Fire Alarm Monitoring System

Group Details:

Group Number: 09

Batch: Third Year (First Semester - Weekend)

|  |  |  |
| --- | --- | --- |
|  | Student Registration Number | Student Name |
| 1 | IT18050318 | M. A. Zeid |
| 2 | IT17029896 | D.S. Jiffry |
| 3 | IT18060690 | M.R.M. Rifan |
| 4 | IT18200034 | M.A.F.Hasna |

Contents

[1.0 Introduction 3](#_Toc39706425)

[2.0 Architectural Diagram 4](#_Toc39706426)

[3.0 Deep Dive into Services and Communications 6](#_Toc39706427)

[3.1 Sensors - Server communication 6](#_Toc39706428)

[3.2. RMI Server: 6](#_Toc39706429)

[3.3. Authentication Service: 7](#_Toc39706430)

[3.4 Alert service: 8](#_Toc39706431)

[3.5 Sensor Service 8](#_Toc39706432)

[4.0 Workflows 10](#_Toc39706433)

[4.1 Sensor Monitor Service 10](#_Toc39706434)

[4.2 RMI Client Server 11](#_Toc39706435)

[5.0 Appendix 12](#_Toc39706436)

[5.1 Alert Service 12](#_Toc39706437)

[5.2 Authentication Service 18](#_Toc39706438)

[5.3 RMI – Client 24](#_Toc39706439)

[5.4 RMI – Server 90](#_Toc39706440)

[5.5 Sensor Service 101](#_Toc39706441)

[5.6 web client 119](#_Toc39706442)

# Introduction

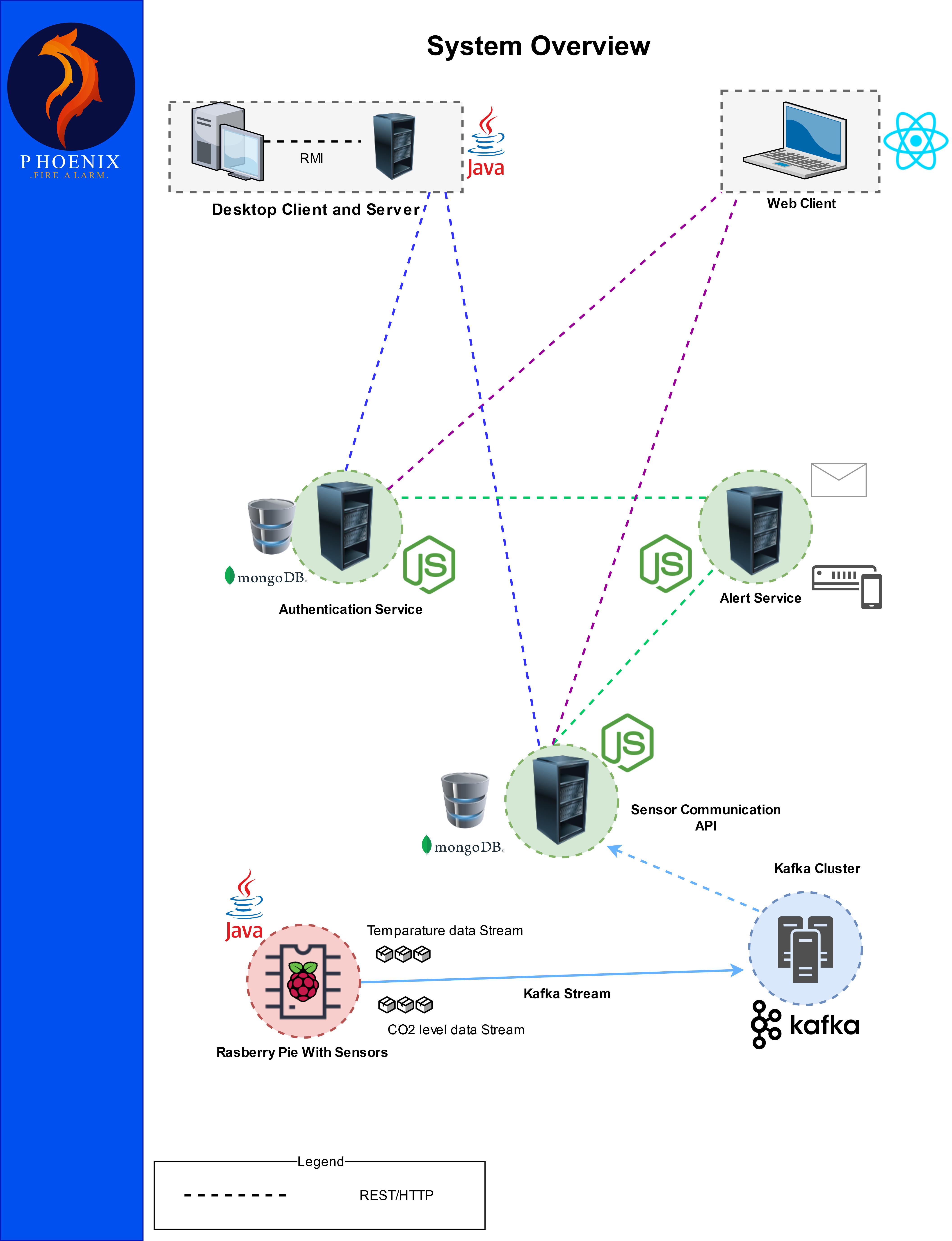
Phoenix Fire is a smoke and co2 sensor alarm system designed to manage an array of smoke and fire sensors. The key functionality of the system is to read sensor data continuously, to alert the user when a specific smoke or co2 level has passed via email or SMS alerts, to check continually which sensors have gone offline and register or remove new sensors to and from the system.

The system is accessible to the user through the desktop interface which is used for administrative purposes and through a web client which is used to monitor the sensors from a remote location. Both of these client applications communicate to the actual backend services through REST calls.

The backend is built using the Service Oriented Architecture principles to provide re-usable services for scalability, modularity and fault tolerance. The key non-functional requirement was identified to be fault tolerance. The system needed to notify the users about an emergency situation even if part of the system was offline, hence the system was broken into smaller services.

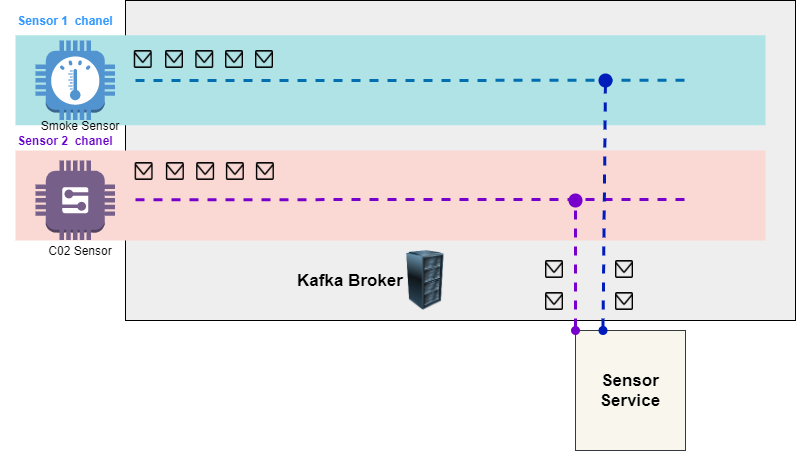
The system consists of three main services which communicated to each other using REST calls. The services include an Authentication service responsible to handle user information and authentication mechanisms, an Alert Service which is responsible for sending emails or SMS`s. and a senor service which is responsible for keeping track of the sensors , processing sensor data to detect emergencies , and continuously checking for sensors which have gone offline. All three services work together to provide the functional requirement of the system.

# 2.0 Architectural Diagram



# Deep Dive into Services and Communications

## 3.1 Sensors - Server communication



The sensor - server communication is done using Kafka streams, every time a new sensor is registered a Kafka topic is created that serves as a channel unique to that sensor ID. Each sensor emits a message every time a reading is done which contains the reading and current time, then the sensor client pushes the message to the Kafka topic under their sensor ID. These messages are consumed by the server and relevant services accordingly.

## 3.2. RMI Server:

* + - * The server exposes the following to the RMI Client:
        + public boolean addSensor(int floorNumber, int roomNumber) throws RemoteException;
        + public boolean removeSensor(int floorNumber, int roomNumber) throws RemoteException;
        + public boolean changeState(int floorNumber, int roomNumber, boolean state) throws RemoteException;
        + public ArrayList<Sensor> viewSensors() throws RemoteException;
        + public boolean login(String username, String password) throws RemoteException;
        + public boolean checkAuthenticationServer() throws RemoteException;
* public HashMap<String,String> getReading(HashMap<String, String> sensorName) throws RemoteException;
  + - * The Client will first use **checkAuthenticationServer()** to make sure the Authentication service is reachable.
      * The **login(…)** will make the RMI Server validate the login by contacting the Authentication Server.
      * The **addSensor(...), removeSensor(…) and changeState(…)** are used to manage the sensors. The client will make the request to the RMI server who then talks to the Sensor Communication API and gets the reply which is then sent to the RMI Client.
      * **ViewSensors()** will make the RMI Server retrieve all the sensors in the Sensor Communication API for the Client. This is called every 30s to keep the data updated.

## 3.3. Authentication Service:

* <http://localhost:8080/loginUser> – Will validate the login of the user, expects a POST request with JSON keys “username” and “password”. Will return status 200 if login is valid otherwise status 404.
* <http://localhost:8080/register> – Will register a new user to the system, expects a POST request with JSON keys "username", "password", "email", "phoneNumber"and "type". (“type” can only have values of either “admin” or “user”). Passwords will be hash-coded before added to the database to avoid storing of plain-text passwords. Will return status 404 if username is taken, otherwise status 200 if register is sucessful.
* <http://localhost:8080/getEmails> – Will return a JSON array containing the emails of all the registered users, this is for the use of the Alert service to send out emails.
* <http://localhost:8080/getPhoneNumbers> – Will return a JSON array containing the phone numbers of all the registered users, this for use of the Alert Service to send out SMS.
* <http://localhost:8080/checkAuthenticationAlive> – Can be used to make sure the Authentication service is reachable. Will return status 200 if called.

## 3.4 Alert service:

* In our system we have identified the Alert Service and the sensor API as the critical services. In case one of the other services was down when a fire started, we wanted to make sure that the alert would still go out to the users. To do this we implemented a local caching mechanism. When the alert service first comes online it would get the emails and phone numbers of the users from the Authentication service and store them locally.
* In case a fire occurred, the sensor API would call on the Alert Service and then the alert service would fetch the emails and numbers from the Authentication service. Now in case the Authentication service is down when the Alert service needs the details. It would send out the emails and messages to the addresses and numbers it has cached. This way we can guarantee that even if all the users don’t get notified at least some of them will be. Since in this kind of safety system it is better to at least notify some people than none at all.
* This mechanism prevents our Alert service from being completely dependent on the Authentication service. When requesting emails and numbers of the Authentication service is reachable then the cache would also be updated.
* <http://localhost:8081/emailAlert> – Will send out an email alert. Expects a POST request with the JSON key “message”.
* <http://localhost:8081/smsAlert> – Will send out an SMS alert. Expects a POST request with the JSON key “message”.

## 3.5 Sensor Service

## 

The sensor Service contains three smaller services that run independently.

1. Sensor API:

The API that manages sensor meta data and information and used as the access point to serve information to other services regarding sensors. The sensor registration, deletion and updating is done in this service along with interfaces that provide information about sensors including status and latest reading.

Service Interfaces

|  |  |  |  |
| --- | --- | --- | --- |
| Interface | Description | Input | Output |
| GET  /registerSensor | Used to register new sensors to the system | SensorUID,  Sensor type,  Room, floor | Registered sensor saved in database |
| PUT  /updateSensor | Updates sensor details | SensorUID,  Sensor type,  Room, floor | Updates existing sensors with new information |
| DELETE  /deleteSensor | Deletes an existing sensor | Sensor UID | Deletes the sensor from the system |
| PUT  /updateSensorStatus | Updates the sensor status to online or offline | SensorUID  status | Updates the sensor status in the system |
| PUT  /updateSensorReading | Updates the sensor reading with latest reading | Sensor UID | Updates the reading value in the database |
| GET  /getAllSensorsByUsername | Gets all the sensor data belonging to one user | Username | List of all the sensors registered under the username |
| GET  /getAllSensors | Gets all the sensors in the system , used in the senor monitoring system |  | List of all the sensors in the system |

1. Sensor monitoring service:

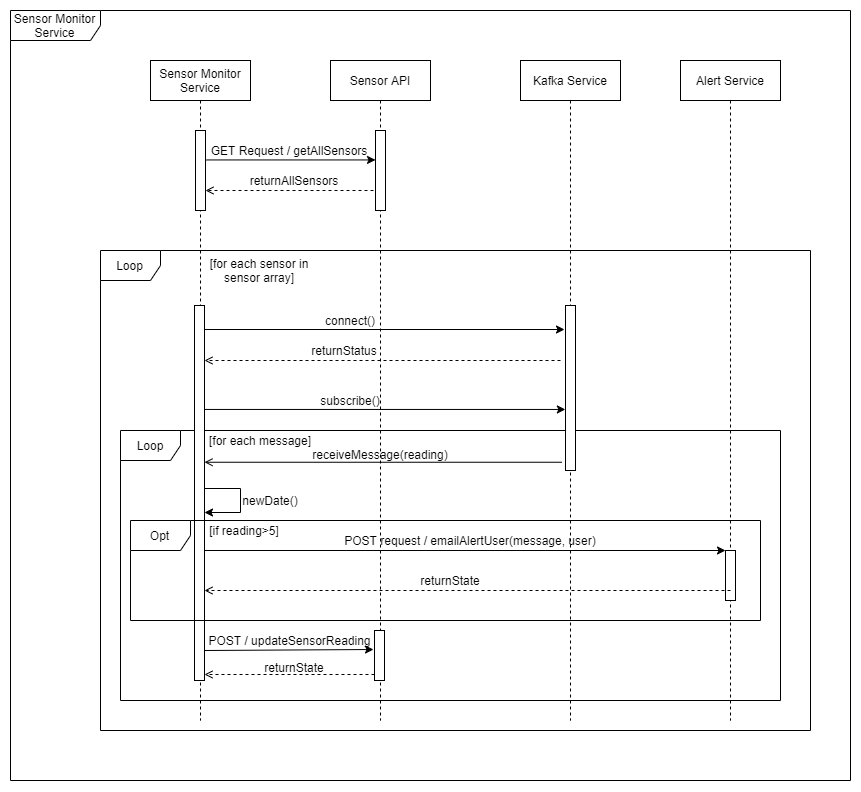
The monitoring service is an autonomous service that continuously listens to the all the sensor data streams. If the reading exceeds value 5 the service calls the alert service to send an email to the user.

1. Sensor Health checking Service

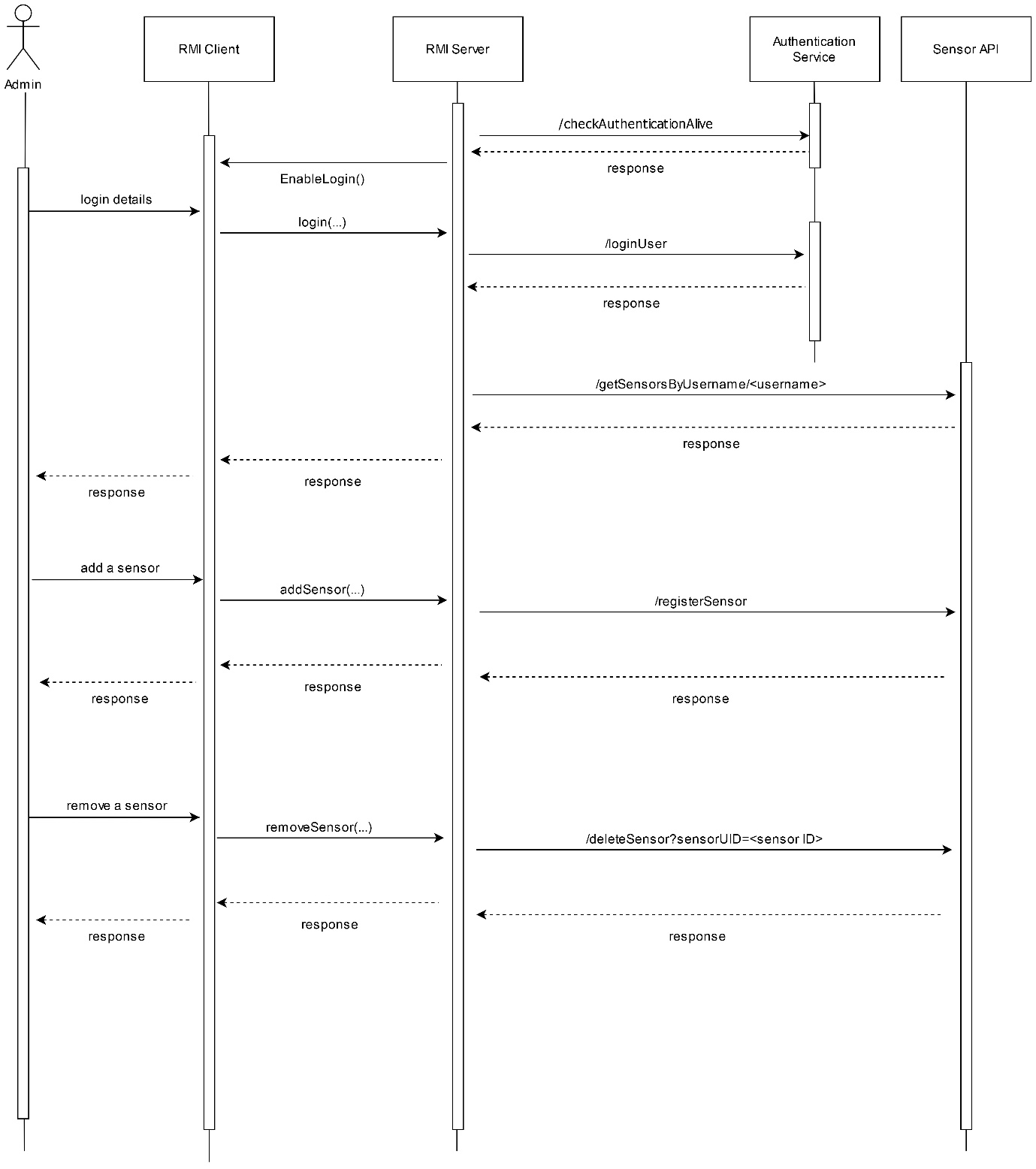
This service is also an autonomous service running which listens to all the sensor data streams. If the sensor doesn’t emit a message for a period of 30 seconds the service will call the senor API and update the status as offline. If the sensor does retransmit message the services will update the services after 30 seconds of the first retransmitted message.

# Workflows

## Sensor Monitor Service



## 4.2 RMI Client Server



# 5.0 Appendix

## 5.1 Alert Service

**server.js**

let express = require("express");

let app = express();

let nodemailer = require('nodemailer');

let fetch = require("node-fetch");

app.use(express.json());

let getEmailsURL = "http://localhost:8080/getEmails";

let getphoneURL = "http://localhost:8080/getPhoneNumbers";

*// In case the authentication server is unreachable, the email/sms will be sent to these.*

let emailCache = [];

let smsCache = [];

*/\*\**

*\* Send email Alert*

*\* POST to http://localhost:8081/emailAlert*

*\* with JSON keys: "message"*

*\*/*

app.post('/emailAlert', async (*req*, *res*) => {

  let message = req.body.message; *// Body of the Email*

  let emailAddresses;

  if (!(message)) {

    return res.status(404).send('Error in JSON body');

  }

  emailAddresses = await fetch(getEmailsURL, {

    method: 'POST',

    headers: {

      'Accept': 'application/JSON',

      'Content-Type': 'application/JSON',

    }

  }).then((*response*) => {

    if (response.ok) {

      return response.json();

    } else {

      throw new Error('Something went wrong');

    }

  }).catch((*err*) => {

  });

  if (!(emailAddresses)) {

    emailAddresses = emailCache.slice();

  }

  else

  {

    emailCache = emailAddresses.slice();

  }

  let testAccount = await nodemailer.createTestAccount();

  let transporter = nodemailer.createTransport({

    host: "smtp.ethereal.email",

    port: 587,

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

    auth: {

      user: testAccount.user, *// generated ethereal user*

      pass: testAccount.pass *// generated ethereal password*

    }

  });

  let info = await transporter.sendMail({

    from: '"Fire Alarm Service" <Alert@FireAlarm.com>', *// sender address*

    to: emailAddresses, *// list of receivers*

    subject: "Fire Alarm Service Alert", *// Subject line*

    text: message, *// plain text body*

  });

  console.log("Email sent");

  return res.status(200).send(nodemailer.getTestMessageUrl(info)); *//Will return a URL to preview the email that was sent*

});

*/\*\**

*\* Send email Alert to a user*

*\* POST to http://localhost:8081/emailAlertToUser*

*\* with JSON keys: "message", "username"*

*\*/*

app.post('/emailAlertToUser', async (*req*, *res*) => {

  let message = req.body.message; *// Body of the Email*

  let username = req.body.username;

  let emailAddresses;

  if (!(message) || !(username) ) {

    return res.status(404).send('Error in JSON body');

  }

  emailAddresses = await fetch(getEmailsURL, {

    method: 'POST',

    headers: {

      'Accept': 'application/JSON',

      'Content-Type': 'application/JSON',

    },

    body: JSON.stringify({"username":username})

  }).then((*response*) => {

    if (response.ok) {

      return response.json();

    } else {

      throw new Error('Something went wrong');

    }

  }).catch((*err*) => {

  });

  if (!(emailAddresses) && (username)) {

    emailAddresses = emailCache.slice();

  }

  else

  {

    emailCache = emailAddresses.slice();

  }

  let testAccount = await nodemailer.createTestAccount();

  let transporter = nodemailer.createTransport({

    host: "smtp.ethereal.email",

    port: 587,

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

    auth: {

      user: testAccount.user, *// generated ethereal user*

      pass: testAccount.pass *// generated ethereal password*

    }

  });

  let info = await transporter.sendMail({

    from: '"Fire Alarm Service" <Alert@FireAlarm.com>', *// sender address*

    to: emailAddresses, *// list of receivers*

    subject: "Fire Alarm Service Alert", *// Subject line*

    text: message, *// plain text body*

  });

  console.log("Email sent");

  return res.status(200).send(nodemailer.getTestMessageUrl(info)); *//Will return a URL to preview the email that was sent*

});

*/\*\**

*\* Send SMS Alert*

*\* POST to http://localhost:8081/smsAlert*

*\* JSON keys: "message"*

*\*/*

app.post('/smsAlert', async (*req*, *res*) => {

  let message = req.body.message;

  let smsNumbers;

  if (!(message)) {

    return res.status(404).send('Error in JSON body');

  }

  smsNumbers = await fetch(getphoneURL, {

    method: 'POST',

    headers: {

      'Accept': 'application/JSON',

      'Content-Type': 'application/JSON',

    }

  }).then((*response*) => {

    if (response.ok) {

      return response.json();

    } else {

      throw new Error('Something went wrong');

    }

  }).catch((*err*) => {

  });

  if (!(smsNumbers)) {

    smsNumbers = smsCache.slice();

  }

  else

  {

    smsCache = smsNumbers.slice();

  }

  console.log("Sms sent");

  return res.status(200).send("Sms message: " + message + " sent to: " + smsNumbers);

});

*/\*\**

*\* Send SMS Alert to user*

*\* POST to http://localhost:8081/smsAlertToUser*

*\* JSON keys: "message", "username"*

*\*/*

app.post('/smsAlertToUser', async (*req*, *res*) => {

  let message = req.body.message;

  let username = req.body.username;

  let smsNumbers;

  if (!(message) || !(username)) {

    return res.status(404).send('Error in JSON body');

  }

  smsNumbers = await fetch(getphoneURL, {

    method: 'POST',

    headers: {

      'Accept': 'application/JSON',

      'Content-Type': 'application/JSON',

    },

    body: JSON.stringify({"username":username})

  }).then((*response*) => {

    if (response.ok) {

      return response.json();

    } else {

      throw new Error('Something went wrong');

    }

  }).catch((*err*) => {

  });

  if (!(smsNumbers)) {

    smsNumbers = smsCache.slice();

  }

  else

  {

    smsCache = smsNumbers.slice();

  }

  console.log("Sms sent");

  return res.status(200).send("Sms message: " + message + " sent to: " + smsNumbers);

});

*/\*\**

*\* Maintaining a cache in case authorization server is down during a fire*

*\*/*

async function updateCache() {

  emailCache = await fetch(getEmailsURL, {

    method: 'POST',

    headers: {

      'Accept': 'application/JSON',

      'Content-Type': 'application/JSON',

    }

  }).then((*response*) => {

    if (response.ok) {

      return response.json();

    } else {

      throw new Error('Something went wrong');

    }

  }).catch((*err*) => {

  });

  smsCache = await fetch(getphoneURL, {

    method: 'POST',

    headers: {

      'Accept': 'application/JSON',

      'Content-Type': 'application/JSON',

    }

  }).then((*response*) => {

    if (response.ok) {

      return response.json();

    } else {

      throw new Error('Something went wrong');

    }

  }).catch((*err*) => {

  });

  if (!(smsCache)) {

    smsCache = ["119"];

  }

  if (!(emailCache)) {

    emailCache = ["damn@weScrewed.com"];

  }

}

*//Keep server running on port*

const port = 8081;

app.listen(port, () => {

  updateCache();

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

});

## **5.2 Authentication Service**

**Server.js**

let express = require("express");

let encode = require('hashcode').hashCode;

let app = express();

let mongoose = require('mongoose/');

app.use(express.json());

*//MongoDB url*

let url = "mongodb://localhost:27017/Fire\_Alarm\_Authentication";

*//Defining a Schema*

let userSchema = mongoose.Schema({

  username: String,

  password: String,

  email: String,

  phoneNumber: String,

  type: String  *// either "user" or "admin"*

});

*//compiling schema to model*

let userModel = mongoose.model('User', userSchema, 'Users');

*/\*\**

*\* Login Function for users*

*\* POST to http://localhost:8080/loginUser*

*\* With JSON Keys : "username", "password"*

*\*/*

app.post('/loginUser', (*req*, *res*) => {

  let username = req.body.username;

*//Hashcoding password before checking with DB*

  let password = encode().value(req.body.password);

  if (!(username) || password == 0) {

    return res.status(404).send('Error in JSON body');

  }

  connectToDB();

  userModel.findOne({ username: username, password: password, type: "user" }, (*err*, *user*) => {

    if (err) {

      console.log(err);

    }

*//If a record is found*

    if (user) {

      return res.status(200).send('Valid Login');

    }

*//If no record found*

    if (!user) {

      console.log('Incorrect Login Details');

      res.status(404).send('Incorrect Login Details');

    }

  });

});

*/\*\**

*\* Login Function for admins*

*\* POST to http://localhost:8080/loginAdmin*

*\* With JSON Keys : "username", "password"*

*\*/*

app.post('/loginAdmin', (*req*, *res*) => {

  let username = req.body.username;

*//Hashcoding password before checking with DB*

  let password = encode().value(req.body.password);

  if (!(username) || password == 0) {

    return res.status(404).send('Error in JSON body');

  }

  connectToDB();

  userModel.findOne({ username: username, password: password, type: "admin" }, (*err*, *user*) => {

    if (err) {

      console.log(err);

    }

*//If a record is found*

    if (user) {

      return res.status(200).send('Valid Login');

    }

*//If no record found*

    if (!user) {

      console.log('Incorrect Login Details');

      res.status(404).send('Incorrect Login Details');

    }

  });

});

*/\*\**

*\* Register New User Function*

*\* POST to http://localhost:8080/register*

*\* With JSON Keys : "username", "password", "email", "phoneNumber", "type"*

*\*/*

app.post('/register', (*req*, *res*) => {

  let username = req.body.username;

  let email = req.body.email;

  let phone = req.body.phoneNumber;

  let type = req.body.type; *// either "user" or "admin"*

*//Hashcoding password for security before storing*

  let password = encode().value(req.body.password);

  if (!(username) || password == 0 || !(email) || !(phone) || !(type)) {

    return res.status(404).send('Error in JSON body');

  }

  if (type != "admin" && type != "user") {

    return res.status(406).send('incorrect type in JSON body. must be either "admin" or "user"');

  }

  connectToDB();

  userModel.findOne({ username: username }, (*err*, *user*) => {

    if (err) {

      console.log(err);

    }

    if (user) {

      console.log('Username is taken');

      res.status(404).send('Username is taken');

    }

    if (!user) {

*//reference to DB*

      let DB = mongoose.connection;

*//Creating a Document*

      let userDoc = new userModel({

        username: username,

        password: password,

        email: email,

        phoneNumber: phone,

        type: type

      });

*//Saving to DB*

      userDoc.save(function (*err*, *user*) {

        if (err) {

          return console.error(err);

        }

        console.log(user.username + " added to DB");

      });

*//Sending client to login page*

      return res.status(200).send(username + ' added to Database');

    }

  });

});

*/\*\**

*\* Get email of user*

*\* POST to http://localhost:8080/getEmails*

*\* with optional JSON keys: "username"*

*\*/*

app.post('/getEmails', (*req*, *res*) => {

  let username = req.body.username;

  connectToDB();

  userModel.find({}, '-\_id username email', function (*err*, *user*) {

    if (err) {

      console.log(err);

    }

    let emails = [];

    if (!(username)) {

      user.forEach(*element* => {

        emails.push(element.email);

      });

    }

    else {

      user.forEach(*element* => {

        if (element.username == username) {

          emails.push(element.email);

        }

      });

    }

    return res.status(200).send(emails);

  });

});

*/\*\**

*\* Get phoneNumber of user*

*\* POST to http://localhost:8080/getPhoneNumbers*

*\* with optional JSON keys: "username"*

*\*/*

app.post('/getPhoneNumbers', (*req*, *res*) => {

  let username = req.body.username;

  connectToDB();

  userModel.find({}, '-\_id username phoneNumber', function (*err*, *user*) {

    if (err) {

      console.log(err);

    }

    let numbers = [];

    if (!(username)) {

      user.forEach(*element* => {

        numbers.push(element.phoneNumber);

      });

    }

    else {

      user.forEach(*element* => {

        if (element.username == username) {

          numbers.push(element.phoneNumber);

        }

      });

    }

    return res.status(200).send(numbers);

  });

});

app.post('/checkAuthenticationAlive', (*req*, *res*) => {

  return res.status(200).send();

});

*/\*\**

*\* Connecting to the Database*

*\*/*

function connectToDB() {

  mongoose.connect(url,

    {

      useNewUrlParser: true ,

      "auth": {"authSource":"admin"} ,

      "user": "root",

      "pass": "rootpassword"

    }

  )

    .then(function () {

      console.log('Connected to MongoDB');

    })

    .catch(function (*err*) {

      console.log('Error in Connecting to MongoDB');

      return;

    });

}

*// Keep server running on port*

const port = 8080;

app.listen(port, () => {

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

});

## 5.3 RMI – Client

Controllers –

**Client.java**

package Controllers;

import forms.Alert;

import java.net.MalformedURLException;

import java.rmi.Naming;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.util.ArrayList;

import java.util.HashMap;

import model.Sensor;

public class Client {

    private ServerInterface service;

    private final String USERNAME;

    private Client() {

        USERNAME = "";

    }

    public Client(String *username*) {

        USERNAME = username;

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

        if (System.getSecurityManager() == null) {

            System.setSecurityManager(new SecurityManager());

        }

        try {

            service = (ServerInterface) Naming.lookup("rmi://localhost/rmiServer");

        } catch (MalformedURLException | NotBoundException | RemoteException *ex*) {

            Alert alert = new Alert("Connection error. Unable to reach RMI server");

        }

    }

*/\*\**

*\* Will as RMI server to add a server*

*\**

*\* @param floorNumber*

*\* @param roomNumber*

*\* @param sensorType "smoke" or "co2"*

*\* @return true if request was successful*

*\*/*

    public boolean addSensor(int *floorNumber*, int *roomNumber*, String *sensorType*) {

        try {

            return service.addSensor(floorNumber, roomNumber, USERNAME, sensorType);

        } catch (RemoteException *ex*) {

            return false;

        }

    }

*/\*\**

*\* Will as RMI server to change the state of a sensor*

*\**

*\* @param floorNumber*

*\* @param roomNumber*

*\* @param state true will make the sensor active, false will make it*

*\* inactive*

*\* @param sensorType "smoke" or "co2"*

*\* @return true if request was successful*

*\*/*

    public boolean changeState(int *floorNumber*, int *roomNumber*, boolean *state*, String *sensorType*) {

        try {

            return service.changeState(floorNumber, roomNumber, state, sensorType);

        } catch (RemoteException *ex*) {

            return false;

        }

    }

*/\*\**

*\* Will make ask RMI server to delete a sensor*

*\**

*\* @param floorNumber*

*\* @param roomNumber*

*\* @param sensorType "smoke" or "co2"*

*\* @return true if request was successful*

*\*/*

    public boolean removeSensor(int *floorNumber*, int *roomNumber*, String *sensorType*) {

        try {

            return service.removeSensor(floorNumber, roomNumber, sensorType);

        } catch (RemoteException *ex*) {

            return false;

        }

    }

*/\*\**

*\* Will request all the sensors in the database from the RMI server*

*\**

*\* @return ArrayList of all the Sensors*

*\*/*

    public ArrayList<Sensor> getSensors() {

        try {

            return service.viewSensors(USERNAME);

        } catch (RemoteException *ignored*) {

        }

        return new ArrayList<>();

    }

*/\*\**

*\* Will check the login of the user*

*\**

*\* @param username*

*\* @param password*

*\* @return true if login was successful*

*\*/*

    public boolean login(String *username*, String *password*) {

        try {

            return service.login(username, password);

        } catch (RemoteException *ex*) {

            return false;

        }

    }

*/\*\**

*\* Will check if the authorization server is reachable*

*\**

*\* @return true if reachable*

*\*/*

    public boolean checkAuthenticationServer() {

        try {

            return service.checkAuthenticationServer();

        } catch (RemoteException *ex*) {

            return false;

        }

    }

*/\*\**

*\* getting the reading of a sensor*

*\**

*\* @param sensorName*

*\* @return HashMap with sensorUID and reading*

*\*/*

    public HashMap<String, String> getReading(HashMap<String, String> *sensorName*) {

        try {

            return service.getReading(sensorName);

        } catch (RemoteException *ex*) {

            ex.printStackTrace();

            return new HashMap<>();

        }

    }

}

**ServerInterface.java**

package Controllers;

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.ArrayList;

import java.util.HashMap;

import model.Sensor;

public interface ServerInterface extends Remote {

    public boolean addSensor(int *floorNumber*, int *roomNumber*, String *username*, String *sensorType*) throws RemoteException;

    public boolean removeSensor(int *floorNumber*, int *roomNumber*, String *sensorType*) throws RemoteException;

    public boolean changeState(int *floorNumber*, int *roomNumber*, boolean *state*, String *sensorType*) throws RemoteException;

    public ArrayList<Sensor> viewSensors(String *username*) throws RemoteException;

    public boolean login(String *username*, String *password*) throws RemoteException;

    public boolean checkAuthenticationServer()  throws RemoteException;

   public HashMap<String,String> getReading(HashMap<String, String> *sensorName*) throws RemoteException;

}

Forms –

**Alert.form**

<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.3" maxVersion="1.9" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">

  <Properties>

    <Property name="defaultCloseOperation" type="int" value="3"/>

    <Property name="alwaysOnTop" type="boolean" value="true"/>

    <Property name="undecorated" type="boolean" value="true"/>

  </Properties>

  <SyntheticProperties>

    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>

    <SyntheticProperty name="generateCenter" type="boolean" value="false"/>

  </SyntheticProperties>

  <AuxValues>

    <AuxValue name="FormSettings\_autoResourcing" type="java.lang.Integer" value="0"/>

    <AuxValue name="FormSettings\_autoSetComponentName" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_generateFQN" type="java.lang.Boolean" value="true"/>

    <AuxValue name="FormSettings\_generateMnemonicsCode" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_i18nAutoMode" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_layoutCodeTarget" type="java.lang.Integer" value="1"/>

    <AuxValue name="FormSettings\_listenerGenerationStyle" type="java.lang.Integer" value="0"/>

    <AuxValue name="FormSettings\_variablesLocal" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_variablesModifier" type="java.lang.Integer" value="2"/>

    <AuxValue name="designerSize" type="java.awt.Dimension" value="-84,-19,0,5,115,114,0,18,106,97,118,97,46,97,119,116,46,68,105,109,101,110,115,105,111,110,65,-114,-39,-41,-84,95,68,20,2,0,2,73,0,6,104,101,105,103,104,116,73,0,5,119,105,100,116,104,120,112,0,0,1,44,0,0,2,44"/>

  </AuxValues>

  <Layout class="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout">

    <Property name="useNullLayout" type="boolean" value="false"/>

  </Layout>

  <SubComponents>

    <Container class="javax.swing.JPanel" name="jPanel1">

      <AuxValues>

        <AuxValue name="JavaCodeGenerator\_InitCodePre" type="java.lang.String" value="jPanel1.setBackground(MaterialColors.BLUE\_GRAY\_800);"/>

      </AuxValues>

      <Constraints>

        <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

          <AbsoluteConstraints x="0" y="0" width="560" height="300"/>

        </Constraint>

      </Constraints>

      <Layout>

        <DimensionLayout dim="0">

          <Group type="103" groupAlignment="0" attributes="0">

              <Group type="102" alignment="0" attributes="0">

                  <Component id="jLabel1" min="-2" pref="561" max="-2" attributes="0"/>

                  <EmptySpace min="0" pref="0" max="32767" attributes="0"/>

              </Group>

              <Group type="102" alignment="0" attributes="0">

                  <EmptySpace min="-2" pref="225" max="-2" attributes="0"/>

                  <Component id="okButton" min="-2" pref="104" max="-2" attributes="0"/>

                  <EmptySpace max="32767" attributes="0"/>

              </Group>

          </Group>

        </DimensionLayout>

        <DimensionLayout dim="1">

          <Group type="103" groupAlignment="0" attributes="0">

              <Group type="102" alignment="0" attributes="0">

                  <EmptySpace min="-2" pref="66" max="-2" attributes="0"/>

                  <Component id="jLabel1" min="-2" pref="95" max="-2" attributes="0"/>

                  <EmptySpace min="-2" pref="45" max="-2" attributes="0"/>

                  <Component id="okButton" min="-2" pref="40" max="-2" attributes="0"/>

                  <EmptySpace pref="54" max="32767" attributes="0"/>

              </Group>

          </Group>

        </DimensionLayout>

      </Layout>

      <SubComponents>

        <Component class="javax.swing.JLabel" name="jLabel1">

          <Properties>

            <Property name="background" type="java.awt.Color" editor="org.netbeans.modules.form.RADConnectionPropertyEditor">

              <Connection component="jLabel1" name="background" type="property"/>

            </Property>

            <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

              <Font name="Tahoma" size="18" style="0"/>

            </Property>

            <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="ff" green="ff" red="ff" type="rgb"/>

            </Property>

            <Property name="horizontalAlignment" type="int" value="0"/>

            <Property name="text" type="java.lang.String" value="jLabel1"/>

          </Properties>

        </Component>

        <Component class="javax.swing.JButton" name="okButton">

          <Properties>

            <Property name="background" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="d2" green="76" red="19" type="rgb"/>

            </Property>

            <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

              <Font name="Tahoma" size="20" style="0"/>

            </Property>

            <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="ff" green="ff" red="ff" type="rgb"/>

            </Property>

            <Property name="text" type="java.lang.String" value="ok"/>

            <Property name="border" type="javax.swing.border.Border" editor="org.netbeans.modules.form.editors2.BorderEditor">

              <Border info="null"/>

            </Property>

            <Property name="horizontalTextPosition" type="int" value="0"/>

          </Properties>

          <Events>

            <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="okButtonActionPerformed"/>

          </Events>

          <AuxValues>

            <AuxValue name="JavaCodeGenerator\_InitCodePost" type="java.lang.String" value="okButton.setBackground (MaterialColors.BLUE\_800);&#xa;okButton.setForeground (Color.WHITE);&#xa;okButton.addMouseListener(MaterialUIMovement.getMovement(okButton, MaterialColors.INDIGO\_900));"/>

          </AuxValues>

        </Component>

      </SubComponents>

    </Container>

  </SubComponents>

</Form>

**Alert.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.awt.Color;

import java.awt.Dimension;

import java.awt.Toolkit;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

import mdlaf.MaterialLookAndFeel;

import mdlaf.animation.MaterialUIMovement;

import mdlaf.utils.MaterialColors;

*/\*\**

*\**

*\* @author Shehan*

*\*/*

public class Alert extends javax.swing.JFrame {

*/\*\**

*\* Creates new form Alert*

*\*/*

*//    public Alert() {*

*//        initComponents();*

*//    }*

    public Alert(String *message*) {

        try {

            UIManager.setLookAndFeel(new MaterialLookAndFeel());

        } catch (UnsupportedLookAndFeelException *e*) {

            e.printStackTrace();

        }

        initComponents();

        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();

        this.setLocation(dim.width / 2 - this.getSize().width / 2, dim.height / 2 - this.getSize().height / 2);

        jLabel1.setText(message);

        this.setVisible(true);

    }

*/\*\**

*\* 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();

        jLabel1 = new javax.swing.JLabel();

        okButton = new javax.swing.JButton();

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

        setAlwaysOnTop(true);

        setUndecorated(true);

        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        jPanel1.setBackground(MaterialColors.BLUE\_GRAY\_800);

        jLabel1.setBackground(jLabel1.getBackground());

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

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

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

        jLabel1.setText("jLabel1");

        okButton.setBackground(new java.awt.Color(25, 118, 210));

        okButton.setFont(new java.awt.Font("Tahoma", 0, 20)); *// NOI18N*

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

        okButton.setText("ok");

        okButton.setBorder(null);

        okButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        okButton.setBackground (MaterialColors.BLUE\_800);

        okButton.setForeground (Color.WHITE);

        okButton.addMouseListener(MaterialUIMovement.getMovement(okButton, MaterialColors.INDIGO\_900));

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

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

                okButtonActionPerformed(evt);

            }

        });

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

        jPanel1.setLayout(jPanel1Layout);

        jPanel1Layout.setHorizontalGroup(

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

            .addGroup(jPanel1Layout.createSequentialGroup()

                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 561, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

            .addGroup(jPanel1Layout.createSequentialGroup()

                .addGap(225, 225, 225)

                .addComponent(okButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 104, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

        );

        jPanel1Layout.setVerticalGroup(

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

            .addGroup(jPanel1Layout.createSequentialGroup()

                .addGap(66, 66, 66)

                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 95, javax.swing.GroupLayout.PREFERRED\_SIZE)

                .addGap(45, 45, 45)

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

                .addContainerGap(54, Short.MAX\_VALUE))

        );

        getContentPane().add(jPanel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 560, 300));

        pack();

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

    private void okButtonActionPerformed(java.awt.event.ActionEvent *evt*) {*//GEN-FIRST:event\_okButtonActionPerformed*

        this.dispose();

    }*//GEN-LAST:event\_okButtonActionPerformed*

*/\*\**

*\* @param args the command line arguments*

*\*/*

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

    private javax.swing.JLabel jLabel1;

    private javax.swing.JPanel jPanel1;

    private javax.swing.JButton okButton;

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

}

**Login.form**

<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.5" maxVersion="1.9" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">

  <Properties>

    <Property name="defaultCloseOperation" type="int" value="3"/>

    <Property name="undecorated" type="boolean" value="true"/>

    <Property name="resizable" type="boolean" value="false"/>

  </Properties>

  <SyntheticProperties>

    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>

    <SyntheticProperty name="generateCenter" type="boolean" value="false"/>

  </SyntheticProperties>

  <AuxValues>

    <AuxValue name="FormSettings\_autoResourcing" type="java.lang.Integer" value="0"/>

    <AuxValue name="FormSettings\_autoSetComponentName" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_generateFQN" type="java.lang.Boolean" value="true"/>

    <AuxValue name="FormSettings\_generateMnemonicsCode" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_i18nAutoMode" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_layoutCodeTarget" type="java.lang.Integer" value="1"/>

    <AuxValue name="FormSettings\_listenerGenerationStyle" type="java.lang.Integer" value="0"/>

    <AuxValue name="FormSettings\_variablesLocal" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_variablesModifier" type="java.lang.Integer" value="2"/>

    <AuxValue name="designerSize" type="java.awt.Dimension" value="-84,-19,0,5,115,114,0,18,106,97,118,97,46,97,119,116,46,68,105,109,101,110,115,105,111,110,65,-114,-39,-41,-84,95,68,20,2,0,2,73,0,6,104,101,105,103,104,116,73,0,5,119,105,100,116,104,120,112,0,0,2,78,0,0,1,-62"/>

  </AuxValues>

  <Layout class="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout">

    <Property name="useNullLayout" type="boolean" value="false"/>

  </Layout>

  <SubComponents>

    <Container class="javax.swing.JPanel" name="jPanel5">

      <Properties>

        <Property name="background" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

          <Color blue="30" green="30" red="30" type="rgb"/>

        </Property>

      </Properties>

      <AuxValues>

        <AuxValue name="JavaCodeGenerator\_InitCodePost" type="java.lang.String" value="jPanel5.setBackground(MaterialColors.BLUE\_GRAY\_900);"/>

      </AuxValues>

      <Constraints>

        <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

          <AbsoluteConstraints x="0" y="0" width="450" height="590"/>

        </Constraint>

      </Constraints>

      <Layout class="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout">

        <Property name="useNullLayout" type="boolean" value="false"/>

      </Layout>

      <SubComponents>

        <Component class="javax.swing.JTextField" name="usernameInput">

          <Properties>

            <Property name="background" type="java.awt.Color" editor="org.netbeans.modules.form.RADConnectionPropertyEditor">

              <Connection component="jPanel5" name="background" type="property"/>

            </Property>

            <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

              <Font name="Tahoma" size="18" style="0"/>

            </Property>

            <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="ff" green="ff" red="ff" type="rgb"/>

            </Property>

            <Property name="toolTipText" type="java.lang.String" value=""/>

            <Property name="border" type="javax.swing.border.Border" editor="org.netbeans.modules.form.editors2.BorderEditor">

              <Border info="null"/>

            </Property>

            <Property name="cursor" type="java.awt.Cursor" editor="org.netbeans.modules.form.editors2.CursorEditor">

              <Color id="Text Cursor"/>

            </Property>

            <Property name="disabledTextColor" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="cc" green="cc" red="cc" type="rgb"/>

            </Property>

            <Property name="selectedTextColor" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="cc" green="cc" red="cc" type="rgb"/>

            </Property>

          </Properties>

          <Events>

            <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="usernameInputActionPerformed"/>

          </Events>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="140" y="222" width="210" height="22"/>

            </Constraint>

          </Constraints>

        </Component>

        <Component class="javax.swing.JPasswordField" name="passwordInput">

          <Properties>

            <Property name="background" type="java.awt.Color" editor="org.netbeans.modules.form.RADConnectionPropertyEditor">

              <Connection component="jPanel5" name="background" type="property"/>

            </Property>

            <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

              <Font name="Tahoma" size="18" style="0"/>

            </Property>

            <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="ff" green="ff" red="ff" type="rgb"/>

            </Property>

            <Property name="border" type="javax.swing.border.Border" editor="org.netbeans.modules.form.editors2.BorderEditor">

              <Border info="null"/>

            </Property>

            <Property name="disabledTextColor" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="cc" green="cc" red="cc" type="rgb"/>

            </Property>

            <Property name="selectedTextColor" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="cc" green="cc" red="cc" type="rgb"/>

            </Property>

          </Properties>

          <Events>

            <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="passwordInputActionPerformed"/>

          </Events>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="140" y="310" width="209" height="20"/>

            </Constraint>

          </Constraints>

        </Component>

        <Component class="java.awt.Label" name="label2">

          <Properties>

            <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

              <Font name="Tahoma" size="36" style="1"/>

            </Property>

            <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="ff" green="ff" red="ff" type="rgb"/>

            </Property>

            <Property name="text" type="java.lang.String" value="Sign In"/>

          </Properties>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="69" y="69" width="135" height="48"/>

            </Constraint>

          </Constraints>

        </Component>

        <Container class="javax.swing.JPanel" name="jPanel6">

          <Properties>

            <Property name="background" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor" postCode="jPanel6.setBackground(MaterialColors.LIGHT\_BLUE\_400);">

              <Color blue="f6" green="b5" red="64" type="rgb"/>

            </Property>

            <Property name="minimumSize" type="java.awt.Dimension" editor="org.netbeans.beaninfo.editors.DimensionEditor">

              <Dimension value="[135, 48]"/>

            </Property>

            <Property name="preferredSize" type="java.awt.Dimension" editor="org.netbeans.beaninfo.editors.DimensionEditor">

              <Dimension value="[135, 48]"/>

            </Property>

          </Properties>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="70" y="120" width="136" height="10"/>

            </Constraint>

          </Constraints>

          <Layout>

            <DimensionLayout dim="0">

              <Group type="103" groupAlignment="0" attributes="0">

                  <EmptySpace min="0" pref="136" max="32767" attributes="0"/>

              </Group>

            </DimensionLayout>

            <DimensionLayout dim="1">

              <Group type="103" groupAlignment="0" attributes="0">

                  <EmptySpace min="0" pref="48" max="32767" attributes="0"/>

              </Group>

            </DimensionLayout>

          </Layout>

        </Container>

        <Component class="javax.swing.JSeparator" name="jSeparator1">

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="140" y="330" width="210" height="20"/>

            </Constraint>

          </Constraints>

        </Component>

        <Component class="javax.swing.JSeparator" name="jSeparator2">

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="140" y="245" width="210" height="20"/>

            </Constraint>

          </Constraints>

        </Component>

        <Component class="javax.swing.JLabel" name="jLabel2">

          <Properties>

            <Property name="horizontalAlignment" type="int" value="0"/>

            <Property name="icon" type="javax.swing.Icon" editor="org.netbeans.modules.form.editors2.IconEditor">

              <Image iconType="3" name="/password\_icon.png"/>

            </Property>

          </Properties>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="80" y="300" width="52" height="40"/>

            </Constraint>

          </Constraints>

        </Component>

        <Component class="javax.swing.JButton" name="signInButton">

          <Properties>

            <Property name="background" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="d2" green="76" red="19" type="rgb"/>

            </Property>

            <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

              <Font name="Tahoma" size="20" style="0"/>

            </Property>

            <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="ff" green="ff" red="ff" type="rgb"/>

            </Property>

            <Property name="text" type="java.lang.String" value="Sign In"/>

            <Property name="border" type="javax.swing.border.Border" editor="org.netbeans.modules.form.editors2.BorderEditor">

              <Border info="null"/>

            </Property>

            <Property name="horizontalTextPosition" type="int" value="0"/>

          </Properties>

          <Events>

            <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="signInButtonActionPerformed"/>

          </Events>

          <AuxValues>

            <AuxValue name="JavaCodeGenerator\_InitCodePost" type="java.lang.String" value="signInButton.setBackground (MaterialColors.BLUE\_800);&#xa;signInButton.setForeground (Color.WHITE);&#xa;signInButton.addMouseListener(MaterialUIMovement.getMovement(signInButton, MaterialColors.INDIGO\_900));"/>

          </AuxValues>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="80" y="420" width="290" height="40"/>

            </Constraint>

          </Constraints>

        </Component>

        <Component class="javax.swing.JLabel" name="jLabel3">

          <Properties>

            <Property name="icon" type="javax.swing.Icon" editor="org.netbeans.modules.form.editors2.IconEditor">

              <Image iconType="3" name="/username\_icon.png"/>

            </Property>

          </Properties>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="80" y="210" width="-1" height="50"/>

            </Constraint>

          </Constraints>

        </Component>

        <Component class="javax.swing.JButton" name="closeButton">

          <Properties>

            <Property name="background" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="30" green="30" red="30" type="rgb"/>

            </Property>

            <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="30" green="30" red="30" type="rgb"/>

            </Property>

            <Property name="icon" type="javax.swing.Icon" editor="org.netbeans.modules.form.editors2.IconEditor">

              <Image iconType="3" name="/cross\_icon.png"/>

            </Property>

            <Property name="toolTipText" type="java.lang.String" value=""/>

            <Property name="border" type="javax.swing.border.Border" editor="org.netbeans.modules.form.editors2.BorderEditor">

              <Border info="null"/>

            </Property>

            <Property name="borderPainted" type="boolean" value="false"/>

            <Property name="contentAreaFilled" type="boolean" value="false"/>

          </Properties>

          <Events>

            <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="closeButtonActionPerformed"/>

          </Events>

          <AuxValues>

            <AuxValue name="JavaCodeGenerator\_InitCodePost" type="java.lang.String" value="closeButton.setBackground (MaterialColors.BLUE\_GRAY\_900);&#xa;&#xa;closeButton.addMouseListener(MaterialUIMovement.getMovement(closeButton, MaterialColors.GRAY\_900));"/>

          </AuxValues>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="410" y="0" width="40" height="-1"/>

            </Constraint>

          </Constraints>

        </Component>

        <Component class="javax.swing.JLabel" name="jLabel1">

          <Properties>

            <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

              <Font name="Tahoma" size="14" style="0"/>

            </Property>

            <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="ff" green="ff" red="ff" type="rgb"/>

            </Property>

            <Property name="horizontalAlignment" type="int" value="0"/>

          </Properties>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="80" y="480" width="290" height="20"/>

            </Constraint>

          </Constraints>

        </Component>

      </SubComponents>

    </Container>

  </SubComponents>

</Form>

**Login.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 Controllers.Client;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Toolkit;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

import mdlaf.MaterialLookAndFeel;

import mdlaf.animation.MaterialUIMovement;

import mdlaf.utils.MaterialColors;

*/\*\**

*\**

*\* @author Shehan*

*\*/*

public class Login extends javax.swing.JFrame implements Runnable {

    public static boolean loggedIn = false;

*/\*\**

*\* Creates new form Login*

*\*/*

    public Login() {

        initComponents();

        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();

        this.setLocation(dim.width / 2 - this.getSize().width / 2, dim.height / 2 - this.getSize().height / 2);

        signInButton.setEnabled(false);

        Thread t1 = new Thread(this);

        t1.start();

    }

    @Override

    public void run() {

        Client tclient = new Client("");

        while (true) {

            jLabel1.setText("trying to reach server...");

            if (!tclient.checkAuthenticationServer()) {

                signInButton.setEnabled(false);

                jLabel1.setText("Please check your connection.");

                try {   *//Try again in 5s*

                    Thread.sleep(5000);

                } catch (InterruptedException *ex*) {

                    ex.printStackTrace();

                }

            } else {

                jLabel1.setText("");

                signInButton.setEnabled(true);

                 try {   *//sleep for 30s*

                    Thread.sleep(30000);

                } catch (InterruptedException *ex*) {

                    ex.printStackTrace();

                }

            }

        }

    }

*/\*\**

*\* 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() {

        jPanel5 = new javax.swing.JPanel();

        usernameInput = new javax.swing.JTextField();

        passwordInput = new javax.swing.JPasswordField();

        label2 = new java.awt.Label();

        jPanel6 = new javax.swing.JPanel();

        jSeparator1 = new javax.swing.JSeparator();

        jSeparator2 = new javax.swing.JSeparator();

        jLabel2 = new javax.swing.JLabel();

        signInButton = new javax.swing.JButton();

        jLabel3 = new javax.swing.JLabel();

        closeButton = new javax.swing.JButton();

        jLabel1 = new javax.swing.JLabel();

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

        setUndecorated(true);

        setResizable(false);

        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        jPanel5.setBackground(new java.awt.Color(48, 48, 48));

        jPanel5.setBackground(MaterialColors.BLUE\_GRAY\_900);

        jPanel5.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        usernameInput.setBackground(jPanel5.getBackground());

        usernameInput.setFont(new java.awt.Font("Tahoma", 0, 18)); *// NOI18N*

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

        usernameInput.setToolTipText("");

        usernameInput.setBorder(null);

        usernameInput.setCursor(new java.awt.Cursor(java.awt.Cursor.TEXT\_CURSOR));

        usernameInput.setDisabledTextColor(new java.awt.Color(204, 204, 204));

        usernameInput.setSelectedTextColor(new java.awt.Color(204, 204, 204));

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

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

                usernameInputActionPerformed(evt);

            }

        });

        jPanel5.add(usernameInput, new org.netbeans.lib.awtextra.AbsoluteConstraints(140, 222, 210, 22));

        passwordInput.setBackground(jPanel5.getBackground());

        passwordInput.setFont(new java.awt.Font("Tahoma", 0, 18)); *// NOI18N*

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

        passwordInput.setBorder(null);

        passwordInput.setDisabledTextColor(new java.awt.Color(204, 204, 204));

        passwordInput.setSelectedTextColor(new java.awt.Color(204, 204, 204));

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

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

                passwordInputActionPerformed(evt);

            }

        });

        jPanel5.add(passwordInput, new org.netbeans.lib.awtextra.AbsoluteConstraints(140, 310, 209, 20));

        label2.setFont(new java.awt.Font("Tahoma", 1, 36)); *// NOI18N*

        label2.setForeground(new java.awt.Color(255, 255, 255));

        label2.setText("Sign In");

        jPanel5.add(label2, new org.netbeans.lib.awtextra.AbsoluteConstraints(69, 69, 135, 48));

        jPanel6.setBackground(new java.awt.Color(100, 181, 246));

        jPanel6.setBackground(MaterialColors.LIGHT\_BLUE\_400);

        jPanel6.setMinimumSize(new java.awt.Dimension(135, 48));

        jPanel6.setPreferredSize(new java.awt.Dimension(135, 48));

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

        jPanel6.setLayout(jPanel6Layout);

        jPanel6Layout.setHorizontalGroup(

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

            .addGap(0, 136, Short.MAX\_VALUE)

        );

        jPanel6Layout.setVerticalGroup(

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

            .addGap(0, 48, Short.MAX\_VALUE)

        );

        jPanel5.add(jPanel6, new org.netbeans.lib.awtextra.AbsoluteConstraints(70, 120, 136, 10));

        jPanel5.add(jSeparator1, new org.netbeans.lib.awtextra.AbsoluteConstraints(140, 330, 210, 20));

        jPanel5.add(jSeparator2, new org.netbeans.lib.awtextra.AbsoluteConstraints(140, 245, 210, 20));

        jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

        jLabel2.setIcon(new javax.swing.ImageIcon(getClass().getResource("/password\_icon.png"))); *// NOI18N*

        jPanel5.add(jLabel2, new org.netbeans.lib.awtextra.AbsoluteConstraints(80, 300, 52, 40));

        signInButton.setBackground(new java.awt.Color(25, 118, 210));

        signInButton.setFont(new java.awt.Font("Tahoma", 0, 20)); *// NOI18N*

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

        signInButton.setText("Sign In");

        signInButton.setBorder(null);

        signInButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        signInButton.setBackground (MaterialColors.BLUE\_800);

        signInButton.setForeground (Color.WHITE);

        signInButton.addMouseListener(MaterialUIMovement.getMovement(signInButton, MaterialColors.INDIGO\_900));

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

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

                signInButtonActionPerformed(evt);

            }

        });

        jPanel5.add(signInButton, new org.netbeans.lib.awtextra.AbsoluteConstraints(80, 420, 290, 40));

        jLabel3.setIcon(new javax.swing.ImageIcon(getClass().getResource("/username\_icon.png"))); *// NOI18N*

        jPanel5.add(jLabel3, new org.netbeans.lib.awtextra.AbsoluteConstraints(80, 210, -1, 50));

        closeButton.setBackground(new java.awt.Color(48, 48, 48));

        closeButton.setForeground(new java.awt.Color(48, 48, 48));

        closeButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/cross\_icon.png"))); *// NOI18N*

        closeButton.setToolTipText("");

        closeButton.setBorder(null);

        closeButton.setBorderPainted(false);

        closeButton.setContentAreaFilled(false);

        closeButton.setBackground (MaterialColors.BLUE\_GRAY\_900);

        closeButton.addMouseListener(MaterialUIMovement.getMovement(closeButton, MaterialColors.GRAY\_900));

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

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

                closeButtonActionPerformed(evt);

            }

        });

        jPanel5.add(closeButton, new org.netbeans.lib.awtextra.AbsoluteConstraints(410, 0, 40, -1));

        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 14)); *// NOI18N*

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

        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

        jPanel5.add(jLabel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(80, 480, 290, 20));

        getContentPane().add(jPanel5, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 450, 590));

        pack();

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

    private void passwordInputActionPerformed(java.awt.event.ActionEvent *evt*) {*//GEN-FIRST:event\_passwordInputActionPerformed*

        signInButtonActionPerformed(evt);

    }*//GEN-LAST:event\_passwordInputActionPerformed*

    private void closeButtonActionPerformed(java.awt.event.ActionEvent *evt*) {*//GEN-FIRST:event\_closeButtonActionPerformed*

        System.exit(0);

    }*//GEN-LAST:event\_closeButtonActionPerformed*

    private void usernameInputActionPerformed(java.awt.event.ActionEvent *evt*) {*//GEN-FIRST:event\_usernameInputActionPerformed*

        signInButtonActionPerformed(evt);

    }*//GEN-LAST:event\_usernameInputActionPerformed*

    private void signInButtonActionPerformed(java.awt.event.ActionEvent *evt*) {*//GEN-FIRST:event\_signInButtonActionPerformed*

        String username = usernameInput.getText();

        String password = new String(passwordInput.getPassword());

        Client client = new Client(username);

        if (client.login(username, password)) {

            for (float i = 1.0f; i > 0.0f; i -= 0.2f) { *//Fade out effect*

                this.setOpacity(i);

                try {

                    Thread.sleep(30);

                } catch (Exception *ignored*) {

                }

            }

            loggedIn = true;

            this.setVisible(false);

            MainPage mainPage = new MainPage(username);

            mainPage.setVisible(true);

        } else {

            loggedIn = false;

            Alert alert = new Alert("Incorrect username/Password");

        }

    }*//GEN-LAST:event\_signInButtonActionPerformed*

*/\*\**

*\* @param args the command line arguments*

*\*/*

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

        try {

            UIManager.setLookAndFeel(new MaterialLookAndFeel());

        } catch (UnsupportedLookAndFeelException *e*) {

            e.printStackTrace();

        }

*/\* Create and display the form \*/*

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

            public void run() {

                new Login().setVisible(true);

            }

        });

    }

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

    private javax.swing.JButton closeButton;

    private javax.swing.JLabel jLabel1;

    private javax.swing.JLabel jLabel2;

    private javax.swing.JLabel jLabel3;

    private javax.swing.JPanel jPanel5;

    private javax.swing.JPanel jPanel6;

    private javax.swing.JSeparator jSeparator1;

    private javax.swing.JSeparator jSeparator2;

    private java.awt.Label label2;

    private javax.swing.JPasswordField passwordInput;

    private javax.swing.JButton signInButton;

    private javax.swing.JTextField usernameInput;

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

}

**MainPage.form**

<?xml version="1.0" encoding="UTF-8" ?>

<Form version="1.6" maxVersion="1.9" type="org.netbeans.modules.form.forminfo.JFrameFormInfo">

  <Properties>

    <Property name="defaultCloseOperation" type="int" value="3"/>

    <Property name="undecorated" type="boolean" value="true"/>

    <Property name="resizable" type="boolean" value="false"/>

  </Properties>

  <SyntheticProperties>

    <SyntheticProperty name="formSizePolicy" type="int" value="1"/>

    <SyntheticProperty name="generateCenter" type="boolean" value="false"/>

  </SyntheticProperties>

  <AuxValues>

    <AuxValue name="FormSettings\_autoResourcing" type="java.lang.Integer" value="0"/>

    <AuxValue name="FormSettings\_autoSetComponentName" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_generateFQN" type="java.lang.Boolean" value="true"/>

    <AuxValue name="FormSettings\_generateMnemonicsCode" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_i18nAutoMode" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_layoutCodeTarget" type="java.lang.Integer" value="1"/>

    <AuxValue name="FormSettings\_listenerGenerationStyle" type="java.lang.Integer" value="0"/>

    <AuxValue name="FormSettings\_variablesLocal" type="java.lang.Boolean" value="false"/>

    <AuxValue name="FormSettings\_variablesModifier" type="java.lang.Integer" value="2"/>

    <AuxValue name="designerSize" type="java.awt.Dimension" value="-84,-19,0,5,115,114,0,18,106,97,118,97,46,97,119,116,46,68,105,109,101,110,115,105,111,110,65,-114,-39,-41,-84,95,68,20,2,0,2,73,0,6,104,101,105,103,104,116,73,0,5,119,105,100,116,104,120,112,0,0,1,-47,0,0,3,65"/>

  </AuxValues>

  <Layout class="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout">

    <Property name="useNullLayout" type="boolean" value="false"/>

  </Layout>

  <SubComponents>

    <Container class="javax.swing.JPanel" name="jPanel1">

      <AuxValues>

        <AuxValue name="JavaCodeGenerator\_InitCodePre" type="java.lang.String" value="jPanel1.setBackground(MaterialColors.BLUE\_GRAY\_900);"/>

      </AuxValues>

      <Constraints>

        <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

          <AbsoluteConstraints x="0" y="0" width="-1" height="-1"/>

        </Constraint>

      </Constraints>

      <Layout class="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout">

        <Property name="useNullLayout" type="boolean" value="false"/>

      </Layout>

      <SubComponents>

        <Component class="javax.swing.JButton" name="closeButton">

          <Properties>

            <Property name="background" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="30" green="30" red="30" type="rgb"/>

            </Property>

            <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="30" green="30" red="30" type="rgb"/>

            </Property>

            <Property name="icon" type="javax.swing.Icon" editor="org.netbeans.modules.form.editors2.IconEditor">

              <Image iconType="3" name="/cross\_icon.png"/>

            </Property>

            <Property name="toolTipText" type="java.lang.String" value=""/>

            <Property name="border" type="javax.swing.border.Border" editor="org.netbeans.modules.form.editors2.BorderEditor">

              <Border info="null"/>

            </Property>

            <Property name="borderPainted" type="boolean" value="false"/>

            <Property name="contentAreaFilled" type="boolean" value="false"/>

          </Properties>

          <Events>

            <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="closeButtonActionPerformed"/>

          </Events>

          <AuxValues>

            <AuxValue name="JavaCodeGenerator\_InitCodePost" type="java.lang.String" value="closeButton.setBackground (MaterialColors.BLUE\_GRAY\_900);&#xa;&#xa;closeButton.addMouseListener(MaterialUIMovement.getMovement(closeButton, MaterialColors.GRAY\_900));"/>

          </AuxValues>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="795" y="0" width="40" height="-1"/>

            </Constraint>

          </Constraints>

        </Component>

        <Container class="javax.swing.JTabbedPane" name="jTabbedPane1">

          <Properties>

            <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

              <Color blue="ff" green="ff" red="ff" type="rgb"/>

            </Property>

          </Properties>

          <AuxValues>

            <AuxValue name="JavaCodeGenerator\_InitCodePost" type="java.lang.String" value="MaterialTabbedPaneUI mtpUI = (MaterialTabbedPaneUI)MaterialTabbedPaneUI.createUI(jTabbedPane1);&#xa;UIManager.put(&quot;TabbedPane.selectionForeground&quot;,MaterialColors.CYAN\_A100);&#xa;UIManager.put(&quot;TabbedPane[focus].colorLine&quot;,MaterialColors.AMBER\_200);&#xa;UIManager.put(&quot;TabbedPane.foreground&quot;,MaterialColors.WHITE);&#xa;UIManager.put(&quot;TabbedPane.border&quot;,MaterialColors.GREEN\_100);&#xa;&#xa;mtpUI.installUI(jTabbedPane1);&#xa;&#xa;&#xa;&#xa;jTabbedPane1.setUI(mtpUI);&#xa;jTabbedPane1.setBackground(MaterialColors.BLUE\_GRAY\_900);"/>

          </AuxValues>

          <Constraints>

            <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout" value="org.netbeans.modules.form.compat2.layouts.DesignAbsoluteLayout$AbsoluteConstraintsDescription">

              <AbsoluteConstraints x="0" y="0" width="830" height="470"/>

            </Constraint>

          </Constraints>

          <Layout class="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLayout"/>

          <SubComponents>

            <Container class="javax.swing.JPanel" name="manageSensor">

              <AuxValues>

                <AuxValue name="JavaCodeGenerator\_InitCodePre" type="java.lang.String" value="manageSensor.setBackground(MaterialColors.BLUE\_GRAY\_900);"/>

              </AuxValues>

              <Constraints>

                <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLayout" value="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLayout$JTabbedPaneConstraintsDescription">

                  <JTabbedPaneConstraints tabName="Manage  Sensors">

                    <Property name="tabTitle" type="java.lang.String" value="Manage  Sensors"/>

                  </JTabbedPaneConstraints>

                </Constraint>

              </Constraints>

              <Layout>

                <DimensionLayout dim="0">

                  <Group type="103" groupAlignment="0" attributes="0">

                      <Group type="102" attributes="0">

                          <EmptySpace max="-2" attributes="0"/>

                          <Group type="103" groupAlignment="0" attributes="0">

                              <Group type="102" attributes="0">

                                  <Group type="103" groupAlignment="0" max="-2" attributes="0">

                                      <Group type="102" alignment="0" attributes="0">

                                          <EmptySpace min="1" pref="1" max="-2" attributes="0"/>

                                          <Component id="jPanel7" max="32767" attributes="0"/>

                                      </Group>

                                      <Component id="label3" alignment="0" min="-2" max="-2" attributes="0"/>

                                  </Group>

                                  <EmptySpace max="32767" attributes="0"/>

                                  <Component id="jLabel3" min="-2" pref="213" max="-2" attributes="0"/>

                              </Group>

                              <Group type="102" attributes="0">

                                  <Component id="jScrollPane2" pref="568" max="32767" attributes="0"/>

                                  <EmptySpace type="separate" max="-2" attributes="0"/>

                                  <Component id="deleteSensorButton" min="-2" pref="182" max="-2" attributes="0"/>

                                  <EmptySpace min="-2" pref="45" max="-2" attributes="0"/>

                              </Group>

                              <Group type="102" alignment="1" attributes="0">

                                  <Component id="tableMessage" min="-2" pref="470" max="-2" attributes="0"/>

                                  <EmptySpace pref="46" max="32767" attributes="0"/>

                                  <Component id="jLabel4" min="-2" max="-2" attributes="0"/>

                                  <EmptySpace max="-2" attributes="0"/>

                                  <Component id="jComboBox1" min="-2" pref="175" max="-2" attributes="0"/>

                              </Group>

                          </Group>

                      </Group>

                  </Group>

                </DimensionLayout>

                <DimensionLayout dim="1">

                  <Group type="103" groupAlignment="0" attributes="0">

                      <Group type="102" alignment="0" attributes="0">

                          <EmptySpace min="-2" pref="19" max="-2" attributes="0"/>

                          <Group type="103" groupAlignment="1" attributes="0">

                              <Group type="102" attributes="0">

                                  <Component id="label3" min="-2" max="-2" attributes="0"/>

                                  <EmptySpace min="3" pref="3" max="-2" attributes="0"/>

                                  <Component id="jPanel7" min="-2" pref="10" max="-2" attributes="0"/>

                              </Group>

                              <Component id="jLabel3" min="-2" max="-2" attributes="0"/>

                          </Group>

                          <Group type="103" groupAlignment="0" attributes="0">

                              <Group type="102" alignment="0" attributes="0">

                                  <EmptySpace type="unrelated" max="-2" attributes="0"/>

                                  <Group type="103" groupAlignment="3" attributes="0">

                                      <Component id="jLabel4" alignment="3" min="-2" max="-2" attributes="0"/>

                                      <Component id="jComboBox1" alignment="3" min="-2" max="-2" attributes="0"/>

                                  </Group>

                                  <EmptySpace min="-2" pref="199" max="-2" attributes="0"/>

                                  <Component id="deleteSensorButton" min="-2" pref="40" max="-2" attributes="0"/>

                              </Group>

                              <Group type="102" alignment="0" attributes="0">

                                  <EmptySpace min="-2" pref="30" max="-2" attributes="0"/>

                                  <Component id="tableMessage" min="-2" pref="17" max="-2" attributes="0"/>

                                  <EmptySpace max="-2" attributes="0"/>

                                  <Component id="jScrollPane2" min="-2" pref="304" max="-2" attributes="0"/>

                              </Group>

                          </Group>

                          <EmptySpace max="32767" attributes="0"/>

                      </Group>

                  </Group>

                </DimensionLayout>

              </Layout>

              <SubComponents>

                <Component class="java.awt.Label" name="label3">

                  <Properties>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="24" style="1"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="text" type="java.lang.String" value="Manage Sensors"/>

                  </Properties>

                </Component>

                <Container class="javax.swing.JPanel" name="jPanel7">

                  <Properties>

                    <Property name="background" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="f6" green="b5" red="64" type="rgb"/>

                    </Property>

                    <Property name="minimumSize" type="java.awt.Dimension" editor="org.netbeans.beaninfo.editors.DimensionEditor">

                      <Dimension value="[135, 48]"/>

                    </Property>

                  </Properties>

                  <Layout>

                    <DimensionLayout dim="0">

                      <Group type="103" groupAlignment="0" attributes="0">

                          <EmptySpace min="0" pref="0" max="32767" attributes="0"/>

                      </Group>

                    </DimensionLayout>

                    <DimensionLayout dim="1">

                      <Group type="103" groupAlignment="0" attributes="0">

                          <EmptySpace min="0" pref="0" max="32767" attributes="0"/>

                      </Group>

                    </DimensionLayout>

                  </Layout>

                </Container>

                <Component class="javax.swing.JLabel" name="jLabel3">

                  <Properties>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="14" style="0"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="text" type="java.lang.String" value="Select a floor to view it&apos;s sensors"/>

                  </Properties>

                </Component>

                <Component class="javax.swing.JComboBox" name="jComboBox1">

                  <Properties>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor" postCode="mtcbUI.installUI(jComboBox1);">

                      <Font name="Tahoma" size="18" style="0"/>

                    </Property>

                    <Property name="model" type="javax.swing.ComboBoxModel" editor="org.netbeans.modules.form.editors2.ComboBoxModelEditor">

                      <StringArray count="0"/>

                    </Property>

                  </Properties>

                  <Events>

                    <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="jComboBox1ActionPerformed"/>

                  </Events>

                  <AuxValues>

                    <AuxValue name="JavaCodeGenerator\_InitCodePre" type="java.lang.String" value="MaterialComboBoxUI mtcbUI = (MaterialComboBoxUI)MaterialComboBoxUI.createUI(jComboBox1);"/>

                    <AuxValue name="JavaCodeGenerator\_TypeParameters" type="java.lang.String" value="&lt;String&gt;"/>

                  </AuxValues>

                </Component>

                <Component class="javax.swing.JButton" name="deleteSensorButton">

                  <Properties>

                    <Property name="background" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="d2" green="76" red="19" type="rgb"/>

                    </Property>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="20" style="0"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="text" type="java.lang.String" value="Delete Sensor"/>

                    <Property name="border" type="javax.swing.border.Border" editor="org.netbeans.modules.form.editors2.BorderEditor">

                      <Border info="null"/>

                    </Property>

                    <Property name="horizontalTextPosition" type="int" value="0"/>

                  </Properties>

                  <Events>

                    <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="deleteSensorButtonActionPerformed"/>

                  </Events>

                  <AuxValues>

                    <AuxValue name="JavaCodeGenerator\_InitCodePost" type="java.lang.String" value="deleteSensorButton.setBackground (MaterialColors.BLUE\_800);&#xa;deleteSensorButton.setForeground (Color.WHITE);&#xa;deleteSensorButton.addMouseListener(MaterialUIMovement.getMovement(deleteSensorButton, MaterialColors.INDIGO\_900));"/>

                  </AuxValues>

                </Component>

                <Component class="javax.swing.JLabel" name="jLabel4">

                  <Properties>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="18" style="0"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="text" type="java.lang.String" value="Floor Number:"/>

                  </Properties>

                </Component>

                <Container class="javax.swing.JScrollPane" name="jScrollPane2">

                  <AuxValues>

                    <AuxValue name="autoScrollPane" type="java.lang.Boolean" value="true"/>

                  </AuxValues>

                  <Layout class="org.netbeans.modules.form.compat2.layouts.support.JScrollPaneSupportLayout"/>

                  <SubComponents>

                    <Component class="javax.swing.JTable" name="jTable1">

                      <Properties>

                        <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                          <Color blue="ff" green="ff" red="ff" type="rgb"/>

                        </Property>

                        <Property name="model" type="javax.swing.table.TableModel" editor="org.netbeans.modules.form.editors2.TableModelEditor">

                          <Table columnCount="4" rowCount="4">

                            <Column editable="false" title="Floor Number" type="java.lang.Integer"/>

                            <Column editable="false" title="Room Number" type="java.lang.Integer"/>

                            <Column editable="false" title="Active" type="java.lang.String"/>

                            <Column editable="false" title="Type" type="java.lang.String"/>

                          </Table>

                        </Property>

                        <Property name="autoscrolls" type="boolean" value="false"/>

                        <Property name="columnModel" type="javax.swing.table.TableColumnModel" editor="org.netbeans.modules.form.editors2.TableColumnModelEditor">

                          <TableColumnModel selectionModel="0">

                            <Column maxWidth="-1" minWidth="-1" prefWidth="-1" resizable="true">

                              <Title/>

                              <Editor/>

                              <Renderer/>

                            </Column>

                            <Column maxWidth="-1" minWidth="-1" prefWidth="-1" resizable="true">

                              <Title/>

                              <Editor/>

                              <Renderer/>

                            </Column>

                            <Column maxWidth="-1" minWidth="-1" prefWidth="-1" resizable="true">

                              <Title/>

                              <Editor/>

                              <Renderer/>

                            </Column>

                            <Column maxWidth="-1" minWidth="-1" prefWidth="-1" resizable="true">

                              <Title/>

                              <Editor/>

                              <Renderer/>

                            </Column>

                          </TableColumnModel>

                        </Property>

                        <Property name="selectionModel" type="javax.swing.ListSelectionModel" editor="org.netbeans.modules.form.editors2.JTableSelectionModelEditor">

                          <JTableSelectionModel selectionMode="0"/>

                        </Property>

                        <Property name="tableHeader" type="javax.swing.table.JTableHeader" editor="org.netbeans.modules.form.editors2.JTableHeaderEditor">

                          <TableHeader reorderingAllowed="true" resizingAllowed="true"/>

                        </Property>

                      </Properties>

                      <AuxValues>

                        <AuxValue name="JavaCodeGenerator\_InitCodePre" type="java.lang.String" value="MaterialTableUI mtUI = (MaterialTableUI)MaterialTableUI.createUI(jTable1);&#xa;&#xa;UIManager.put(&quot;Table.background&quot;, MaterialColors.BLUE\_GRAY\_700);&#xa;UIManager.put(&quot;Table.alternateRowColor&quot;,  MaterialColors.BLUE\_GRAY\_700);&#xa;UIManager.put(&quot;Table.foreground&quot;, MaterialColors.WHITE);&#xa;UIManager.put(&quot;Table.selectionForeground&quot;, MaterialColors.WHITE);&#xa;UIManager.put(&quot;Table.selectionBackground&quot;, MaterialColors.PURPLE\_300);&#xa;&#xa;UIManager.put(&quot;Table.showHorizontalLines&quot;, true);&#xa;UIManager.put(&quot;Table.showVerticalLines&quot;, true);&#xa;UIManager.put(&quot;Table.gridColor&quot;, MaterialColors.WHITE);&#xa;&#xa;UIManager.put(&quot;Table.font&quot;, (new java.awt.Font(&quot;Tahoma&quot;, 0, 16))); &#xa;&#xa;mtUI.installUI(jTable1);"/>

                      </AuxValues>

                    </Component>

                  </SubComponents>

                </Container>

                <Component class="javax.swing.JLabel" name="tableMessage">

                  <Properties>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="14" style="0"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="maximumSize" type="java.awt.Dimension" editor="org.netbeans.beaninfo.editors.DimensionEditor">

                      <Dimension value="[470, 17]"/>

                    </Property>

                    <Property name="minimumSize" type="java.awt.Dimension" editor="org.netbeans.beaninfo.editors.DimensionEditor">

                      <Dimension value="[470, 17]"/>

                    </Property>

                  </Properties>

                </Component>

              </SubComponents>

            </Container>

            <Container class="javax.swing.JPanel" name="addSensor">

              <AuxValues>

                <AuxValue name="JavaCodeGenerator\_InitCodePre" type="java.lang.String" value="addSensor.setBackground(MaterialColors.BLUE\_GRAY\_900);"/>

              </AuxValues>

              <Constraints>

                <Constraint layoutClass="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLayout" value="org.netbeans.modules.form.compat2.layouts.support.JTabbedPaneSupportLayout$JTabbedPaneConstraintsDescription">

                  <JTabbedPaneConstraints tabName="Add Sensor">

                    <Property name="tabTitle" type="java.lang.String" value="Add Sensor"/>

                  </JTabbedPaneConstraints>

                </Constraint>

              </Constraints>

              <Layout>

                <DimensionLayout dim="0">

                  <Group type="103" groupAlignment="0" attributes="0">

                      <Group type="102" attributes="0">

                          <EmptySpace max="-2" attributes="0"/>

                          <Group type="103" groupAlignment="0" max="-2" attributes="0">

                              <Group type="102" attributes="0">

                                  <EmptySpace min="1" pref="1" max="-2" attributes="0"/>

                                  <Component id="jPanel6" max="32767" attributes="0"/>

                              </Group>

                              <Component id="label2" min="-2" max="-2" attributes="0"/>

                          </Group>

                          <EmptySpace max="32767" attributes="0"/>

                      </Group>

                      <Group type="102" alignment="1" attributes="0">

                          <EmptySpace min="0" pref="240" max="32767" attributes="0"/>

                          <Group type="103" groupAlignment="0" attributes="0">

                              <Component id="jLabel1" min="-2" max="-2" attributes="0"/>

                              <Component id="jLabel2" min="-2" max="-2" attributes="0"/>

                              <Component id="jLabel5" alignment="1" min="-2" max="-2" attributes="0"/>

                          </Group>

                          <EmptySpace type="unrelated" max="-2" attributes="0"/>

                          <Group type="103" groupAlignment="0" max="-2" attributes="0">

                              <Component id="jSeparator2" pref="210" max="32767" attributes="0"/>

                              <Component id="jSeparator1" pref="210" max="32767" attributes="0"/>

                              <Component id="floorNumberInput" alignment="0" pref="210" max="32767" attributes="0"/>

                              <Component id="roomNumberInput" pref="210" max="32767" attributes="0"/>

                              <Component id="sensorTypeDropDown" max="32767" attributes="0"/>

                          </Group>

                          <EmptySpace min="-2" pref="237" max="-2" attributes="0"/>

                      </Group>

                      <Group type="102" alignment="1" attributes="0">

                          <EmptySpace max="32767" attributes="0"/>

                          <Component id="addSensorButton" min="-2" pref="172" max="-2" attributes="0"/>

                          <EmptySpace min="-2" pref="307" max="-2" attributes="0"/>

                      </Group>

                  </Group>

                </DimensionLayout>

                <DimensionLayout dim="1">

                  <Group type="103" groupAlignment="0" attributes="0">

                      <Group type="102" alignment="0" attributes="0">

                          <EmptySpace min="-2" pref="19" max="-2" attributes="0"/>

                          <Component id="label2" min="-2" max="-2" attributes="0"/>

                          <EmptySpace min="3" pref="3" max="-2" attributes="0"/>

                          <Component id="jPanel6" min="-2" pref="10" max="-2" attributes="0"/>

                          <EmptySpace min="-2" pref="59" max="-2" attributes="0"/>

                          <Group type="103" groupAlignment="3" attributes="0">

                              <Component id="jLabel1" alignment="3" min="-2" max="-2" attributes="0"/>

                              <Component id="floorNumberInput" alignment="3" min="-2" max="-2" attributes="0"/>

                          </Group>

                          <EmptySpace min="1" pref="1" max="-2" attributes="0"/>

                          <Component id="jSeparator2" min="-2" pref="20" max="-2" attributes="0"/>

                          <EmptySpace type="separate" max="-2" attributes="0"/>

                          <Group type="103" groupAlignment="3" attributes="0">

                              <Component id="jLabel2" alignment="3" min="-2" max="-2" attributes="0"/>

                              <Component id="roomNumberInput" alignment="3" min="-2" max="-2" attributes="0"/>

                          </Group>

                          <EmptySpace min="0" pref="0" max="-2" attributes="0"/>

                          <Component id="jSeparator1" min="-2" pref="20" max="-2" attributes="0"/>

                          <EmptySpace type="separate" max="-2" attributes="0"/>

                          <Group type="103" groupAlignment="3" attributes="0">

                              <Component id="jLabel5" alignment="3" min="-2" max="-2" attributes="0"/>

                              <Component id="sensorTypeDropDown" alignment="3" min="-2" max="-2" attributes="0"/>

                          </Group>

                          <EmptySpace pref="92" max="32767" attributes="0"/>

                          <Component id="addSensorButton" min="-2" pref="40" max="-2" attributes="0"/>

                          <EmptySpace min="-2" pref="37" max="-2" attributes="0"/>

                      </Group>

                  </Group>

                </DimensionLayout>

              </Layout>

              <SubComponents>

                <Component class="java.awt.Label" name="label2">

                  <Properties>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="24" style="1"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="text" type="java.lang.String" value="Add a New Sensor"/>

                  </Properties>

                </Component>

                <Container class="javax.swing.JPanel" name="jPanel6">

                  <Properties>

                    <Property name="background" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="f6" green="b5" red="64" type="rgb"/>

                    </Property>

                    <Property name="minimumSize" type="java.awt.Dimension" editor="org.netbeans.beaninfo.editors.DimensionEditor">

                      <Dimension value="[135, 48]"/>

                    </Property>

                  </Properties>

                  <Layout>

                    <DimensionLayout dim="0">

                      <Group type="103" groupAlignment="0" attributes="0">

                          <EmptySpace min="0" pref="0" max="32767" attributes="0"/>

                      </Group>

                    </DimensionLayout>

                    <DimensionLayout dim="1">

                      <Group type="103" groupAlignment="0" attributes="0">

                          <EmptySpace min="0" pref="0" max="32767" attributes="0"/>

                      </Group>

                    </DimensionLayout>

                  </Layout>

                </Container>

                <Component class="javax.swing.JSeparator" name="jSeparator2">

                </Component>

                <Component class="javax.swing.JSeparator" name="jSeparator1">

                </Component>

                <Component class="javax.swing.JButton" name="addSensorButton">

                  <Properties>

                    <Property name="background" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="d2" green="76" red="19" type="rgb"/>

                    </Property>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="20" style="0"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="text" type="java.lang.String" value="add Sensor"/>

                    <Property name="border" type="javax.swing.border.Border" editor="org.netbeans.modules.form.editors2.BorderEditor">

                      <Border info="null"/>

                    </Property>

                    <Property name="horizontalTextPosition" type="int" value="0"/>

                  </Properties>

                  <Events>

                    <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="addSensorButtonActionPerformed"/>

                  </Events>

                  <AuxValues>

                    <AuxValue name="JavaCodeGenerator\_InitCodePost" type="java.lang.String" value="addSensorButton.setBackground (MaterialColors.BLUE\_800);&#xa;addSensorButton.setForeground (Color.WHITE);&#xa;addSensorButton.addMouseListener(MaterialUIMovement.getMovement(addSensorButton, MaterialColors.INDIGO\_900));"/>

                  </AuxValues>

                </Component>

                <Component class="javax.swing.JLabel" name="jLabel1">

                  <Properties>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="18" style="0"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="text" type="java.lang.String" value="Floor Number:"/>

                  </Properties>

                </Component>

                <Component class="javax.swing.JLabel" name="jLabel2">

                  <Properties>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="18" style="0"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="text" type="java.lang.String" value="Room Number:"/>

                  </Properties>

                </Component>

                <Component class="javax.swing.JFormattedTextField" name="floorNumberInput">

                  <Properties>

                    <Property name="background" type="java.awt.Color" editor="org.netbeans.modules.form.RADConnectionPropertyEditor">

                      <Connection component="addSensor" name="background" type="property"/>

                    </Property>

                    <Property name="border" type="javax.swing.border.Border" editor="org.netbeans.modules.form.editors2.BorderEditor">

                      <Border info="null"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="formatterFactory" type="javax.swing.JFormattedTextField$AbstractFormatterFactory" editor="org.netbeans.modules.form.editors.AbstractFormatterFactoryEditor">

                      <Format format="#0" subtype="-1" type="0"/>

                    </Property>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="18" style="0"/>

                    </Property>

                  </Properties>

                </Component>

                <Component class="javax.swing.JFormattedTextField" name="roomNumberInput">

                  <Properties>

                    <Property name="background" type="java.awt.Color" editor="org.netbeans.modules.form.RADConnectionPropertyEditor">

                      <Connection component="addSensor" name="background" type="property"/>

                    </Property>

                    <Property name="border" type="javax.swing.border.Border" editor="org.netbeans.modules.form.editors2.BorderEditor">

                      <Border info="null"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="formatterFactory" type="javax.swing.JFormattedTextField$AbstractFormatterFactory" editor="org.netbeans.modules.form.editors.AbstractFormatterFactoryEditor">

                      <Format format="#0" subtype="-1" type="0"/>

                    </Property>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="18" style="0"/>

                    </Property>

                  </Properties>

                </Component>

                <Component class="javax.swing.JLabel" name="jLabel5">

                  <Properties>

                    <Property name="font" type="java.awt.Font" editor="org.netbeans.beaninfo.editors.FontEditor">

                      <Font name="Tahoma" size="18" style="0"/>

                    </Property>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="text" type="java.lang.String" value="Type:"/>

                  </Properties>

                </Component>

                <Component class="javax.swing.JComboBox" name="sensorTypeDropDown">

                  <Properties>

                    <Property name="foreground" type="java.awt.Color" editor="org.netbeans.beaninfo.editors.ColorEditor">

                      <Color blue="ff" green="ff" red="ff" type="rgb"/>

                    </Property>

                    <Property name="model" type="javax.swing.ComboBoxModel" editor="org.netbeans.modules.form.editors2.ComboBoxModelEditor">

                      <StringArray count="2">

                        <StringItem index="0" value="smoke"/>

                        <StringItem index="1" value="co2"/>

                      </StringArray>

                    </Property>

                  </Properties>

                  <Events>

                    <EventHandler event="actionPerformed" listener="java.awt.event.ActionListener" parameters="java.awt.event.ActionEvent" handler="sensorTypeDropDownActionPerformed"/>

                  </Events>

                  <AuxValues>

                    <AuxValue name="JavaCodeGenerator\_InitCodePost" type="java.lang.String" value="mtcbUI2.installUI(sensorTypeDropDown);"/>

                    <AuxValue name="JavaCodeGenerator\_InitCodePre" type="java.lang.String" value="MaterialComboBoxUI mtcbUI2 = (MaterialComboBoxUI)MaterialComboBoxUI.createUI(sensorTypeDropDown);"/>

                    <AuxValue name="JavaCodeGenerator\_TypeParameters" type="java.lang.String" value="&lt;String&gt;"/>

                  </AuxValues>

                </Component>

              </SubComponents>

            </Container>

          </SubComponents>

        </Container>

      </SubComponents>

    </Container>

  </SubComponents>

</Form>

**MainPage.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 Controllers.Client;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Vector;

import javax.swing.JOptionPane;

import javax.swing.UIManager;

import javax.swing.UnsupportedLookAndFeelException;

import javax.swing.border.Border;

import javax.swing.border.EmptyBorder;

import javax.swing.event.ListSelectionEvent;

import mdlaf.MaterialLookAndFeel;

import mdlaf.animation.MaterialUIMovement;

import mdlaf.components.combobox.MaterialComboBoxUI;

import mdlaf.components.tabbedpane.MaterialTabbedPaneUI;

import mdlaf.components.table.MaterialTableUI;

import mdlaf.components.togglebutton.MaterialToggleButtonUI;

import mdlaf.utils.MaterialColors;

import model.Sensor;

*/\*\**

*\**

*\* @author Shehan*

*\*/*

public class MainPage extends javax.swing.JFrame implements Runnable {

    private ArrayList<Sensor> sensors;

    private Vector<String> columns;

    private Vector<Vector<Object>> items;

    HashMap<String, String> sensorReadings;

    private Alert alert;

    private Client rmiclient;

    private final String USERNAME;

*/\*\**

*\* Creates new form MainPage*

*\*/*

    private MainPage() {

        USERNAME = "";

    }

    public MainPage(String *username*) {

*//        if(!Login.loggedIn)*

*//        {*

*//            Login.main(new String[] {});*

*//            return;*

*//        }*

        USERNAME = username;

        try {

            UIManager.setLookAndFeel(new MaterialLookAndFeel());

        } catch (UnsupportedLookAndFeelException *e*) {

            e.printStackTrace();

        }

        sensorReadings = new HashMap<>();

        columns = new Vector<>();

        columns.add("Floor");

        columns.add("Room");

        columns.add("Active");

        columns.add("Type");

        columns.add("Reading");

        rmiclient = new Client(USERNAME);

        initComponents();

        Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();

        this.setLocation(dim.width / 2 - this.getSize().width / 2, dim.height / 2 - this.getSize().height / 2);

        jTable1.setEnabled(false);

        Thread t1 = new Thread(this);

        t1.start();

        deleteSensorButton.setEnabled(false);

*//        activeToggle.setEnabled(false);*

        jTable1.getSelectionModel().addListSelectionListener((ListSelectionEvent event) -> {

            if (jTable1.getSelectedRow() > -1) {

                int floorNumber = (int) jTable1.getValueAt(jTable1.getSelectedRow(), 0);

                String state = (String) jTable1.getValueAt(jTable1.getSelectedRow(), 2);

                deleteSensorButton.setEnabled(true);

*//                activeToggle.setEnabled(true);*

*//                if (floorNumber > 0) {*

*//                    if (state.equalsIgnoreCase("active")) {*

*//                        activeToggle.setSelected(true);*

*//                    } else {*

*//                        activeToggle.setSelected(false);*

*//                    }*

*//                }*

            } else {

                deleteSensorButton.setEnabled(false);

*//                activeToggle.setEnabled(false);*

            }

        });

    }

    @Override

    public void run() {

        Client tclient = new Client(USERNAME);

        while (true) *//Updating sensors every 30s*

        {

            tableMessage.setText("updating table...");

            try {

                sensors = tclient.getSensors();

            } catch (Exception *e*) { *// happens if not online. So we try again in 5s*

                tableMessage.setText("Please make sure that you are online.");

                threadSleep(5000);

                continue;

            }

            for (Sensor sensor : sensors) {

                String sensorUID = "sensor" + sensor.getFloor() + sensor.getRoom() + sensor.getSensorType();

                sensorReadings.put(sensorUID, "Waiting");

            }

            if (!sensorReadings.isEmpty()) {

                sensorReadings = rmiclient.getReading(sensorReadings);

            }

            UpdateTable();

            jTable1.setEnabled(true);

            threadSleep(15000); *// sleep for 15s*

        }

    }

    public void UpdateTable() {

        items = new Vector<>();

        List<String> dropDownList = new ArrayList<>();

        dropDownList.add("All");

        for (Sensor sensor : sensors) {

            Vector<Object> row = new Vector<>();*//row*

            row.add(sensor.getFloor());

            if (!dropDownList.contains(String.valueOf(sensor.getFloor()))) {

                dropDownList.add(String.valueOf(sensor.getFloor()));

            }

            row.add(sensor.getRoom());

            if (sensor.isActive()) {

                row.add("Active");

            } else {

                row.add("Inactive");

            }

            row.add(sensor.getSensorType());

            String reading;

            if (sensor.isActive()) {

                String sensorUID = "sensor" + sensor.getFloor() + sensor.getRoom() + sensor.getSensorType();

                reading = sensorReadings.get(sensorUID);

            } else {

                reading = "none";

            }

            row.add(reading);

            items.add(row);

        }

        int selectedRow = jComboBox1.getSelectedIndex();

        Vector<Vector<Object>> filtereditems = new Vector<>();

        if (selectedRow > 0) {

            int floorNumber = Integer.valueOf(jComboBox1.getItemAt(selectedRow));

            for (Sensor sensor : sensors) {

                if (sensor.getFloor() == floorNumber) {

                    Vector<Object> row = new Vector<>();*//row*

                    row.add(sensor.getFloor());

                    row.add(sensor.getRoom());

                    if (sensor.isActive()) {

                        row.add("Active");

                    } else {

                        row.add("Inactive");

                    }

                    row.add(sensor.getSensorType());

                    String reading;

                    if (sensor.isActive()) {

                        String sensorUID = "sensor" + sensor.getFloor() + sensor.getRoom() + sensor.getSensorType();

                        reading = sensorReadings.get(sensorUID);

                    } else {

                        reading = "none";

                    }

                    row.add(reading);

                    filtereditems.add(row);

                }

            }

        } else {

            filtereditems = (Vector<Vector<Object>>) items.clone();

        }

*// Populating the Table*

        jTable1.setModel(new javax.swing.table.DefaultTableModel(

                filtereditems, columns

        ) {

            Class[] types = new Class[]{

                java.lang.Integer.class, java.lang.Integer.class, java.lang.String.class, java.lang.String.class, java.lang.String.class

            };

            boolean[] canEdit = new boolean[]{

                false, false, false, false, false

            };

            public Class getColumnClass(int columnIndex) {

                return types[columnIndex];

            }

            public boolean isCellEditable(int rowIndex, int columnIndex) {

                return canEdit[columnIndex];

            }

        });

*// Populating the Drop down list*

        selectedRow = jComboBox1.getSelectedIndex();

        jComboBox1.setModel(new javax.swing.DefaultComboBoxModel<>(dropDownList.toArray(new String[0])));

        jComboBox1.setSelectedIndex(selectedRow);

        tableMessage.setText("");

    }

*/\*\**

*\* 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();

        closeButton = new javax.swing.JButton();

        jTabbedPane1 = new javax.swing.JTabbedPane();

        manageSensor = new javax.swing.JPanel();

        label3 = new java.awt.Label();

        jPanel7 = new javax.swing.JPanel();

        jLabel3 = new javax.swing.JLabel();

        jComboBox1 = new javax.swing.JComboBox<>();

        deleteSensorButton = new javax.swing.JButton();

        jLabel4 = new javax.swing.JLabel();

        jScrollPane2 = new javax.swing.JScrollPane();

        jTable1 = new javax.swing.JTable();

        tableMessage = new javax.swing.JLabel();

        addSensor = new javax.swing.JPanel();

        label2 = new java.awt.Label();

        jPanel6 = new javax.swing.JPanel();

        jSeparator2 = new javax.swing.JSeparator();

        jSeparator1 = new javax.swing.JSeparator();

        addSensorButton = new javax.swing.JButton();

        jLabel1 = new javax.swing.JLabel();

        jLabel2 = new javax.swing.JLabel();

        floorNumberInput = new javax.swing.JFormattedTextField();

        roomNumberInput = new javax.swing.JFormattedTextField();

        jLabel5 = new javax.swing.JLabel();

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

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

        setUndecorated(true);

        setResizable(false);

        getContentPane().setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        jPanel1.setBackground(MaterialColors.BLUE\_GRAY\_900);

        jPanel1.setLayout(new org.netbeans.lib.awtextra.AbsoluteLayout());

        closeButton.setBackground(new java.awt.Color(48, 48, 48));

        closeButton.setForeground(new java.awt.Color(48, 48, 48));

        closeButton.setIcon(new javax.swing.ImageIcon(getClass().getResource("/cross\_icon.png"))); *// NOI18N*

        closeButton.setToolTipText("");

        closeButton.setBorder(null);

        closeButton.setBorderPainted(false);

        closeButton.setContentAreaFilled(false);

        closeButton.setBackground (MaterialColors.BLUE\_GRAY\_900);

        closeButton.addMouseListener(MaterialUIMovement.getMovement(closeButton, MaterialColors.GRAY\_900));

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

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

                closeButtonActionPerformed(evt);

            }

        });

        jPanel1.add(closeButton, new org.netbeans.lib.awtextra.AbsoluteConstraints(795, 0, 40, -1));

        jTabbedPane1.setForeground(new java.awt.Color(255, 255, 255));

        MaterialTabbedPaneUI mtpUI = (MaterialTabbedPaneUI)MaterialTabbedPaneUI.createUI(jTabbedPane1);

        UIManager.put("TabbedPane.selectionForeground",MaterialColors.CYAN\_A100);

        UIManager.put("TabbedPane[focus].colorLine",MaterialColors.AMBER\_200);

        UIManager.put("TabbedPane.foreground",MaterialColors.WHITE);

        UIManager.put("TabbedPane.border",MaterialColors.GREEN\_100);

        mtpUI.installUI(jTabbedPane1);

        jTabbedPane1.setUI(mtpUI);

        jTabbedPane1.setBackground(MaterialColors.BLUE\_GRAY\_900);

        manageSensor.setBackground(MaterialColors.BLUE\_GRAY\_900);

        label3.setFont(new java.awt.Font("Tahoma", 1, 24)); *// NOI18N*

        label3.setForeground(new java.awt.Color(255, 255, 255));

        label3.setText("Manage Sensors");

        jPanel7.setBackground(new java.awt.Color(100, 181, 246));

        jPanel7.setMinimumSize(new java.awt.Dimension(135, 48));

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

        jPanel7.setLayout(jPanel7Layout);

        jPanel7Layout.setHorizontalGroup(

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

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

        );

        jPanel7Layout.setVerticalGroup(

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

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

        );

        jLabel3.setFont(new java.awt.Font("Tahoma", 0, 14)); *// NOI18N*

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

        jLabel3.setText("Select a floor to view it's sensors");

        MaterialComboBoxUI mtcbUI = (MaterialComboBoxUI)MaterialComboBoxUI.createUI(jComboBox1);

        jComboBox1.setFont(new java.awt.Font("Tahoma", 0, 18)); *// NOI18N*

        mtcbUI.installUI(jComboBox1);

        jComboBox1.addActionListener(new java.awt.event.ActionListener() {

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

                jComboBox1ActionPerformed(evt);

            }

        });

        deleteSensorButton.setBackground(new java.awt.Color(25, 118, 210));

        deleteSensorButton.setFont(new java.awt.Font("Tahoma", 0, 20)); *// NOI18N*

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

        deleteSensorButton.setText("Delete Sensor");

        deleteSensorButton.setBorder(null);

        deleteSensorButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        deleteSensorButton.setBackground (MaterialColors.BLUE\_800);

        deleteSensorButton.setForeground (Color.WHITE);

        deleteSensorButton.addMouseListener(MaterialUIMovement.getMovement(deleteSensorButton, MaterialColors.INDIGO\_900));

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

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

                deleteSensorButtonActionPerformed(evt);

            }

        });

        jLabel4.setFont(new java.awt.Font("Tahoma", 0, 18)); *// NOI18N*

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

        jLabel4.setText("Floor Number:");

        MaterialTableUI mtUI = (MaterialTableUI)MaterialTableUI.createUI(jTable1);

        UIManager.put("Table.background", MaterialColors.BLUE\_GRAY\_700);

        UIManager.put("Table.alternateRowColor",  MaterialColors.BLUE\_GRAY\_700);

        UIManager.put("Table.foreground", MaterialColors.WHITE);

        UIManager.put("Table.selectionForeground", MaterialColors.WHITE);

        UIManager.put("Table.selectionBackground", MaterialColors.PURPLE\_300);

        UIManager.put("Table.showHorizontalLines", true);

        UIManager.put("Table.showVerticalLines", true);

        UIManager.put("Table.gridColor", MaterialColors.WHITE);

        UIManager.put("Table.font", (new java.awt.Font("Tahoma", 0, 16)));

        mtUI.installUI(jTable1);

        jTable1.setForeground(new java.awt.Color(255, 255, 255));

        jTable1.setModel(new javax.swing.table.DefaultTableModel(

            new Object [][] {

                {null, null, null, null},

                {null, null, null, null},

                {null, null, null, null},

                {null, null, null, null}

            },

            new String [] {

                "Floor Number", "Room Number", "Active", "Type"

            }

        ) {

            Class[] types = new Class [] {

                java.lang.Integer.class, java.lang.Integer.class, java.lang.String.class, java.lang.String.class

            };

            boolean[] canEdit = new boolean [] {

                false, false, false, false

            };

            public Class getColumnClass(int columnIndex) {

                return types [columnIndex];

            }

            public boolean isCellEditable(int rowIndex, int columnIndex) {

                return canEdit [columnIndex];

            }

        });

        jTable1.setAutoscrolls(false);

        jTable1.setSelectionMode(javax.swing.ListSelectionModel.SINGLE\_SELECTION);

        jScrollPane2.setViewportView(jTable1);

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

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

        tableMessage.setMaximumSize(new java.awt.Dimension(470, 17));

        tableMessage.setMinimumSize(new java.awt.Dimension(470, 17));

        javax.swing.GroupLayout manageSensorLayout = new javax.swing.GroupLayout(manageSensor);

        manageSensor.setLayout(manageSensorLayout);

        manageSensorLayout.setHorizontalGroup(

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

            .addGroup(manageSensorLayout.createSequentialGroup()

                .addContainerGap()

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

                    .addGroup(manageSensorLayout.createSequentialGroup()

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

                            .addGroup(manageSensorLayout.createSequentialGroup()

                                .addGap(1, 1, 1)

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

                            .addComponent(label3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

                        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 213, javax.swing.GroupLayout.PREFERRED\_SIZE))

                    .addGroup(manageSensorLayout.createSequentialGroup()

                        .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT\_SIZE, 568, Short.MAX\_VALUE)

                        .addGap(18, 18, 18)

                        .addComponent(deleteSensorButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 182, javax.swing.GroupLayout.PREFERRED\_SIZE)

                        .addGap(45, 45, 45))

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

                        .addComponent(tableMessage, javax.swing.GroupLayout.PREFERRED\_SIZE, 470, javax.swing.GroupLayout.PREFERRED\_SIZE)

                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 46, Short.MAX\_VALUE)

                        .addComponent(jLabel4)

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

                        .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED\_SIZE, 175, javax.swing.GroupLayout.PREFERRED\_SIZE))))

        );

        manageSensorLayout.setVerticalGroup(

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

            .addGroup(manageSensorLayout.createSequentialGroup()

                .addGap(19, 19, 19)

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

                    .addGroup(manageSensorLayout.createSequentialGroup()

                        .addComponent(label3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

                        .addGap(3, 3, 3)

                        .addComponent(jPanel7, javax.swing.GroupLayout.PREFERRED\_SIZE, 10, javax.swing.GroupLayout.PREFERRED\_SIZE))

                    .addComponent(jLabel3))

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

                    .addGroup(manageSensorLayout.createSequentialGroup()

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

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

                            .addComponent(jLabel4)

                            .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

                        .addGap(199, 199, 199)

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

                    .addGroup(manageSensorLayout.createSequentialGroup()

                        .addGap(30, 30, 30)

                        .addComponent(tableMessage, javax.swing.GroupLayout.PREFERRED\_SIZE, 17, javax.swing.GroupLayout.PREFERRED\_SIZE)

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

                        .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED\_SIZE, 304, javax.swing.GroupLayout.PREFERRED\_SIZE)))

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

        );

        jTabbedPane1.addTab("Manage  Sensors", manageSensor);

        addSensor.setBackground(MaterialColors.BLUE\_GRAY\_900);

        label2.setFont(new java.awt.Font("Tahoma", 1, 24)); *// NOI18N*

        label2.setForeground(new java.awt.Color(255, 255, 255));

        label2.setText("Add a New Sensor");

        jPanel6.setBackground(new java.awt.Color(100, 181, 246));

        jPanel6.setMinimumSize(new java.awt.Dimension(135, 48));

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

        jPanel6.setLayout(jPanel6Layout);

        jPanel6Layout.setHorizontalGroup(

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

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

        );

        jPanel6Layout.setVerticalGroup(

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

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

        );

        addSensorButton.setBackground(new java.awt.Color(25, 118, 210));

        addSensorButton.setFont(new java.awt.Font("Tahoma", 0, 20)); *// NOI18N*

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

        addSensorButton.setText("add Sensor");

        addSensorButton.setBorder(null);

        addSensorButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);

        addSensorButton.setBackground (MaterialColors.BLUE\_800);

        addSensorButton.setForeground (Color.WHITE);

        addSensorButton.addMouseListener(MaterialUIMovement.getMovement(addSensorButton, MaterialColors.INDIGO\_900));

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

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

                addSensorButtonActionPerformed(evt);

            }

        });

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

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

        jLabel1.setText("Floor Number:");

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

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

        jLabel2.setText("Room Number:");

        floorNumberInput.setBackground(addSensor.getBackground());

        floorNumberInput.setBorder(null);

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

        floorNumberInput.setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(new javax.swing.text.NumberFormatter(new java.text.DecimalFormat("#0"))));

        floorNumberInput.setFont(new java.awt.Font("Tahoma", 0, 18)); *// NOI18N*

        roomNumberInput.setBackground(addSensor.getBackground());

        roomNumberInput.setBorder(null);

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

        roomNumberInput.setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(new javax.swing.text.NumberFormatter(new java.text.DecimalFormat("#0"))));

        roomNumberInput.setFont(new java.awt.Font("Tahoma", 0, 18)); *// NOI18N*

        jLabel5.setFont(new java.awt.Font("Tahoma", 0, 18)); *// NOI18N*

        jLabel5.setForeground(new java.awt.Color(255, 255, 255));

        jLabel5.setText("Type:");

        MaterialComboBoxUI mtcbUI2 = (MaterialComboBoxUI)MaterialComboBoxUI.createUI(sensorTypeDropDown);

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

        sensorTypeDropDown.setModel(new javax.swing.DefaultComboBoxModel<>(new String[] { "smoke", "co2" }));

        mtcbUI2.installUI(sensorTypeDropDown);

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

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

                sensorTypeDropDownActionPerformed(evt);

            }

        });

        javax.swing.GroupLayout addSensorLayout = new javax.swing.GroupLayout(addSensor);

        addSensor.setLayout(addSensorLayout);

        addSensorLayout.setHorizontalGroup(

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

            .addGroup(addSensorLayout.createSequentialGroup()

                .addContainerGap()

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

                    .addGroup(addSensorLayout.createSequentialGroup()

                        .addGap(1, 1, 1)

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

                    .addComponent(label2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

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

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

                .addGap(0, 240, Short.MAX\_VALUE)

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

                    .addComponent(jLabel1)

                    .addComponent(jLabel2)

                    .addComponent(jLabel5, javax.swing.GroupLayout.Alignment.TRAILING))

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

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

                    .addComponent(jSeparator2, javax.swing.GroupLayout.DEFAULT\_SIZE, 210, Short.MAX\_VALUE)

                    .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT\_SIZE, 210, Short.MAX\_VALUE)

                    .addComponent(floorNumberInput, javax.swing.GroupLayout.DEFAULT\_SIZE, 210, Short.MAX\_VALUE)

                    .addComponent(roomNumberInput, javax.swing.GroupLayout.DEFAULT\_SIZE, 210, Short.MAX\_VALUE)

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

                .addGap(237, 237, 237))

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

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

                .addComponent(addSensorButton, javax.swing.GroupLayout.PREFERRED\_SIZE, 172, javax.swing.GroupLayout.PREFERRED\_SIZE)

                .addGap(307, 307, 307))

        );

        addSensorLayout.setVerticalGroup(

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

            .addGroup(addSensorLayout.createSequentialGroup()

                .addGap(19, 19, 19)

                .addComponent(label2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

                .addGap(3, 3, 3)

                .addComponent(jPanel6, javax.swing.GroupLayout.PREFERRED\_SIZE, 10, javax.swing.GroupLayout.PREFERRED\_SIZE)

                .addGap(59, 59, 59)

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

                    .addComponent(jLabel1)

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

                .addGap(1, 1, 1)

                .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED\_SIZE, 20, javax.swing.GroupLayout.PREFERRED\_SIZE)

                .addGap(18, 18, 18)

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

                    .addComponent(jLabel2)

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

                .addGap(0, 0, 0)

                .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED\_SIZE, 20, javax.swing.GroupLayout.PREFERRED\_SIZE)

                .addGap(18, 18, 18)

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

                    .addComponent(jLabel5)

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

                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 92, Short.MAX\_VALUE)

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

                .addGap(37, 37, 37))

        );

        jTabbedPane1.addTab("Add Sensor", addSensor);

        jPanel1.add(jTabbedPane1, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, 830, 470));

        getContentPane().add(jPanel1, new org.netbeans.lib.awtextra.AbsoluteConstraints(0, 0, -1, -1));

        pack();

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

    private void closeButtonActionPerformed(java.awt.event.ActionEvent *evt*) {*//GEN-FIRST:event\_closeButtonActionPerformed*

        System.exit(0);

    }*//GEN-LAST:event\_closeButtonActionPerformed*

    private void addSensorButtonActionPerformed(java.awt.event.ActionEvent *evt*) {*//GEN-FIRST:event\_addSensorButtonActionPerformed*

        int floorNumber = -1;

        int roomNumber = -1;

        String sensorType;

        switch (sensorTypeDropDown.getSelectedIndex()) {

            case 0:

                sensorType = "smoke";

                break;

            case 1:

                sensorType = "co2";

                break;

            default:

                alert = new Alert("Select Sensor Type");

                return;

        }

        try {

            floorNumber = Integer.valueOf(floorNumberInput.getText());

            roomNumber = Integer.valueOf(roomNumberInput.getText());

        } catch (NumberFormatException *e*) {

            alert = new Alert("Invalid inputs");

            return;

        }

        if (floorNumber <= 0 || roomNumber <= 0) {

            alert = new Alert("Invalid inputs");

            return;

        }

        if (rmiclient.addSensor(floorNumber, roomNumber, sensorType)) {

            try {

                sensors = rmiclient.getSensors();

                UpdateTable();

            } catch (Exception *e*) { *// happens if not online.*

                tableMessage.setText("Please make sure that you are online.");

            }

            alert = new Alert("Sensor added sucessfully");

            floorNumberInput.setText("");

            roomNumberInput.setText("");

        } else {

            alert = new Alert("Error when adding Sensor");

        }

    }*//GEN-LAST:event\_addSensorButtonActionPerformed*

    private void deleteSensorButtonActionPerformed(java.awt.event.ActionEvent *evt*) {*//GEN-FIRST:event\_deleteSensorButtonActionPerformed*

        if (jTable1.getSelectedRow() < 0) {

            return;

        }

        int floorNumber = (int) jTable1.getValueAt(jTable1.getSelectedRow(), 0);

        int roomNumber = (int) jTable1.getValueAt(jTable1.getSelectedRow(), 1);

        String sensorType = (String) jTable1.getValueAt(jTable1.getSelectedRow(), 3);

        UIManager.put("Panel.background", MaterialColors.BLUE\_GRAY\_800);

        UIManager.put("OptionPane.messageForeground", MaterialColors.WHITE);

        int result = JOptionPane.showConfirmDialog(jPanel1,

                "    Sure you want to delete sensor on floor " + floorNumber + ", room number " + roomNumber + "    ",

                "Confirm Delete",

                JOptionPane.YES\_NO\_OPTION,

                JOptionPane.QUESTION\_MESSAGE);

        switch (result) {

            case JOptionPane.NO\_OPTION:

*//Do nothing*

                break;

            case JOptionPane.YES\_OPTION:

                if (rmiclient.removeSensor(floorNumber, roomNumber, sensorType)) {

                    try {

                        sensors = rmiclient.getSensors();

                        UpdateTable();

                    } catch (Exception *e*) { *// happens if not online.*

                        tableMessage.setText("Please make sure that you are online.");

                    }

                    alert = new Alert("Sensor deleted sucessfully");

                } else {

                    alert = new Alert("Error Deleting Server");

                }

                break;

        }

    }*//GEN-LAST:event\_deleteSensorButtonActionPerformed*

    private void jComboBox1ActionPerformed(java.awt.event.ActionEvent *evt*) {*//GEN-FIRST:event\_jComboBox1ActionPerformed*

        jComboBox1.addActionListener((ActionEvent e) -> {

            UpdateTable();

        });

    }*//GEN-LAST:event\_jComboBox1ActionPerformed*

    private void sensorTypeDropDownActionPerformed(java.awt.event.ActionEvent *evt*) {*//GEN-FIRST:event\_sensorTypeDropDownActionPerformed*

*// TODO add your handling code here:*

    }*//GEN-LAST:event\_sensorTypeDropDownActionPerformed*

    public void threadSleep(long *millis*) {

        try {

            Thread.sleep(millis);

        } catch (InterruptedException *ex*) {

            ex.printStackTrace();

        }

    }

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

    private javax.swing.JPanel addSensor;

    private javax.swing.JButton addSensorButton;

    private javax.swing.JButton closeButton;

    private javax.swing.JButton deleteSensorButton;

    private javax.swing.JFormattedTextField floorNumberInput;

    private javax.swing.JComboBox<String> jComboBox1;

    private javax.swing.JLabel jLabel1;

    private javax.swing.JLabel jLabel2;

    private javax.swing.JLabel jLabel3;

    private javax.swing.JLabel jLabel4;

    private javax.swing.JLabel jLabel5;

    private javax.swing.JPanel jPanel1;

    private javax.swing.JPanel jPanel6;

    private javax.swing.JPanel jPanel7;

    private javax.swing.JScrollPane jScrollPane2;

    private javax.swing.JSeparator jSeparator1;

    private javax.swing.JSeparator jSeparator2;

    private javax.swing.JTabbedPane jTabbedPane1;

    private javax.swing.JTable jTable1;

    private java.awt.Label label2;

    private java.awt.Label label3;

    private javax.swing.JPanel manageSensor;

    private javax.swing.JFormattedTextField roomNumberInput;

    private javax.swing.JComboBox<String> sensorTypeDropDown;

    private javax.swing.JLabel tableMessage;

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

}

Model –

**Sensor.java**

package model;

import java.io.Serializable;

public class Sensor implements Serializable {

    private static final long serialVersionUID = 1L;

    private String username;

    private String sensorUID;

    private String status = "online";

    private int floor;

    private int room;

    private String sensorType;

    public Sensor(int *floorNumber*, int *roomNumber*) {

        this.setFloor(floorNumber);

        this.setRoom(roomNumber);

        setSensorUID(String.valueOf(floorNumber) + String.valueOf(roomNumber));

    }

    public boolean isActive() {

        if (status.equalsIgnoreCase("online")) {

            return true;

        }

        return false;

    }

    public void setActive() {

        this.status = "online";

    }

    public void setInactive() {

        this.status = "offline";

    }

    public int getFloor() {

        return floor;

    }

    public void setFloor(int *floorNumber*) {

        this.floor = floorNumber;

    }

    public String getSensorUID() {

        return sensorUID;

    }

    public void setSensorUID(String *sensorUID*) {

        this.sensorUID = sensorUID;

    }

    public int getRoom() {

        return room;

    }

    public void setRoom(int *roomNumber*) {

        this.room = roomNumber;

    }

    public String getUsername() {

        return username;

    }

    public void setUsername(String *username*) {

        this.username = username;

    }

    public String getSensorType() {

        return sensorType;

    }

    public void setSensorType(String *sensorType*) {

        this.sensorType = sensorType;

    }

}

© 2020 GitHub, Inc.

## 5.4 RMI – Server

Controllers:

**KafkaConsumers.java**

package Controllers;

import java.time.Duration;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import java.util.Properties;

import java.util.concurrent.atomic.AtomicBoolean;

import org.apache.kafka.clients.consumer.ConsumerConfig;

import org.apache.kafka.clients.consumer.ConsumerRecord;

import org.apache.kafka.clients.consumer.ConsumerRecords;

import org.apache.kafka.clients.consumer.KafkaConsumer;

import org.apache.kafka.common.serialization.StringDeserializer;

public class KafkaConsumers implements Runnable {

    private final String bootstrapServer = "127.0.0.1:9092";

    private final String grp\_id = "desktop-kafka";

    private ArrayList<String> topics; *// Array List Of All the topics*

    private HashMap<String, String> values; *// Name of sensor and reading*

    private Properties properties;

    private AtomicBoolean hasTopicsBeenUpdated;

    public KafkaConsumers() {

        properties = new Properties();

        properties.setProperty(ConsumerConfig.BOOTSTRAP\_SERVERS\_CONFIG, bootstrapServer);

        properties.setProperty(ConsumerConfig.KEY\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class.getName());

        properties.setProperty(ConsumerConfig.VALUE\_DESERIALIZER\_CLASS\_CONFIG, StringDeserializer.class.getName());

        properties.setProperty(ConsumerConfig.GROUP\_ID\_CONFIG, grp\_id);

        properties.setProperty(ConsumerConfig.AUTO\_OFFSET\_RESET\_CONFIG, "earliest");

        topics = new ArrayList<String>();

        values = new HashMap<>();

        hasTopicsBeenUpdated = new AtomicBoolean(true);

    }

    public void addTopic(String *sensorName*) {

        topics.add(sensorName);

        hasTopicsBeenUpdated.set(true);

    }

    public void removeTopic(String *sensorName*) {

        topics.remove(sensorName);

        hasTopicsBeenUpdated.set(true);

    }

    public HashMap<String, String> getReading(HashMap<String, String> *sensorName*) {

        for (Map.Entry<String, String> entry : sensorName.entrySet()) {

            if (!topics.contains(entry.getKey())) {

                topics.add(entry.getKey());

                hasTopicsBeenUpdated.set(true);

            }

            if (values.containsKey(entry.getKey())) {

                String reading = values.get(entry.getKey());

                if (reading.contains("timeStamp")) {

                    reading = reading.split(",")[0].replaceAll("\\{", "").replaceAll("\"", "").split(":")[1];

                }

                sensorName.put(entry.getKey(), reading);

            }

        }

        return sensorName;

    }

    public void run() {

        KafkaConsumer<String, String> consumer = new KafkaConsumer<String, String>(properties);

        while (true) {

            if (topics.isEmpty()) {

                ThreadSleep(1000);

                continue;

            }

            if (hasTopicsBeenUpdated.get()) {

                consumer.subscribe(topics);

                hasTopicsBeenUpdated.set(false);

            }

            ConsumerRecords<String, String> records = consumer.poll(Duration.ofMillis(1000));

            for (ConsumerRecord<String, String> record : records) {

                values.put(record.topic(), record.value());

            }

        }

    }

    private void ThreadSleep(long *milliseconds*) {

        try {

            Thread.sleep(milliseconds);

        } catch (InterruptedException *e*) {

            e.printStackTrace();

        }

    }

}

**Server.java**

package Controllers;

import java.io.IOException;

import java.io.UnsupportedEncodingException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import org.apache.http.HttpResponse;

import org.apache.http.client.ClientProtocolException;

import org.apache.http.client.HttpClient;

import org.apache.http.client.methods.HttpDelete;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.client.methods.HttpPut;

import org.apache.http.entity.StringEntity;

import org.apache.http.impl.client.BasicResponseHandler;

import org.apache.http.impl.client.HttpClientBuilder;

import model.Sensor;

import com.google.gson.Gson;

public class Server extends UnicastRemoteObject implements ServerInterface {

    private final String SENSOR\_API\_URL = "http://localhost:5000/api";

    private final String AUTHENTICATION\_BASE\_URL = "http://localhost:8080";

    private KafkaConsumers consumers;

    public Server() throws RemoteException {

        super();

        consumers = new KafkaConsumers();

        Thread thread = new Thread(consumers);

        thread.start();

    }

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

*// Starting the RMI registry*

        try {

            LocateRegistry.createRegistry(1099);

        } catch (RemoteException *ignored*) {

*// Means RMI registry is already running*

        }

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

        try {

            ServerInterface server = new Server();

            Registry registry = LocateRegistry.getRegistry();

            registry.bind("rmiServer", server);

            System.out.println("Service started....");

            new Server().createUser();

        } catch (Exception *e*) {

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

        }

    }

*/\*\**

*\* Will make a JSON request to add a Sensor to the database*

*\**

*\* @return true if sucessful*

*\*/*

    @Override

    public boolean addSensor(int *floorNumber*, int *roomNumber*, String *username*, String *sensorType*) {

        String sensorUID = "sensor" + floorNumber + roomNumber + sensorType;

        Map<String, String> body = new HashMap<>();

        body.put("username", username);

        body.put("sensorUID", sensorUID);

        body.put("floor", String.valueOf(floorNumber));

        body.put("room", String.valueOf(roomNumber));

        body.put("sensorType", sensorType);

        if(makeRequest(body, "POST", SENSOR\_API\_URL + "/registerSensor"))

        {

            consumers.addTopic(sensorUID);

            runSensor(sensorUID);

            return true;

        }

        return false;

    }

*/\*\**

*\* Will make a JSON request to remove a Sensor from the database*

*\**

*\* @return true if sucessful*

*\*/*

    @Override

    public boolean removeSensor(int *floorNumber*, int *roomNumber*, String *sensorType*) throws RemoteException {

        Map<String, String> body = new HashMap<>();

        String sensorUID = "sensor" + floorNumber + roomNumber + sensorType;

        if(makeRequest(body, "DELETE", SENSOR\_API\_URL + "/deleteSensor?sensorUID=" + sensorUID))

        {

            consumers.removeTopic(sensorUID);

            return true;

        }

        return false;

    }

*/\*\**

*\* Will make a JSON request to change the state of a Sensor (active or inactive)*

*\**

*\* @return true if sucessful*

*\*/*

    @Override

    public boolean changeState(int *floorNumber*, int *roomNumber*, boolean *state*, String *sensorType*) throws RemoteException {

        Map<String, String> body = new HashMap<>();

        String sensorUID = "sensor" + floorNumber + roomNumber + sensorType;

        String status;

        if (state) {

            status = "online";

        } else {

            status = "offline";

        }

        return makeRequest(body, "PUT", SENSOR\_API\_URL + "/updateSensorStatus/" + sensorUID + "?status=" + status);

    }

*/\*\**

*\* Will make a JSON request to retreive all sensors on database*

*\**

*\* @return sensor Arraylist of the sensor details*

*\*/*

    @Override

    public ArrayList<Sensor> viewSensors(String *username*) throws RemoteException {

        ArrayList<Sensor> sensors = new ArrayList<>();

        HttpClient httpClient = HttpClientBuilder.create().build();

        HttpGet get = new HttpGet(SENSOR\_API\_URL + "/getSensorsByUsername/" + username);

        get.setHeader("Content-type", "application/json");

        get.setHeader("Accept", "application/json");

        HttpResponse response = null;

        String responseString = null;

        try {

            response = httpClient.execute(get);

            responseString = new BasicResponseHandler().handleResponse(response);

        } catch (ClientProtocolException *e*) {

            e.printStackTrace();

        } catch (IOException *e*) {

            e.printStackTrace();

        }

        for (String string : responseString.replaceAll("[\\[\\]]", "").split("},")) {

            string = string.replaceAll("[{}]|(\\r\\n|\\r|\\n).\*}.\*,", "");

            if (string.trim().length() == 0) {

                continue;

            }

            string = "{\n" + string.replace("},", "").trim() + "\n}";

            Sensor sensor = new Gson().fromJson(string, Sensor.class);

            if (sensor.getRoom() != 0 && sensor.getFloor() != 0) {

                sensors.add(sensor);

            }

        }

        return sensors;

    }

*/\*\**

*\* Checks the login with the authorization api*

*\**

*\* @param username*

*\* @param password*

*\* @return true if valid login*

*\*/*

    @Override

    public boolean login(String *username*, String *password*) throws RemoteException {

        Map<String, String> body = new HashMap<>();

        body.put("username", String.valueOf(username));

        body.put("password", String.valueOf(password));

        return makeRequest(body, "POST", AUTHENTICATION\_BASE\_URL + "/loginAdmin");

    }

*/\*\**

*\* Makes sure the authorization server is reachable*

*\**

*\* @return true if reachable*

*\*/*

    @Override

    public boolean checkAuthenticationServer() throws RemoteException {

        Map<String, String> body = new HashMap<>();

        return makeRequest(body, "POST", AUTHENTICATION\_BASE\_URL + "/checkAuthenticationAlive");

    }

*/\*\**

*\* getting the reading of a sensor*

*\* @param sensorName*

*\*/*

    @Override

    public HashMap<String,String> getReading(HashMap<String,String> *sensorName*) throws RemoteException {

        return consumers.getReading(sensorName);

    }

*/\*\**

*\* creating the default user for testing*

*\*/*

    public void createUser() {

        Map<String, String> body = new HashMap<>();

        body.put("username", "admin");

        body.put("password", "admin");

        body.put("email", "admin@test.com");

        body.put("phoneNumber", "0112345678");

        body.put("type", "admin");

        makeRequest(body, "POST", "http://localhost:8080/register");

    }

*/\*\**

*\* Used by add/remove/change methods to make requests.*

*\*/*

    public boolean makeRequest(Map<String, String> *body*, String *RequestType*, String *URL*) {

        HttpResponse response = null;

        try {

            HttpClient httpClient = HttpClientBuilder.create().build();

            StringEntity postingString = new StringEntity(new Gson().toJson(body));

            switch (RequestType) {

                case "POST":

                    HttpPost post = new HttpPost(URL);

                    post.setEntity(postingString);

                    post.setHeader("Content-type", "application/json");

                    response = httpClient.execute(post);

                    break;

                case "PUT":

                    HttpPut put = new HttpPut(URL);

                    put.setEntity(postingString);

                    put.setHeader("Content-type", "application/json");

                    response = httpClient.execute(put);

                    break;

                case "DELETE":

                    HttpDelete delete = new HttpDelete(URL);

                    delete.setHeader("Content-type", "application/json");

                    response = httpClient.execute(delete);

                    break;

            }

        } catch (UnsupportedEncodingException *e*) {

            e.printStackTrace();

        } catch (ClientProtocolException *e*) {

            e.printStackTrace();

        } catch (IOException *e*) {

            e.printStackTrace();

        }

        if (response == null || response.getStatusLine().getStatusCode() > 399) {

            return false;

        }

        return true;

    }

    private void runSensor(String *sensorName*)

    {

        Runtime rt = Runtime.getRuntime();

        try {

            rt.exec("cmd.exe /c cd \"..\\..\\Sensor\_Service\\sensor\_client\\ \" & start cmd.exe /k \"node senor\_client "+sensorName+"\"");

        } catch (IOException *e*) {

            e.printStackTrace();

        }

    }

    private static final long serialVersionUID = 1L;

}

**ServerInterface.java**

package Controllers;

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.ArrayList;

import java.util.HashMap;

import model.Sensor;

public interface ServerInterface extends Remote {

    public boolean addSensor(int *floorNumber*, int *roomNumber*, String *username*, String *sensorType*) throws RemoteException;

    public boolean removeSensor(int *floorNumber*, int *roomNumber*, String *sensorType*) throws RemoteException;

    public boolean changeState(int *floorNumber*, int *roomNumber*, boolean *state*, String *sensorType*) throws RemoteException;

    public ArrayList<Sensor> viewSensors(String *username*) throws RemoteException;

    public boolean login(String *username*, String *password*) throws RemoteException;

    public boolean checkAuthenticationServer()  throws RemoteException;

    public HashMap<String,String> getReading(HashMap<String,String> *sensorName*) throws RemoteException;

}

Model –

**Sensor.java**

package model;

import java.io.Serializable;

public class Sensor implements Serializable {

    private static final long serialVersionUID = 1L;

    private String username;

    private String sensorUID;

    private String status = "online";

    private int floor;

    private int room;

    private String sensorType;

    public Sensor(int *floorNumber*, int *roomNumber*) {

        this.setFloor(floorNumber);

        this.setRoom(roomNumber);

        setSensorUID(String.valueOf(floorNumber) + String.valueOf(roomNumber));

    }

    public boolean isActive() {

        if (status.equalsIgnoreCase("online")) {

            return true;

        }

        return false;

    }

    public void setActive() {

        this.status = "online";

    }

    public void setInactive() {

        this.status = "offline";

    }

    public int getFloor() {

        return floor;

    }

    public void setFloor(int *floorNumber*) {

        this.floor = floorNumber;

    }

    public String getSensorUID() {

        return sensorUID;

    }

    public void setSensorUID(String *sensorUID*) {

        this.sensorUID = sensorUID;

    }

    public int getRoom() {

        return room;

    }

    public void setRoom(int *roomNumber*) {

        this.room = roomNumber;

    }

    public String getUsername() {

        return username;

    }

    public void setUsername(String *username*) {

        this.username = username;

    }

    public String getSensorType() {

        return sensorType;

    }

    public void setSensorType(String *sensorType*) {

        this.sensorType = sensorType;

    }

}

## 5.5 Sensor Service

**Sensor.js**

const mongoose = require("mongoose")

*//Mongoose Schema Model for Sensors*

const schema = mongoose.Schema({

  username: String,

  sensorUID : String,

  floor: Number,

  room: Number,

  sensorType: String,

  status : String,

  reading: Number

})

module.exports = mongoose.model("Sensor", schema)

**index.js**

const express = require('express');

const mongoose = require("mongoose");

const {Kafka} = require("kafkajs")

const Sensor = require('./Sensor')

const routes = require("./routes")

const bodyParser = require("body-parser")

mongoose.set('useFindAndModify', false);

*//On the start of the server this method runs*

*//It grabs all the sensors from the Database*

*//Then it will start a kafka topic(channel) for each unique sensor*

*//This is to make sure that even if the kafka server is restarted or lost*

*//the new instance will contain channels for each sensor*

async function startSensorChannels ()

{

    try {

*//Getting all sensors*

        let sensors = await Sensor.find();

        console.log(sensors)

*//Connecting to Kafka*

        const kafka = new Kafka({

            "clientId": "node-sensor-api",

            "brokers" :["localhost:9092"]

       })

        const admin = kafka.admin();

        console.log("Connecting.....")

        await admin.connect()

        console.log("Connected!")

        let topicContetnt = [];

*//creating a kafka topic list to send as parameter for create.topic method*

        sensors.forEach(

            (*sensor*) =>

            {

                topicContetnt.push(

                    {

                        "topic" : sensor.sensorUID,

                        "numPartitions" : 1

                    }

                )

            }

        )

*//Creating topics in kafka cluster*

        await admin.createTopics({

            "topics": topicContetnt

        })

        console.log("Created Successfully!")

        await admin.disconnect();

    }

    catch (error)

    {

        console.error(`Something bad happened ${error}`)

    }

    return 0;

}

const port = process.env.PORT || 5001;

*//Connecting to mongodb instance*

mongoose

  .connect("mongodb://localhost:28017/acmedb",

  {

      useNewUrlParser: true ,

      "auth": {"authSource":"admin"} ,

      "user": "root",

      "pass": "root"

    },

      )

    .then(() => {

    const app = express()

    app.use(bodyParser.json())

    app.use("/api", routes)

*//Starting the sensor Channels*

    startSensorChannels();

*//Starting the server*

    app.listen(5000, () => {

      console.log("Server has started!")

    })

})

**routes.js**

const express = require("express")

const { Kafka } = require("kafkajs")

*//Importing Mongoose Data Model*

const Sensor = require('./Sensor')

const router = express.Router()

*// Method Used To register A Kafka Topic for each unique sensor*

async function addSensorChannel(*sensorUID*) {

*//Creating A kafka connection to the cluster*

    const kafka = new Kafka({

        "clientId": "node-sensor-api",

        "brokers": ["localhost:9092"]

    })

    const admin = kafka.admin();

    console.log("Connecting.....")

    await admin.connect()

    console.log("Connected!")

*//Creating a topic (Channel) for given sensor*

    await admin.createTopics({

        "topics": [{

            "topic": sensorUID,

            "numPartitions": 1

        }]

    })

    console.log("Created Successfully!")

    await admin.disconnect();

}

*// GET endpoint that returns the sensor reading for specific user*

router

    .get("/getSensorReading/:sensorUID",

        async (*req*, *res*) => {

*//Getting the path parameter*

            let sensorUID = req.params.sensorUID;

            let reading = 0

*//Check if sensor Exist in DB If true returns reading else returns 404 error*

            Sensor.exists({ sensorUID: sensorUID },

                async function (*err*, *result*) {

                    reading = "not found";

                    if (result === false) {

                        res.status(404)

                        res.send({reading:reading});

                    }

                    else {

                        const doc = await Sensor.findOne({ sensorUID: sensorUID });

                        res.status(200);

                        res.send({reading:doc.reading});

                    }

                })

        }

    )

*//PUT endpoint to modify sensor readuing*

router

    .put("/updateSensorReading/:sensorUID",

        async (*req*, *res*) => {

*//Getting the path parameter*

            let sensorUID = req.params.sensorUID;

*//Get query parameter from JSON body*

            let reading = req.body.reading;

*//Check if sensor Exist in DB If true updates reading else returns 404 error*

            Sensor.exists({ sensorUID: sensorUID }, async function (*err*, *result*) {

                if (result === false) {

                    res.send({});

                    res.status(404)

                }

                else {

*//Find and Update sensor*

                    const doc = await Sensor.findOne({ sensorUID: sensorUID });

                    doc.reading = parseInt(reading)

                    await doc.save();

                    const sensor = await Sensor.findOne({ sensorUID: req.params.sensorUID });

                    res.send(sensor);

                    res.status(200);

                }

            })

        }

    )

*//PUT Route to modify sensor status to online or offline by sensor UID*

router

    .put("/updateSensorStatus/:sensorUID",

        async (*req*, *res*) => {

*//Get sensor UID from path*

            let sensorUID = req.params.sensorUID;

*//Get status from JSON body*

            let status = req.query.status;

*//If exist find and update sensor*

            Sensor.exists({ sensorUID: sensorUID },

                async function (*err*, *result*) {

                        if (result === false) {

                            res.status(404)

                        }

                        else {

                            const doc = await Sensor.findOne({ sensorUID: sensorUID });

                            doc.status = status

                            await doc.save();

                            res.status(200);

                        }

                    })

            const sensor = await Sensor.findOne({ sensorUID: req.body.sensorUID });

            res.send(sensor);

        }

    )

*//DELETE route for delerting a specific sensor by sensorUID*

router

    .delete("/deleteSensor", async (*req*, *res*) => {

        let sensorUID = req.query.sensorUID;

        Sensor.exists({ sensorUID: sensorUID }, async (*err*, *result*) => {

            if (result === true) {

                let sensor = await Sensor.deleteOne({ sensorUID: sensorUID });

                await Sensor.remove(sensor);

                res.status(200).send("Successfully removes !")

            }

            else {

                res.status(404).send("Sensor Not Found !")

            }

        })

    })

*//GET route to get all sensors from the DB*

router

    .get("/getAllSensors",

        async (*req*, *res*) => {

            let sensors = await Sensor.find();

            res.send(sensors);

        }

)

*//GET route to get all sensors from the DB belonging to one user*

router.get("/getSensorsByUsername/:username", async (*req*, *res*) => {

    var username = req.params.username;

    Sensor.exists({ username: username }, async (*err*, *result*) => {

        if (result === true) {

            let sensors = await Sensor.find({ username: username });

            res.status(200);

            res.send(sensors);

        }

        else {

            res.status(404)

            res.send([]);

        }

    })

})

*//PUT route to modify Sensor information*

router.put("/updateSensor", async (*req*, *res*) => {

    const sensor = new Sensor(

        {

            username: req.body.username,

            sensorUID: req.body.sensorUID,

            floor: req.body.floor,

            room: req.body.room,

            sensorType: req.body.sensorType

        })

    Sensor.exists({ sensorUID: req.body.sensorUID }, async function (*err*, *result*) {

        if (result === false) {

            res.status(404)

        }

        else {

            const doc = await Sensor.findOne({ sensorUID: req.body.sensorUID });

            doc.floor = req.body.floor;

            doc.room = req.body.room;

            doc.sensorType = req.body.sensorType

            await doc.save();

            res.status(200);

        }

        res.send(sensor);

    })

})

*//POST request to register sensor*

*//For each new sensor a kafka topic is created so clients can push to that topic*

router

    .post("/registerSensor",

        async (*req*, *res*) => {

            const sensor = new Sensor(

                {

                    username: req.body.username,

                    sensorUID: req.body.sensorUID,

                    floor: req.body.floor,

                    room: req.body.room,

                    sensorType: req.body.sensorType,

                    status: "online",

                    reading: 0

                })

*//Check if Exist    , if exist return error 409, else save to DB and start kafka chanel*

            Sensor.exists({ sensorUID: req.body.sensorUID },

                async function (*err*, *result*) {

                    if (result === false) {

*//save sensor to DB*

                        await sensor.save()

*//start server*

                        addSensorChannel(req.body.sensorUID);

                        res.status(201)

                    }

                    else {

                        res.status(409)

                    }

                    res.send(sensor);

                }

            )

        }

    )

module.exports = router

**zk-single-kafka-single.yml**

version: '2.1'

services:

  zoo1:

    image: zookeeper:3.4.9

    hostname: zoo1

    ports:

      - "2181:2181"

    environment:

        ZOO\_MY\_ID: 1

        ZOO\_PORT: 2181

        ZOO\_SERVERS: server.1=zoo1:2888:3888

    volumes:

      - ./zk-single-kafka-single/zoo1/data:/data

      - ./zk-single-kafka-single/zoo1/datalog:/datalog

  kafka1:

    image: confluentinc/cp-kafka:5.5.0

    hostname: kafka1

    ports:

      - "9092:9092"

    environment:

      KAFKA\_ADVERTISED\_LISTENERS: LISTENER\_DOCKER\_INTERNAL://kafka1:19092,LISTENER\_DOCKER\_EXTERNAL://${DOCKER\_HOST\_IP:-127.0.0.1}:9092

      KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: LISTENER\_DOCKER\_INTERNAL:PLAINTEXT,LISTENER\_DOCKER\_EXTERNAL:PLAINTEXT

      KAFKA\_INTER\_BROKER\_LISTENER\_NAME: LISTENER\_DOCKER\_INTERNAL

      KAFKA\_ZOOKEEPER\_CONNECT: "zoo1:2181"

      KAFKA\_BROKER\_ID: 1

      KAFKA\_LOG4J\_LOGGERS: "kafka.controller=INFO,kafka.producer.async.DefaultEventHandler=INFO,state.change.logger=INFO"

      KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1

    volumes:

      - ./zk-single-kafka-single/kafka1/data:/var/lib/kafka/data

    depends\_on:

      - zoo1

  mongodb\_container:

    image: mongo:latest

    environment:

      MONGO\_INITDB\_ROOT\_USERNAME: root

      MONGO\_INITDB\_ROOT\_PASSWORD: root

    ports:

      - 28017:27017

    volumes:

      - mongodb\_data\_container:/data/db

volumes:

  mongodb\_data\_container:

**sensor\_client –**

**senor\_client.js**

const {Kafka} = require("kafkajs")

let myArgs = process.argv.slice(2);

function randomNumber(*min*, *max*) {

    return Math.floor(Math.random() \* (max - min) + min);

}

console.log(myArgs[0])

run();

async function run(){

    try

    {

         const kafka = new Kafka({

              "clientId": "myapp",

              "brokers" :["localhost:9092"]

         })

        const producer = kafka.producer();

        console.log("Connecting.....")

        await producer.connect()

        console.log("Connected!")

        setInterval(() => {

            let payload = {

                "reading" : randomNumber(0, 10),

                "timeStamp" : new Date()

            }

            sendMessage(producer,myArgs[0], JSON.stringify(payload))

        }, 1000);

    }

    catch(ex)

    {

        console.error(`Something bad happened ${ex}`)

    }

    finally{

        await producer.disconnect();

        process.exit(0);

    }

}

async function sendMessage(*producer*, *topic* , *message*) {

    const result =  await producer.send({

        "topic": topic,

        "messages": [

            {

                "value": message,

                "partition": 0

            }

        ]

    })

    console.log(`Send Successfully! ${JSON.stringify(result)}`)

}

sensor\_monitoring –

**index.js**

const fetch = require('node-fetch');

const {Kafka} = require("kafkajs");

*//This service listens to the kafka stream and update the sensor values in the sensor API*

*//Also if the value is above 5 the sensor calls yhe alert api to send email to the user*

*//connecting to kafka server*

const kafka = new Kafka({

    clientId: 'sensor-monitor',

    brokers: ['localhost:9092']

  })

*//url to sensor api to get all sensors*

const url = "http://localhost:5000/api/getAllSensors"

*//Methood to fetch All sensors*

const getAllSensors = async () =>

{

    let sensorArray = await fetch(url);

    sensorArray = await sensorArray.json();

    return sensorArray;

}

*//For each sensor start listening to their respective topics using kafkaListnerMethd*

getAllSensors().then(

    (*data*) => {

        let topics = [];

        data.forEach(*element* => {

                kafkaListners({topic:element.sensorUID, fromBeginning: false} , element.username  , element.floor , element.room , element.sensorType)

        });

    }

)

*//Handles reading through each message asynchronusly*

*//Reads the new value*

*//Updates the new value in sensor API*

async function kafkaListners(*topic\_array*,*username*,*floorNum*, *roomNum* , *type*)

{

    console.log("Called KafkaListner Method")

*//url to Alert to send an email when required*

    const url2 = "http://localhost:8081/emailAlertToUser";

*//Starts listenning to a channel*

    const consumer = kafka.consumer({ groupId: `${topic\_array.topic}-alert-monitor` })

    await consumer.connect();

    await consumer.subscribe(topic\_array)

*//This method runs every time a new senor value is added*

    await consumer.run({

        eachMessage: async ({ *topic*, *partition*, *message* }) => {

*//Caturing the sensor reading and timeStamp of each senor emmited*

            let reading = Number(JSON.parse(message.value.toString()).reading);

            let timeStamp =  new Date(JSON.parse(message.value.toString()).timeStamp);

*//Updating the main senosor server reading of corresponding sensor in the sensor API*

            updateSensorReading(topic , reading)

            if( reading > 5  )

            {

*//Compoing the email body*

                    let msg = `${username} Your ${type} sensor ${topic} on floor ${floorNum} room ${roomNum} has shown a reading of ${JSON.parse(message.value.toString()).reading} on ${timeStamp.toDateString()} ${timeStamp.toTimeString()}`;

*//Calling the alert API to send a message*

                    fetch(

                        url2,

                        {

                            method:"POST" ,

                            body: JSON.stringify({"username": `${username}` , "message": `${msg}`}) ,

                            headers: {

                                'Content-Type': 'application/JSON'

                             },

                        }

                    )

                    .then( *res* => console.log(res.status))

            }

        },

      })

}

*//Updating sensor reading field in the main sensor server*

    function updateSensorReading(*sensorUID*, *reading*)

    {

        fetch(`http://localhost:5000/api/updateSensorReading/${sensorUID}`, {

            method: 'put',

            body:    JSON.stringify({"reading" : reading}),

            headers: { 'Content-Type': 'application/json' },

        })

        .then(*res* => console.log(res))

    }

healthCheck –

**index.js**

const fetch = require('node-fetch');

const { Kafka } = require("kafkajs");

*//connecting to a kafka instance*

const kafka = new Kafka({

    clientId: 'sensor-monitor',

    brokers: ['localhost:9092']

})

*//Endpoint to the Sensor API to get all sensors*

const url = "http://localhost:5000/api/getAllSensors"

*//Method to fetch all sensor date*

const getAllSensors = async () => {

    let sensorArray = await fetch(url);

    sensorArray = await sensorArray.json();

    console.log("Getting sensor metadata....");

    return sensorArray;

}

console.log("Initializing ......")

*//Calling async method*

getAllSensors().then(

    (*data*) => {

*//This parameter controls how frequent the service checks the sensors*

        let refetch\_time\_frame = 10 ; *//in seconds*

*//This paramters controls the initial delay*

        let initial\_delay = 10 ; *//in seconds*

        setTimeout(() => {

            console.log("syncing with kafka streams ......")

*//Creating an array with sensorUID and last\_updated , setting last upadated to current Time*

            let sensor\_table = [];

            data.forEach(*element* => {

                sensor\_table.push({ sensorUID: element.sensorUID, lastReading: new Date() })

            });

*//Running Kafka Conumer to check for new messages, If new message is detected  last\_updated would be updated*

*//Passing the topic array to the kafka listner method so It starts listenning*

            data.forEach(*element* => {

                kafkaListners({"topic": element.sensorUID , "fromBeginning": false} , sensor\_table)

            });

*//A delay is introduced to avoid start methods before kafka client has joined the cluster*

            setInterval(() => {

*//Every 10 seconds a method should calculated the time between the health checker service started*

*//and when the last reading is*

                sensor\_table.forEach(*element* => {

                    let currentTime = new Date();

                    console.log(element.sensorUID, Math.floor( (currentTime - element.lastReading)/1000 ) )

                });

*// Check if they have been updated for atleast n Seconds If Not Set status as offline*

                sensor\_table.forEach(*element* => {

                    let currentTime = new Date();

                    let status\_checking\_time\_frame = 30 *//(secs) the minimum gap between now and last reading for a status update online or offline*

*//settting online or off line status depending on the gap between now and last reading*

                    if( ( Math.floor( ( currentTime - element.lastReading ) / 1000) ) >= status\_checking\_time\_frame)

                    {

                        console.log(`sensor ${element.sensorUID} is Offline`)

                        let url = `http://localhost:5000/api/updateSensorStatus/${element.sensorUID}?status=offline`

*//Calling sensor Api to update the sensor status*

                        fetch( url, {method:'put'} )

                            .then( *res* => console.log(res.status))

                    }

                    else

                    {

                        console.log(`sensor ${element.sensorUID} is Online`)

*//Calling sensor Api to update the sensor status*

                        let url = `http://localhost:5000/api/updateSensorStatus/${element.sensorUID}?status=online`

                        fetch( url, {method:'put'} )

                        .   then( *res* => console.log(res.status))

                    }

                });

            }, refetch\_time\_frame \* 1000);

        }, initial\_delay \* 1000)

    }

)

    async function kafkaListners(*topic\_array* , *sensor\_table*) {

*//console.log("Called KafkaListner Method")*

        let val = topic\_array.topic;

        var index = sensor\_table.findIndex(function (*item*, *i*) {

            return item.sensorUID === val

        })

        console.log("Sensor Position " , index)

        const consumer = kafka.consumer({ groupId: `${topic\_array.topic}-alert-health-check` })

        await consumer.connect();

        await consumer.subscribe(topic\_array)

*//This method runs everytime a message is sent to the kafka topic which is the sensorUID from a client*

        await consumer.run({

            eachMessage: async ({ *topic*, *partition*, *message* }) => {

*//console.log(topic, JSON.parse(message.value) )*

                sensor\_table[index] = {sensorUID: topic , lastReading : new Date(JSON.parse(message.value).timeStamp) }

            },

        })

    }

## 5.6 web client

src -

**App.js**

import React from 'react';

import './App.css';

import {

  BrowserRouter as Router,

  Switch,

  Route,

} from "react-router-dom";

import WelcomePage from './Pages/WelcomePage';

import Dashboard from './Pages/Dashboard';

*//Creating User Context to handle authentication functions*

export const AuthContext = React.createContext();

*//Initial state to User context*

const initialState = {

  isAuthenticated: false,

  user: null,

  token: null,

  sensors: []

};

*//Reducer function to Login and store user credential in browser storage and to remove*

const reducer = (*state*, *action*) => {

  switch (action.type) {

    case "LOGIN":

          localStorage.setItem("user", JSON.stringify(action.payload.user));

          localStorage.setItem("sensors", JSON.stringify(action.payload.sensors));

*// localStorage.setItem("token", JSON.stringify(action.payload.token));*

      return {

        ...state,

        isAuthenticated: true,

        user: action.payload.user,

        sensors: action.payload.sensors

      };

    case "LOGOUT":

      localStorage.clear();

      return {

        ...state,

        isAuthenticated: false,

        user: null

      };

    default:

      return state;

  }

};

function App() {

*//Creating The use reducer hook to pass into the Auth Context*

*//State is the initial state we declared ,dispatch is the reducer function we created above*

*//Passing all this to the rest of the applicatin*

  const [state, dispatch] = React.useReducer(reducer, initialState);

  return (

    <Router>

      <AuthContext.Provider

*value*={{

          state,

          dispatch

        }}

      >

        <div *className*="App">

          <Switch>

            <Route *path*="/Dashboard">

              <Dashboard />

            </Route>

            <Route *path*="/">

              <WelcomePage />

            </Route>

          </Switch>

        </div>

      </AuthContext.Provider>

    </Router>

  );

}

export default App;

**index.js**

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

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

ReactDOM.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>,

  document.getElementById('root')

);

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

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

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

serviceWorker.unregister();

Components – Molecules –

**Sensor.jsx**

import React from 'react'

import "../../App.css"

export default function Sensor({*sensorData*}) {

    const [bgColour , setBgColour]  = React.useState("#05020F")

    React.useEffect(()=>{

*//If reading is above 5 set the background color to RED if not keep the default*

        if ( Number( sensorData.reading ) > 5)

        {

            setBgColour("#FF0000")

        }

        else

        {

            setBgColour("#05020F")

        }

    },[sensorData.reading])

    return (

        <div *className*="sensorBox" *style*={{background:bgColour}}>

            <div *className* = "txt1">

                {sensorData.status}

            </div>

            <div *className* = "txt2" >

                <div *className* = "sensorMiniBox">

                    <div>

                        {sensorData.reading}

                    </div>

                    <div *className* = "txt3">

                        {sensorData.sensorType}

                        <br/>

                        level

                    </div>

                </div>

            </div>

            <div *className* = "txt4">

                Room {sensorData.room}

            </div>

            <div *className* = "txt4" *style* = {{fontWeight:"lighter"}}>

                {sensorData.sensorUID}

            </div>

        </div>

    )

}

Components – Organisms –

**LoginForm.jsx**

import React from 'react'

import { Redirect } from 'react-router-dom'

import "../../App.css"

import logo from "../../images/asset.svg"

*//Importing Auth Context from App.js*

import { AuthContext } from "../../App.js"

const initialState = {

  username: "",

  password: "",

  isSubmitting: false,

  errorMessage: null,

  toDashboard: false

};

export default function LoginForm() {

  const [data, setData] = React.useState(initialState);

  const { dispatch } = React.useContext(AuthContext);

  const handleInputChange = *event* => {

    setData({

      ...data,

      [event.target.name]: event.target.value

    });

  }

  const handleFormSubmit = *event* => {

    event.preventDefault();

    setData({

      ...data,

      isSubmitting: true,

      errorMessage: null

    });

*//Loggin In the User*

    fetch("http://localhost:8080/loginAdmin", {

      method: "post",

      mode: 'cors',

      headers: {

        "Content-Type": "application/json"

      },

      body: JSON.stringify({

        username: data.username,

        password: data.password

      })

    })

      .then(*res* => {

        console.log(res)

*//If Credentials valid Fetching the senor Data*

        if (res.ok) {

          fetch(`http://localhost:5000/api/getSensorsByUsername/${data.username}`,

            {

              method: "get",

            })

            .then(

              (*response*) => {

                response.json().then(

                  (*res*) => {

*//Calling the reducer in the Auth context*

                    dispatch({

                      type: "LOGIN",

                      payload: { sensors: res, user: data.username.trim() }

                    })

*//sending to Dashboard*

                    setData({

                      ...data,

                      toDashboard: true

                    });

                  });

              }

            )

            .catch(

              (*err*) => {

                console.log(err)

              }

            )

        }

        else {

          alert("Username or password doesnt match !")

          throw res;

        }

      })

  };

  return (

    <div *className*="loginFormContainer">

      <form *className*="loginFormContainerInternal">

        <img *src*={logo} />

        <div>

          <h3  >Username</h3>

          <br />

          <input

*type*="text"

*value*={data.username}

*onChange*={handleInputChange}

*name*="username"

*id*="username"

*placeholder*="username" />

        </div>

        <div>

          <br />

          <h3 >Password</h3>

          <br />

          <input *type*="password"

*placeholder*="Password"

*value*={data.password}

*onChange*={handleInputChange}

*name*="password"

*id*="password"

          />

        </div>

        <div>

          <br />

          <button *onClick*={handleFormSubmit} *type*="submit">Login</button>

          {data.toDashboard ? <Redirect *to*='/Dashboard' /> : <></>}

        </div>

      </form>

    </div>

  )

}

**SensorBoard.jsx**

import React from 'react'

import Sensor from '../Molecules/Sensor'

import "../../App.css"

import {AuthContext} from '../../App.js'

export default function SensorBoard(*props*) {

    const { state , dispatch } = React.useContext(AuthContext);

    const [sensorElements , setSensorElements] = React.useState([])

    React.useEffect(() => {

*//console.log("You should re render every 10 seconds")*

*//Grab only the sensors for the relevet floor*

        let sensor\_array = state.sensors.filter(

            (*sensor*) =>   sensor.floor === props.floor

        )

        let sensor\_elements = [];

*//For each sensor render a Sensor Element*

        sensor\_array.forEach(*element* => {

            sensor\_elements.push(

                <>

                <Sensor

*key* = {element.sensorUID}

*sensorData* = {element}

                    />

                <br/>

                </>

            )

        });

        setSensorElements(sensor\_elements)

    },[state])

    return (

        <div>

            <h2>{`Floor ${props.floor}`}</h2>

            <br/>

            <div *className*="sensorBoard">

                {sensorElements}

            </div>

        </div>

    )

}

Pages –

**Dashboard.jsx**

import React from 'react'

import "../App.css"

import SensorBoard from '../Components/Organisms/SensorBoard'

import { AuthContext } from "../App.js"

import useInterval from '../hooks/useInterval.js'

export default function Dashboard() {

  const { state, dispatch } = React.useContext(AuthContext);

  const [sensorBoards, setSensorBoards] = React.useState([]);

*// Time between two request to fetch data from api in seconds*

  let refresh\_period = 3;

*//This custom react hook is from Dan abromov personel blog https://overreacted.io/making-setinterval-declarative-with-react-hooks/*

  useInterval(() => {

    console.log(`Run Me every ${refresh\_period} seconds`)

    let user = state.user;

    console.log(state)

*//fetching Data From Sensor API*

    fetch(`http://localhost:5000/api/getSensorsByUsername/${user}`,

            {

              method: "get",

            })

            .then(

              (*response*) => {

                console.log(response)

                response.json().then(

                  (*res*) => {

                    console.log(res)

*//Updating the senors (LOGIN reducer method is called for first time fetch and subsequent periodic refetch)*

                    dispatch({

                      type: "LOGIN",

                      payload: { sensors: res, user: state.user }

                    })

                  });

              }

            )

            .catch(

              (*err*) => {

                console.log(err)

              }

            )

  }, refresh\_period \* 1000);

  React.useEffect(() => {

    let floors = []

    state.sensors.forEach(*element* => {

      floors.push(element.floor)

    });

*//console.log(floors)*

*//Removes duplicates and sorts array*

    function uniq(*a*) {

      return a.sort().filter(function (*item*, *pos*, *ary*) {

        return !pos || item != ary[pos - 1];

      })

    }

    let sorted\_unique\_floors = uniq(floors)

*//console.log(sorted\_unique\_floors)*

    let SensorBoards = [];

    sorted\_unique\_floors.forEach(*element* => {

*//console.log(<SensorBoard key = {element} floor = {element}/>)*

      SensorBoards.push(<SensorBoard *key*={element} *floor*={element} />)

    });

    setSensorBoards(SensorBoards);

*//console.log(SensorBoards)*

  }, [])

  return (

    <div *className*="dashboard">

      <h2 *style*={{fontWeight:"700"}}>{JSON.parse(window.localStorage.getItem('user'))}</h2>

      <br />

      {sensorBoards}

    </div>

  )

}

**WelcomePage.jsx**

import React from 'react'

import LoginForm from '../Components/Organisms/LoginForm'

import ".././App.css"

export default function WelcomePage() {

    return (

        <div *className*="homeGridContainer">

            <LoginForm/>

        </div>

    )

}

hooks –

**useInterval.js**

import React, { useState, useEffect, useRef } from 'react';

export default function useInterval(*callback*, *delay*) {

  const savedCallback = useRef();

*// Remember the latest callback.*

  useEffect(() => {

    savedCallback.current = callback;

  }, [callback]);

*// Set up the interval.*

  useEffect(() => {

    function tick() {

      savedCallback.current();

    }

    if (delay !== null) {

      let id = setInterval(tick, delay);

      return () => clearInterval(id);

    }

  }, [delay]);

}