Sri Lanka Institute of Information Technology

Distributed Systems (SE3020)

Assignment 02-REST API

Assignment Report

IT18053838 H.P Sumudu Lakruwan

IT18027884 Fernando G.U.S

IT18020618 R.P.A.B Somarathne

IT18029246 Saranga S.A.N

Table of Contents

Introduction...........................................................................................................................................3

High Level Architectural Diagram........................................................................................................4

System Workflow Scenario Execution .................................................................................................5

Appendix..............................................................................................................................................

Rest\_Api -----9

Rmi server 13

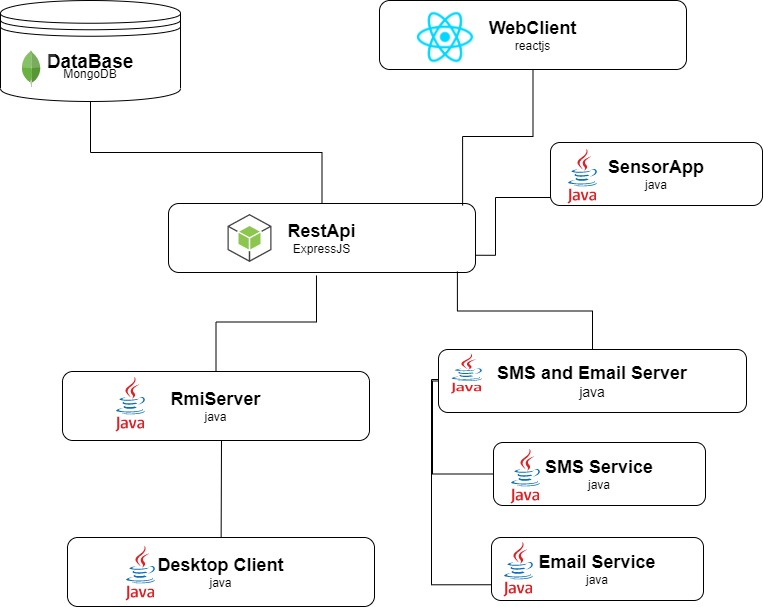
Email and sms server 41

Rmi client

**Introduction**

This System is about fire alarm monitoring system. For develop this system, we design Sensor app, REST API, email service and sms service, RMIi server, web client and desktop client as well. sensor app update the sensor status from REST API is 10seconds.rest api is developed using express and node. REST API use mongodb to save data.All the communications are done using REST API. RMI server develop is developed using java.RMI server checks the sensor status from REST API is 15 seconds.Web client is developed using React .web client Asynchronously display details by calling rest api end point.desktop client can update, delete,a dd and list the details by calling the rmi methods.we create dummy sensor app and it change co2 and smoke levels. If co2 level or smoke level greater than five,rmi sever generate alert.sensr app also can add sensors.

**High Level Architectural Diagram**

****

**System Workflow Scenario Execution**

**Rest Api**

With the help of the rest api,we can send data as Jason objects. Rest api is connected to mongodb.

Api end points

(1) <http://localhost:4000/monitoring/details>

list the all fire alarm details.in here we can list the all details of the fire alarms.

Http method:GET

(2) <http://localhost:4000/monitoring/details>

With help of this endpoint,we can add anew fire alarm details.

HTTP method:Post

(3) <http://localhost:4000/monitoring/details//:id>

This end point helps to update the certain records of the schema.id is the mongodb id

HTTP method:PUT

(4) <http://localhost:4000/monitoring/details//:id>

This end point helps to delete certain record of the mongodb.

HTTP method:DELETE

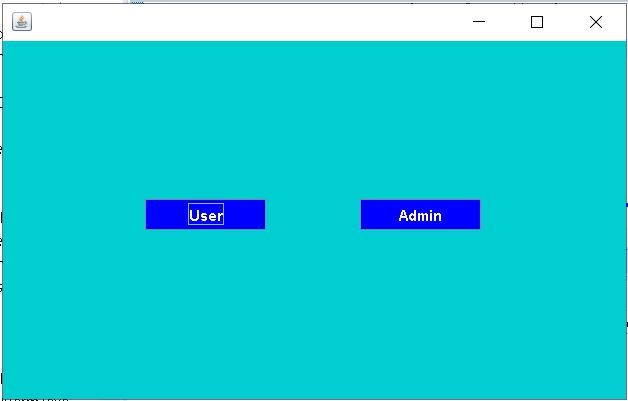
(5) <http://localhost:4000/monitoring/search:id>

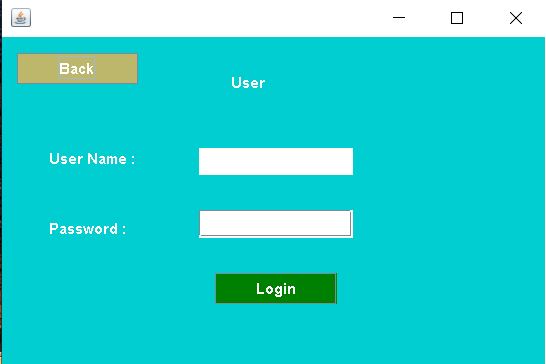
This end point help to find certain record

HTTP method:GET

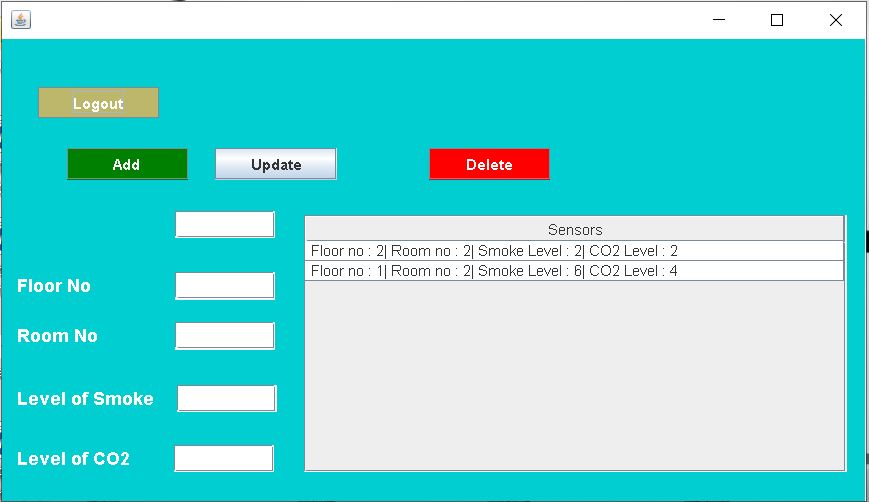
**Desktop Client**

When opening the desktop client, we can see that below ui,you can selete either user or admin.



When clicking the admin,we can see below ui aswell,

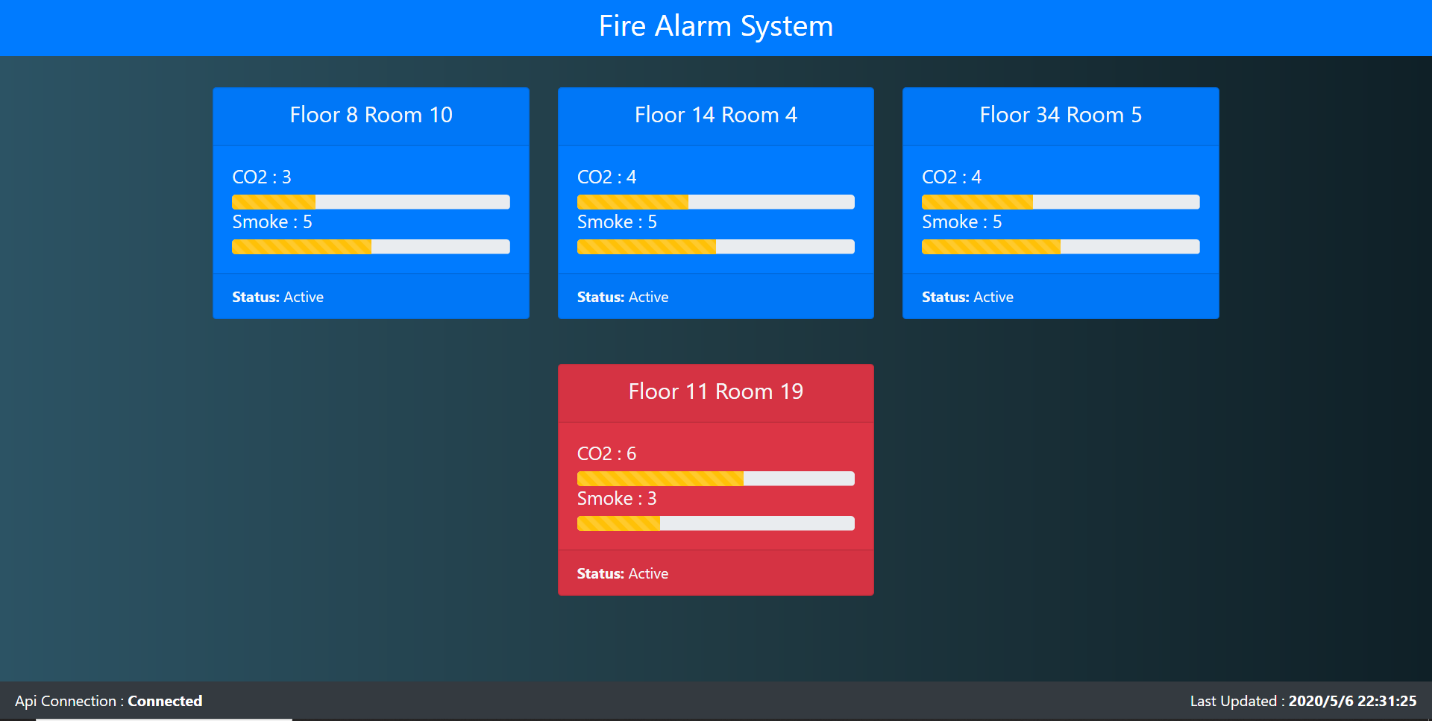
Admin user name is admin and password is admin.when clicking login we can see below ui.

****

With help of the this ui we can add,delete,update and view the records.

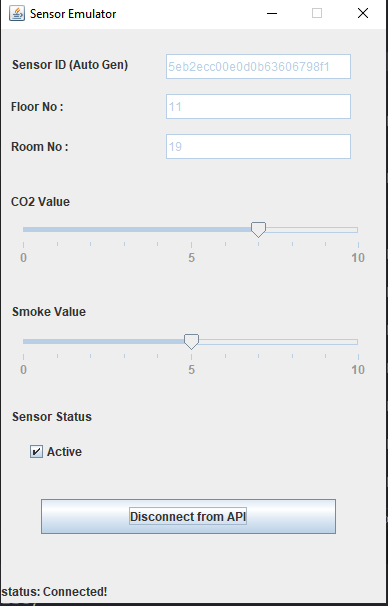
Web Client

Web client can view data Asynchronously.



**Sensor App**

Sensor app Ui is displayed as below.



**Appendix**

**Rest-api**

**(1)Index.js**

const express=require('express');

const fullRoutes=require('./customizedRoutes/RouteFrApi');

const Parsbdy=require('body-parser');

const mongoose=require('mongoose');

const app=express();

const cors = require('cors');

mongoose.connect('mongodb://localhost/monitor', {useUnifiedTopology: true, useNewUrlParser: true }).then(() => console.log('MongoDB Connected...'))

.catch(err => console.log(err));

mongoose.Promise=global.Promise;

app.use(cors());//acess controll\_security feature

app.use(function (req, res, next) {

res.header("Access-Control-Allow-Origin", "\*");

res.header('Access-Control-Allow-Methods: GET, POST, OPTIONS');

res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");

next();

});

app.use(Parsbdy.json());

app.use('/monitoring',fullRoutes);

//custom error handling

app.use(function(erro,rq,res,next){

res.status(422).send({error:erro.message});

});

app.listen(4000,function(){

console.log('Listning the request in port 4000');

});

**(2)RouterFrApi.js**

const express=require('express');

const router=express.Router();

const FireAlarm=require('../schemas/firealarmSchema');

//list the all fire alarm details

router.get('/details',function(req,res,next){

FireAlarm.find({}).then(function(item){

res.send(item);

})

});

//add anew fire alarm details

router.post('/details',function(req,res,next){

FireAlarm.create({

floorNo :parseFloat(req.body.floorNo),

roomNo :parseFloat(req.body.roomNo),

smokelevel :parseFloat(req.body.smokelevel),

co2\_level:parseFloat(req.body.co2\_level),

status:req.body.status,

co2\_Alert:req.body.co2\_Alert,

smoke\_Alert:req.body.smoke\_Alert

}

).then(function(item){

res.send(item);

}).catch(next);

});

//update a fire alarm details

router.put('/details/:id',function(req,res,next){

FireAlarm.findByIdAndUpdate({\_id:req.params.id},{

floorNo :parseFloat(req.body.floorNo),

roomNo :parseFloat(req.body.roomNo),

smokelevel :parseFloat(req.body.smokelevel),

co2\_level:parseFloat(req.body.co2\_level),

status:req.body.status,

co2\_Alert:req.body.co2\_Alert,

smoke\_Alert:req.body.smoke\_Alert

}).then(function(){

FireAlarm.findOne({\_id:req.params.id}).then(function(single){

res.send(single);

});

});

});

//Search a certain record

router.get('/search/:id',function(req,res,next){

FireAlarm.find({\_id:req.params.id}).then(function(item){

res.send(item);

});

});

//delete a fire alarm details

router.delete('/details/:id',function(req,res,next){

FireAlarm.findByIdAndRemove({\_id:req.params.id}).then(function(item){

res.send(item);

});

});

module.exports=router;

**RMI server**

1. FireAlarm Model

import java.io.Serializable;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class fireAlarm implements Serializable  {

    /\*\*

     \*

     \*/

    private static final long serialVersionUID = 1L;

    String id;

    int floor;

    int roomNumber;

    int smokelevel;

    int co2level;

    Boolean status;

    Boolean smokeAlert;

    Boolean co2Alert;

    public fireAlarm() throws RemoteException {

        super();

    }

    public fireAlarm(String id, int floor, int roomNumber, int smokelevel, int co2level) throws RemoteException {

        super();

        this.id = id;

        this.floor = floor;

        this.roomNumber = roomNumber;

        this.smokelevel = smokelevel;

        this.co2level = co2level;

        this.smokeAlert=false;

        this.co2Alert=false;

        this.status=false;

    }

    public void setId(String id) {

        this.id = id;

    }

    public int getFloor() {

        return floor;

    }

    public void setFloor(int floor) {

        this.floor = floor;

    }

    public int getRoomNumber() {

        return roomNumber;

    }

    public void setRoomNumber(int roomNumber) {

        this.roomNumber = roomNumber;

    }

    public int getSmokelevel() {

        return smokelevel;

    }

    public void setSmokelevel(int smokelevel) {

        this.smokelevel = smokelevel;

    }

    public int getCo2level() {

        return co2level;

    }

    public void setCo2level(int co2level) {

        this.co2level = co2level;

    }

    public Boolean getSmokeAlert() {

        return smokeAlert;

    }

    public void setSmokeAlert(Boolean smokeAlert) {

        this.smokeAlert = smokeAlert;

    }

    public Boolean getCo2Alert() {

        return co2Alert;

    }

    public void setCo2Alert(Boolean co2Alert) {

        this.co2Alert = co2Alert;

    }

    public String getId() {

        return id;

    }

    public Boolean getStatus() {

        return status;

    }

    public void setStatus(Boolean status) {

        this.status = status;

    }

}

1. RMI\_Server

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.net.HttpURLConnection;

import java.net.MalformedURLException;

import java.net.URL;

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

import java.util.List;

import java.util.Timer;

import java.util.TimerTask;

import org.json.JSONArray;

import org.json.JSONObject;

public class RMI\_Server extends UnicastRemoteObject implements ServerInt {

    private static List<sensorListener> listeners = new ArrayList<>();

    protected RMI\_Server() throws RemoteException {

        super();

        // TODO Auto-generated constructor stub

    }

    @Override

    public void serverCheck() throws Exception {

        System.out.println("Server is Checked");

    }

    @Override

    public ArrayList<fireAlarm> addItem(fireAlarm item) throws Exception {

        ArrayList<fireAlarm> faList = new ArrayList<>();

        final String POST\_PARAMS =

                "{\n" + "\"floorNo\": "+item.floor+",\r\n" +

                "    \"roomNo\": "+item.roomNumber+",\r\n" +

                "    \"smokelevel\":"+item.smokelevel+",\r\n" +

                "    \"co2\_level\": "+item.co2level+",\r\n" +

                "    \"smoke\_Alert\":"+item.smokeAlert+",\r\n" +

                "    \"co2\_Alert\": "+item.co2Alert + "\n}";

            URL obj = new URL("http://localhost:4000/monitoring/details/");

            HttpURLConnection postConnection = (HttpURLConnection) obj.openConnection();

            postConnection.setRequestMethod("POST");

            postConnection.setRequestProperty("Content-Type", "application/json");

            postConnection.setDoOutput(true);

            OutputStream os = postConnection.getOutputStream();

            os.write(POST\_PARAMS.getBytes());

            os.flush();

            os.close();

            int responseCode = postConnection.getResponseCode();

            int req=HttpURLConnection.HTTP\_CREATED ;

            if (responseCode == (req-1)) { //success

                BufferedReader in = new BufferedReader(new InputStreamReader(

                    postConnection.getInputStream()));

                String inputLine;

                StringBuffer response = new StringBuffer();

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

                    response.append(inputLine);

                } in .close();

                JSONObject sensor = new JSONObject(response.toString());

                    fireAlarm currentSensor = new fireAlarm();

                    currentSensor.setId(String.valueOf(sensor.getString("\_id")));

                    currentSensor.setFloor(Integer.valueOf(sensor.getInt("floorNo")));

                    currentSensor.setRoomNumber(Integer.valueOf(sensor.getInt("roomNo")));

                    currentSensor.setSmokelevel(Integer.valueOf(sensor.getInt("smokelevel")));

                    currentSensor.setCo2level(Integer.valueOf(sensor.getInt("co2\_level")));

                    currentSensor.setSmokeAlert(Boolean.valueOf(sensor.getBoolean("smoke\_Alert")));

                    currentSensor.setCo2Alert(Boolean.valueOf(sensor.getBoolean("co2\_Alert")));

                    faList.add(currentSensor);

            } else {

                System.out.println("POST NOT WORKED");

            }

            return faList;

    }

    @Override

    public ArrayList<fireAlarm> getItem(String id) throws Exception {

        ArrayList<fireAlarm> faList = new ArrayList<>();

        JSONArray myResponse ;

        try {

            String url = "http://localhost:4000/monitoring/search/"+id;

            URL obj = new URL(url);

            HttpURLConnection con = (HttpURLConnection) obj.openConnection();

            con.setRequestMethod("GET");

            //add request header

            con.setRequestProperty("User-Agent", "Mozilla/5.0");

            int responseCode = con.getResponseCode();

            BufferedReader in = new BufferedReader(

                    new InputStreamReader(con.getInputStream()));

            String inputLine;

            StringBuffer response = new StringBuffer();

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

                response.append(inputLine);

            }

            in.close();

              myResponse = new JSONArray(response.toString());

              JSONObject sensor;

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

                    sensor =  myResponse.getJSONObject(i);

                    fireAlarm currentSensor = new fireAlarm();

                    currentSensor.setId(String.valueOf(sensor.getString("\_id")));

                    currentSensor.setFloor(Integer.valueOf(sensor.getInt("floorNo")));

                    currentSensor.setRoomNumber(Integer.valueOf(sensor.getInt("roomNo")));

                    currentSensor.setSmokelevel(Integer.valueOf(sensor.getInt("smokelevel")));

                    currentSensor.setCo2level(Integer.valueOf(sensor.getInt("co2\_level")));

                    currentSensor.setSmokeAlert(Boolean.valueOf(sensor.getBoolean("smoke\_Alert")));

                    currentSensor.setCo2Alert(Boolean.valueOf(sensor.getBoolean("co2\_Alert")));

                    faList.add(currentSensor);

                }

        } catch (Exception e) {

            System.out.println(e);

        }

        return faList;

    }

    @Override

    public ArrayList<fireAlarm> getItems() throws Exception {

        ArrayList<fireAlarm> faList = new ArrayList<>();

        JSONArray myResponse ;

        try {

            String url = "http://localhost:4000/monitoring/details/";

            URL obj = new URL(url);

            HttpURLConnection con = (HttpURLConnection) obj.openConnection();

            con.setRequestMethod("GET");

            con.setRequestProperty("User-Agent", "Mozilla/5.0");

            int responseCode = con.getResponseCode();

            BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));

            String inputLine;

            StringBuffer response = new StringBuffer();

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

                response.append(inputLine);

            }

            in.close();

              myResponse = new JSONArray(response.toString());

                JSONObject sensor;

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

                    sensor =  myResponse.getJSONObject(i);

                    fireAlarm currentSensor = new fireAlarm();

                    currentSensor.setId(String.valueOf(sensor.getString("\_id")));

                    currentSensor.setFloor(Integer.valueOf(sensor.getInt("floorNo")));

                    currentSensor.setRoomNumber(Integer.valueOf(sensor.getInt("roomNo")));

                    currentSensor.setStatus(Boolean.valueOf(sensor.getBoolean("status")));

                    currentSensor.setSmokelevel(Integer.valueOf(sensor.getInt("smokelevel")));

                    currentSensor.setCo2level(Integer.valueOf(sensor.getInt("co2\_level")));

                    currentSensor.setSmokeAlert(Boolean.valueOf(sensor.getBoolean("smoke\_Alert")));

                    currentSensor.setCo2Alert(Boolean.valueOf(sensor.getBoolean("co2\_Alert")));

                    faList.add(currentSensor);

                }

        } catch (Exception e) {

            // TODO: handle exception

        }

        return faList;

    }

    @Override

    public ArrayList<fireAlarm> updateItem(fireAlarm item) throws Exception {

        ArrayList<fireAlarm> faList = new ArrayList<>();

        final String POST\_PARAMS =

                "{\n" + "\"floorNo\": "+item.floor+",\r\n" +

                        "    \"roomNo\": "+item.roomNumber+",\r\n" +

                        "    \"smokelevel\":"+item.smokelevel+",\r\n" +

                        "    \"co2\_level\": "+item.co2level+",\r\n" +

                        "    \"status\":"+item.status+",\r\n" +

                        "    \"smoke\_Alert\":"+item.smokeAlert+",\r\n" +

                        "    \"co2\_Alert\": "+item.co2Alert+ "\n}";

            URL obj = new URL("http://localhost:4000/monitoring/details/"+item.id);

            HttpURLConnection postConnection = (HttpURLConnection) obj.openConnection();

            postConnection.setRequestMethod("PUT");

            //postConnection.setRequestProperty("IdNumber","101");

            postConnection.setRequestProperty("Content-Type", "application/json");

            postConnection.setDoOutput(true);

            OutputStream os = postConnection.getOutputStream();

            os.write(POST\_PARAMS.getBytes());

            os.flush();

            os.close();

            int responseCode = postConnection.getResponseCode();

            int req=HttpURLConnection.HTTP\_CREATED ;

            if (responseCode == (req-1)) { //success

                BufferedReader in = new BufferedReader(new InputStreamReader(

                    postConnection.getInputStream()));

                String inputLine;

                StringBuffer response = new StringBuffer();

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

                    response.append(inputLine);

                } in .close();

                JSONObject sensor = new JSONObject(response.toString());

                fireAlarm currentSensor = new fireAlarm();

                    currentSensor.setId(String.valueOf(sensor.getString("\_id")));

                    currentSensor.setFloor(Integer.valueOf(sensor.getInt("floorNo")));

                    currentSensor.setRoomNumber(Integer.valueOf(sensor.getInt("roomNo")));

                    currentSensor.setSmokelevel(Integer.valueOf(sensor.getInt("smokelevel")));

                    currentSensor.setCo2level(Integer.valueOf(sensor.getInt("co2\_level")));

                    currentSensor.setSmokeAlert(Boolean.valueOf(sensor.getBoolean("smoke\_Alert")));

                    currentSensor.setCo2Alert(Boolean.valueOf(sensor.getBoolean("co2\_Alert")));

                    faList.add(currentSensor);

            } else {

                System.out.println("UPDATE NOT WORKED");

            }

        return faList;

    }

    @Override

    public ArrayList<fireAlarm> deleteItem(String id) throws Exception {

        ArrayList<fireAlarm> faList = new ArrayList<>();

        final String POST\_PARAMS = null;

            URL obj = new URL("http://localhost:4000/monitoring/details/"+id);

            HttpURLConnection postConnection = (HttpURLConnection) obj.openConnection();

            postConnection.setRequestMethod("DELETE");

            postConnection.setRequestProperty("Content-Type", "application/json");

            postConnection.setDoOutput(true);

            OutputStream os = postConnection.getOutputStream();

            os.flush();

            os.close();

            int responseCode = postConnection.getResponseCode();

            int req=HttpURLConnection.HTTP\_CREATED ;

            if (responseCode == (req-1)) { //success

                BufferedReader in = new BufferedReader(new InputStreamReader(

                    postConnection.getInputStream()));

                String inputLine;

                StringBuffer response = new StringBuffer();

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

                    response.append(inputLine);

                } in .close();

                JSONObject sensor =  new JSONObject(response.toString());

                 fireAlarm currentSensor = new fireAlarm();

                    currentSensor.setId(String.valueOf(sensor.getString("\_id")));

                    currentSensor.setFloor(Integer.valueOf(sensor.getInt("floorNo")));

                    currentSensor.setRoomNumber(Integer.valueOf(sensor.getInt("roomNo")));

                    currentSensor.setSmokelevel(Integer.valueOf(sensor.getInt("smokelevel")));

                    currentSensor.setCo2level(Integer.valueOf(sensor.getInt("co2\_level")));

                    currentSensor.setSmokeAlert(Boolean.valueOf(sensor.getBoolean("smoke\_Alert")));

                    currentSensor.setCo2Alert(Boolean.valueOf(sensor.getBoolean("co2\_Alert")));

                    faList.add(currentSensor);

            } else {

                System.out.println("DELETE NOT WORKED");

            }

        return faList;

    }

    public void addSensorListener(sensorListener sl) throws RemoteException

    {

        listeners.add(sl);

    }

    public void smokeAlert(fireAlarm item) throws Exception{

        //System.out.println("smoke Alert"+ item.id);

        this.updateItem(item);

    }

    public void co2Alert(fireAlarm item) throws Exception{

        //System.out.println("Co2 Alert" + item.id);

        this.updateItem(item);

    }

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

System.out.println("Server is started");

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

        RMI\_Server s1 = new RMI\_Server();

        Naming.rebind("server", s1);

        //RMI Server checks the Alert status and send Alerts to Rest every 15 Secs while updating desktop Client

        TimerTask task = new TimerTask() {

              @Override

              public void run() {

                  for (sensorListener lListener : listeners)

                  {

                      try

                      {

                          ArrayList<fireAlarm> array=lListener.sensorStatusChanged();

                          for (fireAlarm item : array) {

                            if(item.co2level>5) {

                                item.setCo2Alert(true);

                                s1.co2Alert(item);

                            }

                            if (item.smokelevel>5) {

                                item.setSmokeAlert(true);

                                s1.smokeAlert(item);

                            }

                        }

                      }

                      catch (Exception aInE)

                      {

                      }

                  }

              }

            };

            Timer timer = new Timer();

            long delay = 0;

            long intevalPeriod = 1 \* 15000;

            timer.scheduleAtFixedRate(task, delay,intevalPeriod);

         System.out.println("Server is running");

    }

}

1. SensorListener interface
2. import java.rmi.Remote;
3. import java.rmi.RemoteException;
4. import java.util.ArrayList;
5. public interface sensorListener extends Remote{
6. ArrayList<fireAlarm> sensorStatusChanged() throws RemoteException;
7. }

4. ServerInt interface

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.ArrayList;

public interface ServerInt extends Remote {

    public void serverCheck() throws Exception;

    public ArrayList<fireAlarm> addItem(fireAlarm Item) throws Exception;

    public ArrayList<fireAlarm> getItem(String id) throws Exception;

    public ArrayList<fireAlarm> getItems() throws Exception;

    public ArrayList<fireAlarm> updateItem(fireAlarm Item) throws Exception;

    public ArrayList<fireAlarm> deleteItem(String id) throws Exception;

    public void addSensorListener(sensorListener sl)throws Exception;

    public void co2Alert(fireAlarm id) throws Exception;

    public void smokeAlert(fireAlarm id) throws Exception;

}

**Web Client (react js)**

**Sensor.js**

import React from 'react'

import { Card, ProgressBar } from 'react-bootstrap';

const Sensor = (props) => {

    let cardStyle = "primary";

    //this will change the sensor card backgrouund color red if smoke or co2 lvls are above 5

    if (props.co2 > 5 || props.smoke > 5) {

        cardStyle = "danger";

    }

    return (

        //if a sensor is inactive yellow border will be added to the card

        <Card border={props.status?null:"warning"} bg={cardStyle} key={props.id}

            style={{ minWidth: '20rem' }} text="light" >

            <Card.Header className="text-center">

                <h4> {"Floor " + props.floor + " Room " + props.room}</h4>

            </Card.Header>

            <Card.Body>

                <h5>CO2 : {props.co2}</h5>

                <ProgressBar srOnly animated striped variant="warning"

                    now={props.co2} min={0} max={10} />

                <h5>Smoke : {props.smoke}</h5>

                <ProgressBar animated striped variant="warning"

                    now={props.smoke} min={0} max={10} />

            </Card.Body>

            <Card.Footer> <strong>Status: </strong>{props.status ? "Active" : "Inactive"}</Card.Footer>

        </Card>

    )

}

export default Sensor;

**Wrapper.js**

const Wrapper = (props) => {

    return props.children;

}

export default Wrapper;

**Footer.js**

import React from 'react'

import { Nav } from 'react-bootstrap';

const Footer = (props) => {

    //Simple function to format date to readable string

    const getFormattedDate = (datePassed) => {

        let date = new Date(datePassed);

        let str = date.getFullYear() + "/" + (date.getMonth() + 1)

            + "/" + date.getDate() + " " + date.getHours()

            + ":" + date.getMinutes() + ":" + date.getSeconds();

        return str;

    }

    return (

        <Nav className=" fixed-bottom  navbar navbar-dark bg-dark text-white ">

            <Nav.Item>

                {/\* this will display api Connection status \*/}

                Api Connection : <strong>{props.apiStatus}</strong>

            </Nav.Item>

            <Nav.Item>

                {/\* This will display the last Updated time \*/}

                Last Updated : <strong>{getFormattedDate(props.lastUpdated)}</strong>

            </Nav.Item>

        </Nav>

    );

}

export default Footer

**Header.js**

import React from 'react';

import { Nav } from 'react-bootstrap';

const Header = () => {

    return (

        <Nav className=" fixed-top justify-content-center navbar navbar-dark bg-primary text-white ">

                <h2>Fire Alarm System</h2>

        </Nav>

    )

}

export default Header

**SensorView.js**

import React from 'react';

import { Container, Row, Col } from 'react-bootstrap';

import Sensor from '../../Components/Sensor/Sensor';

const SensorView = (props) => {

    // this will craete jsx elements for every sensor object in sensors[]

    const sensorRender = props.sensorList.map(sensor => {

        return (<Col key={sensor.\_id} className='mt-5'>

            <Sensor

                floor={sensor.floorNo}

                room={sensor.roomNo}

                co2={sensor.co2\_level}

                smoke={sensor.smokelevel}

                status={sensor.status} />

        </Col>);

    });

    return (

        <Container className='mb-5'>

            <Row xs={1} md={2} lg={2} xl={3} className="justify-content-md-center">

                {sensorRender}

            </Row>

        </Container>

    );

}

export default SensorView;

**APP.JS**

import React, { Component } from 'react';

import 'bootstrap/dist/css/bootstrap.min.css';

import { Container } from 'react-bootstrap';

import './App.css';

import SensorView from './Layouts/SensorView/SensorView';

import Header from './Layouts/Header/Header';

import Footer from './Layouts/Footer/Footer';

import Wrapper from './Hoc/Wrapper';

class App extends Component {

  constructor(props){

    super(props);

    this.state = {

      sensors: [],

      apiStatus: "Disconnected",

      lastUpdated : Date.now()

   };

    this.loadData = this.loadData.bind(this);

    // this method need to be binded 'this' inorder to call setState method

  }

  componentDidMount() {

    //get data from api

    this.loadData()

    //run load data function after every 40 seconds to get latest data from api

    setInterval(this.loadData, 40000);

  }

  //this function will asynchronously load data from rest api

  async loadData() {

    try {

      const res = await fetch('http://localhost:4000/monitoring/details');

      const data = await res.json();

      //updateing state with lastest data

      this.setState({

        sensors: data,

        apiStatus: "Connected",

        lastUpdated : Date.now()

      });

    } catch (e) {

      //if failed to communicate with api this code block will run

      console.log(e);

      this.setState({

        apiStatus:"Disconnected!!"

      });

    }

  }

  render() {

    return (

      <Wrapper>

        <Header />

        <Container className="App mt-5">

          <SensorView sensorList={this.state.sensors}/>

        </Container>

        <Footer apiStatus={this.state.apiStatus} lastUpdated={this.state.lastUpdated}/>

      </Wrapper>

    );

  };

};

  export default App;

**Index.js**

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import \* as serviceWorker from './serviceWorker';

ReactDOM.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>,

  document.getElementById('root')

);

// If you want your app to work offline and load faster, you can change

// unregister() to register() below. Note this comes with some pitfalls.

// Learn more about service workers: https://bit.ly/CRA-PWA

serviceWorker.unregister();

**Sensor App**

**Sensor.java**

package model;  
  
public class Sensor {  
  
 private String id ;  
 private String roomNo;  
 private String floorNo;  
 private int co2Value;  
 private int smokeValue;  
 private boolean status;  
  
 public Sensor(String *id*, String *roomNo*, String *floorNo*, int *co2Value*, int *smokeValue*, boolean *status*) {  
 *this*.id = *id*;  
 *this*.roomNo = *roomNo*;  
 *this*.floorNo = *floorNo*;  
 *this*.co2Value = *co2Value*;  
 *this*.smokeValue = *smokeValue*;  
 *this*.status = *status*;  
 }  
  
 public Sensor() {  
 }  
  
 public String getId() {  
 return id;  
 }  
  
 public void setId(String *id*) {  
 *this*.id = *id*;  
 }  
  
 public String getRoomNo() {  
 return roomNo;  
 }  
  
 public void setRoomNo(String *roomNo*) {  
 *this*.roomNo = *roomNo*;  
 }  
  
 public String getFloorNo() {  
 return floorNo;  
 }  
  
 public void setFloorNo(String *floorNo*) {  
 *this*.floorNo = *floorNo*;  
 }  
  
 public int getCo2Value() {  
 return co2Value;  
 }  
  
 public void setCo2Value(int *co2Value*) {  
 *this*.co2Value = *co2Value*;  
 }  
  
 public int getSmokeValue() {  
 return smokeValue;  
 }  
  
 public void setSmokeValue(int *smokeValue*) {  
 *this*.smokeValue = *smokeValue*;  
 }  
  
 public boolean isStatus() {  
 return status;  
 }  
  
 public void setStatus(boolean *status*) {  
 *this*.status = *status*;  
 }  
  
 @Override  
 public String toString() {  
 return "Sensor{" +  
 "id='" + id + '\'' +  
 ", roomNo='" + roomNo + '\'' +  
 ", floorNo='" + floorNo + '\'' +  
 ", co2Value=" + co2Value +  
 ", smokeValue=" + smokeValue +  
 ", status=" + status +  
 '}';  
 }  
}

**Sensdatatoapi.java**

package controller;  
  
import model.Sensor;  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.*OutputStream*;  
import java.net.*HttpURLConnection*;  
import java.net.URL;  
import java.util.ArrayList;  
  
  
  
public class SendDataToApi {  
  
  
 public String makeRequest(Sensor *sensor*) throws IOException, InterruptedException {  
  
 ArrayList<Sensor> faList = new ArrayList<>();  
 String id = null;  
  
 final String POST\_PARAMS =  
 "{\n" + "\"status\": "+*sensor*.isStatus()+",\r\n" +  
 " \"floorNo\": "+*sensor*.getFloorNo()+",\r\n" +  
 " \"roomNo\": "+*sensor*.getRoomNo()+",\r\n" +  
 " \"smokelevel\":"+*sensor*.getSmokeValue()+",\r\n" +  
 " \"co2\_level\": "+*sensor*.getCo2Value()+"" + "\n}";  
  
 System.*out*.println(POST\_PARAMS);  
  
 URL obj = new URL("http://localhost:4000/monitoring/details/");  
 *HttpURLConnection* postConnection = (*HttpURLConnection*) obj.openConnection();  
 postConnection.setRequestMethod("POST");  
 *//postConnection.setRequestProperty("IdNumber","101");* postConnection.setRequestProperty("Content-Type", "application/json");  
 postConnection.setDoOutput(true);  
 *OutputStream* os = postConnection.getOutputStream();  
 os.write(POST\_PARAMS.getBytes());  
 os.flush();  
 os.close();  
 int responseCode = postConnection.getResponseCode();  
 System.*out*.println("POST Response Code : " + responseCode);  
 System.*out*.println("POST Response Message : " + postConnection.getResponseMessage());  
 int req=*HttpURLConnection*.*HTTP\_CREATED* ;  
  
 if (responseCode == (req-1)) { *//success* BufferedReader in = new BufferedReader(new InputStreamReader(  
 postConnection.getInputStream()));  
 String inputLine;  
 StringBuffer response = new StringBuffer();  
 while ((inputLine = in .readLine()) != null) {  
 response.append(inputLine);  
 } in .close();  
 *// print result* String responseStr = response.toString();  
 System.*out*.println(responseStr);  
 id = responseStr.substring(responseStr.indexOf("\_id\":") + 6, responseStr.indexOf(",\"floorNo\"")-1);  
 System.*out*.println(id);  
  
  
 } else {  
 System.*out*.println("POST NOT WORKED");  
 }  
 return id;  
 }  
  
 public void updateData(Sensor *sensor*) throws Exception {  
  
 final String POST\_PARAMS =  
 "{\n" + "\"status\": "+*sensor*.isStatus()+",\r\n" +  
 " \"floorNo\": "+*sensor*.getFloorNo()+",\r\n" +  
 " \"roomNo\": "+*sensor*.getRoomNo()+",\r\n" +  
 " \"smokelevel\":"+*sensor*.getSmokeValue()+",\r\n" +  
 " \"co2\_Alert\":"+false+",\r\n" +  
 " \"smoke\_Alert\":"+false+",\r\n" +  
 " \"co2\_level\": "+*sensor*.getCo2Value()+"" + "\n}";  
  
 System.*out*.println(POST\_PARAMS);  
  
 URL obj = new URL("http://localhost:4000/monitoring/details/"+*sensor*.getId());  
 *HttpURLConnection* postConnection = (*HttpURLConnection*) obj.openConnection();  
 postConnection.setRequestMethod("PUT");  
 *//postConnection.setRequestProperty("IdNumber","101");* postConnection.setRequestProperty("Content-Type", "application/json");  
 postConnection.setDoOutput(true);  
 *OutputStream* os = postConnection.getOutputStream();  
 os.write(POST\_PARAMS.getBytes());  
 os.flush();  
 os.close();  
 int responseCode = postConnection.getResponseCode();  
 System.*out*.println("POST Response Code : " + responseCode);  
 System.*out*.println("POST Response Message : " + postConnection.getResponseMessage());  
 int req=*HttpURLConnection*.*HTTP\_CREATED* ;  
  
 if (responseCode == (req-1)) { *//success* BufferedReader in = new BufferedReader(new InputStreamReader(  
 postConnection.getInputStream()));  
 String inputLine;  
 StringBuffer response = new StringBuffer();  
 while ((inputLine = in .readLine()) != null) {  
 response.append(inputLine);  
 } in .close();  
  
 *// print result* System.*out*.println(response.toString());  
  
  
 } else {  
 System.*out*.println("UPDATE NOT WORKED");  
 }  
  
  
 }  
  
 public void deleteData(Sensor *sensor*) throws Exception {  
  
  
  
 URL obj = new URL("http://localhost:4000/monitoring/details/"+*sensor*.getId());  
 *HttpURLConnection* postConnection = (*HttpURLConnection*) obj.openConnection();  
 postConnection.setRequestMethod("DELETE");  
 *//postConnection.setRequestProperty("IdNumber","101");* postConnection.setRequestProperty("Content-Type", "application/json");  
 postConnection.setDoOutput(true);  
 *OutputStream* os = postConnection.getOutputStream();  
 os.flush();  
 os.close();  
 int responseCode = postConnection.getResponseCode();  
 System.*out*.println("POST Response Code : " + responseCode);  
 System.*out*.println("POST Response Message : " + postConnection.getResponseMessage());  
 int req=*HttpURLConnection*.*HTTP\_CREATED* ;  
  
 if (responseCode == (req-1)) { *//success* BufferedReader in = new BufferedReader(new InputStreamReader(  
 postConnection.getInputStream()));  
 String inputLine;  
 StringBuffer response = new StringBuffer();  
 while ((inputLine = in .readLine()) != null) {  
 response.append(inputLine);  
 } in .close();  
 *// print result* System.*out*.println(response.toString());  
  
  
 } else {  
 System.*out*.println("DELETE NOT WORKED");  
 }  
  
 }  
  
}

**Home.java**

*//Generated by GuiGenie - Copyright (c) 2004 Mario Awad.  
//Home Page http://guigenie.cjb.net - Check often for new versions!*import controller.SendDataToApi;  
import model.Sensor;  
  
import javax.swing.\*;  
  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.*ActionListener*;  
import java.awt.event.*WindowAdapter*;  
import java.awt.event.WindowEvent;  
import java.io.IOException;  
import java.util.Date;  
import java.util.Random;  
import java.util.*TimerTask*;  
import java.util.Timer;  
  
import java.util.UUID;  
  
  
  
  
public class Home extends JPanel {  
 private JLabel jcomp1;  
 private static JTextField *idField*;  
 private JLabel jcomp3;  
 private static JTextField *floorNo*;  
 private JLabel jcomp5;  
 private static JTextField *roomNo*;  
 private JLabel jcomp7;  
 private static JSlider *co2Value*;  
 private JLabel jcomp9;  
 private static JSlider *smokeValue*;  
 private static JCheckBox *sensorStatus*;  
 private JLabel jcomp12;  
 private static JButton *updateButton*;  
 private static JLabel *appStatus*;  
 private static Sensor *sensor*;  
 private static String *uniqueID*;  
 private static SendDataToApi *sendDataToApi*;  
 static Random *rand*;  
 static Timer *timer*;  
 private static boolean *connected* =false;  
  
 private Home() {  
 *//construct components* jcomp1 = new JLabel ("Sensor ID (Auto Gen)");  
 *idField* = new JTextField (5);  
 jcomp3 = new JLabel ("Floor No :");  
 *floorNo* = new JTextField (5);  
 jcomp5 = new JLabel ("Room No :");  
 *roomNo* = new JTextField (5);  
 jcomp7 = new JLabel ("CO2 Value");  
 *co2Value* = new JSlider (0, 10);  
 jcomp9 = new JLabel ("Smoke Value");  
 *smokeValue* = new JSlider (0, 10);  
 *sensorStatus* = new JCheckBox ("Active");  
 jcomp12 = new JLabel ("Sensor Status");  
 *updateButton* = new JButton ("Connect to REST API");  
 *appStatus* = new JLabel ("status: Disconnected!");  
  
 *//set components properties  
 idField*.setEnabled (false);  
 *roomNo*.setEnabled(false);  
 *floorNo*.setEnabled(false);  
 *smokeValue*.setEnabled(false);  
 *co2Value*.setEnabled(false);  
  
 *co2Value*.setOrientation (JSlider.*HORIZONTAL*);  
 *co2Value*.setMinorTickSpacing (1);  
 *co2Value*.setMajorTickSpacing (5);  
 *co2Value*.setPaintTicks (true);  
 *co2Value*.setPaintLabels (true);  
 *smokeValue*.setOrientation (JSlider.*HORIZONTAL*);  
 *smokeValue*.setMinorTickSpacing (1);  
 *smokeValue*.setMajorTickSpacing (5);  
 *smokeValue*.setPaintTicks (true);  
 *smokeValue*.setPaintLabels (true);  
  
 *//adjust size and set layout* setPreferredSize (new Dimension(385, 574));  
 setLayout (null);  
  
 *//add components* add (jcomp1);  
 add (*idField*);  
 add (jcomp3);  
 add (*floorNo*);  
 add (jcomp5);  
 add (*roomNo*);  
 add (jcomp7);  
 add (*co2Value*);  
 add (jcomp9);  
 add (*smokeValue*);  
 add (*sensorStatus*);  
 add (jcomp12);  
 add (*updateButton*);  
 add (*appStatus*);  
  
 *//set component bounds (only needed by Absolute Positioning)* jcomp1.setBounds (10, 25, 125, 20);  
 *idField*.setBounds (165, 25, 185, 25);  
 jcomp3.setBounds (10, 65, 100, 25);  
 *floorNo*.setBounds (165, 65, 185, 25);  
 jcomp5.setBounds (10, 105, 100, 25);  
 *roomNo*.setBounds (165, 105, 185, 25);  
 jcomp7.setBounds (10, 160, 100, 25);  
 *co2Value*.setBounds (15, 190, 350, 50);  
 jcomp9.setBounds (10, 270, 100, 25);  
 *smokeValue*.setBounds (15, 305, 350, 45);  
 *sensorStatus*.setBounds (25, 410, 100, 25);  
 *sensorStatus*.setSelected(true);  
 jcomp12.setBounds (10, 375, 100, 25);  
 *updateButton*.setBounds (40, 470, 295, 35);  
 *appStatus*.setBounds (0, 550, 160, 25);  
 }  
  
  
 public static void main (String[] *args*) {  
 JFrame frame = new JFrame ("Sensor Emulator");  
 frame.setDefaultCloseOperation (JFrame.*EXIT\_ON\_CLOSE*);  
 frame.getContentPane().add (new Home());  
 frame.pack();  
 frame.setVisible (true);  
 frame.setResizable(false);  
 frame.addWindowListener(new *WindowAdapter*() {  
 public void windowClosing(WindowEvent *e*) {  
 if (*uniqueID*!=null){  
 try {  
 *sendDataToApi*.deleteData(*sensor*);  
 } catch (Exception *ex*) {  
 *ex*.printStackTrace();  
 }}  
 }  
 });  
 *sendDataToApi* = new SendDataToApi();  
 *updateButton*.addActionListener(new *ActionListener*() {  
  
 @Override  
 public void actionPerformed(ActionEvent *e*) {  
 System.*out*.println("Button clicked! ");  
 *makeSensorObject*();  
  
 if (*connected*) {  
 *timer*.cancel();  
 *updateButton*.setText("Connect to REST API");  
 *appStatus*.setText("status: Disconnected!");  
 try {  
 *sendDataToApi*.deleteData(*sensor*);  
 } catch (Exception *ex*) {  
 *ex*.printStackTrace();  
 }  
 *uniqueID* =null;  
 *sensor*=null;  
 } else {  
 try {  
  
 *uniqueID* = *sendDataToApi*.makeRequest(*sensor*);  
 if (*uniqueID* == null) {  
 *appStatus*.setText("status: Error While Connecting");  
 }else {  
 *updateButton*.setText("Disconnect from API");  
 *appStatus*.setText("status: Connected!");  
 *connected* = true;  
 *autoUpdate*();  
 }  
  
  
 } catch (IOException *ex*) {  
 *ex*.printStackTrace();  
 } catch (InterruptedException *ex*) {  
 *ex*.printStackTrace();  
 }  
 }  
 }  
 });  
  
 }  
  
 private static void autoUpdate() {  
  
 *TimerTask* task = new *TimerTask*() {  
  
 @Override  
 public void run() {  
 *makeSensorObject*();  
 try {  
 *sendDataToApi*.updateData(*sensor*);  
 } catch (Exception *e*) {  
 *e*.printStackTrace();  
 }  
  
 }  
 };  
  
 *timer* = new Timer();  
 *timer*.schedule(task, new Date(), 10000);  
 }  
  
 private static void makeSensorObject() {  
  
 if (*sensor* == null) {  
 *sensor* = new Sensor();  
 *rand* = new Random();  
 *sensor*.setFloorNo(String.*valueOf*(*rand*.nextInt(20)));  
 *floorNo*.setText(*sensor*.getFloorNo());  
 *sensor*.setRoomNo(String.*valueOf*(*rand*.nextInt(20)));  
 *roomNo*.setText(*sensor*.getRoomNo());  
 }  
  
  
  
 *sensor*.setSmokeValue(*rand*.nextInt(10)+1);  
 *co2Value*.setValue(*sensor*.getCo2Value());  
  
 *sensor*.setCo2Value(*rand*.nextInt(10)+1);  
 *smokeValue*.setValue(*sensor*.getSmokeValue());  
  
 if (*uniqueID*!=null){  
 *sensor*.setId(*uniqueID*);  
 *idField*.setText(*uniqueID*);  
 }  
  
 *sensor*.setStatus(*sensorStatus*.isSelected());  
 }  
  
  
}

**Email and sms server**

**webserver.java**

import controller.ApiConnection;  
import model.Sensor;  
  
import java.util.ArrayList;  
  
public class WebServer {  
  
 public static void main(String[] *args*) throws Exception {  
 ApiConnection connection = new ApiConnection();  
 System.*out*.println("Email and SMS Server Running! ");  
 while (true) {  
  
 *//get all sensor with alert status* ArrayList<Sensor> alertList = connection.getItemsWithAlert();  
  
 for (Sensor sensor : alertList) {  
  
 *//send emails* EmailServer.*sendEmail*(sensor);  
 *//send sms* SMSServer.*sendSMS*(sensor);  
  
 *//set alert false after sending email and sms* connection.updateAlertStatus(sensor);  
 }  
 }  
 }  
  
  
}

**smsserver.java**

import model.Sensor;

public class SMSServer {

public static void sendSMS(Sensor *sensor*){

if (*sensor*.isSmokeAlertStatus()){

System.*out*.println("SMS Sent :");

System.*out*.println("Detected High Smoke level ( "+*sensor*.getSmokeValue() + " )"+ "in floor "+*sensor*.getFloorNo()+" in room : "

+*sensor*.getSmokeValue());

}

if (*sensor*.isCo2AlertStatus()){

System.*out*.println("SMS Sent :");

System.*out*.println("Detected High C02 level ( "+*sensor*.getCo2Value() + " )"+ "in floor "+*sensor*.getFloorNo()+" in room : "

+*sensor*.getSmokeValue());

}

}

}

**emailservice.java**

import model.Sensor;

public class EmailServer {

public static void sendEmail(Sensor *sensor*){

if (*sensor*.isSmokeAlertStatus()){

System.*out*.println("Email Sent :");

System.*out*.println("Detected High Smoke level ( "+*sensor*.getSmokeValue() + " )"+ "in floor "+*sensor*.getFloorNo()+" in room : "

+*sensor*.getSmokeValue());

}

if (*sensor*.isCo2AlertStatus()){

System.*out*.println("Email Sent :");

System.*out*.println("Detected High C02 level ( "+*sensor*.getCo2Value() + " )"+ "in floor "+*sensor*.getFloorNo()+" in room : "

+*sensor*.getSmokeValue());

}

}

}

**apiconnection.java**

package controller;  
  
import model.Sensor;  
import org.json.JSONArray;  
import org.json.JSONObject;  
  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
import java.io.*OutputStream*;  
import java.net.*HttpURLConnection*;  
import java.net.URL;  
import java.util.ArrayList;  
  
public class ApiConnection {  
  
  
 public ArrayList<Sensor> getItemsWithAlert() throws Exception {  
 ArrayList<Sensor> faList = new ArrayList<>();  
 JSONArray myResponse ;  
 try {  
 String url = "http://localhost:4000/monitoring/details/";  
 URL obj = new URL(url);  
 *HttpURLConnection* con = (*HttpURLConnection*) obj.openConnection();  
 *// optional default is GET* con.setRequestMethod("GET");  
 *//add request header* con.setRequestProperty("User-Agent", "Mozilla/5.0");  
 int responseCode = con.getResponseCode();  
  
 BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));  
 String inputLine;  
 StringBuffer response = new StringBuffer();  
 while ((inputLine = in.readLine()) != null) {  
 response.append(inputLine);  
 }  
 in.close();  
  
  
 myResponse = new JSONArray(response.toString());  
  
 JSONObject sensor;  
  
 for (int i = 0; i < myResponse.length(); i++) {  
 sensor = myResponse.getJSONObject(i);  
  
 Sensor currentSensor = new Sensor();  
  
 currentSensor.setCo2AlertStatus(sensor.getBoolean("co2\_Alert"));  
 currentSensor.setSmokeAlertStatus(sensor.getBoolean("smoke\_Alert"));  
  
 *//fileter objects with alert status true* if (currentSensor.isCo2AlertStatus() || currentSensor.isSmokeAlertStatus()) {  
 currentSensor.setStatus(sensor.getBoolean("status"));  
 currentSensor.setId(String.*valueOf*(sensor.getString("\_id")));  
 currentSensor.setFloorNo(String.*valueOf*(sensor.getInt("floorNo")));  
 currentSensor.setRoomNo(String.*valueOf*(sensor.getInt("roomNo")));  
 currentSensor.setSmokeValue(Integer.*valueOf*(sensor.getInt("smokelevel")));  
 currentSensor.setCo2Value(Integer.*valueOf*(sensor.getInt("co2\_level")));  
 faList.add(currentSensor);  
 }  
 }  
  
 } catch (Exception *e*) {  
 *e*.printStackTrace();  
 }  
  
 return faList;  
 }  
  
 public void updateAlertStatus(Sensor *sensor*) throws Exception {  
  
 final String POST\_PARAMS =  
 "{\n" + "\"status\": "+*sensor*.isStatus()+",\r\n" +  
 " \"floorNo\": "+*sensor*.getFloorNo()+",\r\n" +  
 " \"roomNo\": "+*sensor*.getRoomNo()+",\r\n" +  
 " \"smokelevel\":"+*sensor*.getSmokeValue()+",\r\n" +  
 " \"co2\_Alert\":"+false+",\r\n" +  
 " \"smoke\_Alert\":"+false+",\r\n" +  
 " \"co2\_level\": "+*sensor*.getCo2Value()+"" + "\n}";  
  
  
 URL obj = new URL("http://localhost:4000/monitoring/details/"+*sensor*.getId());  
 *HttpURLConnection* postConnection = (*HttpURLConnection*) obj.openConnection();  
 postConnection.setRequestMethod("PUT");  
 postConnection.setRequestProperty("Content-Type", "application/json");  
 postConnection.setDoOutput(true);  
 *OutputStream* os = postConnection.getOutputStream();  
 os.write(POST\_PARAMS.getBytes());  
 os.flush();  
 os.close();  
 int responseCode = postConnection.getResponseCode();  
  
 int req=*HttpURLConnection*.*HTTP\_CREATED* ;  
  
 if (responseCode == (req-1)) { *//success* BufferedReader in = new BufferedReader(new InputStreamReader(  
 postConnection.getInputStream()));  
 String inputLine;  
 StringBuffer response = new StringBuffer();  
 while ((inputLine = in .readLine()) != null) {  
 response.append(inputLine);  
 } in .close();  
  
  
 } else {  
 System.*out*.println("UPDATE NOT WORKED");  
 }  
  
  
 }  
}

fireAlam.java

import java.io.Serializable;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class fireAlarm implements Serializable {

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

String id;

int floor;

int roomNumber;

int smokelevel;

int co2level;

Boolean status;

Boolean smokeAlert;

Boolean co2Alert;

public fireAlarm() throws RemoteException {

super();

}

public fireAlarm(String id, int floor, int roomNumber, int smokelevel, int co2level) throws RemoteException {

super();

this.id = id;

this.floor = floor;

this.roomNumber = roomNumber;

this.smokelevel = smokelevel;

this.co2level = co2level;

this.smokeAlert=false;

this.co2Alert=false;

this.status=false;

}

public void setId(String id) {

this.id = id;

}

public int getFloor() {

return floor;

}

public void setFloor(int floor) {

this.floor = floor;

}

public int getRoomNumber() {

return roomNumber;

}

public void setRoomNumber(int roomNumber) {

this.roomNumber = roomNumber;

}

public int getSmokelevel() {

return smokelevel;

}

public void setSmokelevel(int smokelevel) {

this.smokelevel = smokelevel;

}

public int getCo2level() {

return co2level;

}

public void setCo2level(int co2level) {

this.co2level = co2level;

}

public Boolean getSmokeAlert() {

return smokeAlert;

}

public void setSmokeAlert(Boolean smokeAlert) {

this.smokeAlert = smokeAlert;

}

public Boolean getCo2Alert() {

return co2Alert;

}

public void setCo2Alert(Boolean co2Alert) {

this.co2Alert = co2Alert;

}

public String getId() {

return id;

}

public Boolean getStatus() {

return status;

}

public void setStatus(Boolean status) {

this.status = status;

}

}

Frame1.java

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JButton;

**import** java.awt.Color;

**import** java.awt.event.ActionListener;

**import** java.rmi.Naming;

**import** java.rmi.RemoteException;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.util.ArrayList;

**import** java.awt.event.ActionEvent;

**import** javax.swing.JLabel;

**import** javax.swing.JOptionPane;

**import** javax.swing.JTextField;

**import** javax.swing.event.ListSelectionEvent;

**import** javax.swing.event.ListSelectionListener;

**import** javax.swing.table.DefaultTableModel;

**import** com.sun.tools.javac.code.Attribute.Array;

**import** jdk.internal.org.objectweb.asm.tree.IntInsnNode;

**import** java.awt.Font;

**import** java.awt.Frame;

**import** javax.swing.JTable;

**import** javax.swing.JScrollPane;

**public** **class** Frame1 **extends** JFrame{

**public** Frame1() {

}

**private** **static** JFrame *frame*;

**private** **static** JTextField *textField*;

**private** **static** JTextField *textField\_1*;

**private** **static** JTextField *textField\_2*;

**private** **static** JTextField *textField\_3*;

**private** **static** JTextField *textField\_4*;

**private** **static** JTable *table*;

**static** ArrayList<fireAlarm> *list* = **new**

ArrayList<fireAlarm>();

**static** **void** updateArrayList(ArrayList<fireAlarm> mlist) {

*list*=mlist;

*notifyDataSetChanged*();

}

**static** **void** notifyDataSetChanged() {

DefaultTableModel model = (DefaultTableModel) *table*.getModel();

model.getDataVector().removeAllElements();

Object rowData[] = **new** Object[4];

**for**(**int** i = 0; i < *list*.size(); i++)

{

model.addColumn("ID");

model.addColumn("fLOOR nO");

model.addRow(**new** Object[]{*list*.get(i).toString()});

}

// add header of the table

String header[] = **new** String[] { "Sensors" };

// add header in table model

model.setColumnIdentifiers(header);

//set model into the table object

*table*.setModel(model);

}

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

// EventQueue.invokeLater(new Runnable() {

// public void run() {

// try {

// Frame1 window = new Frame1();

// window.frame.setVisible(true);

// } catch (Exception e) {

// e.printStackTrace();

// }

// }

// });

*initialize*();

RMI\_Client rmc=**new** RMI\_Client();

*list*=rmc.readAll();

*notifyDataSetChanged*();

}

/\*\*

\* Create the application.

\*/

/\*\*

\* Initialize the contents of the frame.

\*/

**private** **static** **void** initialize() {

*frame* = **new** JFrame();

*frame*.getContentPane().setBackground(**new** Color(0, 206, 209));

*frame*.setBounds(100, 100, 706, 407);

*frame*.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

*frame*.getContentPane().setLayout(**null**);

*frame*.setVisible(**true**);

JButton btnNewButton = **new** JButton("Add");

btnNewButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent arg0) {

**try** {

RMI\_Client client=**new** RMI\_Client();

// DefaultTableModel model = (DefaultTableModel)table.getModel();

// model.addRow(new Object [] {Integer.parseInt(textField.getText()),Integer.parseInt(textField\_1.getText()),Integer.parseInt(textField\_2.getText()),Integer.parseInt(textField\_3.getText()),Integer.parseInt(textField\_4.getText()),});

client.add((*textField*.getText()), Integer.*parseInt*(*textField\_1*.getText()), Integer.*parseInt*(*textField\_2*.getText()), Integer.*parseInt*(*textField\_3*.getText()), Integer.*parseInt*(*textField\_4*.getText()));

JOptionPane .*showMessageDialog*(*frame*, "Data added");

*textField*.setText(**null**);

*textField\_1*.setText(**null**);

*textField\_2*.setText(**null**);

*textField\_3*.setText(**null**);

*textField\_4*.setText(**null**);

*list*=client.readAll();

*notifyDataSetChanged*();

} **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

});

// btnNewButton.setBounds(35, 41, 130, 36);

// frame.getContentPane().add(btnNewButton);

//

btnNewButton.setForeground(**new** Color(255, 255, 255));

btnNewButton.setBackground(**new** Color(0, 128, 0));

btnNewButton.setBounds(52, 88, 97, 25);

*frame*.getContentPane().add(btnNewButton);

// try {

// Class.forName("com.mysql.cj.jdbc.Driver");

// Connection con2 = DriverManager.getConnection("jdbc:mysql://localhost:3307/firealarm?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC","root","");

//

//

//

//

// String query1 = "Update sensors set ID = '"+textField.getText()+"', floorNo = '"+textField\_1.getText()+"', roomNo = '"+textField\_2.getText()+"', smokeLvl = '"+textField\_3.getText()+"', CO2Lvl = '"+textField\_4.getText()+"' where ID = '"+textField.getText()+"' ";

// PreparedStatement stm = con2.prepareStatement(query1);

//

//

// stm.execute();

// JOptionPane.showMessageDialog(null,"Data Updated");

// Frame1 h1 = new Frame1();

// h1.setVisible(true);

//

//

// stm.close();

//

//

// }catch(Exception e1) {

//

// System.out.println(e1);

//

// }

// btnUpdate.setBounds(564, 451, 122, 35);

// frame.getContentPane().add(btnUpdate);

JButton btnNewButton\_1 = **new** JButton("Delete");

btnNewButton\_1.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

//

**try** {

RMI\_Client client=**new** RMI\_Client();

client.delete(*textField*.getText());

// System.out.println("selected :"+(textField.getText()));

//

// textField.getText();

//

// System.out.println(textField.getText());

// int a = table.getSelectedRow();

// ((DefaultTableModel)table.getModel()).removeRow(a);

**for**( **int** a = *table*.getSelectedRow() - 1; a >= 0; a-- )

{

*table*.remove(a);

}

RMI\_Client rmc=**new** RMI\_Client();

*list*=rmc.readAll();

*notifyDataSetChanged*();

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

// try {

// Class.forName("com.mysql.cj.jdbc.Driver");

// Connection con3 = DriverManager.getConnection("jdbc:mysql://localhost:3307/firealarm?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC","root","");

//

//

//

//

// String query2 = "delete from sensors where ID = '"+textField.getText()+"' ";

// PreparedStatement stm = con3.prepareStatement(query2);

//

//

// stm.execute();

// JOptionPane.showMessageDialog(null,"Data Deleted");

// Frame1 h3 = new Frame1();

// h3.setVisible(true);

//

//

// stm.close();

//

//

// }catch(Exception e1) {

//

// System.out.println(e1);

//

// }

}

});

// btnDeleteSensor.setBounds(698, 450, 122, 36);

// contentPane.add(btnDeleteSensor);

btnNewButton\_1.setForeground(**new** Color(255, 255, 255));

btnNewButton\_1.setBackground(Color.***RED***);

btnNewButton\_1.setBounds(342, 88, 97, 25);

*frame*.getContentPane().add(btnNewButton\_1);

JLabel lblFloorNo = **new** JLabel("Floor No");

lblFloorNo.setFont(**new** Font("SansSerif", Font.***BOLD***, 14));

lblFloorNo.setForeground(**new** Color(255, 255, 255));

lblFloorNo.setBounds(12, 189, 87, 16);

*frame*.getContentPane().add(lblFloorNo);

JLabel lblRoomNo = **new** JLabel("Room No");

lblRoomNo.setFont(**new** Font("SansSerif", Font.***BOLD***, 14));

lblRoomNo.setForeground(**new** Color(255, 255, 255));

lblRoomNo.setBounds(12, 229, 87, 16);

*frame*.getContentPane().add(lblRoomNo);

JLabel lblNewLabel\_1 = **new** JLabel("Level of Smoke");

lblNewLabel\_1.setFont(**new** Font("SansSerif", Font.***BOLD***, 14));

lblNewLabel\_1.setForeground(**new** Color(255, 255, 255));

lblNewLabel\_1.setBounds(12, 279, 116, 16);

*frame*.getContentPane().add(lblNewLabel\_1);

JLabel lblNewLabel\_2 = **new** JLabel("Level of CO2");

lblNewLabel\_2.setFont(**new** Font("SansSerif", Font.***BOLD***, 14));

lblNewLabel\_2.setForeground(**new** Color(255, 255, 255));

lblNewLabel\_2.setBounds(12, 327, 97, 16);

*frame*.getContentPane().add(lblNewLabel\_2);

*textField* = **new** JTextField();

*textField*.setBounds(139, 138, 80, 22);

*frame*.getContentPane().add(*textField*);

*textField*.setColumns(10);

*textField\_1* = **new** JTextField();

*textField\_1*.setBounds(139, 187, 80, 22);

*frame*.getContentPane().add(*textField\_1*);

*textField\_1*.setColumns(10);

*textField\_2* = **new** JTextField();

*textField\_2*.setBounds(139, 227, 80, 22);

*frame*.getContentPane().add(*textField\_2*);

*textField\_2*.setColumns(10);

*textField\_3* = **new** JTextField();

*textField\_3*.setBounds(140, 277, 80, 22);

*frame*.getContentPane().add(*textField\_3*);

*textField\_3*.setColumns(10);

*textField\_4* = **new** JTextField();

*textField\_4*.setBounds(138, 325, 80, 22);

*frame*.getContentPane().add(*textField\_4*);

*textField\_4*.setColumns(10);

JButton btnBack = **new** JButton("Logout");

btnBack.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

*frame*.dispose();

login b2 = **new** login();

b2.*main*(**null**);

}

});

btnBack.setForeground(**new** Color(255, 255, 255));

btnBack.setBackground(**new** Color(189, 183, 107));

btnBack.setBounds(29, 39, 97, 25);

*frame*.getContentPane().add(btnBack);

JScrollPane scrollPane = **new** JScrollPane();

scrollPane.setBounds(242, 141, 434, 206);

*frame*.getContentPane().add(scrollPane);

*table* = **new** JTable();

scrollPane.setViewportView(*table*);

//DefaultTableModel model = (DefaultTableModel) table.getModel();

JButton btnNewButton\_3 = **new** JButton("Update");

btnNewButton\_3.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**try** {

// DefaultTableModel model = (DefaultTableModel)table.getModel();

// model.addRow(new Object [] {Integer.parseInt(textField.getText()),Integer.parseInt(textField\_1.getText()),Integer.parseInt(textField\_2.getText()),Integer.parseInt(textField\_3.getText()),Integer.parseInt(textField\_4.getText()),});

RMI\_Client client=**new** RMI\_Client();

fireAlarm fa1=**new** fireAlarm((*textField*.getText()), Integer.*parseInt*(*textField\_1*.getText()), Integer.*parseInt*(*textField\_2*.getText()), Integer.*parseInt*(*textField\_3*.getText()), Integer.*parseInt*(*textField\_4*.getText()));

client.update(fa1);

JOptionPane .*showMessageDialog*(*frame*, "Updated");

*textField*.setText(**null**);

*textField\_1*.setText(**null**);

*textField\_2*.setText(**null**);

*textField\_3*.setText(**null**);

*textField\_4*.setText(**null**);

RMI\_Client rmc=**new** RMI\_Client();

*list*=rmc.readAll();

*notifyDataSetChanged*();

} **catch** (Exception ee) {

// **TODO** Auto-generated catch block

ee.printStackTrace();

}

}

});

btnNewButton\_3.setBounds(171, 88, 97, 25);

*frame*.getContentPane().add(btnNewButton\_3);

*table*.getSelectionModel().addListSelectionListener(**new** ListSelectionListener() {

**public** **void** valueChanged(ListSelectionEvent lse) {

**if** (!lse.getValueIsAdjusting()) {

System.***out***.println("Selection Changed");

fireAlarm alarm = *list*.get(*table*.getSelectedRow());

*textField*.setText(alarm.id);

*textField\_1*.setText(String.*valueOf*(alarm.floor));

*textField\_2*.setText(String.*valueOf*(alarm.roomNumber));

*textField\_3*.setText(String.*valueOf*(alarm.smokelevel));

*textField\_4*.setText(String.*valueOf*(alarm.co2level));

}

}

});

}

}

Login.java

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JButton;

**import** java.awt.event.ActionListener;

**import** java.awt.event.ActionEvent;

**import** java.awt.Color;

**public** **class** login {

**private** JFrame frame;

/\*\*

\* Launch the application.

\*/

**public** **static** **void** main(String[] args) {

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

login window = **new** login();

window.frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the application.

\*/

**public** login() {

initialize();

}

/\*\*

\* Initialize the contents of the frame.

\*/

**private** **void** initialize() {

frame = **new** JFrame();

frame.getContentPane().setBackground(**new** Color(0, 206, 209));

frame.setBounds(100, 100, 513, 324);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

frame.getContentPane().setLayout(**null**);

JButton btnNewButton = **new** JButton("User");

btnNewButton.setBackground(Color.***BLUE***);

btnNewButton.setForeground(Color.***WHITE***);

btnNewButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

frame.dispose();

loginUser n = **new** loginUser();

n.*main*(**null**);

}

});

btnNewButton.setBounds(114, 127, 97, 25);

frame.getContentPane().add(btnNewButton);

JButton btnNewButton\_1 = **new** JButton("Admin");

btnNewButton\_1.setBackground(Color.***BLUE***);

btnNewButton\_1.setForeground(Color.***WHITE***);

btnNewButton\_1.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent arg0) {

frame.dispose();

loginAdmin n1 = **new** loginAdmin();

n1.*main*(**null**);

}

});

btnNewButton\_1.setBounds(286, 127, 97, 25);

frame.getContentPane().add(btnNewButton\_1);

}

}

loginAdmin.java

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.ArrayList;

public interface sensorListener extends Remote{

ArrayList<fireAlarm> sensorStatusChanged() throws RemoteException;

}

loginUser.java

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.ArrayList;

public interface sensorListener extends Remote{

ArrayList<fireAlarm> sensorStatusChanged() throws RemoteException;

}

RMI\_Client

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.ArrayList;

public interface ServerInt extends Remote {

public void serverCheck() throws Exception;

public ArrayList<fireAlarm> addItem(fireAlarm Item) throws Exception;

public ArrayList<fireAlarm> getItem(String id) throws Exception;

public ArrayList<fireAlarm> getItems() throws Exception;

public ArrayList<fireAlarm> updateItem(fireAlarm Item) throws Exception;

public ArrayList<fireAlarm> deleteItem(String id) throws Exception;

public void addSensorListener(sensorListener sl)throws Exception;

public void co2Alert(fireAlarm id) throws Exception;

public void smokeAlert(fireAlarm id) throws Exception;

}

sensorListner.java

**import** java.rmi.Remote;

**import** java.rmi.RemoteException;

**import** java.util.ArrayList;

**public** **interface** sensorListener **extends** Remote{

ArrayList<fireAlarm> sensorStatusChanged() **throws** RemoteException;

}

ServerInt.java

**import** java.rmi.Remote;

**import** java.util.ArrayList;

**public** **interface** ServerInt **extends** Remote {

**public** **void** serverCheck() **throws** Exception;

**public** ArrayList<fireAlarm> addItem(fireAlarm Item) **throws** Exception;

**public** ArrayList<fireAlarm> getItem(String id) **throws** Exception;

**public** ArrayList<fireAlarm> getItems() **throws** Exception;

**public** ArrayList<fireAlarm> updateItem(fireAlarm Item) **throws** Exception;

**public** ArrayList<fireAlarm> deleteItem(String id) **throws** Exception;

**public** **void** addSensorListener(sensorListener sl)**throws** Exception;

**public** **void** co2Alert(fireAlarm id) **throws** Exception;

**public** **void** smokeAlert(fireAlarm id) **throws** Exception;

}

userDisplay.java

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** java.awt.Color;

**import** javax.swing.JButton;

**import** java.awt.event.ActionListener;

**import** java.rmi.Naming;

**import** java.util.ArrayList;

**import** java.awt.event.ActionEvent;

**import** javax.swing.JTable;

**import** javax.swing.table.DefaultTableModel;

**import** javax.swing.table.TableModel;

**import** net.proteanit.sql.DbUtils;

**import** javax.swing.JScrollBar;

**import** javax.swing.JScrollPane;

**public** **class** userDisplay {

**private** JFrame frame;

**private** **static** JTable *table*;

**static** ArrayList<fireAlarm> *list* = **new**

ArrayList<fireAlarm>();

/\*\*

\* Launch the application.

\*/

**public** **static** **void** main(String[] args) {

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

userDisplay window = **new** userDisplay();

window.frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

**static** **void** updateArrayList(ArrayList<fireAlarm> mlist) {

*list*=mlist;

*notifyDataSetChanged*();

}

**static** **void** notifyDataSetChanged() {

DefaultTableModel model = (DefaultTableModel) *table*.getModel();

model.getDataVector().removeAllElements();

Object rowData[] = **new** Object[4];

**for**(**int** i = 0; i < *list*.size(); i++)

{

model.addColumn("ID");

model.addColumn("fLOOR nO");

model.addRow(**new** Object[]{*list*.get(i).toString()});

}

// add header of the table

String header[] = **new** String[] { "Sensors" };

// add header in table model

model.setColumnIdentifiers(header);

//set model into the table object

*table*.setModel(model);

}

/\*\*

\* Create the application.

\* **@throws** Exception

\*/

**public** userDisplay() **throws** Exception {

initialize();

RMI\_Client rmc=**new** RMI\_Client();

*list*=rmc.readAll();

*notifyDataSetChanged*();

}

/\*\*

\* Initialize the contents of the frame.

\*/

**private** **void** initialize() {

frame = **new** JFrame();

frame.getContentPane().setBackground(**new** Color(0, 206, 209));

frame.getContentPane().setLayout(**null**);

JButton btnNewButton = **new** JButton("Logout");

btnNewButton.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

frame.dispose();

login b3 = **new** login();

b3.*main*(**null**);

}

});

btnNewButton.setForeground(**new** Color(255, 255, 255));

btnNewButton.setBackground(**new** Color(189, 183, 107));

btnNewButton.setBounds(12, 13, 97, 25);

frame.getContentPane().add(btnNewButton);

JScrollPane scrollPane = **new** JScrollPane();

scrollPane.setBounds(23, 87, 386, 153);

frame.getContentPane().add(scrollPane);

*table* = **new** JTable();

scrollPane.setViewportView(*table*);

DefaultTableModel model = (DefaultTableModel) *table*.getModel();

frame.setBounds(100, 100, 450, 300);

frame.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

}

}