set the values for the sensor

const id = "5eadc15c189bdd4098045a76";

const sensorCode = "B2";

const floorNumber = 2;

const roomNumber = 15;

const min = 1;

const max = 10;

//Create random value for smoke level and co2 level.

//that auto generated value in between 1 and 10

const smokeLevel = Math.floor(Math.random() \* max) + min;

const co2Level = Math.floor(Math.random() \* max) + min;

//if generated smoke level or co2 level grater than 5,then status set as true.

if(smokeLevel > 5 || co2Level > 5){

const status = "true";

await axios.request({

method: "PUT", //use put method for get access with api route

headers: {

"Content-Type": "application/json" // set header values

},

url: `http://localhost:5000/alarmSensors/update/${id}`, //then call the rest api update fire alarm route.

//then pass the new data.

data: JSON.stringify({ sensorCode, floorNumber, roomNumber, smokeLevel, co2Level, status }),

});

}

else {

//if generated smoke level or co2 level less than 5,then status set as false.

const status = "false";

await axios.request({

method: "PUT", //use put method for get access with api route

headers: {

"Content-Type": "application/json" // set header values

},

url: `http://localhost:5000/alarmSensors/update/${id}`, //then call the rest api update fire alarm route.

//then pass the new data.

data: JSON.stringify({ sensorCode, floorNumber, roomNumber, smokeLevel, co2Level, status }),

});

}

}

catch (error) { //if get any error then show error message.

console.log(error);

}

}

async componentDidMount() {

try {

this.AlarmSensorFiveSendData();

setInterval(async () => { //set interval for the process

this.AlarmSensorFiveSendData();

}, 10000); // in every 10 seconds pass the values.

}

catch (error) { // if get any error then print error message.

console.log(error);

}

}

//render simple user interface

//set simple design

render() {

return (

<div className="col-md-12 m-auto">

<div className="card border-primary card-body">

<h2>Fire Alarm Sensor Five Sending Data... <i class="fas fa-arrow-alt-circle-right"></i></h2>

</div>

</div>

);

}

}

export default SensorFiveSimulate;