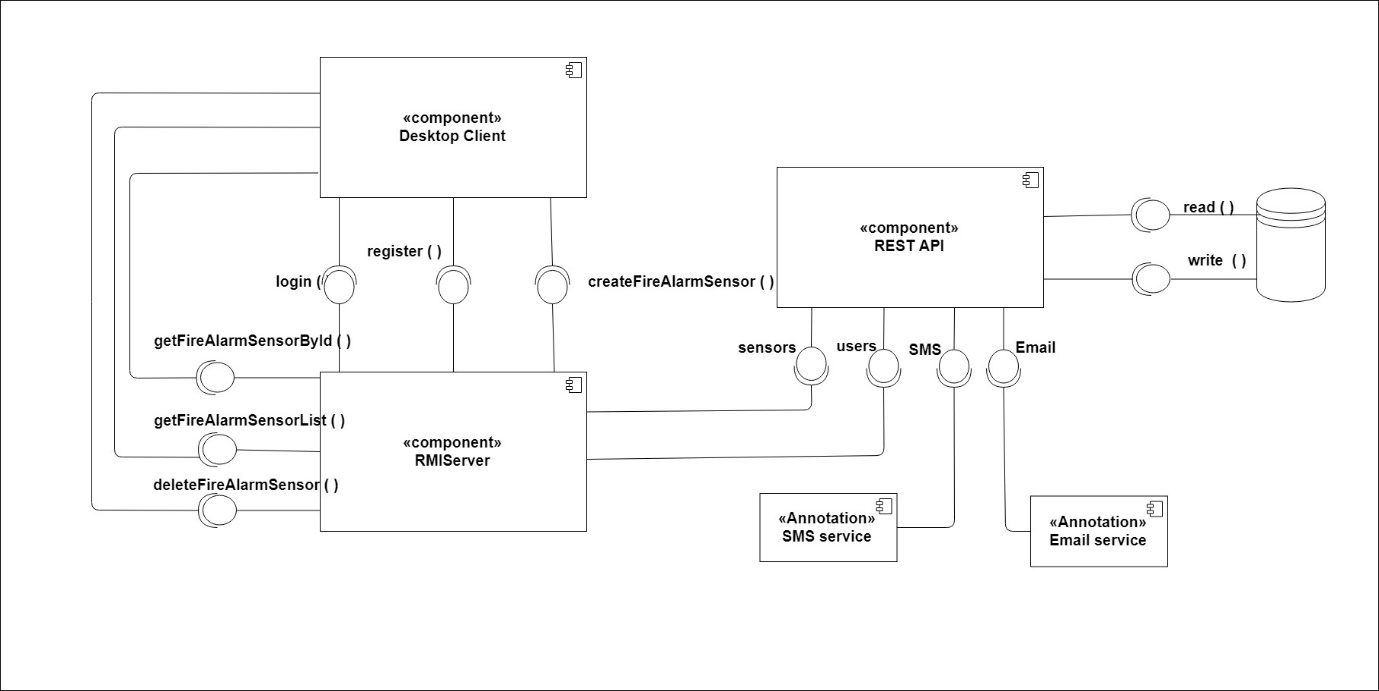
**Assignment 2 – Fire Alarm Service**

**Group Details**

|  |  |  |
| --- | --- | --- |
|  | **Student Registration Number** | **Student Name** |
| **1** | **IT18141016** | **Wickramasinghe H.C.P.** |
| **2** | **IT18158496** | **Wijekoon W.M.S.R.** |
| **3** | **IT18135480** | **Wijesinghe G.K.H.S.** |
| **4** | **IT18057270** | **Dissanayake U.N.** |

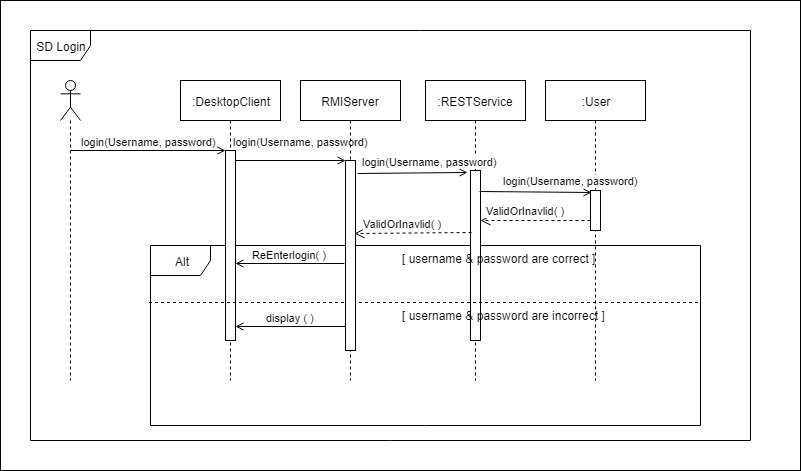
**Component diagram**

****

In this diagram shows how the system interacts with each service. Web client get all services provided by RMI service. RMI server needs to get details from database, for that RMI server makes a connection through REST API. All the database handlers are maintained by REST API. In REST API, for each request from RMI has a response contains either success message or failure message.

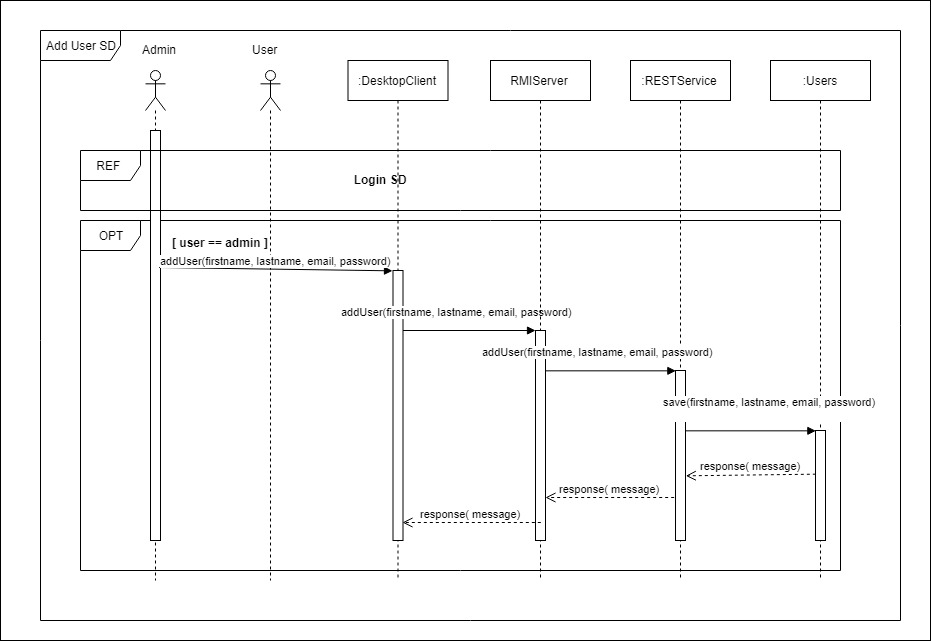
**Sequence diagrams**

1. Process of user login to the fire alarm system



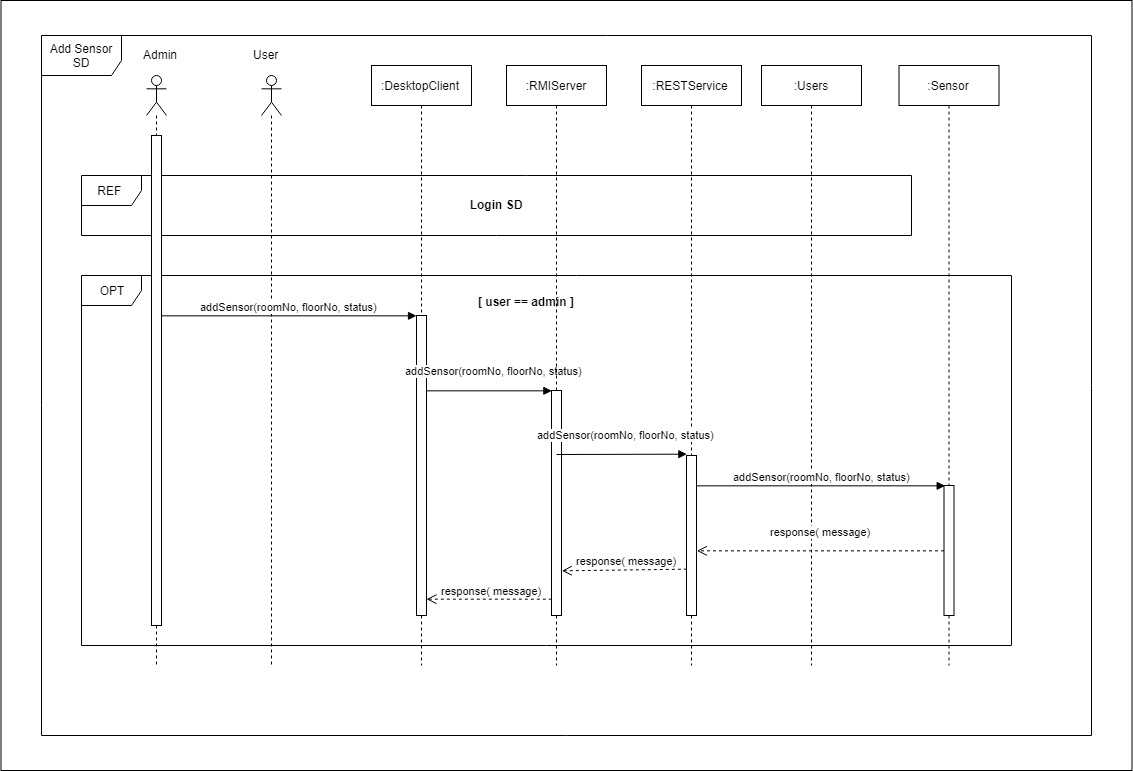
Registered user can login to the system by using their authentications credentials. For the authentication user needs to input their valid email and password. Whenever user insert the credentials system sends email and password through REST API by using RMI service as a JSON format. Those details are checked by REST API and send an authentication response message including authentication is success or not. When authentication is success user can view the all sensors details and alerts messages.

1. Process of new user registration to the fire alarm system

****

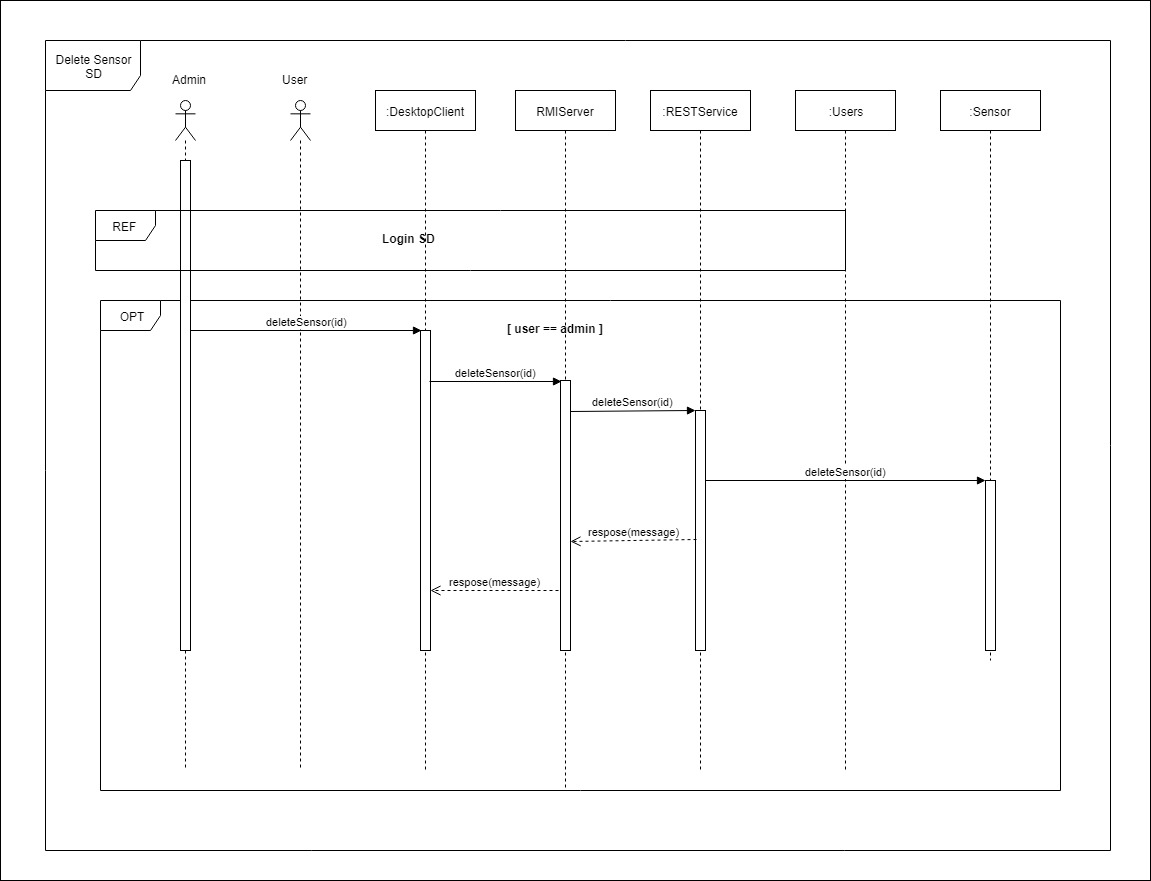
Whenever user login to the system checks whether the user has administration access to the system or not. When the user has administration access system grants permission to add new users to system. For registering a new user client needs to enter user first name, last name, email address and the password. Desktop client takes all new user details and send all those details to REST API by using RMI service. REST API sends response message regarding user is successfully registered or not. Once user is successfully registered, that user can log in to the system with accurate login credentials.

1. Process of adding new sensor to the fire alarm system



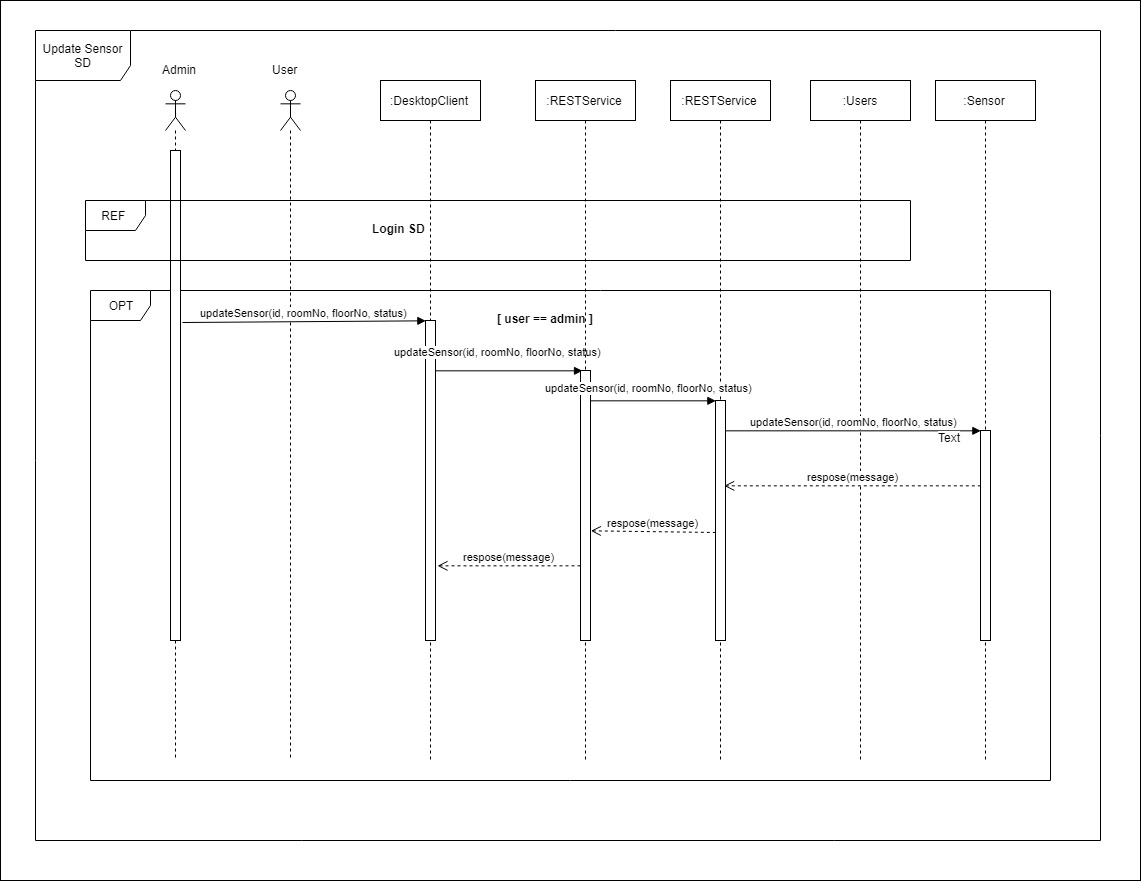
Administration can add new sensors to the system. For adding a new sensor to the system admin needs to add room number, floor number, and status. Those details are checked by RMI server and send those details to REST API in JSON format. REST API inserts those details to database and send a response message including successfully added or not. If new sensor is succesfully added , new sensor will be shown in the all sensors table.

1. Process of deleting existing sensor from the fire alarm system



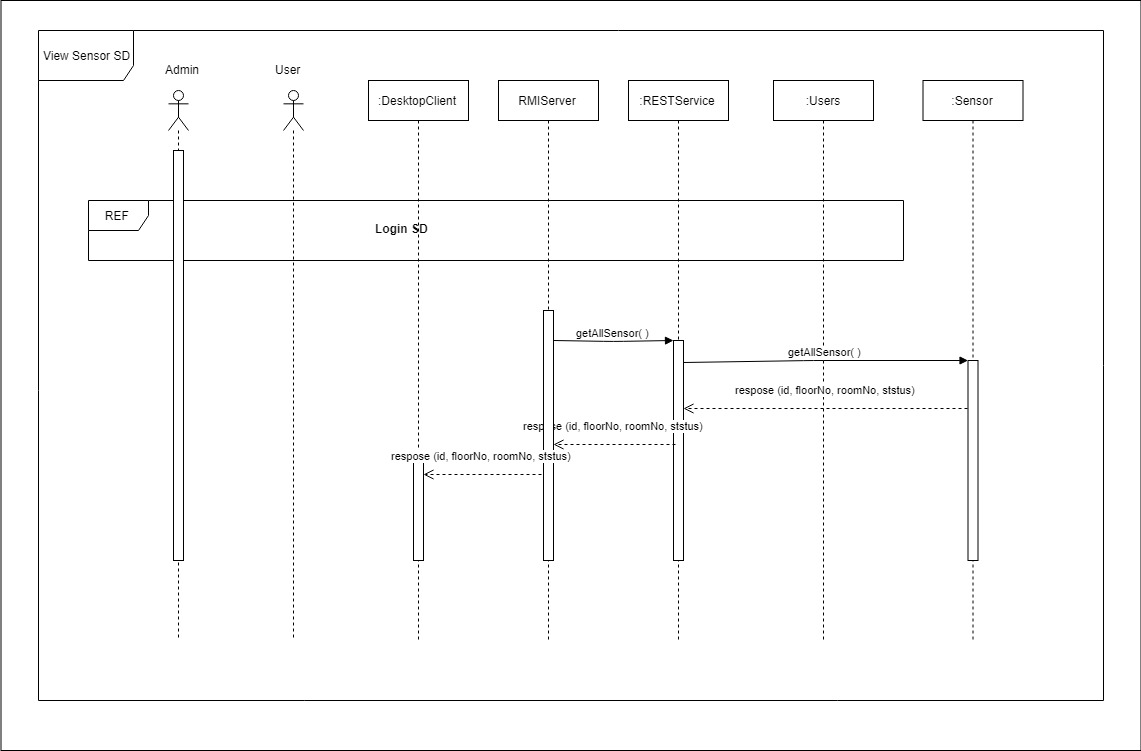
Administration can delete existing sensor from the system. They can select sensor from list. For that they need to know related floor number and room number where sensor is located. When the user selects the relevant sensor, system finds the relevant id of the sensor and send the delete request from RMI service to REST API with relevant sensor id. After that REST API send a response message regarding sensor has been deleted or not. Whenever sensor is deleted it is removed in the sensor list as well.

1. Process of updating existing sensor in the fire alarm system



Whenever administration needs to update existing sensor in the system, they can select sensor from list. For that they need to know related floor number and room number where sensor is located. When the user selects the relevant sensor and need to enter the relevant fields to be updated. System finds the relevant id of the sensor and send the update request from RMI service to REST API with relevant sensor id and data which is going to be updated in JSON format. After that REST API send a response message regarding sensor has been updated or not. Whenever sensor is updated it is updated in the sensor list as well.

1. Process of getting sensor in the fire alarm system



When a user logged in to the system, they can view all sensors in the systems. Whenever user logged into system, system calls REST API to get all sensors details (sensor id, floor number, room number, status) in JSON format. After that all those retrieved details are displayed in desktop client.

**Authentication Implementaion code**

// Authentication function (cb means callback function)

authUser = (userObject, cb) => {

this.checkUser({ email: userObject.username }, (err, result) => {

if (err) {

return cb(err);

} else {

// Comparing with bcrypt with user password

if (bcrypt.compareSync(userObject.password, result.password) === true) {

return cb(null, true);

} else {

return cb("Unauthorized");

}

}

})

};

We are using bcrypt to hash user password. Whenever user login to the system, this function compare hash password with user input. If matches, user will be allowed to login to the system. Otherwise, user will get a Unauthorized message.

**Appendix**

1. **REST API Service**

**Index.js**

import mongoose from 'mongoose';

import app from './bootstrap/routes-config';

import { port, mongoUri } from "./config/core.config";

// Connect to MongoDB

mongoose.connect(mongoUri, { useUnifiedTopology: true, useNewUrlParser: true })

.then(() => {

console.log("DB Connection has been established successfully");

})

.catch((err) => {

console.log(`Unable to connect DB: ${err}`);

});

// Starting Server

app.listen(port, (err) => {

console.log(`Server has started on ${port}`);

});

**Express-config.js**

import express from 'express';

import compress from 'compression';

import cors from 'cors';

import helmet from 'helmet';

const app = express();

// Configure express middlewares

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use(compress());

app.use(cors());

app.use(helmet());

export default app;

**Routes-config.js**

// Main Route Files

import app from './express-config';

import userRoutes from "../app/routes/user/user.routes";

import authRoutes from "../app/routes/auth/auth.routes";

import fireAlarmRoutes from "../app/routes/fire-alarm/fire-alarm.routes";

import emailRoutes from "../app/routes/email/email.routes";

import smsRoutes from "../app/routes/sms/sms.routes";

// Auth routes

app.use('/api/auth', authRoutes);

// User routes

app.use('/api/users', userRoutes);

// Fire-alarm routes

app.use('/api/fire-alarms', fireAlarmRoutes);

// Email routes

app.use('/api/emails', emailRoutes);

// SMS routes

app.use('/api/sms', smsRoutes);

// Web route

app.get('\*', (req, res) => {

res.status(200).send("Hello");

});

export default app;

**auth.controller.js**

import UserService from "../../services/user/UserService";

// Login Function

const signIn = (req, res) => {

const userObject = {

username: req.body.username,

password: req.body.password

};

// Calling authuser method

UserService.authUser(userObject, (err, user) => {

if (err) {

return res.status(400).json({ status: "error", message: "Unauthorized" });

} else {

return res.status(200).json({ status: "success", message: "Authorized" });

}

});

};

export default {

signIn

}

**Email.controller.js**

import axios from 'axios';

// Send Email function

const sendEmail = (req, res) => {

const body = {

floor: req.body.floor,

room: req.body.room

};

mailFn(body).then((result) => {

return res.status(200).json({ status: "success", message: "Success" });

}).catch((err) => {

return res.status(400).json({ status: "error", message: err });

})

};

// Sending Ajax request

const mailFn = (data) => {

// Returning Promise

return new Promise((resolve, reject) => {

const axiosObject = {

headers: {

"Content-Type": "application/json"

},

method: 'post',

url: "http://localhost:4005/api/emails",

data: data

};

// Using axios

axios(axiosObject)

.then((result) => {

resolve(true)

})

.catch((err) => {

reject(err);

})

})

};

export default {

sendEmail

}

**Fire-alarm.controller.js**

import FireAlarmService from "../../services/fire-alarm/FireAlarmService";

// Creating a fire-alarm

const createFireAlarm = (req, res) => {

const newFireAlarm = {

floor\_no: req.body.floor\_no,

room\_no: req.body.room\_no,

smoke\_level: req.body.smoke\_level,

co2\_level: req.body.co2\_level,

status: req.body.status,

};

// Calling service method

FireAlarmService.create(newFireAlarm, (err, result) => {

if (err) {

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

}

return res.status(201).json({ status: "success", data: result });

})

};

// Getting fire-alarm list

const getFireAlarmList = (req, res) => {

// Calling service method

FireAlarmService.find({}, (err, result) => {

if (err) {

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

}

return res.status(200).json({ status: "success", data: result });

})

};

// Get fire-alarm by id

const getFireAlarmById = (req, res) => {

// Calling service method

FireAlarmService.findOne({ \_id: req.params.id }, (err, result) => {

if (err) {

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

}

return res.status(200).json({ status: "success", data: result });

})

};

// Update fire-alarm

const updateFireAlarm = (req, res) => {

const updateFireAlarm = {

floor\_no: req.body.floor\_no,

room\_no: req.body.room\_no,

smoke\_level: req.body.smoke\_level,

co2\_level: req.body.co2\_level,

status: req.body.status,

};

// Calling service method

FireAlarmService.updateFireAlarm(updateFireAlarm, req.params.id, (err, result) => {

if (err) {

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

} else {

return res.status(201).json({ status: "success", data: result });

}

})

};

// Delete fire-alarm

const deleteFireAlarm = (req, res) => {

// Calling service method

FireAlarmService.delete({ \_id: req.params.id }, (err, result) => {

if (err) {

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

}

return res.status(200).json({ status: "success", data: {} });

})

};

export default {

createFireAlarm,

getFireAlarmById,

getFireAlarmList,

updateFireAlarm,

deleteFireAlarm

}

**Sms.controller.js**

import axios from 'axios';

// Send Sms function

const sendSMS = (req, res) => {

const body = {

floor: req.body.floor,

room: req.body.room

};

smsFn(body).then((result) => {

return res.status(200).json({ status: "success", message: "Success" });

}).catch((err) => {

return res.status(400).json({ status: "error", message: err });

})

};

// Sending Ajax request

const smsFn = (data) => {

// Returning Promise

return new Promise((resolve, reject) => {

const axiosObject = {

headers: {

"Content-Type": "application/json"

},

method: 'post',

url: "http://localhost:4010/api/sms",

data: data

};

// Using axios

axios(axiosObject)

.then((result) => {

resolve(true)

})

.catch((err) => {

reject(err);

})

})

};

export default {

sendSMS

}

**user.controller.js**

import UserService from "../../services/user/UserService";

import bcrypt from "bcrypt";

// Registering a User

const createUser = (req, res) => {

const newUser = {

first\_name: req.body.first\_name,

last\_name: req.body.last\_name,

email: req.body.email,

password: bcrypt.hashSync(req.body.password, 10)

};

// Calling service method

UserService.create(newUser, (err, result) => {

if (err) {

return res.status(400).json({ status: "error" });

}

return res.status(201).json({ status: "success", data: result });

})

};

export default {

createUser

}

**Fire-alarm.model.js**

import mongoose from 'mongoose';

// Fire-alarm model

const FireAlarmSchema = mongoose.Schema({

floor\_no: {

type: Number,

required: "Floor no is required"

},

room\_no: {

type: Number,

required: "Room no is required"

},

smoke\_level: {

type: Number,

min: 1,

max: 10,

default: 1

},

co2\_level: {

type: Number,

min: 1,

max: 10,

default: 1

},

status: {

type: Boolean,

default: true

},

created\_at: {

type: Date,

default: Date.now

},

updated\_at: Date

});

export default mongoose.model('FireAlarm', FireAlarmSchema);

**user.model.js**

import mongoose from 'mongoose';

// User model

const UserSchema = mongoose.Schema({

first\_name: {

type: String,

trim: true,

required: "First Name is required"

},

last\_name: {

type: String,

trim: true,

required: "Last Name is required"

},

email: {

type: String,

trim: true,

unique: 'Email already exists',

match: [/.+\@.+\..+/, 'Please fill a valid email address'],

required: 'Email is required'

},

password: {

type: String,

required: "Password is required"

},

created\_at: {

type: Date,

default: Date.now

},

updated\_at: Date

});

export default mongoose.model('User', UserSchema);

**Auth.routes.js**

import express from "express";

import authCtrl from "../../controllers/auth/auth.controller";

// Auth routes individually

const router = express.Router();

router.route('/')

.post(authCtrl.signIn);

export default router;

**email.routes.js**

import express from "express";

import emailCtrl from "../../controllers/email/email.controller";

// Email routes individually

const router = express.Router();

router.route('/')

.post(emailCtrl.sendEmail);

export default router;

**fire-alarm.routes.js**

import express from "express";

import fireAlarmCtrl from "../../controllers/fire-alarm/fire-alarm.controller";

// Fire-alarm routes individually

const router = express.Router();

router.route('/')

.post(fireAlarmCtrl.createFireAlarm)

.get(fireAlarmCtrl.getFireAlarmList);

router.route('/:id')

.get(fireAlarmCtrl.getFireAlarmById)

.put(fireAlarmCtrl.updateFireAlarm)

.delete(fireAlarmCtrl.deleteFireAlarm);

export default router;

**sms.routes.js**

import express from "express";

import smsCtrl from "../../controllers/sms/sms.controller";

// Sms routes individually

const router = express.Router();

router.route('/')

.post(smsCtrl.sendSMS);

export default router;

**user.routes.js**

import express from "express";

import userCtrl from "../../controllers/user/user.controller";

// User routes individually

const router = express.Router();

router.route('/')

.post(userCtrl.createUser);

export default router;

**BaseService.js**

// BaseService (Repository design pattern)

export class BaseService {

constructor(model) {

this.model = model;

}

// Creating a document

create(userObject={}, cb) {

this.model.create(userObject, cb);

}

// Finding a document

findOne(filterOption={}, cb) {

this.model.findOne(filterOption, cb);

}

// Finding multiple documents

find(filterOption={}, cb) {

this.model.find(filterOption, cb);

}

// Delete a document

delete(filterOption={},cb) {

this.model.deleteOne(filterOption, cb);

}

}

**FireAlarmService.js**

import { BaseService } from "../BaseService";

import FireAlarm from "../../models/fire-alarm/fire-alarm.model";

// Fire-alarm service

class FireAlarmService extends BaseService {

constructor() {

super(FireAlarm);

}

// Updating a fire-alarm

updateFireAlarm(object, id, cb) {

// Find alarm by id

this.findOne({ \_id: id }, (err, result) => {

if (err) {

cb(err);

} else {

if (result) {

// If found do the updating

const updatedObject = result;

if (object.floor\_no !== undefined) {

updatedObject.floor\_no = object.floor\_no;

}

if (object.room\_no !== undefined) {

updatedObject.room\_no = object.room\_no;

}

if (object.smoke\_level !== undefined) {

updatedObject.smoke\_level = object.smoke\_level;

}

if (object.co2\_level !== undefined) {

updatedObject.co2\_level = object.co2\_level;

}

if (object.status !== undefined) {

updatedObject.status = object.status;

}

updatedObject.updated\_at = new Date();

updatedObject.save((err, result) => {

if (err) {

cb(err);

} else {

cb(null, result);

}

})

}

}

})

}

}

export default new FireAlarmService();

**UserService.js**

import { BaseService } from "../BaseService";

import User from "../../models/user/user.model";

import bcrypt from "bcrypt";

// User service class

class UserService extends BaseService {

constructor() {

super(User);

}

// Check user whether the user in the database (cb means callback function)

checkUser = (userObject, cb) => {

try {

this.findOne(userObject, (err, result) => {

if (err) {

return cb(err);

} else {

if (result) {

cb(null, result);

} else {

cb("Unauthorized");

}

}

})

} catch (err) {

return cb(err);

}

};

// Authentication function (cb means callback function)

authUser = (userObject, cb) => {

this.checkUser({ email: userObject.username }, (err, result) => {

if (err) {

return cb(err);

} else {

// Comparing with bcrypt with user password

if (bcrypt.compareSync(userObject.password, result.password) === true) {

return cb(null, true);

} else {

return cb("Unauthorized");

}

}

})

};

}

export default new UserService();

**core.config.js**

// Configuration file

const port = process.env.APP\_PORT|| 4000;

const mongoUri = "mongodb+srv://testuser:testuser@firealarm-service-tjtd3.mongodb.net/test?retryWrites=true&w=majority";

const salt = "125r25dsrqcsywdt73fd6t5f78gft5bgyh9u79jh83uehrt538idhf5657";

export {

port,

mongoUri,

salt

}

1. **RMI Server Client Service**

**RMIServer.java**

package rmi.server;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

import interfaces.IFireAlarmService;

import services.FireAlarmServiceImpl;

public class RMIServer extends FireAlarmServiceImpl {

static int port = 5099;

public RMIServer() throws RemoteException {

}

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

try {

// create register for relavent port number

Registry registry = LocateRegistry.createRegistry(port);

// bind the FireAlarmService in registry

registry.rebind("FireAlarmService", new FireAlarmServiceImpl());

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

} catch (Exception ex) {

System.out.println("Server Error: " + ex.toString());

ex.printStackTrace();

}

}

}

**Client.java**

package rmi.client;

import java.io.Serializable;

import java.rmi.Naming;

import interfaces.IFireAlarmService;

public class Client implements Serializable {

private IFireAlarmService stub = null;

// initialize stub

public Client() {

this.setStub();

}

//get the stub

public IFireAlarmService getStub() {

return stub;

}

// set the stub

public void setStub() {

try {

// Looking up the registry for the remote object

this.stub = (IFireAlarmService) Naming.lookup("rmi://localhost:5099/FireAlarmService");

} catch (Exception e) {

System.out.println("Client exception: " + e.toString());

e.printStackTrace();

}

}

}

**AuthResponse.java**

package response.models;

import java.io.Serializable;

// store authentication response message details

public class AuthResponse implements Serializable {

private String status;

private String message;

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

}

**EmailSmsResponse.java**

package response.models;

import java.io.Serializable;

public class EmailSmsResponse implements Serializable {

private String status;

private String message;

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

}

**FireAlarmSensorMultipleResponse.java**

package response.models;

import java.io.Serializable;

import java.util.List;

import dtos.FireAlarmSensor;

// store multiple response

public class FireAlarmSensorMultipleResponse implements Serializable {

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;

}

}

**FireAlarmSensorSingleResponse.java**

package response.models;

import java.io.Serializable;

import dtos.FireAlarmSensor;

// store single response

public class FireAlarmSensorSingleResponse implements Serializable {

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;

}

}

**UserResponse.java**

package response.models;

import java.io.Serializable;

import java.util.List;

import dtos.FireAlarmSensor;

import dtos.User;

// store registration response

public class UserResponse implements Serializable {

private String status;

private User data;

public User getData() {

return data;

}

public void setData(User data) {

this.data = data;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

}

**Auth.java**

package dtos;

import java.io.Serializable;

// set authentication details

public class Auth implements Serializable {

private String username;

private String password;

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**EmailSms.java**

package dtos;

import java.io.Serializable;

public class EmailSms implements Serializable {

private int floor;

private int room;

public int getFloor() {

return floor;

}

public void setFloor(int floor) {

this.floor = floor;

}

public int getRoom() {

return room;

}

public void setRoom(int room) {

this.room = room;

}

}

**FireAlarmSensor.java**

package dtos;

import java.io.Serializable;

// model class for fire alarm sensor details

public class FireAlarmSensor implements Serializable {

private String \_id;

private int floor\_no;

private int room\_no;

private int smoke\_level;

private int co2\_level;

private boolean status;

public String get\_id() {

return \_id;

}

public void set\_id(String \_id) {

this.\_id = \_id;

}

public int getFloor\_no() {

return floor\_no;

}

public void setFloor\_no(int floor\_no) {

this.floor\_no = floor\_no;

}

public int getRoom\_no() {

return room\_no;

}

public void setRoom\_no(int room\_no) {

this.room\_no = room\_no;

}

public int getSmoke\_level() {

return smoke\_level;

}

public void setSmoke\_level(int smoke\_level) {

this.smoke\_level = smoke\_level;

}

public int getCo2\_level() {

return co2\_level;

}

public void setCo2\_level(int co2\_level) {

this.co2\_level = co2\_level;

}

public boolean isStatus() {

return status;

}

public void setStatus(boolean status) {

this.status = status;

}

**User.java**

package dtos;

import java.io.Serializable;

// model class for user details

public class User implements Serializable {

private String first\_name;

private String last\_name;

private String email;

private String password;

public String getFirst\_name() {

return first\_name;

}

public void setFirst\_name(String first\_name) {

this.first\_name = first\_name;

}

public String getLast\_name() {

return last\_name;

}

public void setLast\_name(String last\_name) {

this.last\_name = last\_name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**IFireAlarmService.java**

package interfaces;

import java.rmi.Remote;

import java.rmi.RemoteException;

import dtos.Auth;

import dtos.EmailSms;

import dtos.FireAlarmSensor;

import dtos.User;

import response.models.AuthResponse;

import response.models.EmailSmsResponse;

import response.models.FireAlarmSensorMultipleResponse;

import response.models.FireAlarmSensorSingleResponse;

import response.models.UserResponse;

public interface IFireAlarmService extends Remote {

public AuthResponse login(Auth auth) throws RemoteException;

public FireAlarmSensorSingleResponse createFireAlarmSensor(FireAlarmSensor alarmSensor) throws RemoteException;

public FireAlarmSensorMultipleResponse getFireAlarmSensorList() throws RemoteException;

public FireAlarmSensorSingleResponse getFireAlarmSensorById(String id) throws RemoteException;

public FireAlarmSensorSingleResponse updateFireAlarmSensor(FireAlarmSensor alarmSensor, String id) throws RemoteException;

public FireAlarmSensorSingleResponse deleteFireAlarmSensor(String id) throws RemoteException;

public UserResponse register(User user) throws RemoteException;

public EmailSmsResponse sendEmail(EmailSms email) throws RemoteException;

public EmailSmsResponse sendSms(EmailSms sms) throws RemoteException;

}

**FireAlarmServiceImpl.java**

package services;

import java.io.Serializable;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.util.List;

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 dtos.Auth;

import dtos.EmailSms;

import dtos.FireAlarmSensor;

import dtos.User;

import interfaces.IFireAlarmService;

import response.models.AuthResponse;

import response.models.EmailSmsResponse;

import response.models.UserResponse;

import response.models.FireAlarmSensorMultipleResponse;

import response.models.FireAlarmSensorSingleResponse;

public class FireAlarmServiceImpl extends UnicastRemoteObject implements IFireAlarmService, Serializable {

public Client client;

public FireAlarmServiceImpl() throws RemoteException {}

// Register New User

public UserResponse register(User user) throws RemoteException {

try {

// create new client

client = ClientBuilder.newClient();

// get the users response values from REST api

Response response = client.target("http://localhost:4000/api/users").request().post(Entity.json(user));

// set the response in RegistrationResponse

UserResponse regResponse = response.readEntity(UserResponse.class);

// return registration response

return regResponse;

} catch (Exception e) {

System.out.println(e);

} finally {

client.close(); // close newly created client

}

return null;

}

// Login

public AuthResponse login(Auth auth) throws RemoteException {

try {

// create new client

client = ClientBuilder.newClient();

// get the authentication response values from REST api

Response response = client.target("http://localhost:4000/api/auth").request().post(Entity.json(auth));

// set the response in authResponse

AuthResponse authResponse = response.readEntity(AuthResponse.class);

return authResponse;

} catch (Exception e) {

System.out.println(e);

} finally {

client.close(); // close newly created client

}

return null;

}

// Create a Fire Alarm Sensor

public FireAlarmSensorSingleResponse createFireAlarmSensor(FireAlarmSensor alarmSensor) throws RemoteException {

try {

// create new client

client = ClientBuilder.newClient();

// get the fire alarm sensors response values values from REST api

Response response = client.target("http://localhost:4000/api/fire-alarms").request().post(Entity.json(alarmSensor));

// set the response in authResponse

FireAlarmSensorSingleResponse fireAlarmSensorSingleResponse = response.readEntity(FireAlarmSensorSingleResponse.class);

return fireAlarmSensorSingleResponse;

} catch (Exception e) {

System.out.println(e);

} finally {

client.close(); // close newly created client

}

return null;

}

// Get Fire Alarm Sensors

public FireAlarmSensorMultipleResponse getFireAlarmSensorList() throws RemoteException {

try {

// create new client

client = ClientBuilder.newClient();

// set the multiple fire alarm sensors response values from REST api

FireAlarmSensorMultipleResponse response = client.target("http://localhost:4000/api/fire-alarms").request().get().readEntity(FireAlarmSensorMultipleResponse.class);

return response;

} catch (Exception e) {

System.out.println(e);

} finally {

client.close(); // close newly created client

}

return null;

}

// Get Fire Alarm Sensor by id

public FireAlarmSensorSingleResponse getFireAlarmSensorById(String id) throws RemoteException {

try {

// create new client

client = ClientBuilder.newClient();

// set the single fire alarm sensors response values from REST api

FireAlarmSensorSingleResponse response = client.target("http://localhost:4000/api/fire-alarms/" + id).request().get().readEntity(FireAlarmSensorSingleResponse.class);

return response;

} catch (Exception e) {

System.out.println(e);

} finally {

client.close(); // close newly created client

}

return null;

}

// Update Fire Alarm Sensor

public FireAlarmSensorSingleResponse updateFireAlarmSensor(FireAlarmSensor alarmSensor, String id) throws RemoteException {

try {

// create new client

client = ClientBuilder.newClient();

// get the given id fire alarm sensors response values from REST api

Response response = client.target("http://localhost:4000/api/fire-alarms/" + id).request().put(Entity.json(alarmSensor));

// set the response in fireAlarmSensorSingleResponse

FireAlarmSensorSingleResponse fireAlarmSensorSingleResponse = response.readEntity(FireAlarmSensorSingleResponse.class);

return fireAlarmSensorSingleResponse;

} catch (Exception e) {

System.out.println(e);

} finally {

client.close(); // close newly created client

}

return null;

}

// Delete Fire Alarm Sensor

public FireAlarmSensorSingleResponse deleteFireAlarmSensor(String id) throws RemoteException {

try {

// create new client

client = ClientBuilder.newClient();

// delete the given id fire alarm sensors response values from REST api

Response response = client.target("http://localhost:4000/api/fire-alarms/" + id).request().delete();

// set the response in fireAlarmSensorSingleResponse

FireAlarmSensorSingleResponse fireAlarmSensorSingleResponse = response.readEntity(FireAlarmSensorSingleResponse.class);

return fireAlarmSensorSingleResponse;

} catch (Exception e) {

System.out.println(e);

} finally {

client.close(); // close newly created client

}

return null;

}

// Send Email

public EmailSmsResponse sendEmail(EmailSms email) throws RemoteException {

try {

// create new client

client = ClientBuilder.newClient();

// get the email response values from REST api

Response response = client.target("http://localhost:4000/api/emails").request().post(Entity.json(email));

// set the response in EmailSmsResponse

EmailSmsResponse emailResponse = response.readEntity(EmailSmsResponse.class);

// return email response

return emailResponse;

} catch (Exception e) {

System.out.println(e);

} finally {

client.close(); // close newly created client

}

return null;

}

// Send SMS

public EmailSmsResponse sendSms(EmailSms sms) throws RemoteException {

try {

// create new client

client = ClientBuilder.newClient();

// get the sms response values from REST api

Response response = client.target("http://localhost:4000/api/sms").request().post(Entity.json(sms));

// set the response in EmailSmsResponse

EmailSmsResponse smsResponse = response.readEntity(EmailSmsResponse.class);

// return sms response

return smsResponse;

} catch (Exception e) {

System.out.println(e);

} finally {

client.close(); // close newly created client

}

return null;

}

}

**loginUI.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package forms;

import java.rmi.RemoteException;

import javax.swing.JOptionPane;

import dtos.Auth;

import interfaces.IFireAlarmService;

import response.models.AuthResponse;

import rmi.client.Client;

import javax.swing.GroupLayout.Alignment;

import javax.swing.GroupLayout;

import javax.swing.LayoutStyle.ComponentPlacement;

import java.awt.Font;

import javax.swing.JLabel;

/\*\*

\*

\* @author Supun Randima

\*/

public class loginUI extends javax.swing.JFrame {

private IFireAlarmService stub = null;

private Client client = new Client();

/\*\*

\* Creates new form loginUI

\*/

public loginUI() {

initComponents();

this.stub = client.getStub();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

messageBox = new javax.swing.JDialog();

jPanel2 = new javax.swing.JPanel();

jLabel4 = new javax.swing.JLabel();

jLabel6 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();

txtUserName = new javax.swing.JTextField();

btnLogin = new javax.swing.JButton();

jLabel8 = new javax.swing.JLabel();

txtPassword = new javax.swing.JPasswordField();

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

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(

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

.addGap(0, 100, Short.MAX\_VALUE)

);

jPanel1Layout.setVerticalGroup(

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

.addGap(0, 100, Short.MAX\_VALUE)

);

javax.swing.GroupLayout messageBoxLayout = new javax.swing.GroupLayout(messageBox.getContentPane());

messageBox.getContentPane().setLayout(messageBoxLayout);

messageBoxLayout.setHorizontalGroup(

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

.addGap(0, 400, Short.MAX\_VALUE)

);

messageBoxLayout.setVerticalGroup(

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

.addGap(0, 300, Short.MAX\_VALUE)

);

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

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

setForeground(java.awt.Color.darkGray);

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

jLabel4.setFont(new Font("Sitka Banner", Font.BOLD, 21)); // NOI18N

jLabel4.setForeground(new java.awt.Color(255, 255, 255));

jLabel4.setText("User Name");

jLabel6.setFont(new Font("Sitka Banner", Font.BOLD, 21)); // NOI18N

jLabel6.setForeground(new java.awt.Color(255, 255, 255));

jLabel6.setText("Password");

jLabel7.setFont(new java.awt.Font("Sitka Display", 1, 36)); // NOI18N

jLabel7.setForeground(new java.awt.Color(255, 255, 255));

jLabel7.setText("Welcome");

txtUserName.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

txtUserName.setName("txtUserName"); // NOI18N

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

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

txtUserNameActionPerformed(evt);

}

});

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

btnLogin.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

btnLogin.setText("Login");

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

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

jButton3ActionPerformed(evt);

}

});

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

jLabel8.setText("Please Login to continue");

txtPassword.setFont(new java.awt.Font("Tahoma", 0, 12));

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

jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(Alignment.TRAILING)

.addGroup(Alignment.LEADING, jPanel2Layout.createSequentialGroup()

.addGap(31)

.addGroup(jPanel2Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jLabel8, GroupLayout.PREFERRED\_SIZE, 192, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(Alignment.LEADING)

.addComponent(jLabel4, GroupLayout.PREFERRED\_SIZE, 107, GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel6, GroupLayout.PREFERRED\_SIZE, 107, GroupLayout.PREFERRED\_SIZE))

.addGap(30)

.addGroup(jPanel2Layout.createParallelGroup(Alignment.LEADING)

.addComponent(txtUserName, GroupLayout.DEFAULT\_SIZE, 213, Short.MAX\_VALUE)

.addComponent(txtPassword, GroupLayout.DEFAULT\_SIZE, 213, Short.MAX\_VALUE)

.addComponent(btnLogin, GroupLayout.PREFERRED\_SIZE, 108, GroupLayout.PREFERRED\_SIZE))

.addGap(39))

.addGroup(jPanel2Layout.createSequentialGroup()

.addComponent(jLabel7, GroupLayout.PREFERRED\_SIZE, 139, GroupLayout.PREFERRED\_SIZE)

.addContainerGap(448, Short.MAX\_VALUE))))

);

jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(50)

.addComponent(jLabel7)

.addGroup(jPanel2Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel2Layout.createSequentialGroup()

.addGap(30)

.addGroup(jPanel2Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(jLabel4, GroupLayout.PREFERRED\_SIZE, 32, GroupLayout.PREFERRED\_SIZE)

.addComponent(txtUserName, GroupLayout.PREFERRED\_SIZE, 28, GroupLayout.PREFERRED\_SIZE)))

.addGroup(jPanel2Layout.createSequentialGroup()

.addPreferredGap(ComponentPlacement.UNRELATED)

.addComponent(jLabel8, GroupLayout.PREFERRED\_SIZE, 24, GroupLayout.PREFERRED\_SIZE)))

.addGap(31)

.addGroup(jPanel2Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(txtPassword, GroupLayout.PREFERRED\_SIZE, 27, GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel6, GroupLayout.PREFERRED\_SIZE, 31, GroupLayout.PREFERRED\_SIZE))

.addGap(53)

.addComponent(btnLogin, GroupLayout.PREFERRED\_SIZE, 33, GroupLayout.PREFERRED\_SIZE)

.addContainerGap(85, Short.MAX\_VALUE))

);

jPanel2.setLayout(jPanel2Layout);

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

layout.setHorizontalGroup(

layout.createParallelGroup(Alignment.LEADING)

.addComponent(jPanel2, Alignment.TRAILING, GroupLayout.DEFAULT\_SIZE, 618, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

layout.createParallelGroup(Alignment.LEADING)

.addComponent(jPanel2, Alignment.TRAILING, GroupLayout.DEFAULT\_SIZE, 391, Short.MAX\_VALUE)

);

getContentPane().setLayout(layout);

pack();

setLocationRelativeTo(null);

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

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

// TODO add your handling code here:

}//GEN-LAST:event\_txtUserNameActionPerformed

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton3ActionPerformed

// TODO add your handling code here:

// set authentication details

Auth auth = new Auth();

auth.setUsername(txtUserName.getText().trim().toString());

auth.setPassword(new String(txtPassword.getPassword()).trim());

try {

// get the response for login success or fail

AuthResponse res = this.stub.login(auth);

String check = res.getMessage();

JOptionPane.showMessageDialog(null, check);

// if login success redirect to maiUI

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

this.setVisible(false);

String[] user = new String[1];

user[0] = auth.getUsername();

mainUI.main(user);

} else { // if login failure show error messaage

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

}

} catch (Exception e) {

System.out.println(e);

}

}//GEN-LAST:event\_jButton3ActionPerformed

/\*\*

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

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

});

}

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

private javax.swing.JButton btnLogin;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JLabel jLabel8;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JDialog messageBox;

private javax.swing.JPasswordField txtPassword;

private javax.swing.JTextField txtUserName;

}

**mainUI.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package forms;

import java.util.Calendar;

import java.util.List;

import java.util.Timer;

import java.util.TimerTask;

import javax.swing.JTable;

import javax.swing.table.DefaultTableModel;

import dtos.EmailSms;

import dtos.FireAlarmSensor;

import interfaces.IFireAlarmService;

import response.models.EmailSmsResponse;

import response.models.FireAlarmSensorMultipleResponse;

import rmi.client.Client;

import javax.swing.GroupLayout.Alignment;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.SwingUtilities;

import javax.swing.GroupLayout;

import javax.swing.JButton;

import javax.swing.JLabel;

import java.awt.Color;

import java.awt.Font;

import java.awt.event.ActionListener;

import java.rmi.RemoteException;

import java.sql.Date;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.awt.event.ActionEvent;

/\*\*

\*

\* @author Supun Randima Wijekoon

\*/

public class mainUI extends javax.swing.JFrame {

public static String user;

private IFireAlarmService stub = null;

private Client client = new Client();

private List<FireAlarmSensor> list;

private javax.swing.JTable tblAlerts = new javax.swing.JTable();

private static javax.swing.JTable tblSensors = new javax.swing.JTable();

private javax.swing.JLabel activeSensors = new javax.swing.JLabel();

/\*\*

\* Creates new form mainUI

\*/

public mainUI() {

// get RMI client

this.stub = client.getStub();

this.getAndSetData();

this.refreshData();

initComponents();

activeSensors.setForeground(Color.WHITE);

activeSensors.setFont(new Font("Sitka Heading", Font.BOLD, 14));

}

public void getAndSetData() {

// setting tables values

try {

FireAlarmSensorMultipleResponse res = this.stub.getFireAlarmSensorList();

this.list = res.getData();

// columns array for storing table columns names

String[] cols = {"Sensor", "Floor No", "Room No", "CO2 Level", "Smoke Level", "Status"};

// table model for all sensors table

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

// table model for alert sensors table

final DefaultTableModel alertsModel = new DefaultTableModel(cols, 0);

// assign all sensors for sensors tables

for (FireAlarmSensor s : this.list) {

Object[] row = {s.get\_id(), s.getFloor\_no(), s.getRoom\_no(), s.getCo2\_level(), s.getSmoke\_level(), s.isStatus()};

// add sensors for all sensors table

model.addRow(row);

// if co2 level is high add the sensor in alerts table

if((s.getCo2\_level() >= 5) || (s.getSmoke\_level() >= 5)) {

alertsModel.addRow(row);

EmailSms emailSms = new EmailSms();

emailSms.setFloor(s.getFloor\_no());

emailSms.setRoom(s.getRoom\_no());

// Send Email and SMS to Admin

this.stub.sendEmail(emailSms);

this.stub.sendSms(emailSms);

}

}

// set alerts to alert table

tblAlerts.setModel(alertsModel);

// display all sensors

tblSensors.setModel(model);

//set active sensors

activeSensors.setText(Integer.toString(this.list.size()));

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

} catch (Exception e) {

// TODO: handle exception

System.out.println(e);

}

}

// ########################## refresh form ###############################

public void refreshData() {

Timer time = new Timer();

TimerTask timerTask = new TimerTask() {

@Override

public void run() {

getAndSetData();

}

};

// refresh form in every 15 seconds

time.scheduleAtFixedRate(timerTask, 15000, 15000);

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jDialog1 = new javax.swing.JDialog();

jPanel1 = new javax.swing.JPanel();

jLabel6 = new javax.swing.JLabel();

jLabel6.setForeground(Color.WHITE);

jLabel10 = new javax.swing.JLabel();

jScrollPane1 = new javax.swing.JScrollPane();

jLabel3 = new javax.swing.JLabel();

btnAddSensor = new javax.swing.JButton();

btnEditSensor = new javax.swing.JButton();

jLabel11 = new javax.swing.JLabel();

jScrollPane2 = new javax.swing.JScrollPane();

btnlogout = new javax.swing.JButton();

lblTime = new JLabel("");

javax.swing.GroupLayout jDialog1Layout = new javax.swing.GroupLayout(jDialog1.getContentPane());

jDialog1.getContentPane().setLayout(jDialog1Layout);

jDialog1Layout.setHorizontalGroup(

jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 400, Short.MAX\_VALUE)

);

jDialog1Layout.setVerticalGroup(

jDialog1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGap(0, 300, Short.MAX\_VALUE)

);

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

jPanel1.setBackground(new java.awt.Color(153, 153, 153));

jPanel1.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N

jLabel6.setFont(new Font("Sitka Heading", Font.BOLD, 14)); // NOI18N

jLabel6.setText("Total Sensors");

jLabel10.setFont(new java.awt.Font("Sitka Display", 1, 36)); // NOI18N

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

jLabel10.setText("Fire Alarm System");

// ################### setting clock date ##############################

Timer timer1 = new Timer();

timer1.scheduleAtFixedRate(new TimerTask() {

@Override

public void run() {

// format the date time in dd:MM:yyyy HH:mm:ss

SimpleDateFormat dateFormat = new SimpleDateFormat("dd:MM:yyyy HH:mm:ss");

System.out.println(dateFormat.format(Calendar.getInstance().getTime()) );

String time = dateFormat.format(Calendar.getInstance().getTime()) ;

// set the time in lblTime

lblTime.setText(dateFormat.format(Calendar.getInstance().getTime()));

}

}, 1000, 1000); // run per second

jScrollPane1.setViewportView(tblAlerts);

jLabel3.setFont(new java.awt.Font("Sitka Small", 1, 18)); // NOI18N

jLabel3.setForeground(new java.awt.Color(255, 0, 0));

jLabel3.setText("Alerts");

btnAddSensor.setBackground(new java.awt.Color(153, 0, 204));

btnAddSensor.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N

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

btnAddSensor.setText("Add Fire Alarm ");

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

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

btnAddSensorActionPerformed(evt);

}

});

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

btnEditSensor.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N

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

btnEditSensor.setText("Edit Fire Alarm ");

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

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

btnEditSensorActionPerformed(evt);

}

});

jLabel11.setFont(new java.awt.Font("Sitka Small", 1, 18)); // NOI18N

jLabel11.setForeground(new java.awt.Color(0, 255, 0));

jLabel11.setText("Sensors");

jScrollPane2.setViewportView(tblSensors);

btnAddNewUser = new JButton();

btnAddNewUser.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// show registrationUI

new registrationUI().setVisible(true);

}

});

btnAddNewUser.setText("Add New User");

btnAddNewUser.setForeground(Color.WHITE);

btnAddNewUser.setFont(new Font("Tahoma", Font.BOLD, 11));

btnAddNewUser.setBackground(new Color(153, 0, 204));

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

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

btnlogoutActionPerformed(evt);

}

});

btnlogout.setText("Log out");

btnlogout.setForeground(Color.WHITE);

btnlogout.setFont(new Font("Tahoma", Font.BOLD, 11));

btnlogout.setBackground(Color.BLACK);

lblTime.setForeground(Color.WHITE);

lblTime.setFont(new Font("Tahoma", Font.BOLD, 12));

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

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(72)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jScrollPane2, GroupLayout.PREFERRED\_SIZE, 828, GroupLayout.PREFERRED\_SIZE)

.addContainerGap())

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jLabel11)

.addContainerGap())

.addGroup(jPanel1Layout.createParallelGroup(Alignment.TRAILING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(lblTime, GroupLayout.PREFERRED\_SIZE, 253, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.RELATED, 465, Short.MAX\_VALUE)

.addComponent(btnlogout, GroupLayout.PREFERRED\_SIZE, 99, GroupLayout.PREFERRED\_SIZE)

.addGap(38))

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jLabel6)

.addGap(32)

.addComponent(activeSensors)

.addGap(730))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.TRAILING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(btnAddNewUser, GroupLayout.PREFERRED\_SIZE, 140, GroupLayout.PREFERRED\_SIZE)

.addGap(115)

.addComponent(jLabel10)

.addPreferredGap(ComponentPlacement.RELATED, 141, Short.MAX\_VALUE)

.addComponent(btnAddSensor, GroupLayout.PREFERRED\_SIZE, 140, GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addPreferredGap(ComponentPlacement.RELATED, 678, Short.MAX\_VALUE)

.addComponent(btnEditSensor, GroupLayout.PREFERRED\_SIZE, 140, GroupLayout.PREFERRED\_SIZE)))

.addGap(37))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addComponent(jLabel3)

.addComponent(jScrollPane1, GroupLayout.PREFERRED\_SIZE, 828, GroupLayout.PREFERRED\_SIZE))

.addContainerGap(27, Short.MAX\_VALUE))))))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(122)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(jLabel6)

.addComponent(activeSensors))

.addGap(18)

.addComponent(jLabel3))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(32)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(btnAddSensor, GroupLayout.PREFERRED\_SIZE, 34, GroupLayout.PREFERRED\_SIZE)

.addComponent(btnAddNewUser, GroupLayout.PREFERRED\_SIZE, 34, GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel10))

.addGap(9)

.addComponent(btnEditSensor, GroupLayout.PREFERRED\_SIZE, 34, GroupLayout.PREFERRED\_SIZE)))

.addGap(18)

.addComponent(jScrollPane1, GroupLayout.PREFERRED\_SIZE, 133, GroupLayout.PREFERRED\_SIZE)

.addGap(50)

.addComponent(jLabel11)

.addGap(18)

.addComponent(jScrollPane2, GroupLayout.PREFERRED\_SIZE, 133, GroupLayout.PREFERRED\_SIZE)

.addGap(11)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(btnlogout, GroupLayout.PREFERRED\_SIZE, 34, GroupLayout.PREFERRED\_SIZE)

.addComponent(lblTime))

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

);

jPanel1.setLayout(jPanel1Layout);

if(user.equals("admin@gmail.com")) btnAddSensor.setVisible(true);

else btnAddSensor.setVisible(false);

if(user.equals("admin@gmail.com")) btnEditSensor.setVisible(true);

else btnEditSensor.setVisible(false);

if(user.equals("admin@gmail.com")) btnAddNewUser.setVisible(true);

else btnAddNewUser.setVisible(false);

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

layout.setHorizontalGroup(

layout.createParallelGroup(Alignment.LEADING)

.addComponent(jPanel1, GroupLayout.DEFAULT\_SIZE, 927, Short.MAX\_VALUE)

);

layout.setVerticalGroup(

layout.createParallelGroup(Alignment.LEADING)

.addComponent(jPanel1, GroupLayout.DEFAULT\_SIZE, 609, Short.MAX\_VALUE)

);

getContentPane().setLayout(layout);

pack();

setLocationRelativeTo(null);

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

protected void btnlogoutActionPerformed(ActionEvent evt) {

// TODO Auto-generated method stub

// dispose the mainUI and show loginUI again

this.dispose();

new loginUI().setVisible(true);

}

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

// TODO add your handling code here:

// show editSensorUI while hiding mainUI

this.dispose();

new editSensorUI().setVisible(true);

}//GEN-LAST:event\_btnEditSensorActionPerformed

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

// TODO add your handling code here:

// show newSensorUI while hiding mainUI

this.dispose();

new newSensorUI().setVisible(true);

}//GEN-LAST:event\_btnAddSensorActionPerformed

/\*\*

\* @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

\*/

// user = args[0];

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

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

//</editor-fold>

//user = args[0];

// create a table model and set a Column Identifiers to this model

Object[] columns = {"Sensor","Floor","Room No","CO2 Level", "Smoke Level", "Temperature"};

DefaultTableModel model = new DefaultTableModel();

model.setColumnIdentifiers(columns);

JTable jTable = new JTable();

jTable.setModel(model);

user = args[0];

/\* Create and display the form \*/

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

public void run() {

new mainUI().setVisible(true);

}

});

}

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

private javax.swing.JButton btnAddSensor;

private javax.swing.JButton btnEditSensor;

private javax.swing.JDialog jDialog1;

private javax.swing.JLabel jLabel10;

private javax.swing.JLabel jLabel11;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel6;

private javax.swing.JPanel jPanel1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private JButton btnAddNewUser;

private JButton btnlogout;

private JLabel lblTime;

}

**registrationUI.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package forms;

import javax.swing.JOptionPane;

import javax.swing.GroupLayout.Alignment;

import javax.swing.GroupLayout;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.JLabel;

import java.awt.Font;

import java.rmi.RemoteException;

import java.awt.Color;

import javax.swing.JTextField;

import dtos.Auth;

import dtos.User;

import interfaces.IFireAlarmService;

import response.models.UserResponse;

import rmi.client.Client;

/\*\*

\*

\* @author Supun Randima Wijekoon

\*/

public class registrationUI extends javax.swing.JFrame {

private IFireAlarmService stub = null;

private Client client = new Client();

/\*\*

\* Creates new form registrationUI

\*/

public registrationUI() {

// get RMI client

this.stub = client.getStub();

initComponents();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jPanel1 = new javax.swing.JPanel();

jLabel7 = new javax.swing.JLabel();

lblFirstNAme = new javax.swing.JLabel();

jLabel6 = new javax.swing.JLabel();

jLabel8 = new javax.swing.JLabel();

jLabel9 = new javax.swing.JLabel();

txtFirstName = new javax.swing.JTextField();

txtEmail = new javax.swing.JTextField();

txtPassword1 = new javax.swing.JPasswordField();

txtPassword2 = new javax.swing.JPasswordField();

btnRegister = new javax.swing.JButton();

btnCancel = new javax.swing.JButton();

jLabel1.setText("jLabel1");

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

jPanel1.setBackground(new java.awt.Color(153, 153, 153));

jPanel1.setPreferredSize(new java.awt.Dimension(500, 400));

jLabel7.setFont(new java.awt.Font("Sitka Display", 1, 28)); // NOI18N

jLabel7.setForeground(new java.awt.Color(255, 255, 255));

jLabel7.setText("Registration");

lblFirstNAme.setFont(new java.awt.Font("Segoe UI Semilight", 1, 14)); // NOI18N

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

lblFirstNAme.setText("First Name");

jLabel6.setFont(new java.awt.Font("Segoe UI Semilight", 1, 14)); // NOI18N

jLabel6.setForeground(new java.awt.Color(255, 255, 255));

jLabel6.setText("Email Address");

jLabel8.setFont(new java.awt.Font("Segoe UI Semilight", 1, 14)); // NOI18N

jLabel8.setForeground(new java.awt.Color(255, 255, 255));

jLabel8.setText("Password");

jLabel9.setFont(new java.awt.Font("Segoe UI Semilight", 1, 14)); // NOI18N

jLabel9.setForeground(new java.awt.Color(255, 255, 255));

jLabel9.setText("Repeat Password");

txtFirstName.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

txtFirstName.setName("txtFirstName"); // NOI18N

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

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

txtUserNameActionPerformed(evt);

}

});

txtEmail.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

txtEmail.setName("txtUserName"); // NOI18N

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

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

txtEmailActionPerformed(evt);

}

});

txtPassword1.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

txtPassword2.setFont(new java.awt.Font("Tahoma", 0, 12)); // NOI18N

txtPassword2.addActionListener(new java.awt.event.ActionListener() {

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

txtPassword2ActionPerformed(evt);

}

});

txtPassword2.addKeyListener(new java.awt.event.KeyAdapter() {

public void keyReleased(java.awt.event.KeyEvent evt) {

txtPassword2KeyReleased(evt);

}

});

btnRegister.setBackground(new java.awt.Color(0, 0, 153));

btnRegister.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

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

btnRegister.setText("Register");

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

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

btnRegisterActionPerformed(evt);

}

});

btnCancel.setBackground(new java.awt.Color(0, 0, 0));

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

btnCancel.setText("Cancel");

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

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

btnCancelActionPerformed(evt);

}

});

JLabel lblLastName = new JLabel();

lblLastName.setText("Last Name");

lblLastName.setForeground(Color.WHITE);

lblLastName.setFont(new Font("Segoe UI Semilight", Font.BOLD, 14));

txtLastName = new JTextField();

txtLastName.setName("txtLastName");

txtLastName.setFont(new Font("Tahoma", Font.PLAIN, 12));

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

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(Alignment.TRAILING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(86, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(btnRegister, GroupLayout.PREFERRED\_SIZE, 108, GroupLayout.PREFERRED\_SIZE)

.addGap(42)

.addComponent(btnCancel, GroupLayout.PREFERRED\_SIZE, 113, GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(lblLastName, GroupLayout.PREFERRED\_SIZE, 107, GroupLayout.PREFERRED\_SIZE)

.addGap(40)

.addComponent(txtLastName, GroupLayout.PREFERRED\_SIZE, 198, GroupLayout.PREFERRED\_SIZE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(lblFirstNAme, GroupLayout.PREFERRED\_SIZE, 107, GroupLayout.PREFERRED\_SIZE)

.addGap(40)

.addComponent(txtFirstName, 198, 198, 198))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addComponent(jLabel9)

.addComponent(jLabel8, GroupLayout.PREFERRED\_SIZE, 107, GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel6, GroupLayout.PREFERRED\_SIZE, 107, GroupLayout.PREFERRED\_SIZE))

.addGap(29)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addComponent(txtPassword1, 198, 198, 198)

.addComponent(txtPassword2, 198, 198, 198)

.addComponent(txtEmail, 198, 198, 198))))

.addGap(69))

.addGroup(Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addComponent(jLabel7, GroupLayout.PREFERRED\_SIZE, 158, GroupLayout.PREFERRED\_SIZE)

.addGap(160))))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(25)

.addComponent(jLabel7)

.addGap(18)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(lblFirstNAme, GroupLayout.PREFERRED\_SIZE, 32, GroupLayout.PREFERRED\_SIZE)

.addComponent(txtFirstName, GroupLayout.PREFERRED\_SIZE, 28, GroupLayout.PREFERRED\_SIZE))

.addGap(23)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addComponent(lblLastName, GroupLayout.PREFERRED\_SIZE, 32, GroupLayout.PREFERRED\_SIZE)

.addComponent(txtLastName, GroupLayout.PREFERRED\_SIZE, 28, GroupLayout.PREFERRED\_SIZE))

.addGap(18)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(jLabel6, GroupLayout.PREFERRED\_SIZE, 32, GroupLayout.PREFERRED\_SIZE)

.addComponent(txtEmail, GroupLayout.PREFERRED\_SIZE, 28, GroupLayout.PREFERRED\_SIZE))

.addGap(18)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(txtPassword1, GroupLayout.PREFERRED\_SIZE, 30, GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel8, GroupLayout.PREFERRED\_SIZE, 32, GroupLayout.PREFERRED\_SIZE))

.addGap(13)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(txtPassword2, GroupLayout.PREFERRED\_SIZE, 30, GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel9, GroupLayout.PREFERRED\_SIZE, 32, GroupLayout.PREFERRED\_SIZE))

.addGap(18)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(btnCancel, GroupLayout.PREFERRED\_SIZE, 35, GroupLayout.PREFERRED\_SIZE)

.addComponent(btnRegister, GroupLayout.PREFERRED\_SIZE, 33, GroupLayout.PREFERRED\_SIZE))

.addGap(43))

);

jPanel1.setLayout(jPanel1Layout);

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.Alignment.TRAILING, 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, 373, Short.MAX\_VALUE)

);

pack();

setLocationRelativeTo(null);

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

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

// TODO add your handling code here:

}//GEN-LAST:event\_txtUserNameActionPerformed

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

// TODO add your handling code here:

}//GEN-LAST:event\_txtEmailActionPerformed

private void txtPassword2ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_txtPassword2ActionPerformed

// TODO add your handling code here:

}//GEN-LAST:event\_txtPassword2ActionPerformed

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

// check input is empty show a message

if(!isEmpty("First Name", txtFirstName.getText())){ }

if(!isEmpty("Last Name", txtLastName.getText())){ }

if(!isEmpty("Email Address", txtEmail.getText())){}

if(!isEmpty("Password", new String(txtPassword1.getPassword()))) {}

if(!isEmpty("Password", new String(txtPassword2.getPassword())) ){}

else if ( checkPassword() ) // if passwords are matched

{

// create registration model and set values

User reg = new User();

reg.setFirst\_name( txtFirstName.getText());

reg.setLast\_name(txtLastName.getText());

reg.setEmail(txtEmail.getText());

reg.setPassword(new String(txtPassword2.getPassword()));

try {

// reg the registration model in RMI server

UserResponse regRes = this.stub.register(reg);

String response = regRes.getStatus();

JOptionPane.showMessageDialog(null,response);

if(response.equals("success")) {

JOptionPane.showMessageDialog(null, "Registered Successfully !");

this.dispose();

}else {

JOptionPane.showMessageDialog(null, "Registered Failed !");

}

}catch(Exception e) {

System.out.println(e.getMessage());

}

}

}//GEN-LAST:event\_btnRegisterActionPerformed

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

// TODO add your handling code here:

// close this ui

this.dispose();

}//GEN-LAST:event\_btnCancelActionPerformed

private void txtPassword2KeyReleased(java.awt.event.KeyEvent evt) {//GEN-FIRST:event\_txtPassword2KeyReleased

// TODO add your handling code here:

}//GEN-LAST:event\_txtPassword2KeyReleased

// check passwords are matching or not

private boolean checkPassword(){

// get textpassword values

String password1 = new String(txtPassword1.getPassword()).trim();

String password2 = new String(txtPassword2.getPassword()).trim();

// when password are matching return true

if(password1.equals(password2)){

return true;

}

// when password are not matching return false

JOptionPane.showMessageDialog(null, "Password are not matching !");

return false;

}

/\*\*

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

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

});

}

public boolean isEmpty(String name, String value){

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

JOptionPane.showMessageDialog(null, name + " Required");

return false;

}

else return true;

}

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

private javax.swing.JButton btnCancel;

private javax.swing.JButton btnRegister;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel lblFirstNAme;

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.JTextField txtEmail;

private javax.swing.JPasswordField txtPassword1;

private javax.swing.JPasswordField txtPassword2;

private javax.swing.JTextField txtFirstName;

private JTextField txtLastName;

}

**newSensorUI.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package forms;

import javax.swing.GroupLayout.Alignment;

import javax.swing.GroupLayout;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import java.awt.Font;

import java.awt.Color;

import javax.swing.JRadioButton;

import javax.swing.LayoutStyle.ComponentPlacement;

import dtos.FireAlarmSensor;

import interfaces.IFireAlarmService;

import response.models.FireAlarmSensorSingleResponse;

import rmi.client.Client;

import java.awt.event.ActionListener;

import java.rmi.RemoteException;

import java.util.List;

import java.awt.event.ActionEvent;

/\*\*

\*

\* @author Supun Randima Wijekoon

\*/

public class newSensorUI extends javax.swing.JFrame {

/\*\*

\* Creates new form newSensorUI

\*/

private IFireAlarmService stub = null;

private Client client = new Client();

private FireAlarmSensor fireAlarmSensor = new FireAlarmSensor();

public newSensorUI() {

// get RMI client

this.stub = client.getStub();

initComponents();

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

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jLabel10 = new javax.swing.JLabel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

txtRoomNo = new javax.swing.JTextField();

txtRoomNo.setToolTipText("Enter room no");

txtFloorNo = new javax.swing.JTextField();

txtFloorNo.setToolTipText("Enter floor no");

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

rdbtnInactive = new JRadioButton("Inactive");

rdbtnInactive.setFont(new Font("Tahoma", Font.PLAIN, 14));

rdbtnInactive.setToolTipText("Sensor is in inactive status");

rdbtnInactive.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// create radio button connection with other

// if rdbtnInactive selected make the rdbtnActive unselected

if(rdbtnInactive.isSelected()) {

rdbtnActive.setSelected(false);

}

}

});

rdbtnActive = new JRadioButton("Active");

rdbtnActive.setFont(new Font("Tahoma", Font.PLAIN, 14));

rdbtnActive.setToolTipText("Sensor is in active status");

rdbtnActive.setSelected(true);

jButton2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String floorNo = txtFloorNo.getText().toString();

String roomNo = txtRoomNo.getText().toString();

boolean status;

// check status button is clicked or not

if(rdbtnActive.isSelected()) {

// if it is clicked set status true

status = true;

}else {

// if it is not clicked set status false

status = false;

}

// check floor no is not empty if it is empty show a message

if(!isEmpty("Floor No", floorNo.trim())){ }

// check room no is not empty if it is empty show a message

if(!isEmpty("Room No", roomNo.trim())) {}

else {

try {

// get floor number and room number

int fNo = Integer.parseInt(floorNo.trim());

int rNo = Integer.parseInt(roomNo.trim());

// set fire alarm sensor details

fireAlarmSensor.setFloor\_no(fNo);

fireAlarmSensor.setRoom\_no(rNo);

fireAlarmSensor.setStatus(true);

// initalized default values in co2 and smoke level

fireAlarmSensor.setCo2\_level(1);

fireAlarmSensor.setSmoke\_level(1);

try {

// create new fire alarm sensor

FireAlarmSensorSingleResponse response = stub.createFireAlarmSensor(fireAlarmSensor);

// show the message successfully created or not

JOptionPane.showMessageDialog(null, response.getStatus());

} catch (RemoteException ex) {

// TODO Auto-generated catch block

ex.printStackTrace();

}

}catch (Exception exc) {

System.out.println(exc);

}

}

}

});

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

jPanel1.setBackground(new java.awt.Color(153, 153, 153));

jPanel1.setPreferredSize(new java.awt.Dimension(948, 602));

jLabel10.setFont(new java.awt.Font("Sitka Display", 1, 24)); // NOI18N

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

jLabel10.setText("Register Fire Alarm");

jLabel1.setFont(new java.awt.Font("Sylfaen", 0, 18)); // NOI18N

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

jLabel1.setText("Floor Number");

jLabel2.setFont(new java.awt.Font("Sylfaen", 0, 18)); // NOI18N

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

jLabel2.setText("Room Number");

txtRoomNo.setBackground(new java.awt.Color(204, 204, 204));

txtRoomNo.setFont(new java.awt.Font("Sylfaen", 0, 18)); // NOI18N

txtFloorNo.setBackground(new java.awt.Color(204, 204, 204));

txtFloorNo.setFont(new java.awt.Font("Sylfaen", 0, 18)); // NOI18N

jButton1.setBackground(new java.awt.Color(0, 0, 0));

jButton1.setForeground(new java.awt.Color(255, 255, 255));

jButton1.setText("Home");

jButton1.addActionListener(new java.awt.event.ActionListener() {

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

jButton1ActionPerformed(evt);

}

});

jButton2.setBackground(new java.awt.Color(0, 0, 0));

jButton2.setForeground(new java.awt.Color(255, 255, 255));

jButton2.setText("Save");

label = new JLabel();

label.setText("Status");

label.setForeground(Color.WHITE);

label.setFont(new Font("Sylfaen", Font.PLAIN, 18));

rdbtnActive.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

// create radio button connection with other

// if rdbtnActive selected make the rdbtnInactive unselected

if(rdbtnActive.isSelected()) {

rdbtnInactive.setSelected(false);

}

}

});

rdbtnActive.setBackground(Color.LIGHT\_GRAY);

rdbtnInactive.setBackground(Color.LIGHT\_GRAY);

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

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addContainerGap(65, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addComponent(jLabel2)

.addComponent(jLabel1)

.addComponent(label, GroupLayout.PREFERRED\_SIZE, 109, GroupLayout.PREFERRED\_SIZE))

.addGap(44))

.addGroup(Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addComponent(jButton2, GroupLayout.PREFERRED\_SIZE, 113, GroupLayout.PREFERRED\_SIZE)

.addGap(5)))

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING, false)

.addComponent(txtFloorNo, GroupLayout.PREFERRED\_SIZE, 228, GroupLayout.PREFERRED\_SIZE)

.addComponent(txtRoomNo, GroupLayout.PREFERRED\_SIZE, 228, GroupLayout.PREFERRED\_SIZE)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(rdbtnActive, GroupLayout.PREFERRED\_SIZE, 77, GroupLayout.PREFERRED\_SIZE)

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

.addComponent(rdbtnInactive, GroupLayout.PREFERRED\_SIZE, 81, GroupLayout.PREFERRED\_SIZE)

.addGap(34)))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(54)

.addComponent(jButton1, GroupLayout.PREFERRED\_SIZE, 113, GroupLayout.PREFERRED\_SIZE)))

.addContainerGap(48, Short.MAX\_VALUE))

.addGroup(Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addComponent(jLabel10)

.addGap(149))))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(26)

.addComponent(jLabel10)

.addGap(38)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(jLabel1)

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

.addGap(32)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(jLabel2)

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

.addGap(27)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(label, GroupLayout.PREFERRED\_SIZE, 24, GroupLayout.PREFERRED\_SIZE)

.addComponent(rdbtnActive)

.addComponent(rdbtnInactive))

.addGap(59)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(jButton1, GroupLayout.PREFERRED\_SIZE, 35, GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton2, GroupLayout.PREFERRED\_SIZE, 35, GroupLayout.PREFERRED\_SIZE))

.addGap(70))

);

jPanel1.setLayout(jPanel1Layout);

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

layout.setHorizontalGroup(

layout.createParallelGroup(Alignment.TRAILING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jPanel1, GroupLayout.DEFAULT\_SIZE, 510, Short.MAX\_VALUE)

.addContainerGap())

);

layout.setVerticalGroup(

layout.createParallelGroup(Alignment.TRAILING)

.addGroup(Alignment.LEADING, layout.createSequentialGroup()

.addContainerGap()

.addComponent(jPanel1, GroupLayout.DEFAULT\_SIZE, 413, Short.MAX\_VALUE)

.addContainerGap())

);

getContentPane().setLayout(layout);

pack();

setLocationRelativeTo(null);

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

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton1ActionPerformed

// TODO add your handling code here:

// redirect to homeUI and dispose this

this.dispose();

new mainUI().setVisible(true);

}//GEN-LAST:event\_jButton1ActionPerformed

/\*\*

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

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

});

}

// check empty fields in inputs

public boolean isEmpty(String name, String value){

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

// if there is empty field show an error message including field name

JOptionPane.showMessageDialog(null, name + " Required");

return false;

}

else return true; // if no empty field detected return true

}

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

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel10;

private javax.swing.JLabel jLabel2;

private javax.swing.JPanel jPanel1;

private javax.swing.JTextField txtRoomNo;

private javax.swing.JTextField txtFloorNo;

private JLabel label;

private JRadioButton rdbtnActive;

private JRadioButton rdbtnInactive;

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

}

**editSensorUI.java**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package forms;

import java.util.List;

import java.util.Vector;

import javax.swing.JComboBox;

import dtos.FireAlarmSensor;

import interfaces.IFireAlarmService;

import response.models.FireAlarmSensorMultipleResponse;

import response.models.FireAlarmSensorSingleResponse;

import rmi.client.Client;

import java.awt.event.ActionListener;

import java.rmi.RemoteException;

import java.awt.event.ActionEvent;

import javax.swing.GroupLayout.Alignment;

import javax.swing.GroupLayout;

import javax.swing.LayoutStyle.ComponentPlacement;

import javax.swing.JTextField;

import java.awt.Color;

import java.awt.Font;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JRadioButton;

import javax.swing.JButton;

import java.awt.SystemColor;

/\*\*

\*

\* @author Supun Randima Wijekoon

\*/

public class editSensorUI extends javax.swing.JFrame {

private IFireAlarmService stub = null;

private Client client = new Client();

private List<FireAlarmSensor> list;

private FireAlarmSensor fireAlarmSensor;

/\*\*

\* Creates new form editSensorUI

\*/

public editSensorUI() {

// get RMI client

this.stub = client.getStub();

// get alarm sensor details from RMI

this.getData();

initComponents();

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

}

public void getData() {

try {

// get alarm sensor details from RMI

FireAlarmSensorMultipleResponse res = this.stub.getFireAlarmSensorList();

// set alarm list

this.list = res.getData();

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

} catch (Exception e) {

// TODO: handle exception

}

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

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

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jLabel10 = new javax.swing.JLabel();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

txtRoomNo = new javax.swing.JTextField();

txtFloorNo = new javax.swing.JTextField();

btnHome = new javax.swing.JButton();

btnUpdate = new javax.swing.JButton();

rdbtnActive = new JRadioButton("Active");

rdbtnActive.setFont(new Font("Tahoma", Font.PLAIN, 14));

rdbtnActive.setToolTipText("Sensor is in active status");

rdbtnInactive = new JRadioButton("Inactive");

rdbtnInactive.setFont(new Font("Tahoma", Font.PLAIN, 14));

rdbtnInactive.setToolTipText("Sensor is in inactive status");

btnUpdate.addActionListener(new ActionListener() {

@SuppressWarnings("null")

public void actionPerformed(ActionEvent arg0) {

String floorNo = txtFloorNo.getText().toString();

String roomNo = txtRoomNo.getText().toString();

Boolean status;

// check status button is clicked or not

if(rdbtnActive.isSelected()) {

status = true;

}else {

status = false;

}

// Database id

String uniqueId = fireAlarmSensor.get\_id();

System.out.println(uniqueId + " " + status +" "+ Integer.parseInt(floorNo));

// get the roomNo and floorNo from inputs

int fNo = Integer.parseInt(floorNo.trim());

int rNo = Integer.parseInt(roomNo.trim());

// set the values in fireAlarmSensor model

fireAlarmSensor.setFloor\_no(fNo);

fireAlarmSensor.setRoom\_no(rNo);

fireAlarmSensor.setStatus(status);

try {

// update the new fireAlarmSensor details using RMI server

FireAlarmSensorSingleResponse response = stub.updateFireAlarmSensor(fireAlarmSensor, uniqueId);

// show the response and show success message

System.out.println(response);

JOptionPane.showMessageDialog(null, "Succesfully Updated");

} catch (RemoteException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

});

jLabel3 = new javax.swing.JLabel();

final JComboBox listSensors = new JComboBox();

listSensors.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent arg0) {

JComboBox c = (JComboBox) arg0.getSource();

Item item = (Item) c.getSelectedItem();

// assign selected item

// get values from all sensors list

fireAlarmSensor = list.get(item.getId());

// assign values to text field

txtFloorNo.setText(Integer.toString(fireAlarmSensor.getFloor\_no()));

txtRoomNo.setText(Integer.toString(fireAlarmSensor.getRoom\_no()));

String status = "Not Active";

System.out.println(fireAlarmSensor.isStatus() == true);

// set the connection between btnActive and btnInactive

if(fireAlarmSensor.isStatus() == true) { //if status is true make the rdbtnActive selected

rdbtnInactive.setSelected(false);

rdbtnActive.setSelected(true);

}

else { //if status is false make the rdbtninActive selected

rdbtnActive.setSelected(false);

rdbtnInactive.setSelected(true);

}

System.out.println(item.getId() + " : " + item.getDescription());

}

});

int index = 0;

Vector model = new Vector();

for (FireAlarmSensor s : this.list) {

// add sensors values in lists sensors

listSensors.addItem(new Item( index++ , "Floor No: " + s.getFloor\_no() + ", Room No: " + s.getRoom\_no() ));

}

btnDelete = new javax.swing.JButton();

btnDelete.addActionListener(new ActionListener() {

// delete a sensor

public void actionPerformed(ActionEvent e) {

// get selected item details

String uniqueId = fireAlarmSensor.get\_id();

// show message request from user to delete

int result = JOptionPane.showConfirmDialog( null, "Do you want to delete ?",

"alert", JOptionPane.OK\_CANCEL\_OPTION);

System.out.println(result);

if(result == 0) { // if user accepted to delete

try {

// delete selected sensor from list by using RMI server

FireAlarmSensorSingleResponse response = stub.deleteFireAlarmSensor(uniqueId);

JOptionPane.showMessageDialog(null, response.getStatus());

// update sensors list

List<FireAlarmSensor> list;

Client client = new Client();

IFireAlarmService stub = null;

stub = client.getStub();

FireAlarmSensorMultipleResponse res;

try {

Vector model = new Vector();

// remove all items in current list

int itemCount = listSensors.getItemCount();

System.out.println("size" +itemCount);

int i =0;

while(i <= itemCount) {

System.out.println("Item " + i);

int x = i+1;

System.out.println(listSensors.getItemAt(i));

listSensors.removeItemAt(i);

}

// get new sensors details after deleteing

res = stub.getFireAlarmSensorList();

list = res.getData();

System.out.println(list.size());

int index = 0;

for (FireAlarmSensor s : list) {

// add newly updated sensors list values

System.out.println(index);

listSensors.addItem(new Item( index++ , "Floor No: " + s.getFloor\_no() + ", Room No: " + s.getRoom\_no() ));

}

}catch (Exception e2) {

// TODO: handle exception

System.out.println(e2.getMessage());

}

} catch (RemoteException ex) {

// TODO Auto-generated catch block

ex.printStackTrace();

}

}

}

});

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

jPanel1.setBackground(new java.awt.Color(153, 153, 153));

jPanel1.setPreferredSize(new java.awt.Dimension(948, 602));

jLabel10.setFont(new java.awt.Font("Sitka Display", 1, 24)); // NOI18N

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

jLabel10.setText("Edit Fire Alarm");

jLabel1.setFont(new java.awt.Font("Sylfaen", 0, 18)); // NOI18N

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

jLabel1.setText("Floor Number");

jLabel2.setFont(new java.awt.Font("Sylfaen", 0, 18)); // NOI18N

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

jLabel2.setText("Room Number");

txtRoomNo.setBackground(new java.awt.Color(204, 204, 204));

txtRoomNo.setFont(new java.awt.Font("Sylfaen", 0, 18)); // NOI18N

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

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

jTextField1ActionPerformed(evt);

}

});

txtFloorNo.setBackground(new java.awt.Color(204, 204, 204));

txtFloorNo.setFont(new java.awt.Font("Sylfaen", 0, 18)); // NOI18N

btnHome.setBackground(new java.awt.Color(0, 0, 0));

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

btnHome.setText("Home");

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

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

jButton1ActionPerformed(evt);

}

});

btnUpdate.setBackground(new java.awt.Color(0, 0, 0));

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

btnUpdate.setText("Update");

jLabel3.setFont(new java.awt.Font("Sylfaen", 0, 18)); // NOI18N

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

jLabel3.setText("Sensor");

listSensors.setFont(new java.awt.Font("Sylfaen", 0, 18)); // NOI18N

btnDelete.setBackground(new java.awt.Color(255, 0, 0));

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

btnDelete.setText("Delete");

lblStatus = new JLabel();

lblStatus.setText("Status");

lblStatus.setForeground(Color.WHITE);

lblStatus.setFont(new Font("Sylfaen", Font.PLAIN, 18));

rdbtnActive.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(rdbtnActive.isSelected()) {

rdbtnInactive.setSelected(false);

}

}

});

rdbtnActive.setBackground(Color.LIGHT\_GRAY);

rdbtnInactive.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if(rdbtnInactive.isSelected()) { // if rdbtnInctive selected make the rdbtnActive unselected

rdbtnActive.setSelected(false);

}

}

});

rdbtnInactive.setBackground(Color.LIGHT\_GRAY);

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

jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(Alignment.TRAILING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(161)

.addComponent(jLabel10)

.addContainerGap(165, Short.MAX\_VALUE))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(54)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(btnUpdate, GroupLayout.PREFERRED\_SIZE, 113, GroupLayout.PREFERRED\_SIZE)

.addGap(30)

.addComponent(btnDelete, GroupLayout.PREFERRED\_SIZE, 113, GroupLayout.PREFERRED\_SIZE)

.addGap(18)

.addComponent(btnHome, GroupLayout.PREFERRED\_SIZE, 113, GroupLayout.PREFERRED\_SIZE)

.addGap(43))

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addComponent(jLabel2)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addComponent(jLabel3)

.addComponent(jLabel1)

.addComponent(lblStatus, GroupLayout.PREFERRED\_SIZE, 109, GroupLayout.PREFERRED\_SIZE))

.addGap(44)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addComponent(listSensors, 0, 228, Short.MAX\_VALUE)

.addComponent(txtFloorNo, GroupLayout.DEFAULT\_SIZE, 228, Short.MAX\_VALUE)

.addComponent(txtRoomNo, GroupLayout.DEFAULT\_SIZE, 228, Short.MAX\_VALUE)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(rdbtnActive, GroupLayout.PREFERRED\_SIZE, 75, GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(ComponentPlacement.RELATED, 35, Short.MAX\_VALUE)

.addComponent(rdbtnInactive, GroupLayout.PREFERRED\_SIZE, 78, GroupLayout.PREFERRED\_SIZE)

.addGap(40)))))

.addContainerGap(49, Short.MAX\_VALUE))))

);

jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.LEADING)

.addGroup(jPanel1Layout.createSequentialGroup()

.addGap(58)

.addComponent(jLabel10)

.addGap(71))

.addGroup(Alignment.TRAILING, jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(jLabel3, GroupLayout.PREFERRED\_SIZE, 31, GroupLayout.PREFERRED\_SIZE)

.addComponent(listSensors, GroupLayout.PREFERRED\_SIZE, GroupLayout.DEFAULT\_SIZE, GroupLayout.PREFERRED\_SIZE))

.addGap(18)))

.addGap(18)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(lblStatus, GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(rdbtnActive)

.addComponent(rdbtnInactive))

.addGap(32)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

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

.addComponent(jLabel1))

.addGap(24)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(jLabel2)

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

.addGap(50)

.addGroup(jPanel1Layout.createParallelGroup(Alignment.BASELINE)

.addComponent(btnUpdate, GroupLayout.PREFERRED\_SIZE, 35, GroupLayout.PREFERRED\_SIZE)

.addComponent(btnDelete, GroupLayout.PREFERRED\_SIZE, 35, GroupLayout.PREFERRED\_SIZE)

.addComponent(btnHome, GroupLayout.PREFERRED\_SIZE, 35, GroupLayout.PREFERRED\_SIZE))

.addGap(47))

);

jPanel1.setLayout(jPanel1Layout);

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

layout.setHorizontalGroup(

layout.createParallelGroup(Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jPanel1, GroupLayout.DEFAULT\_SIZE, 484, Short.MAX\_VALUE)

.addContainerGap())

);

layout.setVerticalGroup(

layout.createParallelGroup(Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jPanel1, GroupLayout.PREFERRED\_SIZE, 449, Short.MAX\_VALUE)

.addContainerGap())

);

getContentPane().setLayout(layout);

pack();

setLocationRelativeTo(null);

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

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jButton1ActionPerformed

// TODO add your handling code here:

this.dispose();

new mainUI().setVisible(true);

}//GEN-LAST:event\_jButton1ActionPerformed

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {//GEN-FIRST:event\_jTextField1ActionPerformed

// TODO add your handling code here:

}//GEN-LAST:event\_jTextField1ActionPerformed

/\*\*

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

} catch (InstantiationException ex) {

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

} catch (IllegalAccessException ex) {

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

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

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

}

});

}

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

private javax.swing.JButton btnHome;

private javax.swing.JButton btnUpdate;

private javax.swing.JButton btnDelete;

private javax.swing.JComboBox<String> listSensors;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel10;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JPanel jPanel1;

private javax.swing.JTextField txtRoomNo;

private javax.swing.JTextField txtFloorNo;

private JLabel lblStatus;

private JRadioButton rdbtnInactive;

private JRadioButton rdbtnActive;

}

// set selected sensor values in Item model

class Item

{

private int id;

private String description;

public Item(int id, String description)

{

this.id = id;

this.description = description;

}

public int getId()

{

return id;

}

public String getDescription()

{

return description;

}

public String toString()

{

return description;

}

}

1. **Web Client**

Index.js

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from "./components/bootstrap/App";

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

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

ReactDOM.render(<App />, 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();

App.js

import React, { Fragment } from "react";

import Routes from "./Routes";

const App = () => {

return (

<Fragment>

<Routes />

</Fragment>

)

};

export default App;

Routes.js

import React, { Fragment } from "react";

import { BrowserRouter, Switch, Route } from "react-router-dom";

import Login from "../modules/login/Login";

import FireAlarmList from "../modules/fire-alarm-list/FireAlarmList";

const Routes = () => {

return (

<Fragment>

<BrowserRouter>

<Switch>

<Route path={"/"} exact={true} component={Login} />

<Route path={"/fire-alarm-list"} exact={true} component={FireAlarmList} />

</Switch>

</BrowserRouter>

</Fragment>

)

};

export default Routes;

FireAlarmList.js

import React, { Fragment, useState, useEffect } from "react";

import axios from "axios";

import DataTable from "../../ui-elements/table/Table";

import "./fire-alarm-list.css";

const FireAlarmList = () => {

const headers = ["Id", "Floor No", "Room No", "Smoke Level", "CO2 Level", "Status"];

const [data, setData] = useState([]);

useEffect(() => {

getSensorData();

setTimerFn();

}, []);

const setTimerFn = () => {

setInterval(() => {

getSensorData();

}, 5000)

};

const getSensorData = () => {

const axiosObject = {

headers: {

"Content-Type": "application/json"

},

method: 'get',

url: 'http://localhost:4000/api/fire-alarms'

};

axios(axiosObject)

.then((result) => {

if (result.data.data.length === 0) {

setData(null);

} else {

setData(result.data.data);

}

})

.catch((err) => {

setData(null);

})

};

return (

<Fragment>

<header className="header">

<h1>Fire Alarm Service</h1>

</header>

<div className={"container"}>

<div className={"row"}>

<div className={"col-md-12"}>

<div className={"tableWrapper mt-4"}>

<DataTable

headers={headers}

body={data}

/>

</div>

</div>

</div>

</div>

</Fragment>

)

};

export default FireAlarmList;

Login.js

import React, { useState } from "react";

import axios from "axios";

import { UICard } from "../../ui-elements/card/UICard";

import InputBox from "../../ui-elements/text-field/InputBox";

import InputButton from "../../ui-elements/button/InputButton";

import { CircleLoaderElement } from "../../ui-elements/circle-loader/CircleLoader";

import FlashMessageBox from "../../ui-elements/flash-message/FlashMessageBox";

import "./login.css";

const Login = (props) => {

const [form, setForm] = useState({});

const [loader, setLoader] = useState(false);

const [flash, setFlash] = useState(false);

const onChangeFn = (event) => {

setForm({

...form,

[event.name]: event.value

})

};

const onSubmit = () => {

setLoader(true);

const axiosObject = {

headers: {

"Content-Type": "application/json"

},

method: 'post',

url: 'http://localhost:4000/api/auth',

data: {

username: form.username,

password: form.password

}

};

axios(axiosObject)

.then((result) => {

setLoader(false);

if (result.data.message === "Authorized") {

props.history.push('/fire-alarm-list');

}

})

.catch((err) => {

setLoader(false);

setFlash(true);

})

};

return (

<div className={"container"}>

<div className={"row justify-content-center"}>

<div className={"col-md-6"}>

<div className={"loginWrapper text-center"}>

<UICard>

<h4>LOGIN</h4>

<p>Enter your Username and Password</p>

<InputBox

lableText={"Username"}

isRequired={true}

inputName={"username"}

inputValue={form.username}

inputPlaceholder={'test@gmail.com'}

onChangeTxtFn={onChangeFn}

/>

<InputBox

elementStyle={"mt-2"}

lableText={"Password"}

isRequired={true}

inputName={"password"}

inputType={"password"}

inputValue={form.password}

inputPlaceholder={'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'}

onChangeTxtFn={onChangeFn}

/>

<InputButton

elementStyle={"mt-4 mb-2"}

btnText={"Login"}

isFullWidth={true}

btnSize={"medium"}

onClickBtnFn={onSubmit}

/>

</UICard>

</div>

</div>

</div>

<CircleLoaderElement

isStatus={loader}

loaderSize={50}

loaderThickness={3}

/>

<FlashMessageBox

status={flash}

content={{

icon: "fas fa-times-circle",

cssClass: "errorFlashMessage",

message: "Invalid Username or Password!!"

}}

handleClose={() => setFlash(false)}

/>

</div>

)

};

export default Login;

InputButton.js

import React from "react";

import PropTypes from "prop-types";

import Button from '@material-ui/core/Button';

const emptyFun = (...para) => undefined;

/\*\*

\* --------------------------------------------

\* @Author: Chanaka Wickramasinghe

\* @Description: Input button Wrapper

\* --------------------------------------------

\*/

// InputButton color options

const inputBtnColors = {

default: "default",

inherit: "inherit",

primary: "primary",

secondary: "secondary"

};

// InputButton size options

const inputBtnSizes = {

small: "small",

medium: "medium",

large: "large"

};

// InputButton variant options

const inputBtnVariants = {

text: "text",

outlined: "outlined",

contained: "contained"

};

const InputButton = ({

elementWrapperStyle = "",

elementStyle = "",

btnText = "",

btnName = "",

btnSize = inputBtnSizes.small,

btnColor = inputBtnColors.primary,

isFullWidth = false,

isBtnDisabled = false,

startIcon = null,

endIcon = null,

variant = inputBtnVariants.contained,

onClickBtnFn = emptyFun

}) => {

return (

<span className={`${elementWrapperStyle}`}>

<Button

variant={variant}

color={btnColor}

size={btnSize}

className={`${elementStyle}`}

name={btnName}

fullWidth={isFullWidth}

disabled={isBtnDisabled}

onClick={(event) => onClickBtnFn({ name: btnName, eventInfo: event })}

startIcon={(startIcon != null) ? (<i className={startIcon}/>) : null}

endIcon={(endIcon != null) ? (<i className={endIcon}/>) : null}

>

<span style={{ textTransform: "none"}}>{btnText}</span>

</Button>

</span>

);

};

/\*\*

\* --------------------------------------------

\* @Author: Chanaka Wickramasinghe

\* @Description: Input button Wrapper

\* --------------------------------------------

\*/

InputButton.propTypes = {

/\*\* element Wrapper css class \*/

elementWrapperStyle: PropTypes.string,

/\*\* button element css class \*/

elementStyle: PropTypes.string,

/\*\* button text \*/

btnText: PropTypes.string,

/\*\* button name \*/

btnName: PropTypes.string,

/\*\* button size \*/

btnSize: PropTypes.oneOf(['small', 'medium', 'large']),

/\*\* button color \*/

btnColor: PropTypes.oneOf(['default', 'inherit', 'primary', 'secondary']),

/\*\* Is full width set or not \*/

isFullWidth: PropTypes.bool,

/\*\* Is button disable or not \*/

isBtnDisabled: PropTypes.bool,

/\*\* button left icon \*/

startIcon: PropTypes.string,

/\*\* button right icon \*/

endIcon: PropTypes.string,

/\*\* button variant \*/

variant: PropTypes.oneOf(['text', 'outlined', 'contained']),

/\*\* Onclick button function \*/

onClickBtnFn: PropTypes.func

};

export default InputButton;

UICard.js

import React, { Fragment } from "react";

import PropTypes from "prop-types";

import Card from '@material-ui/core/Card';

import "./ui-card.css";

/\*\*

\* --------------------------------------------

\* @Author: Chanaka Wickramasinghe

\* @Description: card Wrapper

\* --------------------------------------------

\*/

const UICard = ({

elementStyle = "",

children = null

}) => {

return (

<Card className={`defaultCardWrapper ${elementStyle}`}>

<Fragment>

{children}

</Fragment>

</Card>

);

};

/\*\*

\* --------------------------------------------

\* @Author: Chanaka Wickramasinghe

\* @Description: card Wrapper

\* --------------------------------------------

\*/

UICard.propTypes = {

/\*\* card element css class \*/

elementStyle: PropTypes.string,

/\*\* card children \*/

children: PropTypes.node

};

//----------------UICard---------------------

export { UICard };

CircleLoader.js

import React, { Fragment } from "react";

import CircularProgress from '@material-ui/core/CircularProgress';

import "./circle-loader.css";

const CircleLoaderElement = ({

isStatus=false,

loaderSize=40,

loaderThickness=2

}) => {

return (

<Fragment>

{

(isStatus) ? (

<div className={"pageLoader"}>

<CircularProgress

className={"loader"}

size={loaderSize}

thickness={loaderThickness}

/>

</div>

) : null

}

</Fragment>

);

};

export {

CircleLoaderElement

}

FlashMessageBox.js

import React from "react";

import Snackbar from '@material-ui/core/Snackbar';

import "./flash-message.css";

const emptyFun = (...para) => undefined;

const FlashMessageBox = ({

status = true,

handleClose = emptyFun,

content = {},

}) => {

const defaultClass = "defaultFlashMessageBox ";

const defaultContentClass = "defaultFlashContent ";

return (

<div>

<Snackbar

anchorOrigin={{

vertical: 'top',

horizontal: 'right',

}}

open={status}

autoHideDuration={2500}

transitionDuration={1000}

onClose={handleClose}

className={defaultClass + content.cssClass}

ContentProps={{

'aria-describedby': 'message-id',

}}

message={

<span id="message-id" className={defaultContentClass}>

<i className={content.icon}></i>

<span className="messageBoxTxt">{content.message}</span>

</span>

}

/>

</div>

);

};

export default FlashMessageBox;

DataTable.js

import React, { Fragment } from "react";

import Table from '@material-ui/core/Table';

import TableBody from '@material-ui/core/TableBody';

import TableCell from '@material-ui/core/TableCell';

import TableHead from '@material-ui/core/TableHead';

import TableRow from '@material-ui/core/TableRow';

import CircularProgress from '@material-ui/core/CircularProgress';

import Switch from '@material-ui/core/Switch';

import { UICard } from "../card/UICard";

import "./table.css";

const DataTable = ({

elementStyle = "",

headers = [],

body = []

}) => {

return (

<Fragment>

<UICard elementStyle={"p-0"}>

<Table className={elementStyle}>

<TableHead className={"tableHeader"}>

<TableRow>

{

headers.map((value, index) => {

return (

<TableCell key={index} className={"headerValue"}>{value}</TableCell>

)

})

}

</TableRow>

</TableHead>

<TableBody>

{

(body.length > 0) ? (

body.map((value, index) => {

return (

<TableRow

key={index}

hover={true}

className={

(value.smoke\_level >= 5 || value.co2\_level >= 5) ? "alertClass" : "normalClass"

}

>

<TableCell>{value.\_id}</TableCell>

<TableCell align={"center"}>{value.floor\_no}</TableCell>

<TableCell align={"center"}>{value.room\_no}</TableCell>

<TableCell align={"center"}>{value.smoke\_level}</TableCell>

<TableCell align={"center"}>{value.co2\_level}</TableCell>

<TableCell>

<Switch

checked={value.status}

color={

(value.smoke\_level >= 5 || value.co2\_level >= 5) ? "primary" : "secondary"

}

/>

</TableCell>

</TableRow>

)

})

) : (

(body === null) ? (

<TableRow hover={true}>

<TableCell>No records to show..</TableCell>

</TableRow>

) : (

<TableRow>

<TableCell colSpan={6} align={"center"}>

<CircularProgress

color={"primary"}

size={90}

thickness={3}

/>

</TableCell>

</TableRow>

)

)

}

</TableBody>

</Table>

</UICard>

</Fragment>

)

};

export default DataTable;

InputBox.js

import React from "react";

import PropTypes from "prop-types";

import TextField from '@material-ui/core/TextField';

import './input-text.css';

const emptyFun = (...para) => undefined;

/\*\*

\* --------------------------------------------

\* @Author: Chanaka Wickramasinghe

\* @Description: Material TextField Wrapper

\* --------------------------------------------

\*/

const InputBox = ({

elementStyle = "",

isFullWidth = true,

lableText = "",

inputType = "text",

inputValue = "",

inputName = "",

inputError = false,

isMultiline = false,

rows = 1,

isAutoFocus = false,

inputPlaceholder = "",

isDisabled = false,

isRequired = false,

onChangeTxtFn = emptyFun

}) => {

return (

<TextField

className={`defaultInputWrapper ${elementStyle}`}

label={lableText}

type={inputType}

value={inputValue}

required={isRequired}

fullWidth={isFullWidth}

name={inputName}

autoFocus={isAutoFocus}

placeholder={inputPlaceholder}

disabled={isDisabled}

error={inputError}

multiline={isMultiline}

rows={rows}

onChange={event =>

onChangeTxtFn({

name: inputName,

value: event.target.value,

eventInfo: event

})

}

/>

);

};

/\*\*

\* --------------------------------------------

\* @Author: Chanaka Wickramasinghe

\* @Description: Material TextField Wrapper

\* --------------------------------------------

\*/

InputBox.propTypes = {

/\*\* Button element css class \*/

elementStyle: PropTypes.string,

/\*\* is set full width or not \*/

isFullWidth: PropTypes.bool,

/\*\* lable text \*/

lableText: PropTypes.string,

/\*\* input field type \*/

inputType: PropTypes.oneOf(['text', 'password','number','email']),

/\*\* input field type \*/

inputValue: PropTypes.string,

/\*\* input field name \*/

inputName: PropTypes.string,

/\*\* input field error message \*/

inputError: PropTypes.bool,

/\*\* is text area or not \*/

isMultiline: PropTypes.bool,

/\*\* is text area or not \*/

rows: PropTypes.number,

/\*\* is auto Focus or not \*/

isAutoFocus: PropTypes.bool,

/\*\* input field placeholder text \*/

inputPlaceholder: PropTypes.string,

/\*\* enable/disabled field \*/

isDisabled: PropTypes.bool,

/\*\* set required \* mark \*/

isRequired: PropTypes.bool,

/\*\* onchange text event Function \*/

onChangeTxtFn: PropTypes.func,

};

//----------------InputBox---------------------

export default InputBox;

1. **Email Service**

Index.js

const express = require('express');

const compress = require('compression');

const cors = require('cors');

const helmet = require('helmet');

const { mailFn } = require('./helpers/email.helper');

const app = express();

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.use(compress());

app.use(cors());

app.use(helmet());

app.post('/api/emails', (req, res) => {

const html = `<h4>Dear Sir/Madam</h4><br><h4>There is an alert from sensor (Floor: ${req.body.floor}, Room: ${req.body.room})</h4><br><h4>Thanks & Regards,<br>Fire Alarm Service</h4>`;

mailFn("blcchanaka@gmail.com", "firalarmservice@gmail.com", "Sensor Alert", html);

res.send("Email Send");

});

app.listen(4005, (err) => {

console.log("Server has started");

});

Email.helper.js

const nodemailer = require('nodemailer');

const mailFn = async (to, from, subject, body) => {

const transporter = await nodemailer.createTransport({

host: "smtp.gmail.com",

port: "587",

secure: false, // true for 465, false for other ports

auth: {

user: "ras@inovaitsys.com", // generated ethereal user

pass: "inoras123" // generated ethereal password

}

});

return await transporter.sendMail({

from: from, // sender address

to: to, // list of receivers

subject: subject, // Subject line

html: body // html body

});

};

module.exports = {

mailFn: mailFn

};

1. **SMS Service**

Index.js

const express = require('express');

const app = express();

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

app.post('/api/sms', (req, res) => {

//sendMsj({});

res.send("Success");

});

const sendMsj = (data) => {

console.log("SMS has sent successfully");

};

app.listen(4010, (err) => {

console.log("Server has started");

});

1. **Simulate Application**

Index.js

const axios = require('axios');

const getSensorData = () => {

return new Promise((resolve, reject) => {

const data = [];

const axiosObject = {

headers: {

"Content-Type": "application/json"

},

method: 'get',

url: 'http://localhost:4000/api/fire-alarms'

};

axios(axiosObject)

.then((result) => {

result.data.data.forEach((value) => {

data.push(value.\_id);

});

resolve(data);

})

.catch((err) => {

reject(err);

})

})

};

const getRandomInt = (min, max) => {

min = Math.ceil(min);

max = Math.floor(max);

return Math.floor(Math.random() \* (max - min)) + min; //The maximum is exclusive and the minimum is inclusive

};

const updateSensorData = (id) => {

const fields = ["smoke\_level", "co2\_level"];

let selectedField = fields[Math.floor(Math.random() \* fields.length)];

const randomValue = getRandomInt(1, 10);

const axiosObject = {

headers: {

"Content-Type": "application/json"

},

method: 'put',

url: `http://localhost:4000/api/fire-alarms/${id}`,

data: {

[selectedField]: randomValue

}

};

axios(axiosObject)

.then((result) => {

if (result.data.status === "success") {

return result.data.status;

}

})

};

const mainMethod = async () => {

let sensorsToBeSimulated = Number(process.argv[2]) || 1;

getSensorData()

.then((result) => {

if (result.length < sensorsToBeSimulated) {

sensorsToBeSimulated = result.length;

}

for (let i = 0; i < sensorsToBeSimulated; i++) {

setInterval(() => {

updateSensorData(result[i]);

}, 10000)

}

});

};

mainMethod();