Assignment 2

Fire Alarm Monitoring System

Distributed Systems

Group ID: WE_09

Group Members

- IT18004182 Perera H.C.S
- IT18113228 Wasala W.M.D.C
- IT18133400 De Silva T.S.D
- IT18165258 Rajapakshe R.M.P.R.L

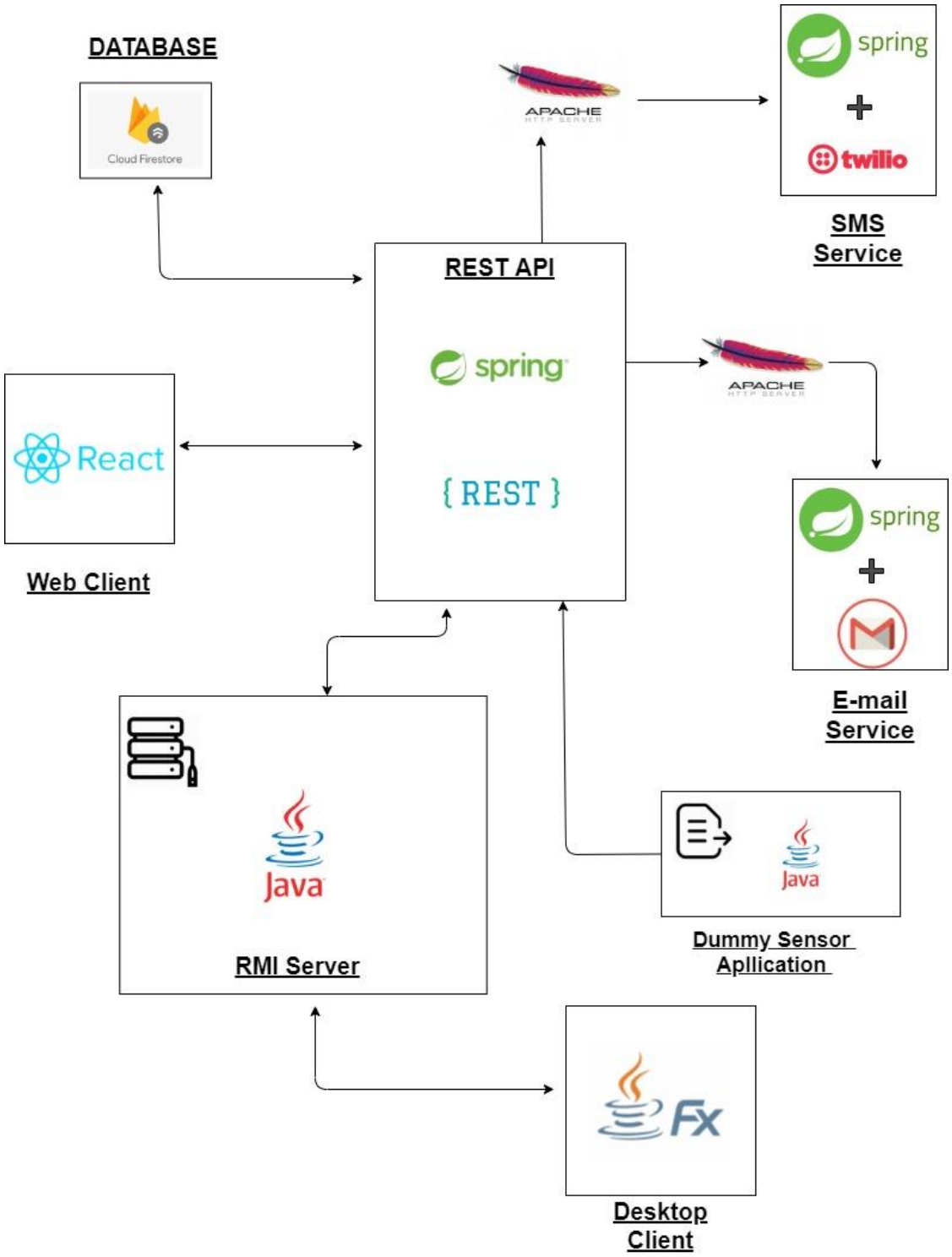
Content **page**

1.0 Introduction

The Fire Alarm Monitoring System (FAMS) is implemented using Spring Boot for the Rest API, Java and JavaFX for RMI Server and Desktop Client, ReactJS for Web Client and Spring for SMS Server and Email Server and the Firebase as the database. This system allows admin to manage the fire alarm system in the building. When the smoke level or CO2 Level goes more than 5 then admin will be notified about that. Admin will get the notification to his dashboard and also, he gets SMS and Email as well. Then admin can see which sensor has gone high with number of sensor level increased.

New user can register to the system using valid email, phone number and other required details and then user can log into system and in lead to dashboard. In that dashboard admin can see all existing sensor overview.

2.0 High Level Architectural Diagram


DATABASE
Cloud Firestore
APACHE
spring
twilio
SMS
Service
REST API
spring
{ REST }
React
Web Client
APACHE
spring
E-mail
Service
Java
RMI Server
Java
Dummy Sensor
Apllication
Fx
Desktop
Client

3.0 Tools & Technologies

- Rest API
  - Spring framework
  - Postman
  - IntelliJ IDEA
- Web Client
  - VS Code
  - Semantic UI
  - Axios HTTP Client
- RMI Server
  - Java
  - IntelliJ
- Database
  - Firebase Firestore
- Desktop Application
  - JavaFX
  - Jfoenix UI library
  - IntelliJ IDEA
- Email Service
  - Spring
  - Gmail
  - Eclipse
  - Apache
- SMS Service
  - Spring
  - Twilio
  - Eclipse

4.0 System Components

4.1 Rest API

Rest API is developed using Java and Spring Boot Framework. It runs on port 8080. Rest API has all Admin, Sensor, SensorData, Room, Floor CRUD operations and send-email notification as well. Rest API has access to Firebase database. RMI Server and Web client could access Firebase through this Rest API

Rest API End Points

End point	Description
http://localhost:8080/createAdmin	Creates New admin
http://localhost:8080/deleteAdmin	Delete existing by admin ID
http://localhost:8080/updateAdmin	Updates admin
http://localhost:8080/getAdmin	Get admin by admin ID
http://localhost:8080/getAllAdmins	Get admins list
http://localhost:8080/login	Admin login
http://localhost:8080/createSensor	Create new sensor
http://localhost:8080/deleteSensor	Delete sensor by ID
http://localhost:8080/updateSensor	Update sensor
http://localhost:8080/getSensor	Get sensor by id
http://localhost:8080/getAllSensors	Get sensors list
http://localhost:8080/createRoom	Create new room
http://localhost:8080/deleteRoom	Delete room by id
http://localhost:8080/updateRoom	Update room
http://localhost:8080/getRoom	Get room by id
http://localhost:8080/getAllRooms	Get rooms list
http://localhost:8080/createSensorData	Create new sensor data
http://localhost:8080/deleteSensorData	Delete sensor data by id
http://localhost:8080/updateSensorData	Update sensor data
http://localhost:8080/getSensorData	Get sensor data by id
http://localhost:8080/getAllSensorData	Get sensor data list
http://localhost:8080/getAllLatestSensorData	Get latest sensor data
http://localhost:9090/send-mail	Send request to email server

4.2 Web Client

Web Client has implemented in ReactJS and Sementic UI has used as css framework. Through Rest API which runs on port 8080. Web client gets sensor data from that rest api. In an every 40 seconds it's automatically fetch sensors data again. Web clients runs on port 3000. To request http request web client has used Axios http client

4.3 RMI Server

RMI Server implemented using Java with design patterns such as Factory, Repository, Façade, and singleton

4.4 Desktop Application

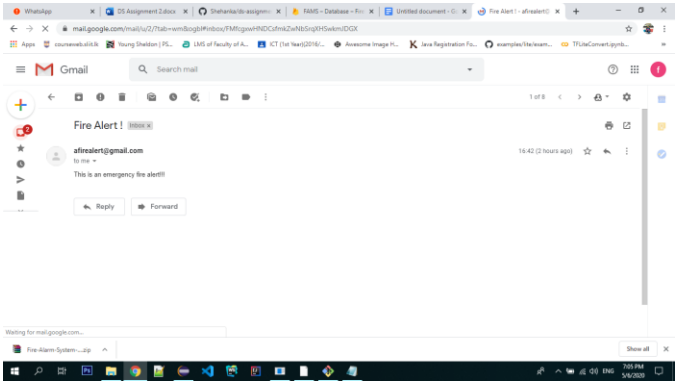
The desktop application has an Admin login. So, the unauthorized users cannot access the system where it provides a security for the application. This provides a live dashboard which displays the available sensors, their details and status. This status updates in every 15 sec. The sensor become active when CO2 or the smoke value reaches above 5. The admin can add sensors by proving the sensor name, room number and floor number. The admin can update and delete details. Also, the admin is privileged to add new room and floors also.

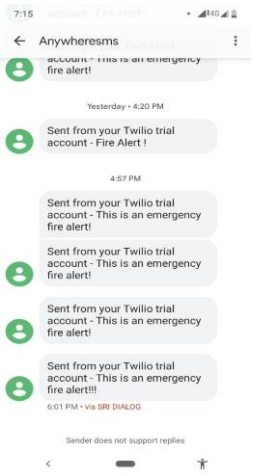
4.5 Email Service

When CO2 level or smoke level reaches above 5 then automatically Email will be sent to Admin' Gmail account. The email service component developed using spring boot and it is connected to the Rest API. In here spring email dependency and web dependency are used to initialize the email service component and google Gmail service is used to send email alert to admin.

4.6 SMS Service

A web API called Twilio is used to implement the SMS Service in this fire alarm system. This service also runs on the spring boot framework which is connected to the REST API. Same as the email service, when CO2 level or smoke level reaches above level 5 admin will be notified by SMS. In here also spring web dependency and Twilio SMS dependency is used to implement this SMS component.





5.0 Sequence of Execusion

When a service is called from the Desktop Client it will be gone through the service interface inside the common module which is exposed by the server according to the service class whether it is a sensor service, admin service or a room service. It is done by the Service Factory implemented inside RMI Server which is exposed to the outside. All the request go through this factory implementation. Then the repository factory will forward the request to the endpoint of Rest Api according to the service type explained earlier.

Web Client will communicate only with the Rest Api.

6.0 Authentication and Security

The Fire Alarm System Application has provided an admin login which is validated for authorized user only. Then the RMI Server and Client calls are protected using privacy policies separately. They are allowed only to communicate through the port defined.

7.0 Appendix

Rest API

Controller Classes

AdminController.java

```
package com.fantastic4.restapi.controller;
import com.fantastic4.restapi.dto.Admin;
import com.fantastic4.restapi.service.FirebaseInitialize;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.concurrent.ExecutionException;


@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class AdminController {
    FirebaseInitialize firebaseInitialize = new FirebaseInitialize();
    @PostMapping("/createAdmin")
    public String createAdmin(@RequestBody Admin admin) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.addAdmin(admin);
    }

    @DeleteMapping("/deleteAdmin")
    public String deleteAdmin(@RequestParam String id) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.deleteAdmin(id);
    }
```

```
    @PutMapping("/updateAdmin")
    public String updateAdmin(@RequestBody Admin admin) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.updateAdmin(admin);
    }
    @GetMapping("/getAdmin")
    public Admin getAdmin(@RequestParam String id) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getAdminByID(id);
    }

    @GetMapping("/getAllAdmins")
    public ArrayList<Admin> getAllAdmins() throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getAllAdmins();
    }

    @PutMapping("/login")
    public String login(@RequestBody Admin admin) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.login(admin);
    }}
```

EmailController.java

```
package com.fantastic4.restapi.controller;import
org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.RestController;

@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class EmailController {
}
```

FloorController.java

```
package com.fantastic4.restapi.controller;
import com.fantastic4.restapi.dto.Floor;
import com.fantastic4.restapi.service.FirebaseInitialize;
import org.springframework.web.bind.annotation.*;
import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class FloorController {
    FirebaseInitialize firebaseInitialize = new FirebaseInitialize();
    @PostMapping("/createFloor")
```

```
    public String addFloor(@RequestBody Floor floor) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.addFloor(floor);
    }

    @DeleteMapping("/deleteFloor")
    public String deleteFloor(@RequestParam String floorID) throws
ExecutionException, InterruptedException {
        return firebaseInitialize.deleteFloor(floorID);
    }
    @PutMapping("/updateFloor")
    public String updateFloor(@RequestBody Floor floor) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.updateFloor(floor);
    }
    @GetMapping("/getFloor")
    public Floor getFloor(@RequestParam String floorID) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getFloorByID(floorID);
    }

    @GetMapping("/getAllFloors")
    public ArrayList<Floor> getAllFloors() throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getAllFloors();
    }}
```

RoomController.java

```
package com.fantastic4.restapi.controller;

import com.fantastic4.restapi.dto.Room;
import com.fantastic4.restapi.service.FirebaseInitialize;
import org.springframework.web.bind.annotation.*;


import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class RoomController {

    FirebaseInitialize firebaseInitialize = new FirebaseInitialize();

    @PostMapping("/createRoom")
    public String addRoom(@RequestBody Room room) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.addRoom(room);
```

```
    }
    @DeleteMapping("/deleteRoom")
    public String deleteRoom(@RequestParam String roomID) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.deleteRoom(roomID);
    }
    @PutMapping("/updateRoom")
    public String updateRoom(@RequestBody Room room) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.updateRoom(room);
    }
    @GetMapping("/getRoom")
    public Room getRoom(@RequestParam String roomID) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getRoomByID(roomID);
    }
    @GetMapping("/getAllRooms")
    public ArrayList<Room> getAllRooms() throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getAllRooms();
    }}
```

SensorController.java

```
package com.fantastic4.restapi.controller;

import com.fantastic4.restapi.dto.Room;
import com.fantastic4.restapi.dto.Sensor;
import com.fantastic4.restapi.dto.SensorData;
import com.fantastic4.restapi.service.FirebaseInitialize;
import org.springframework.web.bind.annotation.*;
import java.util.ArrayList;
import java.util.concurrent.ExecutionException;

@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class SensorController {

    FirebaseInitialize firebaseInitialize = new FirebaseInitialize();

    @PostMapping("/createSensor")
    public String addSensor(@RequestBody Sensor sensor) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.addSensor(sensor);
    }
    @DeleteMapping("/deleteSensor")
    public String removeSensor(@RequestBody String sensorID) throws
ExecutionException, InterruptedException {
        return firebaseInitialize.deleteSensor(sensorID);
```

```
    }

    @PutMapping("/updateSensor")
    public String updateSensor(@RequestBody Sensor sensor) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.updateSensor(sensor);
    }

    @GetMapping("/getSensor")
    public Sensor getSensor(@RequestBody String sensorID) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getSensorByID(sensorID);
    }

    @GetMapping("/getAllSensors")
    public ArrayList<Sensor> getAllSensors() throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getAllSensors();
    }

    @PostMapping("/createRoom")
    public String addRoom(@RequestBody Room room) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.addRoom(room);
    }

    @DeleteMapping("/deleteRoom")
    public String deleteRoom(@RequestParam String roomID) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.deleteRoom(roomID);
    }

    @PutMapping("/updateRoom")
    public String updateRoom(@RequestBody Room room) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.updateRoom(room);
    }

    @GetMapping("/getRoom")
    public Room getRoom(@RequestParam String roomID) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getRoomByID(roomID);
    }

    @GetMapping("/getAllRooms")
    public ArrayList<Room> getAllRooms() throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getAllRooms();
    }
```

```
    @PostMapping("/createSensorData")
    public String createSensorData(SensorData sensorData) throws ExecutionException,
InterruptedException {
        return firebaseInitialize.addSensorData(sensorData);
    }

    @DeleteMapping("/deleteSensorData")
    public String deleteSensorData(@RequestParam String id) throws
ExecutionException, InterruptedException {
        return firebaseInitialize.deleteSensorData(id);
    }

    @PutMapping("/updateSensorData")
    public String updateSensorData(@RequestBody SensorData sensorData) throws
ExecutionException, InterruptedException {
        return firebaseInitialize.updateSensorData(sensorData);
    }

    @GetMapping("/getSensorData")
    public SensorData getSensorData(@RequestParam String id) throws
ExecutionException, InterruptedException {
        return firebaseInitialize.getSensorDataByID(id);
    }

    @GetMapping("/getAllSensorData")
    public ArrayList<SensorData> getAllSensorData() throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getAllSensorData();
    }

    @GetMapping("/getAllLatestSensorData")
    public ArrayList<SensorData> getAllLatestSensorData() throws ExecutionException,
InterruptedException {
        return firebaseInitialize.getAllLatestSensorData();
    }

}
```

SMSController.java

```
package com.fantastic4.restapi.controller;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.RestController;

@CrossOrigin(origins = "*", allowedHeaders = "*")
@RestController
public class SMSController {
}
```

DTO

Admin.java

```
package com.fantastic4.restapi.dto;

public class Admin {
    private String adminID;
    private String name;
    private String address;
    private String email;
    private String contactNo;
    private String password;

    public Admin() {
    }

    public Admin(String adminID, String name, String address, String email, String
contactNo, String password) {
        this.adminID = adminID;
        this.name = name;
        this.address = address;
        this.email = email;
        this.contactNo = contactNo;
        this.password = password;
    }

    public String getAdminID() {
        return adminID;
    }

    public void setAdminID(String adminID) {
        this.adminID = adminID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getAddress() {
        return address;
    }
```

```
    public void setAddress(String address) {
        this.address = address;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getContactNo() {
        return contactNo;
    }

    public void setContactNo(String contactNo) {
        this.contactNo = contactNo;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Floor.java

```
package com.fantastic4.restapi.dto;
public class Floor {
    private String floorID;
    public Floor() {
    }
    public Floor(String floorID) {
        this.floorID = floorID;
    }
    public String getFloorID() {
        return floorID;
    }
    public void setFloorID(String floorID) {
        this.floorID = floorID;
    }
}
```

Room.java

```
package com.fantastic4.restapi.dto;

public class Room {
    private String roomNo;
    private int floorNo;
    private String sensorID;

    public Room() {
    }

    public Room(String roomNo, int floorNo, String sensorID) {
        this.roomNo = roomNo;
        this.floorNo = floorNo;
        this.sensorID = sensorID;
    }

    public String getRoomNo() {
        return roomNo;
    }

    public void setRoomNo(String roomNo) {
        this.roomNo = roomNo;
    }

    public int getFloorNo() {
        return floorNo;
    }

    public void setFloorNo(int floorNo) {
        this.floorNo = floorNo;
    }

    public String getSensorID() {
        return sensorID;
    }
    public void setSensorID(String sensorID) {
        this.sensorID = sensorID;
    }
}
```

Sensor.java

```
package com.fantastic4.restapi.dto;

import java.util.List;
```

```
public class Sensor {

    private String sensorID;
     private int floorNo;
     private int roomNo;
     private int latestCO2Level;
     private int latestSmokeLevel;
     private boolean status;

    public Sensor() {
     }

    public Sensor(String sensorID, int floorNo, int roomNo, int
latestCO2Level, int latestSmokeLevel, boolean status) {
          this.sensorID = sensorID;
          this.floorNo = floorNo;
          this.roomNo = roomNo;
          this.latestCO2Level = latestCO2Level;
          this.latestSmokeLevel = latestSmokeLevel;
          this.status = status;
     }

    public String getSensorID() {
          return sensorID;
     }

    public void setSensorID(String sensorID) {
          this.sensorID = sensorID;
     }

    public int getFloorNo() {
          return floorNo;
     }

    public void setFloorNo(int floorNo) {
          this.floorNo = floorNo;
     }

    public int getRoomNo() {
          return roomNo;
     }
```

```
    public void setRoomNo(int roomNo) {
         this.roomNo = roomNo;
     }

    public int getLatestCO2Level() {
         return latestCO2Level;
     }

    public void setLatestCO2Level(int co2Level) {
         this.latestCO2Level = co2Level;
     }

    public int getLatestSmokeLevel() {
         return latestSmokeLevel;
     }

    public void setLatestSmokeLevel(int smokeLevel) {
         this.latestSmokeLevel = smokeLevel;
     }

    public void setStatus(boolean status) {
         this.status = status;
     }

    public boolean getStatus() {
         return status;
     }

}
```

SensorData.java

```
package com.fantastic4.restapi.dto;

public class SensorData {
     private String sensorDataID;
     private String sensorID;
     private int floorNo;
     private int roomNo;
     private int co2Level;
     private int smokeLevel;
     private String date;
```

```
     private boolean status;

    public SensorData() {
     }

    public SensorData(String sensorDataID, String sensorID, int floorNo, int
roomNo, int co2Level, int smokeLevel, String date, boolean status) {
          this.sensorDataID = sensorDataID;
          this.sensorID = sensorID;
          this.floorNo = floorNo;
          this.roomNo = roomNo;
          this.co2Level = co2Level;
          this.smokeLevel = smokeLevel;
          this.date = date;
          this.status = status;
     }

    public String getSensorDataID() {
          return sensorDataID;
     }

    public void setSensorDataID(String sensorDataID) {
          this.sensorDataID = sensorDataID;
     }

    public String getSensorID() {
          return sensorID;
     }

    public void setSensorID(String sensorID) {
          this.sensorID = sensorID;
     }

    public int getFloorNo() {
          return floorNo;
     }

    public void setFloorNo(int floorNo) {
          this.floorNo = floorNo;
     }

    public int getRoomNo() {
          return roomNo;
```

```
    }

    public void setRoomNo(int roomNo) {
        this.roomNo = roomNo;
    }

    public int getCo2Level() {
        return co2Level;
    }

    public void setCo2Level(int co2Level) {
        this.co2Level = co2Level;
    }

    public int getSmokeLevel() {
        return smokeLevel;
    }

    public void setSmokeLevel(int smokeLevel) {
        this.smokeLevel = smokeLevel;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

    public boolean isStatus() {
        return status;
    }

    public void setStatus(boolean status) {
        this.status = status;
    }
}
```

Service

FirebaseConfig.java

```
package com.fantastic4.restapi.service;

import com.google.cloud.firestore.Firestore;
 import com.google.firebase.cloud.FirestoreClient;
 import org.springframework.context.annotation.Bean;
 import org.springframework.context.annotation.Configuration;

@Configuration
 public class FirebaseConfig {

    @Bean
     public Firestore getFirestore() {
         return FirestoreClient.getFirestore();
     }

}
```

FirebaseInitialize.java

```
package com.fantastic4.restapi.service;

import com.fantastic4.restapi.dto.Admin;
 import com.fantastic4.restapi.dto.Room;
 import com.fantastic4.restapi.dto.Sensor;
 import com.fantastic4.restapi.dto.SensorData;
 import com.google.api.core.ApiFuture;
 import com.google.auth.oauth2.GoogleCredentials;
 import com.google.cloud.firestore.*;
 import com.google.firebase.FirebaseApp;
 import com.google.firebase.FirebaseOptions;
 import com.google.firebase.cloud.FirestoreClient;
 import org.springframework.stereotype.Service;

import javax.annotation.PostConstruct;
 import java.io.FileInputStream;
 import java.util.ArrayList;
 import java.util.List;
 import java.util.concurrent.ExecutionException;

@Service
 public class FirebaseInitialize {
```

```
    static Firestore firestore;

    @PostConstruct
     public void initialize() {
         try {
             FileInputStream serviceAccount =
                     new FileInputStream("./serviceAccountKey.json");

            FirebaseOptions options = new FirebaseOptions.Builder()

.setCredentials(GoogleCredentials.fromStream(serviceAccount))
                     .setDatabaseUrl("https://online-organizer-
caefa.firebaseio.com")
                     .build();

            FirebaseApp.initializeApp(options);
             firestore = FirestoreClient.getFirestore();
         } catch (Exception e) {
             e.printStackTrace();
         }
     }

    public String addSensor(Sensor sensor) throws ExecutionException,
InterruptedException {
         ApiFuture<WriteResult> collectionApiFuture = firestore
                 .collection("sensors")
                 .document(sensor.getSensorID())
                 .set(sensor);

        return collectionApiFuture.get().getUpdateTime().toString();
     }

    public String addSensorData(SensorData sensorData) throws
ExecutionException, InterruptedException {
         ApiFuture<WriteResult> collectionApiFuture = firestore
                 .collection("sensorData")
                 .document(sensorData.getSensorDataID())
                 .set(sensorData);

        return collectionApiFuture.get().getUpdateTime().toString();
     }

    public String addRoom(Room room) throws ExecutionException,
```

```
InterruptedException {
         ApiFuture<WriteResult> collectionApiFuture = firestore
                 .collection("rooms")
                 .document(room.getRoomNo())
                 .set(room);

        return collectionApiFuture.get().getUpdateTime().toString();
     }

    public String addAdmin(Admin admin) throws ExecutionException,
InterruptedException {
         ApiFuture<WriteResult> collectionApiFuture = firestore
                 .collection("admins")
                 .document(admin.getAdminID())
                 .set(admin);

        return collectionApiFuture.get().getUpdateTime().toString();
     }

    public Sensor getSensorByID(String sensorID) throws ExecutionException,
InterruptedException {
         DocumentReference documentReference = firestore
                 .collection("sensors")
                 .document(sensorID);
         ApiFuture<DocumentSnapshot> future = documentReference.get();
         Sensor sensor = null;

        DocumentSnapshot document = future.get();
         if (document.exists()) {
             sensor = document.toObject(Sensor.class);
         }

        return sensor;
     }

    public SensorData getSensorDataByID(String id) throws ExecutionException,
InterruptedException {
         DocumentReference documentReference = firestore
                 .collection("sensorData")
                 .document("sensorData")
                 .collection(id)
                 .document("PBMFtgsGm3aivlibW54b");
         ApiFuture<DocumentSnapshot> future = documentReference.get();
```

```
        SensorData sensorData = null;

        DocumentSnapshot document = future.get();
        if (document.exists()) {
            sensorData = document.toObject(SensorData.class);
        }
        return sensorData;
    }

    public Room getRoomByID(String id) throws ExecutionException,
InterruptedException {
        DocumentReference documentReference = firestore
                .collection("rooms")
                .document(id);
        ApiFuture<DocumentSnapshot> future = documentReference.get();
        Room room = null;

        DocumentSnapshot document = future.get();
        if (document.exists()) {
            room = document.toObject(Room.class);
        }

        return room;
    }

    public Admin getAdminByID(String id) throws ExecutionException,
InterruptedException {
        DocumentReference documentReference = firestore
                .collection("admins")
                .document(id);
        ApiFuture<DocumentSnapshot> future = documentReference.get();
        Admin admin = null;

        DocumentSnapshot document = future.get();
        if (document.exists()) {
            admin = document.toObject(Admin.class);
        }

        return admin;
    }

    public String updateSensor(Sensor sensor) throws ExecutionException,
InterruptedException {
```

```
        ApiFuture<WriteResult> future = firestore
                .collection("sensors")
                .document(sensor.getSensorID())
                .set(sensor);
        return future.get().getUpdateTime().toString();
    }

    public String updateSensorData(SensorData sensorData) throws
ExecutionException, InterruptedException {
        ApiFuture<WriteResult> future = firestore
                .collection("sensorData")
                .document(sensorData.getSensorDataID())
                .set(sensorData);
        return future.get().getUpdateTime().toString();
    }

    public String updateRoom(Room room) throws ExecutionException,
InterruptedException {
        ApiFuture<WriteResult> future = firestore
                .collection("rooms")
                .document(room.getRoomNo())
                .set(room);
        return future.get().getUpdateTime().toString();
    }

    public String updateAdmin(Admin admin) throws ExecutionException,
InterruptedException {
        ApiFuture<WriteResult> future = firestore
                .collection("admins")
                .document(admin.getAdminID())
                .set(admin);
        return future.get().getUpdateTime().toString();
    }

    public String deleteSensor(String sensorID) throws ExecutionException,
InterruptedException {
        ApiFuture<WriteResult> future = firestore
                .collection("sensors")
                .document(sensorID).delete();

        return future.get().getUpdateTime().toString();
    }
```

```
    public String deleteSensorData(String id) throws ExecutionException,
InterruptedException {
         ApiFuture<WriteResult> future = firestore
                 .collection("sensorData")
                 .document(id).delete();

        return future.get().getUpdateTime().toString();
     }

    public String deleteRoom(String id) throws ExecutionException,
InterruptedException {
         ApiFuture<WriteResult> future = firestore
                 .collection("rooms")
                 .document(id).delete();

        return future.get().getUpdateTime().toString();
     }

    public String deleteAdmin(String id) throws ExecutionException,
InterruptedException {
         ApiFuture<WriteResult> future = firestore
                 .collection("admins")
                 .document(id).delete();

        return future.get().getUpdateTime().toString();
     }

    public ArrayList<Sensor> getAllSensors() throws ExecutionException,
InterruptedException {
         ApiFuture<QuerySnapshot> futures = firestore
                 .collection("sensors").get();

        ArrayList<Sensor> sensorArrayList = new ArrayList<>();

        List<QueryDocumentSnapshot> documents = futures.get().getDocuments();
         documents.forEach(doc -> {
             System.out.println(doc.get("sen"));
             sensorArrayList.add(doc.toObject(Sensor.class));
         });

        return sensorArrayList;
     }
```

```
    public ArrayList<SensorData> getAllSensorData() throws
ExecutionException, InterruptedException {
         ApiFuture<QuerySnapshot> futures = firestore
                 .collection("sensors")
                 .document("S001")
                 .collection("sensorData").orderBy("date")
                 .get();

        ArrayList<SensorData> sensorDataArrayList = new ArrayList<>();

        List<QueryDocumentSnapshot> documents = futures.get().getDocuments();
         documents.forEach(doc -> {
             sensorDataArrayList.add(doc.toObject(SensorData.class));
         });

        return sensorDataArrayList;
     }

    public ArrayList<SensorData> getAllLatestSensorData() throws
ExecutionException, InterruptedException {
         ArrayList<SensorData> sensorDataLatestArrayList = new ArrayList<>();

        ArrayList<Sensor> sensorArrayList = getAllSensors();
         sensorArrayList.forEach((sensor) -> {
             ApiFuture<QuerySnapshot> futures = firestore
                     .collection("sensors")
                     .document(sensor.getSensorID())
                     .collection("sensorData").orderBy("date")
                     .limit(1)
                     .get();

            List<QueryDocumentSnapshot> documents = null;
             try {
                 documents = futures.get().getDocuments();
             } catch (InterruptedException | ExecutionException e) {
                 e.printStackTrace();
             }
             assert documents != null;
             documents.forEach(doc -> {

sensorDataLatestArrayList.add(doc.toObject(SensorData.class));
             });
         });
```

```
        return sensorDataLatestArrayList;
     }

    public ArrayList<Room> getAllRooms() throws ExecutionException,
InterruptedException {
         ApiFuture<QuerySnapshot> futures = firestore
                 .collection("rooms").get();

        ArrayList<Room> roomArrayList = new ArrayList<>();

        List<QueryDocumentSnapshot> documents = futures.get().getDocuments();
         documents.forEach(doc -> {
             roomArrayList.add(doc.toObject(Room.class));
         });

        return roomArrayList;
     }

    public ArrayList<Admin> getAllAdmins() throws ExecutionException,
InterruptedException {
         ApiFuture<QuerySnapshot> futures = firestore
                 .collection("admins").get();

        ArrayList<Admin> adminArrayList = new ArrayList<>();

        List<QueryDocumentSnapshot> documents = futures.get().getDocuments();
         documents.forEach(doc -> {
             adminArrayList.add(doc.toObject(Admin.class));
         });

        return adminArrayList;
     }

    public String login(Admin admin) {

        return "true";
     }
 }
```

RestAPI Application

RestAPIApplication.java

```
package com.fantastic4.restapi;

import org.springframework.boot.SpringApplication;
 import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
 public class RestAPIApplication {

    public static void main(String[] args) {
         SpringApplication.run(RestAPIApplication.class, args);
     }

}
```

WEB-CLIENT

index.html

```
<!DOCTYPE html>
 <html lang="en">
   <head>
     <meta charset="utf-8" />
     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
     <meta name="viewport" content="width=device-width, initial-scale=1" />
     <meta name="theme-color" content="#000000" />
     <meta
       name="description"
       content="Web site created using create-react-app"
     />
     <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
     <!--
       manifest.json provides metadata used when your web app is installed on
a
       user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
     -->
     <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
     <!--
       Notice the use of %PUBLIC_URL% in the tags above.
       It will be replaced with the URL of the `public` folder during the
```

```
build.
       Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
       work correctly both with client-side routing and a non-root public
URL.
       Learn how to configure a non-root public URL by running `npm run
build`.
     -->
     <link
       rel="stylesheet"
       href="https://cdn.jsdelivr.net/npm/semantic-
ui@2.4.2/dist/semantic.min.css"
     />

    <style>

    </style>
     <title>FAMS Web</title>
   </head>
   <body>
     <noscript>You need to enable JavaScript to run this app.</noscript>
     <div id="root"></div>
     <!--
       This HTML file is a template.
       If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
       The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
       To create a production bundle, use `npm run build` or `yarn build`.
     -->
     <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></scri
pt>
     <script src="https://cdn.jsdelivr.net/npm/semantic-
ui@2.4.2/dist/semantic.min.js"></script>
   </body>
 </html>
```

App.js

```
import React from 'react';
 import './App.css';
 import Sensors from './components/sensors/Sensors';
 import Header from './components/header/Header';

function App() {
   return (
     <div className='App'>
       <Header />
       <br/>
       <Sensors />
     </div>
   );
 }

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')
 );

serviceWorker.unregister();
```

Components

Header.jsx

```
import React, { Component } from 'react';

class Header extends Component {
```

```
  state = {};
  render() {
    return (
      <header>
        <div className='ui blue inverted menu'>
          <div className='header item'>FAMS Web </div>
          <a className='item' href='!#'>
            About Us
          </a>
          <a className='item' href='!#'>
            Sensors
          </a>
          <a className='item' href='!#'>
            Locations
          </a>
        </div>
      </header>
    );
  }
}

export default Header;
```

Sensors.jsx

```
import React, { Component } from 'react';
 import Axios from 'axios';

class Sensors extends Component {
  interval = null;
  state = {
    sensors: [],
  };

  componentDidMount() {
    this.interval = setInterval(this.getData, 40000);
    this.getData();
  }

  componentWillUnmount() {
    clearInterval(this.interval);
  }
```

```
  getData = () => {
     Axios.get('http://localhost:8080/getAllSensors')
       .then((res) => {
         console.log(res.data);
         this.setState({
           sensors: res.data,
         });
       })
       .catch((err) => console.error(err));
   };

  render() {
     return (
       <div className='ui container'>
         <table className='ui celled inverted grey table'>
           <thead>
             <tr>
               <th>Sensor ID</th>
               <th>Status</th>
               <th>Location</th>
               <th>Smoke Level</th>
               <th>
                 CO<sub>2</sub> Level
               </th>
             </tr>
           </thead>
           <tbody>
             {this.state.sensors.map((sensorData, i) => (
               <tr
                 key={i}
                 style={{
                   backgroundColor:
                     sensorData.latestCO2Level > 5 ||
                     sensorData.latestSmokeLevel > 5
                       ? '#e84118'
                       : 'teal',
                 }}
               >
                 <td data-label='Sensor ID'>{sensorData.sensorID}</td>
                 <td>{sensorData.status === true ? 'Active' :
'Deactivated'}</td>
                 <td>
```

```
                  Floor No: {sensorData.floorNo}
                  <br /> Room No: {sensorData.roomNo}
                </td>
                <td>{sensorData.latestSmokeLevel}</td>
                <td>{sensorData.latestCO2Level}</td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    );
  }
}

export default Sensors;
```

RMI Serer

Desktop Application

AdminController.java

```
package com.fantastic4.desktop.controller;

import com.fantastic4.common.dto.AdminDTO;
import com.fantastic4.common.services.ServicesFactory;
import com.fantastic4.common.services.custom.AdminService;
import com.fantastic4.desktop.proxy.ProxyHandler;

import java.util.List;
```

```
public class AdminController {

   private static AdminService adminService;

   public static boolean createAdmin(AdminDTO adminDTO) throws
Exception {
      adminService = (AdminService) ProxyHandler.getInstance()
            .getService(ServicesFactory.ServicesType.ADMIN);

      return adminService.addAdmin(adminDTO);
   }

   public static boolean updateAdmin(AdminDTO adminDTO) throws
Exception {
      adminService = (AdminService) ProxyHandler.getInstance()
            .getService(ServicesFactory.ServicesType.ADMIN);

      return adminService.updateAdmin(adminDTO);
   }

   public static boolean deleteAdmin(String adminID) throws Exception {
      adminService = (AdminService) ProxyHandler.getInstance()
            .getService(ServicesFactory.ServicesType.ADMIN);
```

```
        return adminService.deleteAdmin(adminID);
    }

    public static List<AdminDTO> getAllAdmins() throws Exception {
        adminService = (AdminService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.ADMIN);

        return adminService.getAllAdmins();
    }

    public static AdminDTO findAdmin(String adminID) throws Exception
{
        adminService = (AdminService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.ADMIN);

        return adminService.findAdminByID(adminID);
    }

    public static AdminDTO login(AdminDTO adminDTO) throws
Exception {
        adminService = (AdminService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.ADMIN);
```

```
        return adminService.login(adminDTO);
    }

}
```

FloorController.java

```
package com.fantastic4.desktop.controller;

import com.fantastic4.common.dto.FloorDTO;
import com.fantastic4.common.services.ServicesFactory;
import com.fantastic4.common.services.custom.FloorService;
import com.fantastic4.desktop.proxy.ProxyHandler;

import java.util.List;

public class FloorController {

    private static FloorService floorService;

    public static boolean addFloor(FloorDTO floorDTO) throws Exception
{
        floorService = (FloorService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.FLOOR);
```

```
        return floorService.addFloor(floorDTO);
    }

    public static boolean updateFloor(FloorDTO floorDTO) throws
Exception {
        floorService = (FloorService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.FLOOR);

        return floorService.updateFloor(floorDTO);
    }

    public static boolean deleteFloor(String id) throws Exception {
        floorService = (FloorService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.FLOOR);

        return floorService.deleteFloor(id);
    }

    public static FloorDTO getFloorByID(String id) throws Exception {
        floorService = (FloorService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.FLOOR);

        return floorService.findFloorByID(id);
```

```
    }

    public static List<FloorDTO> getAllFloors() throws Exception {
        floorService = (FloorService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.FLOOR);

        return floorService.getAllFloors();
    }
}
```

RoomController.java

```
package com.fantastic4.desktop.controller;

import com.fantastic4.common.dto.RoomDTO;
import com.fantastic4.common.services.ServicesFactory;
import com.fantastic4.common.services.custom.RoomService;
import com.fantastic4.desktop.proxy.ProxyHandler;

import java.util.List;

public class RoomController {

    private static RoomService roomService;
```

```
    public static boolean addRoom(RoomDTO roomDTO) throws
Exception {
        roomService = (RoomService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.ROOM);

        return roomService.addRoom(roomDTO);
    }

    public static boolean updateRoom(RoomDTO roomDTO) throws
Exception {
        roomService = (RoomService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.ROOM);

        return roomService.updateRoom(roomDTO);
    }

    public static boolean deleteRoom(String id) throws Exception {
        roomService = (RoomService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.ROOM);

        return roomService.deleteRoom(id);
    }

    public static RoomDTO getRoomByID(String id) throws Exception {
```

```
        roomService = (RoomService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.ROOM);

        return roomService.findRoomByID(id);
    }

    public static List<RoomDTO> getAllRooms() throws Exception {
        roomService = (RoomService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.ROOM);

        return roomService.getAllRooms();
    }

}
```

SensorController.java

```
package com.fantastic4.desktop.controller;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.common.dto.SensorDataDTO;
import com.fantastic4.desktop.proxy.ProxyHandler;
import com.fantastic4.common.services.ServicesFactory;
import com.fantastic4.common.services.custom.SensorService;
```

```
import java.util.List;

public class SensorController {

   private static SensorService sensorService;

   public static boolean createSensor(SensorDTO sensorDTO) throws
Exception {
      sensorService = (SensorService) ProxyHandler.getInstance()
            .getService(ServicesFactory.ServicesType.SENSOR);

      return sensorService.addSensor(sensorDTO);
   }

   public static boolean updateSensor(SensorDTO sensorDTO) throws
Exception {
      sensorService = (SensorService) ProxyHandler.getInstance()
            .getService(ServicesFactory.ServicesType.SENSOR);

      return sensorService.updateSensor(sensorDTO);
   }

   public static List<SensorDTO> getAllSensors() throws Exception {
```

```
        sensorService = (SensorService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.SENSOR);

        return sensorService.getAllSensors();
    }


    public static SensorDTO getSensorByID(String id) throws Exception {
        sensorService = (SensorService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.SENSOR);

        return sensorService.findSensorByID(id);
    }

    public static boolean deleteSensor(String id) throws Exception {
        sensorService = (SensorService) ProxyHandler.getInstance()
                .getService(ServicesFactory.ServicesType.SENSOR);

        return sensorService.deleteSensor(id);
    }
}
```

AdminFloorResgister.java

```
package com.fantastic4.desktop.fxml.controller;

import com.fantastic4.common.dto.FloorDTO;
import com.fantastic4.desktop.controller.FloorController;
import com.fantastic4.desktop.main.App;
import javafx.fxml.FXML;
import javafx.scene.control.TextField;
import java.io.IOException;
import javafx.fxml.Initializable;
import java.util.ResourceBundle;
import java.net.URL;

public class AdminFloorResgister implements Initializable{

    @FXML
    private TextField floorNoText;

    @FXML
    private void addFloor() throws Exception{

        int floor =  Integer.parseInt(floorNoText.getText());
        FloorDTO floorDTO = new FloorDTO();
        floorDTO.setFloorNo(floor);
```

```
        boolean status = FloorController.addFloor(floorDTO);
        if(status){
            App.setRoot("roommanager");
        }
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {

    }

    @FXML
    private void switchToRoomManager() throws IOException {
        App.setRoot("roommanager");
    }
}
```

AdminRoomRegister.java

```
package com.fantastic4.desktop.fxml.controller;

import com.fantastic4.common.dto.RoomDTO;
import com.fantastic4.common.dto.FloorDTO;
```

```
import com.fantastic4.desktop.controller.FloorController;
import com.fantastic4.desktop.controller.RoomController;
import com.fantastic4.desktop.main.App;
import javafx.fxml.FXML;
import javafx.scene.control.TextField;
import javafx.scene.control.ComboBox;
import java.io.IOException;
import javafx.fxml.Initializable;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import java.util.ResourceBundle;
import java.net.URL;

public class AdminRoomRegister implements Initializable{

    @FXML
    private TextField roomRegText;

    @FXML
    private ComboBox<String> roomFloorSelect;

    FloorController floorDTO;
```

```
    ObservableList<String> floorData =
FXCollections.observableArrayList();

    @FXML
    private void addRoom() throws Exception {

        int selectedFloor =  Integer.parseInt(roomFloorSelect.getValue());

        RoomDTO roomDTO = new RoomDTO();
        roomDTO.setRoomNo(roomRegText.getText());
        roomDTO.setFloorNo(selectedFloor);

        boolean status = RoomController.addRoom(roomDTO);
        if(status){
            App.setRoot("roommanager");
        }
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        try {
            floorData = (ObservableList)floorDTO.getAllFloors();
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
        roomFloorSelect.setItems(floorData);
    }

    @FXML
    private void switchToRoomManager() throws IOException {
        App.setRoot("roommanager");
    }

}
```

AdminSensorRegister.java

```
package com.fantastic4.desktop.fxml.controller;

import com.fantastic4.common.dto.RoomDTO;
import com.fantastic4.common.dto.FloorDTO;
import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.desktop.controller.FloorController;
import com.fantastic4.desktop.controller.RoomController;
import com.fantastic4.desktop.controller.SensorController;
import com.fantastic4.desktop.main.App;
import javafx.fxml.FXML;
import javafx.scene.control.TextField;
```

```
import javafx.scene.control.ComboBox;
import java.io.IOException;
import javafx.fxml.Initializable;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import java.util.ResourceBundle;
import java.net.URL;

public class AdminSensorRegister implements Initializable {

    @FXML
    private TextField sensorNameText;

    @FXML
    private ComboBox sensorFloorSelect, sensorRoomSelect;

    FloorController floorDTO;
    RoomController roomDTO;

    ObservableList<String> floorData =
FXCollections.observableArrayList();
    ObservableList<String> roomData =
FXCollections.observableArrayList();
```

```
    @FXML
    private void addSensor() throws Exception {

        int selectedFloor =  Integer.parseInt((String)
sensorFloorSelect.getValue());
        int selectedRoom =  Integer.parseInt((String)
sensorRoomSelect.getValue());

        SensorDTO sensorDTO = new SensorDTO();
        sensorDTO.setSensorID(sensorNameText.getText());
        sensorDTO.setFloorNo(selectedFloor);
        sensorDTO.setRoomNo(selectedRoom);

        boolean status = SensorController.createSensor(sensorDTO);
        if(status){
            App.setRoot("roommanager");
        }
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        try {
            floorData = (ObservableList)floorDTO.getAllFloors();
            roomData = (ObservableList)roomDTO.getAllRooms();
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        sensorFloorSelect.setItems(floorData);
        sensorRoomSelect.setItems(roomData);
    }

    @FXML
    private void switchToRoomManager() throws IOException {
        App.setRoot("roommanager");
    }
}
```

AdminSensorUpdate.java

```
package com.fantastic4.desktop.fxml.controller;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.desktop.controller.FloorController;
import com.fantastic4.desktop.controller.RoomController;
import com.fantastic4.desktop.controller.SensorController;
import com.fantastic4.desktop.main.App;
```

```
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

public class AdminSensorUpdate implements Initializable {

    SensorController sensorController;
    SensorDTO sensorDTO;

    FloorController floorDTO;
    RoomController roomDTO;

    ObservableList<String> floorData =
FXCollections.observableArrayList();
    ObservableList<String> roomData =
FXCollections.observableArrayList();
```

```
    @FXML
    TextField sensorNameText;

    @FXML
    private ComboBox sensorFloorSelect, sensorRoomSelect;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        try {
            floorData = (ObservableList)floorDTO.getAllFloors();
            roomData = (ObservableList)roomDTO.getAllRooms();
        } catch (Exception e) {
            e.printStackTrace();
        }
        sensorFloorSelect.setItems(floorData);
        sensorRoomSelect.setItems(roomData);
    }

    @FXML
    private void updateSensor() throws Exception {

        int selectedFloor =  Integer.parseInt((String)
sensorFloorSelect.getValue());
```

```
        int selectedRoom =  Integer.parseInt((String)
sensorRoomSelect.getValue());

        SensorDTO sensorDTO = new SensorDTO();
        sensorDTO.setSensorID(sensorNameText.getText());
        sensorDTO.setFloorNo(selectedFloor);
        sensorDTO.setRoomNo(selectedRoom);

        boolean status = SensorController.updateSensor(sensorDTO);
        if(status){
            App.setRoot("roommanager");
        }
    }

    public void getSensorID(String id) throws Exception {
        sensorDTO = sensorController.getSensorByID(id);
        sensorNameText.setText(sensorDTO.getSensorID());
        sensorFloorSelect.setValue(sensorDTO.getFloorNo());
        sensorRoomSelect.setValue(sensorDTO.getRoomNo());

    }

    @FXML
    private void switchToRoomManager() throws IOException {
```

```
        App.setRoot("roommanager");
    }
}
```

AlertBox.java

```
package com.fantastic4.desktop.fxml.controller;

import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class AlertBox {

    public static void display(String title, String msg) {
        Stage stage = new Stage();

        stage.initModality(Modality.APPLICATION_MODAL);
        stage.setTitle(title);
        stage.setMaxWidth(300);
```

```
        stage.setMaxHeight(250);

        Label label = new Label();
        label.setText(msg);
        Button close = new Button("OK");
        close.setOnAction(e -> stage.close());

        VBox box = new VBox(10);
        box.getChildren().addAll(label, close);
        box.setAlignment(Pos.CENTER);

        Scene scene = new Scene(box);
        stage.setScene(scene);
        stage.showAndWait();
    }
}
```

DashboardUIController.java

```
package com.fantastic4.desktop.fxml.controller;

import com.jfoenix.controls.JFXDrawer;
import com.jfoenix.controls.JFXHamburger;
```

```
import
com.jfoenix.transitions.hamburger.HamburgerSlideCloseTransition;
import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.input.MouseEvent;
import javafx.scene.layout.AnchorPane;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

public class DashboardUIController implements Initializable{
    @FXML
    public AnchorPane rootPane;

    @FXML
    public AnchorPane parameterizedPane;

    @FXML
    public AnchorPane paneTopHeader;

    @FXML
```

```
    private JFXDrawer drawer;

    @FXML
    private JFXHamburger hamburger;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        sideMenuActions();
        loadMenuUI();
    }

    private void sideMenuActions(){
        HamburgerSlideCloseTransition transition = new
HamburgerSlideCloseTransition(hamburger);
        try {
            AnchorPane sideAnchorPane = FXMLLoader.load(getClass()
                    .getResource("/fxml/SideMenuUI.fxml"));
            drawer.setSidePane(sideAnchorPane);
            drawer.close();
            drawer.setVisible(false);
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
        transition.setRate(-0.7);
        hamburger.addEventHandler(MouseEvent.MOUSE_PRESSED, (e)->
{
            transition.setRate(transition.getRate() * -1);
            transition.play();

            if (!drawer.isOpened()) {
                AnchorPane sideAnchorPane;
                try {
                    sideAnchorPane = FXMLLoader.load(getClass()
                            .getResource("/fxml/SideMenuUI.fxml"));
                    drawer.setVisible(true);
                    drawer.setSidePane(sideAnchorPane);
                    drawer.open();
                } catch (IOException e1) {
                    e1.printStackTrace();
                }
            } else {
                drawer.close();
                drawer.setVisible(false);
            }
        });
```

```
}

private void loadMenuUI(){
   try {
      AnchorPane paneLogin = FXMLLoader
            .load(getClass().getResource("/fxml/MenuUI.fxml"));
      parameterizedPane.getChildren().setAll(paneLogin);
   } catch (IOException e) {
      e.printStackTrace();
   }
}

void setAnchorPaneTo(AnchorPane anchorPane) {

   if (anchorPane != null) {
      try {
         parameterizedPane.getChildren().setAll(anchorPane);
      } catch (Exception e) {
         e.printStackTrace();
      }
   } else {
      System.out.println("Anchor pane is null");
   }
```

```
    }

    void logOutAction(){
        AnchorPane loginPane = null;
        try {
            loginPane = FXMLLoader
                    .load(getClass().getResource("/fxml/LoginUI.fxml"));
        } catch (IOException e) {
            e.printStackTrace();
        }
        rootPane.getChildren().setAll(loginPane);
    }

    @FXML
    private void windowClose() {
        Platform.exit();
    }

    @FXML
    private void windowMinimize() {
//        primaryStage.setIconified(false);
    }
```

```
}
```

FXMLDocumentController.java

```
package com.fantastic4.desktop.fxml.controller;


import javafx.animation.FadeTransition;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import javafx.util.Duration;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;
```

```
public class FXMLDocumentController implements Initializable{

    @FXML
    private AnchorPane holderPane;
    @FXML
    private Button btnDashboard;
    @FXML
    private Button btnProfile;
    @FXML
    private Button btnSensors;
    @FXML
    private Button btnRoom;
    @FXML
    private AnchorPane profile,sensors,dashboard,rooms;

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        //setUIComponents();
    }


//    private void setUIComponents(){
```

```
//
//      //Reservation Button
//      Image reservationImage = new Image(getClass()
//              .getResourceAsStream("/images/reservation.png"));
//      btnRes.setGraphic(new ImageView(reservationImage));
//
//      //Payment Button
//      Image paymentImage = new Image(getClass()
//              .getResourceAsStream("/images/payment.png"));
//      btnPayment.setGraphic(new ImageView(paymentImage));
//
//      //Update Button
//      Image updateImage = new Image(getClass()
//              .getResourceAsStream("/images/update.png"));
//      btnUpdate.setGraphic(new ImageView(updateImage));
//
//      //Get Vehicle
//      Image vehicleImage = new Image(getClass()
//              .getResourceAsStream("/images/car.png"));
//      btnGetVehicle.setGraphic(new ImageView(vehicleImage));
//
//  }
```

```
    @FXML
    private void switchDashboard(ActionEvent event) throws IOException
{
        try {
            FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("/com/fantastic4/desktop/fxml/ui/
livedashboard.fxml"));
            Parent root1 = (Parent) fxmlLoader.load();
            Stage stage = new Stage();
            stage.setTitle("Live Sensor Data Dashboard");
            stage.setScene(new Scene(root1));
            stage.show();
        }catch (Exception e){
            System.out.println("Cannot Load Dashboard");
        }
    }

    @FXML
    private void switchRooms(ActionEvent event) {
        try {
            FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("/com/fantastic4/desktop/fxml/ui/
roommanager.fxml"));
```

```
            Parent root1 = (Parent) fxmlLoader.load();
            Stage stage = new Stage();
            stage.setTitle("Manage Rooms");
            stage.setScene(new Scene(root1));
            stage.show();
        }catch (Exception e){
            System.out.println("Cannot Load Rooms Manager");
        }
    }

    @FXML
    private void switchSensors(ActionEvent event) {
        try {
            FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("/com/fantastic4/desktop/fxml/ui/
sensormanager.fxml"));
            Parent root1 = (Parent) fxmlLoader.load();
            Stage stage = new Stage();
            stage.setTitle("Manage Sensors");
            stage.setScene(new Scene(root1));
            stage.show();
        }catch (Exception e){
            System.out.println("Cannot Load Sensors Manager");
```

```
        }
    }

    @FXML
    private void switchProfile(ActionEvent event) {
        try {
            FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("/com/fantastic4/desktop/fxml/ui/
profile.fxml"));
            Parent root1 = (Parent) fxmlLoader.load();
            Stage stage = new Stage();
            stage.setTitle("Admin Profile");
            stage.setScene(new Scene(root1));
            stage.show();
        }catch (Exception e){
            System.out.println("Cannot Load Admin Profile");
        }
    }

}
```

LiveDashboardController.java

```
package com.fantastic4.desktop.fxml.controller;
```

```
import com.fantastic4.common.dto.AdminDTO;
import com.fantastic4.common.dto.RoomDTO;
import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.desktop.controller.AdminController;
import com.fantastic4.desktop.controller.RoomController;
import com.fantastic4.desktop.controller.SensorController;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.AnchorPane;
import javafx.scene.text.Text;

import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.ResourceBundle;
```

```
public class LiveDashboardController implements Initializable {

    @FXML
    private Text totalSensors,inactiveSensors,totalAdmins,totalRooms;

    @FXML
    private TableView<SensorTableModel> tblSensor;

    @FXML
    private TableColumn<SensorTableModel, String> colSensorID;

    @FXML
    private TableColumn<SensorTableModel, Integer> colFloor;

    @FXML
    private TableColumn<SensorTableModel, String> colRoom;

    @FXML
    private TableColumn<SensorTableModel, Integer> colSmoke;

    @FXML
```

```
    private TableColumn<SensorTableModel, Integer> colCO2;

    @FXML
    private TableColumn<SensorTableModel, String> colStatus;

    @FXML
    private TableColumn<SensorTableModel, String> colSensorStatus;

    private List<SensorDTO> iSensors = new ArrayList<>();
    private List<SensorDTO> tSensors = new ArrayList<>();
    private List<RoomDTO> tRooms = new ArrayList<>();
    private List<AdminDTO> tAdmins = new ArrayList<>();
    private ObservableList<SensorTableModel>
sensorTableModelObservableList = FXCollections.observableArrayList();

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {

        try {
            tSensors = SensorController.getAllSensors();
            for (SensorDTO sensor:tSensors
                ) {
                System.out.println(sensor.getSensorID());
                if(sensor.getStatus() == false){
```

```
                iSensors.add(sensor);
            }
        }
        //tAdmins = AdminController.getAllAdmins();
        //tRooms = RoomController.getAllRooms();

        totalSensors.setText(Integer.toString(tSensors.size()));
        totalRooms.setText(Integer.toString(tRooms.size()));
        totalAdmins.setText(Integer.toString(tAdmins.size()));
        inactiveSensors.setText(Integer.toString(iSensors.size()));

        loadSensorTableView();

    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void loadSensorTableView(){

    colSensorID.setCellValueFactory(new
PropertyValueFactory<>("sensorID"));
    colFloor.setCellValueFactory(new
PropertyValueFactory<>("floor"));
```

```
        colRoom.setCellValueFactory(new
PropertyValueFactory<>("room"));
        colSmoke.setCellValueFactory(new
PropertyValueFactory<>("smoke"));
        colCO2.setCellValueFactory(new PropertyValueFactory<>("CO2"));
        colStatus.setCellValueFactory(new
PropertyValueFactory<>("status"));
        colSensorStatus.setCellValueFactory(new
PropertyValueFactory<>("sensorStatus"));

        tblSensor.setItems(sensorTableModelObservableList);

        try {
            List<SensorDTO> sensorDTOList =
SensorController.getAllSensors();
            for (SensorDTO sensorDTO: sensorDTOList
            ) {
                SensorTableModel sensorTableModel = new
SensorTableModel();
                sensorTableModel.setSensorID(sensorDTO.getSensorID());
                sensorTableModel.setFloor(sensorDTO.getFloorNo());

sensorTableModel.setRoom(String.valueOf(sensorDTO.getRoomNo()));

sensorTableModel.setSmoke(sensorDTO.getLatestSmokeLevel());
```

```
            sensorTableModel.setCO2(sensorDTO.getLatestCO2Level());
            if(sensorDTO.getLatestSmokeLevel() >= 5 ||
sensorDTO.getLatestCO2Level() >= 5){
               sensorTableModel.setStatus("DANGER");
            }else{
               sensorTableModel.setStatus("NORMAL");
            }
            if(sensorDTO.getStatus()==true){
               sensorTableModel.setSensorStatus("ACTIVE");
            }else{
               sensorTableModel.setSensorStatus("INACTIVE");
            }

            sensorTableModelObservableList.add(sensorTableModel);
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
   }

   private void refreshTableView(){
      tblSensor.setItems(null);
      sensorTableModelObservableList.clear();
```

```
        loadSensorTableView();
    }

    @FXML
    private void getSelectedItem() {
        String branchId =
tblSensor.getSelectionModel().getSelectedItem().getSensorID();
    }
}
```

ManageSensorsController.java

```
package com.fantastic4.desktop.fxml.controller;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.desktop.controller.SensorController;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
```

```
import javafx.scene.control.cell.PropertyValueFactory;

import java.net.URL;
import java.util.List;
import java.util.ResourceBundle;

public class ManageSensorsController implements Initializable {

    @FXML
    public TextField sensorID;

    @FXML
    public TextField floorNo;

    @FXML
    public TextField roomNo;


    @FXML
    private TableView<SensorsTableModel> tblSensors;

    @FXML
    private TableColumn<SensorsTableModel, String> colSensorID;
```

```
    @FXML
    private TableColumn<SensorsTableModel, Integer> colFloor;

    @FXML
    private TableColumn<SensorsTableModel, Integer> colRoom;

    private ObservableList<SensorsTableModel>
sensorsTableModelObservableList =
FXCollections.observableArrayList();

    @Override
    public void initialize(URL location, ResourceBundle resources) {
        loadSensorsTable();
    }


    @FXML
    private void addAction() {
        SensorDTO sensorDTO = new SensorDTO(
                sensorID.getText(),
                Integer.parseInt(floorNo.getText()),
                Integer.parseInt(roomNo.getText()),
                0,
                0,
```

```
            true

        );

        try {
            boolean isAdded = SensorController.createSensor(sensorDTO);
            if (isAdded){
//                new AlertBuilder("info","ManageCustomer","Customer
Add",
//                        "Customer added successfully");
                refreshTableView();

                sensorID.setText("");
                roomNo.setText("");
                floorNo.setText("");
            }else {
//                new AlertBuilder("error","Manage Customer", "Customer
Add",
//                        "Customer couldn't add");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

```
    }

    @FXML
    private void deleteAction() {
        try {
            boolean isDeleted =
SensorController.deleteSensor(sensorID.getText());

            if (isDeleted){
//                new AlertBuilder("info","ManageCustomer","Customer
Delete",
//                        "Customer deleted successfully");
                refreshTableView();
            }else {
//                new AlertBuilder("error","ManageCustomer","Customer
Delete",
//                        "Customer couldn't delete");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @FXML
```

```
    private void searchById() {
        try {
            SensorDTO sensorDTO =
SensorController.getSensorByID(sensorID.getText());
            if (sensorDTO != null){
                sensorID.setText(sensorDTO.getSensorID());
                floorNo.setText(String.valueOf(sensorDTO.getFloorNo()));
                roomNo.setText(String.valueOf(sensorDTO.getRoomNo()));

            }else {
//                new AlertBuilder("warn","Manage Customer","Customer
Search",
//                        "Customer couldn't found");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @FXML
    private void updateAction() {
        SensorDTO sensorDTO = new SensorDTO(
                sensorID.getText(),
```

```
            Integer.parseInt(floorNo.getText()),
            Integer.parseInt(roomNo.getText()),
            0,
            0,
            true
        );

        try {
            boolean isUpdated =
SensorController.updateSensor(sensorDTO);

            if (isUpdated){
//                new AlertBuilder("info","Manage Customer","Customer
Update",
//                        "Customer updated successfully");
                refreshTableView();
            }else {
//                new AlertBuilder("info","Manage Customer","Customer
Update",
//                        "Customer couldn't update.");
            }
        } catch (Exception e) {
            e.printStackTrace();
```

```
    }
  }

  private void loadSensorsTable(){
    colSensorID.setCellValueFactory(new
PropertyValueFactory<>("sensorID"));
    colFloor.setCellValueFactory(new
PropertyValueFactory<>("floor"));
    colRoom.setCellValueFactory(new
PropertyValueFactory<>("room"));

    tblSensors.setItems(sensorsTableModelObservableList);

    try {
      List<SensorDTO> sensorDTOList =
SensorController.getAllSensors();
      for (SensorDTO sensorDTO: sensorDTOList
      ) {
        SensorsTableModel sensorsTableModel = new
SensorsTableModel();
        sensorsTableModel.setSensorID(sensorDTO.getSensorID());
        sensorsTableModel.setFloor(sensorDTO.getFloorNo());
        sensorsTableModel.setRoom(sensorDTO.getRoomNo());
```

```
            sensorsTableModelObservableList.add(sensorsTableModel);
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
    }

    @FXML
    private void getSelectedItem() {
      String sensor =
tblSensors.getSelectionModel().getSelectedItem().getSensorID();
      sensorID.setText(sensor);
      searchById();
    }


    private void refreshTableView(){
      tblSensors.setItems(null);
      sensorsTableModelObservableList.clear();
      loadSensorsTable();
    }
}
```

PrimaryController.java

```
package com.fantastic4.desktop.fxml.controller;

import com.fantastic4.common.dto.AdminDTO;
import com.fantastic4.desktop.controller.AdminController;
import com.fantastic4.desktop.main.App;
import javafx.fxml.FXML;
import javafx.scene.control.TextField;

import java.io.IOException;

public class PrimaryController {
    @FXML
    private TextField email;
    @FXML
    private TextField password;

    @FXML
    private void switchToSecondary() throws Exception {
            App.setRoot("fxmldocument");

    }
```

```
    @FXML
    private void login() throws Exception{
        AdminDTO adminDTO = new AdminDTO();
        adminDTO.setEmail(email.getText());
        adminDTO.setPassword(email.getText());
        AdminDTO admin = AdminController.login(adminDTO);
        if(admin.getName()!=null){
            App.setRoot("fxmldocument");
        }
    }

}
```

Roommanager.java

```
package com.fantastic4.desktop.fxml.controller;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.desktop.controller.SensorController;
import com.fantastic4.desktop.main.App;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
```

```
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Stage;

import java.io.IOException;
import java.net.URL;
import java.util.ResourceBundle;

public class roommanager implements Initializable {

    @FXML
    TableView tableView;

    @FXML
    TextField searchSensorText;

    @FXML
```

```
    Button editSensor, deleteSensor, addSensor, addRoom, addFloor;

    @FXML
    TableColumn id, sensorName, roomNumber, floorNumber;

    SensorController sensorController;
    AlertBox alertBox;

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {

        SensorController sensorController = null;

        TableColumn id = new TableColumn("id");
        TableColumn sensor = new TableColumn("sensorName");
        TableColumn room = new TableColumn("roomNumber");
        TableColumn floor = new TableColumn("floorNumber");
        tableView.getColumns().addAll(id, sensor, room, floor);

        ObservableList<SensorDTO> sensorDetails =
FXCollections.observableArrayList();
        try {
            sensorDetails = (ObservableList)
sensorController.getAllSensors();
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        id.setCellValueFactory(new PropertyValueFactory<SensorDTO,
String>("id"));
        sensor.setCellValueFactory(new PropertyValueFactory<SensorDTO,
String>("name"));
        room.setCellValueFactory(new PropertyValueFactory<SensorDTO,
String>("room"));
        floor.setCellValueFactory(new PropertyValueFactory<SensorDTO,
String>("floor"));

        tableView.setItems(sensorDetails);

    }

    @FXML
    private void switchToAddRoom() throws IOException {
        App.setRoot("AdminRoomRegister");
    }

    @FXML
    private void switchToAddFloor() throws IOException{
        App.setRoot("AdminFloorResgister");
```

```
    }

    @FXML
    private void switchToAddSensor() throws IOException{
        App.setRoot("AdminSensorRegister");
    }

    @FXML
    private void switchToDeleteSensor() throws Exception {
        String sensor = searchSensorText.getText();
        boolean result = sensorController.deleteSensor(sensor);
        if(result) {
            alertBox.display("Deleted","Sensor Successfully deleted!");
        }

    }

    @FXML
    private void switchToEditSensor() throws Exception {
        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/com/fantastic4/desktop/fxml/ui/
AdminSensorUpdate.fxml"));
        Parent root = loader.load();
```

```
        AdminSensorUpdate sensorUpdate = loader.getController();
        sensorUpdate.getSensorID(searchSensorText.getText());

        Stage stage = new Stage();
        stage.setScene(new Scene(root));
        stage.show();
    }
}
```

SecondaryController.java

```
package com.fantastic4.desktop.fxml.controller;

import com.fantastic4.desktop.main.App;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.layout.Pane;

import java.io.IOException;

public class SecondaryController {
    @FXML
    private Pane pnl_home,pnl_sensors,pnl_rooms;
```

```
    @FXML
    private Button btn_home,btn_manage_sensors,btn_manage_rooms;

    @FXML
    private void handleButtonAction(ActionEvent event){
        if(event.getSource() == btn_home){
            pnl_home.toFront();
        }
        else if (event.getSource() == btn_manage_rooms){
            pnl_rooms.toFront();
        }
        else if(event.getSource() == btn_manage_sensors){
            pnl_sensors.toFront();
        }
    }

    @FXML
    private void switchToPrimary() throws IOException {
        App.setRoot("primary");
    }
}
```

SensorDataController.java

```
package com.fantastic4.desktop.fxml.controller;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.desktop.controller.SensorController;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.control.ListCell;
import javafx.scene.control.ProgressIndicator;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Color;
import javafx.scene.text.Text;

import java.io.IOException;
import java.net.URL;
import java.util.List;
import java.util.ResourceBundle;

public class SensorDataController extends ListCell<SensorDTO>{

    @FXML
    private Text sensorName,floorNo,roomNo,status;
    @FXML
```

```
    private ProgressIndicator smokeValue,co2Value;


    @Override
    protected void updateItem(SensorDTO sensorDTO, boolean empty) {
        super.updateItem(sensorDTO, empty);
        if(empty || sensorDTO == null) {

            setText(null);
            setGraphic(null);

        } else {
            FXMLLoader mLLoader = null;
            if (mLLoader == null) {
                mLLoader = new
FXMLLoader(getClass().getResource("/com/fantastic4/desktop/fxml/ui/
sensormanager.fxml"));
                mLLoader.setController(this);

                try {
                    mLLoader.load();
                } catch (IOException e) {
                    e.printStackTrace();
                }
```

```
        }

        sensorName.setText(sensorDTO.getSensorID());
        floorNo.setText(String.valueOf(sensorDTO.getFloorNo()));
        roomNo.setText(String.valueOf(sensorDTO.getRoomNo()));

smokeValue.setProgress(sensorDTO.getLatestSmokeLevel()/10.0);
        co2Value.setProgress(sensorDTO.getLatestCO2Level()/10.0);


      }
   }
}
```

SensorsTableModel.java

```
package com.fantastic4.desktop.fxml.controller;

import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;

public class SensorsTableModel {
   private SimpleStringProperty sensorID = new
SimpleStringProperty("");
```

```
    private SimpleIntegerProperty floor = new SimpleIntegerProperty(0);
    private SimpleIntegerProperty room = new SimpleIntegerProperty(0);

    public SensorsTableModel(){
    }

    public SensorsTableModel(SimpleStringProperty sensorID,
SimpleIntegerProperty floor, SimpleIntegerProperty room) {
        this.sensorID = sensorID;
        this.floor = floor;
        this.room = room;
    }

    public String getSensorID() {
        return sensorID.get();
    }

    public SimpleStringProperty sensorIDProperty() {
        return sensorID;
    }

    public void setSensorID(String sensorID) {
        this.sensorID.set(sensorID);
    }
```

```
public int getFloor() {
   return floor.get();
}

public SimpleIntegerProperty floorProperty() {
   return floor;
}

public void setFloor(int floor) {
   this.floor.set(floor);
}

public int getRoom() {
   return room.get();
}

public SimpleIntegerProperty roomProperty() {
   return room;
}

public void setRoom(int room) {
   this.room.set(room);
```

```
    }
}
```

SensorTableModel.java

```
package com.fantastic4.desktop.fxml.controller;

import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;

public class SensorTableModel {

    private SimpleStringProperty sensorID = new
SimpleStringProperty("");
    private SimpleIntegerProperty floor = new SimpleIntegerProperty(0);
    private SimpleStringProperty room = new SimpleStringProperty("");
    private SimpleIntegerProperty smoke = new
SimpleIntegerProperty(0);
    private SimpleIntegerProperty CO2 = new SimpleIntegerProperty(0);
    private SimpleStringProperty status = new SimpleStringProperty("");
    private SimpleStringProperty sensorStatus = new
SimpleStringProperty("");

    public SensorTableModel(){
    }
```

```
    public SensorTableModel(SimpleStringProperty sensorID,
SimpleIntegerProperty floor, SimpleStringProperty room,
SimpleIntegerProperty smoke, SimpleIntegerProperty CO2,
SimpleStringProperty status, SimpleStringProperty sensorStatus) {
        this.sensorID = sensorID;
        this.floor = floor;
        this.room = room;
        this.smoke = smoke;
        this.CO2 = CO2;
        this.status = status;
        this.sensorStatus = sensorStatus;
    }

    public String getSensorID() {
        return sensorID.get();
    }

    public SimpleStringProperty sensorIDProperty() {
        return sensorID;
    }

    public void setSensorID(String sensorID) {
        this.sensorID.set(sensorID);
    }
```

```
public int getFloor() {
   return floor.get();
}

public SimpleIntegerProperty floorProperty() {
   return floor;
}

public void setFloor(int floor) {
   this.floor.set(floor);
}

public String getRoom() {
   return room.get();
}

public SimpleStringProperty roomProperty() {
   return room;
}

public void setRoom(String room) {
   this.room.set(room);
```

```
}

public int getSmoke() {
   return smoke.get();
}

public SimpleIntegerProperty smokeProperty() {
   return smoke;
}

public void setSmoke(int smoke) {
   this.smoke.set(smoke);
}

public int getCO2() {
   return CO2.get();
}

public SimpleIntegerProperty CO2Property() {
   return CO2;
}

public void setCO2(int CO2) {
```

```
    this.CO2.set(CO2);
}

public String getStatus() {
    return status.get();
}

public SimpleStringProperty statusProperty() {
    return status;
}

public void setStatus(String status) {
    this.status.set(status);
}

public String getSensorStatus() {
    return sensorStatus.get();
}

public SimpleStringProperty sensorStatusProperty() {
    return sensorStatus;
}
```

```
    public void setSensorStatus(String sensorStatus) {
        this.sensorStatus.set(sensorStatus);
    }
}
```

App.java

```
package com.fantastic4.desktop.main;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

/**
* JavaFX App
*/
public class App extends Application {

    private double xOffset = 0;
```

```
private double yOffset = 0;
private static Scene scene;

@Override
public void start(Stage stage) throws IOException {
    scene = new Scene(loadFXML("primary"));
    stage.setScene(scene);
    stage.show();

    scene.setOnMousePressed(event -> {
        xOffset = event.getSceneX();
        yOffset = event.getSceneY();
    });

    scene.setOnMouseDragged(event -> {
        stage.setX(event.getScreenX() - xOffset);
        stage.setY(event.getScreenY() - yOffset);
    });

}

public static void setRoot(String fxml) throws IOException {
    scene.setRoot(loadFXML(fxml));
```

```
    }

    private static Parent loadFXML(String fxml) throws IOException {
        FXMLLoader fxmlLoader = new
FXMLLoader(App.class.getResource("/com/fantastic4/desktop/fxml/ui/
"+fxml +".fxml"));
        return fxmlLoader.load();
    }

    public static void main(String[] args) {
        launch(args);
    }

}
```

Dashboard.java

```
package com.fantastic4.desktop.main;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
```

```
public class Dashboard extends Application {

    @Override
    public void start(Stage stage) throws Exception {
        Parent root =
FXMLLoader.load(getClass().getResource("com/fantastic4/fxmldocume
nt.fxml"));

        Scene scene = new Scene(root);

        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }

}
```

ProxyHandler.java

```
package com.fantastic4.desktop.proxy;
```

```
import com.fantastic4.common.services.ServicesFactory;
import com.fantastic4.common.services.SuperService;
import com.fantastic4.common.services.custom.AdminService;
import com.fantastic4.common.services.custom.SensorService;

import java.rmi.Naming;
import java.rmi.RMISecurityManager;

public class ProxyHandler implements ServicesFactory{

   private static ProxyHandler proxyHandler;
   private SensorService sensorService;
   private AdminService adminService;
   String host = "localhost";

   private ProxyHandler() {
      try {
         System.setProperty("java.security.policy", "file:./client.policy");
//          if(System.getSecurityManager() == null ){
//             System.setSecurityManager( new RMISecurityManager() );
//          }
         ServicesFactory servicesFactory = (ServicesFactory)
Naming.lookup("//localhost:5050/fams");
```

```
            sensorService = (SensorService)
servicesFactory.getService(ServicesType.SENSOR);
            adminService = (AdminService)
servicesFactory.getService(ServicesType.ADMIN);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static ProxyHandler getInstance(){
        if (proxyHandler == null)
            proxyHandler = new ProxyHandler();
        return proxyHandler;
    }

    @Override
    public SuperService getService(ServicesType servicesType) throws
Exception {
        switch (servicesType) {
            case SENSOR: return sensorService;
            case ADMIN: return adminService;
            default: return null;
        }
```

```
    }
}
```

Module-info.java

```
module com.fantastic4 {
    requires javafx.controls;
    requires javafx.fxml;
    requires java.rmi;
    requires FAMSCommon;
    requires com.jfoenix;
    //requires kotlin.stdlib;

    opens com.fantastic4.desktop.main to javafx.graphics;
    opens com.fantastic4.desktop.fxml.controller to javafx.fxml;
    exports com.fantastic4.desktop.fxml.controller;
}
```

FAMSCommon

Module-info.java

```
module FAMSCommon {
    requires java.rmi;
    exports com.fantastic4.common.dto;
```

```
    exports com.fantastic4.common.observer;
    exports com.fantastic4.common.reservation;
    exports com.fantastic4.common.services.custom;
    exports com.fantastic4.common.services;
}
```

AdminDTO.java

```
package com.fantastic4.common.dto;

public class AdminDTO implements SuperDTO{
    private String adminID;
    private String name;
    private String address;
    private String email;
    private String contactNo;
    private String password;

    public AdminDTO() {
    }

    public AdminDTO(String adminID, String name, String address, String
email, String contactNo, String password) {
```

```
    this.adminID = adminID;
    this.name = name;
    this.address = address;
    this.email = email;
    this.contactNo = contactNo;
    this.password = password;
}

public String getAdminID() {
    return adminID;
}

public void setAdminID(String adminID) {
    this.adminID = adminID;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}
```

```
public String getAddress() {
   return address;
}

public void setAddress(String address) {
   this.address = address;
}

public String getEmail() {
   return email;
}

public void setEmail(String email) {
   this.email = email;
}

public String getContactNo() {
   return contactNo;
}

public void setContactNo(String contactNo) {
   this.contactNo = contactNo;
```

```
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

FloorDTO.java

```
package com.fantastic4.common.dto;

public class FloorDTO implements SuperDTO {
    private int floorNo;
    private String sensorID;

    public FloorDTO() {
    }

    public FloorDTO(int floorNo) {
```

```
    this.floorNo = floorNo;
}

public int getFloorNo() {
    return floorNo;
}

public void setFloorNo(int floorNo) {
    this.floorNo = floorNo;
}

public String getSensorID() {
    return sensorID;
}

public void setSensorID(String sensorID) {
    this.sensorID = sensorID;
}

@java.lang.Override
public java.lang.String toString() {
    return "FloorDTO{" +
            "floorNo=" + floorNo +
```

```
            ", sensorID='" + sensorID + '\'' +
            '}';
    }
}
```

RoomDTO.java

```
package com.fantastic4.common.dto;

public class RoomDTO implements SuperDTO{
    private String roomNo;
    private int floorNo;
    private String sensorID;

    public RoomDTO() {
    }

    public RoomDTO(String roomNo, int floorNo, String sensorID) {
        this.roomNo = roomNo;
        this.floorNo = floorNo;
        this.sensorID = sensorID;
    }

    public String getRoomNo() {
```

```
        return roomNo;
    }

    public void setRoomNo(String roomNo) {
        this.roomNo = roomNo;
    }

    public int getFloorNo() {
        return floorNo;
    }

    public void setFloorNo(int floorNo) {
        this.floorNo = floorNo;
    }

    public String getSensorID() {
        return sensorID;
    }

    public void setSensorID(String sensorID) {
        this.sensorID = sensorID;
    }
}
```

SensorDataDTO.java

```
package com.fantastic4.common.dto;

public class SensorDataDTO implements SuperDTO{

    private String sensorDataID;
    private String sensorID;
    private int co2Level;
    private int smokeLevel;
    private String date;

    public SensorDataDTO() {
    }

    public SensorDataDTO(String sensorDataID, String sensorID, int
co2Level, int smokeLevel, String date, boolean status) {
        this.sensorDataID = sensorDataID;
        this.sensorID = sensorID;
        this.co2Level = co2Level;
        this.smokeLevel = smokeLevel;
        this.date = date;
    }
```

```
public String getSensorDataID() {
   return sensorDataID;
}

public void setSensorDataID(String sensorDataID) {
   this.sensorDataID = sensorDataID;
}

public String getSensorID() {
   return sensorID;
}

public void setSensorID(String sensorID) {
   this.sensorID = sensorID;
}

public int getCo2Level() {
   return co2Level;
}

public void setCo2Level(int co2Level) {
   this.co2Level = co2Level;
}
```

```
    public int getSmokeLevel() {
        return smokeLevel;
    }

    public void setSmokeLevel(int smokeLevel) {
        this.smokeLevel = smokeLevel;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }
}
```

SensorDTO.java

```
package com.fantastic4.common.dto;

public class SensorDTO implements SuperDTO {
```

```
    private String sensorID;
    private int floorNo;
    private int roomNo;
    private int latestCO2Level;
    private int latestSmokeLevel;
    private boolean status;

    public SensorDTO() {
    }

    public SensorDTO(String sensorID, int floorNo, int roomNo, int co2,
int smoke,boolean status) {
        this.sensorID = sensorID;
        this.floorNo = floorNo;
        this.roomNo = roomNo;
        this.latestCO2Level = co2;
        this.latestSmokeLevel = smoke;
        this.status = status;
    }

    @Override
    public String toString() {
        return "{" +
```

```
            "sensorID:'" + sensorID + '\'' +
            ", floorNo:" + floorNo +
            ", roomNo:" + roomNo +
            ", latestCO2Level:" + latestCO2Level +
            ", latestSmokeLevel:" + latestSmokeLevel +
            ", status=" + status +
            '}';
}

public String getSensorID() {
   return sensorID;
}

public void setSensorID(String sensorID) {
   this.sensorID = sensorID;
}

public int getFloorNo() {
   return floorNo;
}

public void setFloorNo(int floorNo) {
   this.floorNo = floorNo;
```

```
}

public int getRoomNo() {
   return roomNo;
}

public void setRoomNo(int roomNo) {
   this.roomNo = roomNo;
}

public void setLatestCO2Level(int co2Level) {
   this.latestCO2Level = co2Level;
}

public int getLatestCO2Level() {
   return latestCO2Level;
}

public void setLatestSmokeLevel(int smokeLevel) {
   this.latestSmokeLevel = smokeLevel;
}

public int getLatestSmokeLevel() {
```

```
        return latestSmokeLevel;
    }

    public void setStatus(boolean status) {
        this.status = status;
    }

    public boolean getStatus() {
        return status;
    }

}
```

SuperDTO.java

```
package com.fantastic4.common.dto;

import java.io.Serializable;

public interface SuperDTO extends Serializable {
}
```

SuperObserver.java

```
package com.fantastic4.common.observer;
```

```
import java.rmi.Remote;

public interface SuperObserver extends Remote {
}
```

Reservation.java

```
package com.fantastic4.common.reservation;

import java.rmi.Remote;

public interface Reservation extends Remote {

   public boolean reserve(Object id)throws Exception;

   public boolean release(Object id)throws Exception;

}
```

ServicesFactory.java

```
package com.fantastic4.common.services;

import java.rmi.Remote;
```

```
public interface ServicesFactory extends Remote {
   enum ServicesType {
      SENSOR,
      ADMIN,
      ROOM,
      FLOOR,
      SENSOR_DATA
   }

   SuperService getService(ServicesType servicesType) throws
Exception;
}
```

SuperService.java

```
package com.fantastic4.common.services;

import com.fantastic4.common.reservation.Reservation;

import java.rmi.Remote;

public interface SuperService extends Remote, Reservation {
}
```

AdminService.java

```
package com.fantastic4.common.services.custom;

import com.fantastic4.common.dto.AdminDTO;
import com.fantastic4.common.services.SuperService;

import java.util.List;

public interface AdminService extends SuperService{
    boolean addAdmin(AdminDTO adminDTO) throws Exception;

    boolean updateAdmin(AdminDTO adminDTO) throws Exception;

    boolean deleteAdmin(String adminID) throws Exception;

    AdminDTO findAdminByID(String adminID) throws Exception;

    List<AdminDTO> getAllAdmins() throws Exception;

    AdminDTO login(AdminDTO adminDTO) throws Exception;
}
```

FloorService.java

```
package com.fantastic4.common.services.custom;

import com.fantastic4.common.dto.FloorDTO;
import com.fantastic4.common.services.SuperService;

import java.util.List;

public interface FloorService extends SuperService {

    boolean addFloor(FloorDTO floorDTO) throws Exception;

    boolean updateFloor(FloorDTO floorDTO) throws Exception;

    boolean deleteFloor(String id) throws Exception;

    FloorDTO findFloorByID(String id) throws Exception;

    List<FloorDTO> getAllFloors() throws Exception;

}
```

RoomService.java

```
package com.fantastic4.common.services.custom;
```

```
import com.fantastic4.common.dto.RoomDTO;
import com.fantastic4.common.services.SuperService;

import java.util.List;

public interface RoomService extends SuperService {

    boolean addRoom(RoomDTO roomDTO) throws Exception;

    boolean updateRoom(RoomDTO roomDTO) throws Exception;

    boolean deleteRoom(String id) throws Exception;

    RoomDTO findRoomByID(String id) throws Exception;

    List<RoomDTO> getAllRooms() throws Exception;

}
```

SensorService.java

```
package com.fantastic4.common.services.custom;

import com.fantastic4.common.dto.SensorDTO;
```

```
import com.fantastic4.common.dto.SensorDataDTO;
import com.fantastic4.common.services.SuperService;

import java.util.List;

public interface SensorService extends SuperService {

    boolean addSensor(SensorDTO sensorDTO) throws Exception;

    boolean updateSensor(SensorDTO sensorDTO) throws Exception;

    boolean deleteSensor(String sensorID) throws Exception;

    SensorDTO findSensorByID(String sensorID) throws Exception;

    List<SensorDTO> getAllSensors() throws Exception;


    // Add Observers later

}
```

FAMSServer

Module-info.java

```
module FAMSServer {
    requires java.net.http;
    requires java.desktop;
    requires java.rmi;
    requires java.naming;
    requires FAMSCommon;
    requires com.fasterxml.jackson.databind;
    requires json;
}
```

BOFactory.java

```
package com.fantastic4.server.business;

import com.fantastic4.server.business.custom.impl.AdminBOImpl;
import com.fantastic4.server.business.custom.impl.FloorBOImpl;
import com.fantastic4.server.business.custom.impl.RoomBOImpl;
import com.fantastic4.server.business.custom.impl.SensorBOImpl;

public class BOFactory {
    public enum BOTypes{
        SENSOR,
        ADMIN,
        ROOM,
```

```
    FLOOR,
    SENSOR_DATA
}

private BOFactory() {
}

private static BOFactory boFactory;

public static BOFactory getInstance(){
    if (boFactory == null)
        boFactory = new BOFactory();
    return boFactory;
}

public SuperBO getBOFactory(BOTypes boTypes){
    switch (boTypes){
        case SENSOR: return new SensorBOImpl();
        case ADMIN: return new AdminBOImpl();
        case ROOM: return new RoomBOImpl();
        case FLOOR: return new FloorBOImpl();
        default:return null;
    }
```

```
    }
}
```

SuperBO.java

```
package com.fantastic4.server.business;

public interface SuperBO {
}
```

AdminBO.java

```
package com.fantastic4.server.business.custom;

import com.fantastic4.common.dto.AdminDTO;
import com.fantastic4.server.business.SuperBO;

import java.util.List;

public interface AdminBO extends SuperBO {

    boolean addAdmin(AdminDTO adminDTO) throws Exception;

    boolean updateAdmin(AdminDTO adminDTO) throws Exception;

    boolean deleteAdmin(String adminID) throws Exception;
```

```
    AdminDTO findAdminByID(String adminID) throws Exception;

    List<AdminDTO> getAllAdmins() throws Exception;

    AdminDTO login(AdminDTO adminDTO) throws Exception;
}
```

FloorBO.java

```
package com.fantastic4.server.business.custom;

import com.fantastic4.common.dto.FloorDTO;
import com.fantastic4.server.business.SuperBO;

import java.util.List;

public interface FloorBO extends SuperBO {

    boolean addFloor(FloorDTO floorDTO) throws Exception;

    boolean updateFloor(FloorDTO floorDTO) throws Exception;

    boolean deleteFloor(String id) throws Exception;
```

```
    FloorDTO findFloorByID(String id) throws Exception;

    List<FloorDTO> getAllFloors() throws Exception;

}
```

RoomBO.java

```
package com.fantastic4.server.business.custom;

import com.fantastic4.common.dto.RoomDTO;
import com.fantastic4.server.business.SuperBO;

import java.util.List;

public interface RoomBO extends SuperBO {

    boolean addRoom(RoomDTO roomDTO) throws Exception;

    boolean updateRoom(RoomDTO roomDTO) throws Exception;

    boolean deleteRoom(String id) throws Exception;

    RoomDTO findRoomByID(String id) throws Exception;
```

```
    List<RoomDTO> getAllRooms() throws Exception;

}
```

SensorBO.java

```
package com.fantastic4.server.business.custom;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.common.dto.SensorDataDTO;
import com.fantastic4.server.business.SuperBO;

import java.util.List;

public interface SensorBO extends SuperBO {

    boolean addSensor(SensorDTO sensorDTO) throws Exception;

    boolean updateSensor(SensorDTO sensorDTO) throws Exception;

    boolean deleteSensor(String sensorID) throws Exception;

    SensorDTO getSensorByID(String sensorID) throws Exception;
```

```
    List<SensorDTO> getAllSensors() throws Exception;


}
```

AdminBOImpl.java

```
package com.fantastic4.server.business.custom.impl;

import com.fantastic4.common.dto.AdminDTO;
import com.fantastic4.server.business.custom.AdminBO;
import com.fantastic4.server.repository.RepositoryFactory;
import com.fantastic4.server.repository.custom.AdminRepository;

import java.util.List;

public class AdminBOImpl implements AdminBO {

    private final AdminRepository adminRepository;

    public AdminBOImpl() {
        adminRepository = (AdminRepository)
RepositoryFactory.getInstance()

.getRepository(RepositoryFactory.RepositoryFactoryTypes.ADMIN);
```

```
    }

    @Override
    public boolean addAdmin(AdminDTO adminDTO) throws Exception {
        return adminRepository.save(adminDTO);
    }

    @Override
    public boolean updateAdmin(AdminDTO adminDTO) throws
Exception {
        return adminRepository.update(adminDTO);
    }

    @Override
    public boolean deleteAdmin(String adminID) throws Exception {
        return adminRepository.delete(adminID);
    }

    @Override
    public AdminDTO findAdminByID(String adminID) throws Exception {
        return adminRepository.findById(adminID);
    }

    @Override
```

```
    public List<AdminDTO> getAllAdmins() throws Exception {
        return adminRepository.findAll();
    }

    @Override
    public AdminDTO login(AdminDTO adminDTO) throws Exception {
        return adminRepository.login(adminDTO);
    }
}
```

FloorBOImpl.java

```
package com.fantastic4.server.business.custom.impl;

import com.fantastic4.common.dto.FloorDTO;
import com.fantastic4.server.business.custom.FloorBO;
import com.fantastic4.server.repository.RepositoryFactory;
import com.fantastic4.server.repository.custom.FloorRepository;

import java.util.List;

public class FloorBOImpl implements FloorBO {

    private final FloorRepository floorRepository;
```

```
    public FloorBOImpl() {
        floorRepository = (FloorRepository)
RepositoryFactory.getInstance()

.getRepository(RepositoryFactory.RepositoryFactoryTypes.FLOOR);
    }

    @Override
    public boolean addFloor(FloorDTO floorDTO) throws Exception {
        return floorRepository.save(floorDTO);
    }

    @Override
    public boolean updateFloor(FloorDTO floorDTO) throws Exception {
        return floorRepository.update(floorDTO);
    }

    @Override
    public boolean deleteFloor(String id) throws Exception {
        return floorRepository.delete(id);
    }

    @Override
```

```
    public FloorDTO findFloorByID(String id) throws Exception {
        return floorRepository.findById(id);
    }

    @Override
    public List<FloorDTO> getAllFloors() throws Exception {
        return floorRepository.findAll();
    }
}
```

RoomBOImpl.java

```
package com.fantastic4.server.business.custom.impl;

import com.fantastic4.common.dto.RoomDTO;
import com.fantastic4.server.business.custom.RoomBO;
import com.fantastic4.server.repository.RepositoryFactory;
import com.fantastic4.server.repository.custom.RoomRepository;

import java.util.List;

public class RoomBOImpl implements RoomBO {

    private final RoomRepository roomRepository;
```

```
    public RoomBOImpl() {
        roomRepository = (RoomRepository)
RepositoryFactory.getInstance()

.getRepository(RepositoryFactory.RepositoryFactoryTypes.ROOM);
    }

    @Override
    public boolean addRoom(RoomDTO roomDTO) throws Exception {
        return roomRepository.save(roomDTO);
    }

    @Override
    public boolean updateRoom(RoomDTO roomDTO) throws Exception {
        return roomRepository.update(roomDTO);
    }

    @Override
    public boolean deleteRoom(String id) throws Exception {
        return roomRepository.delete(id);
    }

    @Override
    public RoomDTO findRoomByID(String id) throws Exception {
```

```
        return roomRepository.findById(id);
    }

    @Override
    public List<RoomDTO> getAllRooms() throws Exception {
        return roomRepository.findAll();
    }
}
```

SensorBOImpl.java

```
package com.fantastic4.server.business.custom.impl;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.common.dto.SensorDataDTO;
import com.fantastic4.server.business.custom.SensorBO;
import com.fantastic4.server.repository.RepositoryFactory;
import com.fantastic4.server.repository.custom.SensorRepository;

import java.util.List;

public class SensorBOImpl implements SensorBO {

    private final SensorRepository sensorRepository;
```

```
    public SensorBOImpl() {
        sensorRepository = (SensorRepository)
RepositoryFactory.getInstance()

.getRepository(RepositoryFactory.RepositoryFactoryTypes.SENSOR);
    }

    @Override
    public boolean addSensor(SensorDTO sensorDTO) throws Exception {
        return sensorRepository.save(sensorDTO);
    }

    @Override
    public boolean updateSensor(SensorDTO sensorDTO) throws
Exception {
        return sensorRepository.update(sensorDTO);
    }

    @Override
    public boolean deleteSensor(String sensorID) throws Exception {
        return sensorRepository.delete(sensorID);
    }

    @Override
```

```
    public SensorDTO getSensorByID(String sensorID) throws Exception {
        return sensorRepository.findById(sensorID);
    }

    @Override
    public List<SensorDTO> getAllSensors() throws Exception {
        System.out.println("Inside getAllSensors SensorBO");
        return sensorRepository.findAll();
    }

}
```

RepositoryFactory.java

```
package com.fantastic4.server.repository;

import
com.fantastic4.server.repository.custom.impl.AdminRepositoryImpl;
import
com.fantastic4.server.repository.custom.impl.FloorRepositoryImpl;
import
com.fantastic4.server.repository.custom.impl.RoomRepositoryImpl;
import
com.fantastic4.server.repository.custom.impl.SensorRepositoryImpl;

public class RepositoryFactory {
```

```
    public enum RepositoryFactoryTypes{
        SENSOR,
        ADMIN,
        ROOM,
        FLOOR,
        SENSOR_DATA
    }

    private RepositoryFactory() {
    }

    private static RepositoryFactory repositoryFactory;

    public static RepositoryFactory getInstance(){
        if (repositoryFactory == null)
            repositoryFactory = new RepositoryFactory();
        return repositoryFactory;
    }

    public SuperRepository getRepository(RepositoryFactoryTypes
repositoryFactoryTypes){
        switch (repositoryFactoryTypes){
            case SENSOR: return new SensorRepositoryImpl();
```

```
            case ADMIN: return new AdminRepositoryImpl();
            case ROOM: return new RoomRepositoryImpl();
            case FLOOR: return new FloorRepositoryImpl();
            default: return null;
        }
    }
}
```

SuperRepository.java

```
package com.fantastic4.server.repository;

import java.util.List;

public interface SuperRepository<T, ID> {

    boolean save(T t)throws Exception;

    boolean delete(ID id)throws Exception;

    boolean update(T t)throws Exception;

    T findById(ID id)throws Exception;

    List<T> findAll() throws Exception;
```

```
}
```

AdminRepository.java

```
package com.fantastic4.server.repository.custom;

import com.fantastic4.common.dto.AdminDTO;
import com.fantastic4.server.repository.SuperRepository;

public interface AdminRepository extends SuperRepository<AdminDTO,
String> {

    AdminDTO login(AdminDTO adminDTO) throws Exception;
}
```

FloorRepository.java

```
package com.fantastic4.server.repository.custom;

import com.fantastic4.common.dto.FloorDTO;
import com.fantastic4.server.repository.SuperRepository;

public interface FloorRepository extends SuperRepository<FloorDTO,
String>{
}
```

RoomRepository.java

```
package com.fantastic4.server.repository.custom;

import com.fantastic4.common.dto.RoomDTO;
import com.fantastic4.server.repository.SuperRepository;

public interface RoomRepository extends SuperRepository<RoomDTO,
String> {
}
```

SensorRepository.java

```
package com.fantastic4.server.repository.custom;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.common.dto.SensorDataDTO;
import com.fantastic4.server.repository.SuperRepository;

import java.io.IOException;
import java.util.List;

public interface SensorRepository extends
SuperRepository<SensorDTO, String> {

}
```

AdminRepositoryImpl.java

```
package com.fantastic4.server.repository.custom.impl;
```

```
import com.fantastic4.common.dto.AdminDTO;
import com.fantastic4.server.repository.custom.AdminRepository;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.json.JSONArray;
import org.json.JSONObject;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.util.ArrayList;
import java.util.List;

public class AdminRepositoryImpl implements AdminRepository {

    private final HttpClient client;

    public AdminRepositoryImpl() {
        client =  HttpClient.newHttpClient();
    }
```

```
    @Override
    public boolean save(AdminDTO adminDTO) throws Exception {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create("http://localhost:8080/createAdmin"))

.POST(HttpRequest.BodyPublishers.ofString(adminDTO.toString()))
                .build();

        HttpResponse<?> response = client.send(request,
HttpResponse.BodyHandlers.discarding());
        System.out.println(response.statusCode());
        return false;
    }

    @Override
    public boolean delete(String s) throws Exception {
        return false;
    }

    @Override
    public boolean update(AdminDTO adminDTO) throws Exception {
        return false;
    }
```

```
    @Override
    public AdminDTO findById(String s) throws Exception {
        return null;
    }

    @Override
    public List<AdminDTO> findAll() throws Exception {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create("http://localhost:8080/getAllAdmins"))
                .build();

        List<AdminDTO> adminDTOList = new ArrayList<>();
        HttpResponse<String> response =
                client.send(request, HttpResponse.BodyHandlers.ofString());
        JSONArray jsonArray = new JSONArray(response.body());
        ObjectMapper mapper = new ObjectMapper();
        jsonArray.forEach(object -> {
            try {
                adminDTOList.add(mapper.readValue(object.toString(),
AdminDTO.class));
            } catch (JsonProcessingException e) {
                e.printStackTrace();
            }
```

```
        });

        return adminDTOList;
    }

    @Override
    public AdminDTO login(AdminDTO adminDTO) throws Exception{
        HttpRequest request = HttpRequest.newBuilder()

.uri(URI.create("http://localhost:8080/login?email="+adminDTO.getEm
ail()+"&pass="+adminDTO.getPassword()))
                .build();
        HttpResponse<String> response =
                client.send(request, HttpResponse.BodyHandlers.ofString());
        JSONObject jsonObject = new JSONObject(response.body());
        ObjectMapper mapper = new ObjectMapper();
        AdminDTO admin =
mapper.readValue(jsonObject.toString(),AdminDTO.class);

        return admin;
    }
}
```

FloorRepositoryImpl.java

```
package com.fantastic4.server.repository.custom.impl;
```

```
import com.fantastic4.common.dto.FloorDTO;
import com.fantastic4.server.repository.custom.FloorRepository;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.json.JSONArray;
import java.net.URI;

import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.util.ArrayList;
import java.util.List;

public class FloorRepositoryImpl implements FloorRepository {

    private final HttpClient client;

    public FloorRepositoryImpl() {
        client = HttpClient.newHttpClient();
    }

    @Override
```

```
    public boolean save(FloorDTO floorDTO) throws Exception {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create("http://localhost:8080/createFloor"))
.POST(HttpRequest.BodyPublishers.ofString(floorDTO.toString()))
                .build();

        HttpResponse<?> response = client.send(request,
HttpResponse.BodyHandlers.discarding());
        System.out.println(response.statusCode());
        return false;
    }

    @Override
    public boolean delete(String s) throws Exception {
        return false;
    }

    @Override
    public boolean update(FloorDTO floorDTO) throws Exception {
        return false;
    }

    @Override
```

```
    public FloorDTO findById(String s) throws Exception {
        return null;
    }

    @Override
    public List<FloorDTO> findAll() throws Exception {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create("http://localhost:8080/getAllFloors"))
                .build();

        List<FloorDTO> floorDTOList = new ArrayList<>();
        HttpResponse<String> response =
                client.send(request, HttpResponse.BodyHandlers.ofString());
        JSONArray jsonArray = new JSONArray(response.body());
        ObjectMapper mapper = new ObjectMapper();
        jsonArray.forEach(object -> {
            try {
                floorDTOList.add(mapper.readValue(object.toString(),
FloorDTO.class));
            } catch (JsonProcessingException e) {
                e.printStackTrace();
            }
        });
```

```
        return floorDTOList;
    }
}
```

RoomRepositoryImpl.java

```
package com.fantastic4.server.repository.custom.impl;

import com.fantastic4.common.dto.AdminDTO;
import com.fantastic4.common.dto.RoomDTO;
import com.fantastic4.server.repository.custom.RoomRepository;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.json.JSONArray;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.util.ArrayList;
import java.util.List;

public class RoomRepositoryImpl implements RoomRepository {
```

```
    private final HttpClient client;

    public RoomRepositoryImpl() {
        client = HttpClient.newHttpClient();
    }

    @Override
    public boolean save(RoomDTO roomDTO) throws Exception {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create("http://localhost:8080/createRoom"))

.POST(HttpRequest.BodyPublishers.ofString(roomDTO.toString()))
                .build();

        HttpResponse<?> response = client.send(request,
HttpResponse.BodyHandlers.discarding());
        System.out.println(response.statusCode());
        return false;
    }

    @Override
    public boolean delete(String s) throws Exception {
        return false;
```

```
}

@Override
public boolean update(RoomDTO roomDTO) throws Exception {
   return false;
}

@Override
public RoomDTO findById(String s) throws Exception {
   return null;
}

@Override
public List<RoomDTO> findAll() throws Exception {

   HttpRequest request = HttpRequest.newBuilder()
         .uri(URI.create("http://localhost:8080/getAllRooms"))
         .build();

   List<RoomDTO> roomDTOList = new ArrayList<>();
   HttpResponse<String> response =
         client.send(request, HttpResponse.BodyHandlers.ofString());
   JSONArray jsonArray = new JSONArray(response.body());
```

```
        ObjectMapper mapper = new ObjectMapper();
        jsonArray.forEach(object -> {
            try {
                roomDTOList.add(mapper.readValue(object.toString(),
RoomDTO.class));
            } catch (JsonProcessingException e) {
                e.printStackTrace();
            }
        });

        return roomDTOList;
    }
}
```

SensorRepositoryImpl.java

```
package com.fantastic4.server.repository.custom.impl;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.server.repository.custom.SensorRepository;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.json.JSONArray;
```

```
import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.util.ArrayList;
import java.util.List;

public class SensorRepositoryImpl implements SensorRepository {

    private final HttpClient client;

    public SensorRepositoryImpl() {
        client = HttpClient.newHttpClient();
    }

    @Override
    public boolean save(SensorDTO sensorDTO) throws Exception {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create("http://localhost:8080/createSensor"))

.POST(HttpRequest.BodyPublishers.ofString(sensorDTO.toString()))
                .build();
```

```
        HttpResponse<?> response = client.send(request,
HttpResponse.BodyHandlers.discarding());
        System.out.println(response.statusCode());
        return false;
    }

    @Override
    public boolean delete(String id) throws Exception {
        HttpRequest request = HttpRequest.newBuilder()

.uri(URI.create("http://localhost:8080/deleteSensor?sensorID=" + id))
                .DELETE().build();
        HttpResponse<?> response = client.send(request,
HttpResponse.BodyHandlers.discarding());

        return response.statusCode() == 200;
    }

    @Override
    public boolean update(SensorDTO sensorDTO) throws Exception {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create("http://localhost:8080/updateSensor"))
                .header("Content-Type", "application/json")
```

```
.PUT(HttpRequest.BodyPublishers.ofString(sensorDTO.toString()))
            .build();
        HttpResponse<?> response = client.send(request,
HttpResponse.BodyHandlers.discarding());
        System.out.println(response.statusCode());
        return false;
    }

    @Override
    public SensorDTO findById(String s) throws Exception {
        return null;
    }

    @Override
    public List<SensorDTO> findAll() throws IOException,
InterruptedException {
        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create("http://localhost:8080/getAllSensors"))
            .build();

        List<SensorDTO> sensorDTOList = new ArrayList<>();
        HttpResponse<String> response =
            client.send(request, HttpResponse.BodyHandlers.ofString());
```

```
        JSONArray jsonArray = new JSONArray(response.body());
        ObjectMapper mapper = new ObjectMapper();
        jsonArray.forEach(object -> {
            try {
                sensorDTOList.add(mapper.readValue(object.toString(),
SensorDTO.class));
            } catch (JsonProcessingException e) {
                e.printStackTrace();
            }
        });

        return sensorDTOList;
    }
}
```

ServicesFactory.java

```
package com.fantastic4.server.services.impl;

import com.fantastic4.common.services.ServicesFactory;
import com.fantastic4.common.services.SuperService;
import com.fantastic4.server.services.impl.custom.AdminServiceImpl;
import com.fantastic4.server.services.impl.custom.FloorServiceImpl;
```

```
import com.fantastic4.server.services.impl.custom.RoomServiceImpl;
import com.fantastic4.server.services.impl.custom.SensorServiceImpl;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class ServicesFactoryImpl extends UnicastRemoteObject
implements ServicesFactory {
   private static ServicesFactory servicesFactory;

   private ServicesFactoryImpl() throws RemoteException {
   }

   public static ServicesFactory getInstance()throws RemoteException{
      if (servicesFactory == null)
         servicesFactory = new ServicesFactoryImpl();
      return servicesFactory;
   }

   @Override
   public SuperService getService(ServicesType servicesType) throws
Exception {
      switch (servicesType){
         case SENSOR: return new SensorServiceImpl();
```

```
            case ADMIN: return new AdminServiceImpl();
            case ROOM: return new RoomServiceImpl();
            case FLOOR: return new FloorServiceImpl();
            default: return null;
        }
    }
}
```

AdminServiceImpl.java

```
package com.fantastic4.server.services.impl.custom;

import com.fantastic4.common.dto.AdminDTO;
import com.fantastic4.common.services.custom.AdminService;
import com.fantastic4.server.business.BOFactory;
import com.fantastic4.server.business.custom.AdminBO;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

public class AdminServiceImpl extends UnicastRemoteObject
implements AdminService{

    private final AdminBO adminBO;
```

```
    public AdminServiceImpl() throws RemoteException {
        adminBO = (AdminBO) BOFactory.getInstance()
                .getBOFactory(BOFactory.BOTypes.ADMIN);
    }



    @Override
    public boolean addAdmin(AdminDTO adminDTO) throws Exception {
        return false;
    }

    @Override
    public boolean updateAdmin(AdminDTO adminDTO) throws
Exception {
        return false;
    }

    @Override
    public boolean deleteAdmin(String adminID) throws Exception {
        return false;
    }

    @Override
```

```
public AdminDTO findAdminByID(String adminID) throws Exception {
   return null;
}

@Override
public List<AdminDTO> getAllAdmins() throws Exception {
   return null;
}

@Override
public AdminDTO login(AdminDTO adminDTO) throws Exception {
   return adminBO.login(adminDTO);
}

@Override
public boolean reserve(Object id) throws Exception {
   return false;
}

@Override
public boolean release(Object id) throws Exception {
   return false;
}
```

```
}
```

FloorServiceImpl.java

```
package com.fantastic4.server.services.impl.custom;

import com.fantastic4.common.dto.FloorDTO;
import com.fantastic4.common.services.custom.FloorService;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

public class FloorServiceImpl extends UnicastRemoteObject implements
FloorService {

    public FloorServiceImpl() throws RemoteException {
    }

    @Override
    public boolean addFloor(FloorDTO floorDTO) throws Exception {
        return false;
    }

    @Override
    public boolean updateFloor(FloorDTO floorDTO) throws Exception {
```

```
    return false;
}

@Override
public boolean deleteFloor(String id) throws Exception {
    return false;
}

@Override
public FloorDTO findFloorByID(String id) throws Exception {
    return null;
}

@Override
public List<FloorDTO> getAllFloors() throws Exception {
    return null;
}

@Override
public boolean reserve(Object id) throws Exception {
    return false;
}
```

```
    @Override
    public boolean release(Object id) throws Exception {
        return false;
    }
}
```

RoomServiceImpl.java

```
package com.fantastic4.server.services.impl.custom;

import com.fantastic4.common.dto.RoomDTO;
import com.fantastic4.common.services.custom.RoomService;
import com.fantastic4.server.business.BOFactory;
import com.fantastic4.server.business.custom.RoomBO;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

public class RoomServiceImpl extends UnicastRemoteObject
implements RoomService {

    private final RoomBO roomBO;

    public RoomServiceImpl() throws RemoteException {
```

```
    roomBO = (RoomBO) BOFactory.getInstance()
            .getBOFactory(BOFactory.BOTypes.ROOM);
}


@Override
public boolean addRoom(RoomDTO roomDTO) throws Exception {
    return roomBO.addRoom(roomDTO);
}

@Override
public boolean updateRoom(RoomDTO roomDTO) throws Exception {
    return roomBO.updateRoom(roomDTO);
}

@Override
public boolean deleteRoom(String id) throws Exception {
    return roomBO.deleteRoom(id);
}

@Override
public RoomDTO findRoomByID(String id) throws Exception {
    return roomBO.findRoomByID(id);
```

```
    }

    @Override
    public List<RoomDTO> getAllRooms() throws Exception {
        return roomBO.getAllRooms();
    }

    @Override
    public boolean reserve(Object id) throws Exception {
        return false;
    }

    @Override
    public boolean release(Object id) throws Exception {
        return false;
    }
}
```

SensorServiceImpl.java

```
package com.fantastic4.server.services.impl.custom;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.common.dto.SensorDataDTO;
import com.fantastic4.common.services.custom.SensorService;
```

```
import com.fantastic4.server.business.BOFactory;
import com.fantastic4.server.business.custom.SensorBO;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.List;

public class SensorServiceImpl extends UnicastRemoteObject
implements SensorService {

    private final SensorBO sensorBO;

    public SensorServiceImpl() throws RemoteException {
        sensorBO = (SensorBO) BOFactory.getInstance()
                .getBOFactory(BOFactory.BOTypes.SENSOR);
    }

    @Override
    public boolean addSensor(SensorDTO sensorDTO) throws Exception {
        return sensorBO.addSensor(sensorDTO);
    }

    @Override
```

```
    public boolean updateSensor(SensorDTO sensorDTO) throws
Exception {
        return sensorBO.updateSensor(sensorDTO);
    }

    @Override
    public boolean deleteSensor(String sensorID) throws Exception {
        return sensorBO.deleteSensor(sensorID);
    }

    @Override
    public SensorDTO findSensorByID(String sensorID) throws Exception {
        return sensorBO.getSensorByID(sensorID);
    }

    @Override
    public List<SensorDTO> getAllSensors() throws Exception {
        System.out.println("Inside getAllSensors SensorServiceImpl");
        return sensorBO.getAllSensors();
    }

    @Override
    public boolean reserve(Object id) throws Exception {
```

```
        return false;
    }

    @Override
    public boolean release(Object id) throws Exception {
        return false;
    }
}
```

ServerStart.java

```
package com.fantastic4.server.startup;

import com.fantastic4.common.dto.SensorDTO;
import com.fantastic4.common.services.custom.SensorService;
import com.fantastic4.server.services.impl.ServicesFactoryImpl;
import com.fantastic4.server.services.impl.custom.SensorServiceImpl;


import javax.swing.*;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.List;
```

```
import java.util.Timer;
import java.util.TimerTask;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class ServerStart {

    public static void main(String[] args){
        System.setProperty("java.rmi.server.hostname","localhost");
        System.setProperty("java.security.policy", "file:./server.policy");

//        System.setProperty("java.rmi.server.hostname","127.0.0.1");
//        System.setProperty("java.security.policy","security.policy");
//        System.setSecurityManager(new RMISecurityManager());
        try {
            if (true) {
//                if(System.getSecurityManager() == null ){
//                    System.setSecurityManager( new RMISecurityManager() );
//                }
                Registry registry = LocateRegistry.createRegistry(5050);
                registry.rebind("fams", ServicesFactoryImpl.getInstance());
```

```
            System.out.println("Server has been started successfully");
            SensorServiceImpl sensorService = new SensorServiceImpl();
            SensorDTO sensorDTO = new
                        SensorDTO("S004", 3, 3,3,3, false);
            System.out.println(sensorDTO.toString());

            System.out.println(sensorService.addSensor(sensorDTO));
            Timer timer = new Timer();
            timer.schedule(new checkStatus(),0,5000);


        } else {
            JOptionPane.showMessageDialog(null,
                    "API is not connected", "Error",
JOptionPane.INFORMATION_MESSAGE);
        }
      } catch (Exception e) {
        e.printStackTrace();
      }
   }

   static class checkStatus extends TimerTask{
      private final HttpClient client;
```

```
    public checkStatus() {
       client = HttpClient.newHttpClient();
    }

    @Override
    public void run() {
       try {
          SensorServiceImpl sensorService = new SensorServiceImpl();
          List<SensorDTO> sensorDTOList =
sensorService.getAllSensors();
          for (SensorDTO sensor:sensorDTOList
          ) {
             if(sensor.getLatestCO2Level() >= 5 ||
sensor.getLatestSmokeLevel() >= 5){
                //Implement and Call Endpoint
                HttpRequest request1 = HttpRequest.newBuilder()
                      .uri(URI.create("http://localhost:9090/send-mail"))
                      .build();

                HttpRequest request2 = HttpRequest.newBuilder()
                      .uri(URI.create("http://localhost:9091/send-sms"))
                      .build();
```

```
                //HttpResponse response1 = client.send(request1,
HttpResponse.BodyHandlers.discarding());
                //HttpResponse response2 = client.send(request2,
HttpResponse.BodyHandlers.discarding());
                }
            }
        } catch (RemoteException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

DEMO Application

App.java

```
package com.fantastic4;

import javafx.application.Application;
import javafx.scene.Scene;
```

```
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

import java.util.Timer;

/**
* JavaFX App
*/
public class App extends Application {

    @Override
    public void start(Stage stage) {

        Timer timer = new Timer();
        timer.schedule(new SendSensorData(),0,5000);
        var label = new Label("Hello, Demo App Started");
        var scene = new Scene(new StackPane(label), 640, 480);
        stage.setScene(scene);
        stage.show();
    }

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

```
        launch();
    }

}
```

SendSensorData.java

```
package com.fantastic4;

import com.fantastic4.models.SensorDTO;
import com.fantastic4.models.SensorData;

import java.io.IOException;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.util.*;

public class SendSensorData extends TimerTask{

    private final HttpClient client;
```

```
    public SendSensorData() {
        client = HttpClient.newHttpClient();
    }

    public void run(){
        try {
            List<SensorDTO> sensorDTOS = getSensors();
            for (SensorDTO sensorDTO : sensorDTOS
                ) {
                SensorData sensorData = new SensorData();
                sensorData.setSensorID(sensorDTO.getSensorID());
                sensorData.setCo2Level(new Random().nextInt((10-1)+1)+1);
                sensorData.setSmokeLevel(new Random().nextInt((10-
1)+1)+1);

sensorData.setDate(Long.toString(System.currentTimeMillis()));
            }
        } catch (IOException | InterruptedException e) {
            e.printStackTrace();
        }
    }
```

```
   public List<SensorDTO> getSensors() throws
IOException,InterruptedException{
      HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create("http://localhost:8080/getAllSensors"))
            .build();

      List<SensorDTO> sensorDTOList = new ArrayList<>();
//       HttpResponse<String> response =
//             client.send(request, HttpResponse.BodyHandlers.ofString());
//       JSONArray jsonArray = new JSONArray(response.body());
//       ObjectMapper mapper = new ObjectMapper();
//       jsonArray.forEach(object -> {
//          try {
//             sensorDTOList.add(mapper.readValue(object.toString(),
SensorDTO.class));
//          } catch (JsonProcessingException e) {
//             e.printStackTrace();
//          }
//       });


      return sensorDTOList;
   }
```

```
    public void sendSensorData(SensorData sensorData){
        //Implement HTTP
    }

}
```

SensorData.java

```
package com.fantastic4.models;

public class SensorData{

    private String sensorDataID;
    private String sensorID;
    private int co2Level;
    private int smokeLevel;
    private String date;

    public SensorData() {
    }

    public SensorData(String sensorDataID, String sensorID, int co2Level,
int smokeLevel, String date) {
        this.sensorDataID = sensorDataID;
        this.sensorID = sensorID;
        this.co2Level = co2Level;
```

```
    this.smokeLevel = smokeLevel;
    this.date = date;
}

public String getSensorDataID() {
    return sensorDataID;
}

public void setSensorDataID(String sensorDataID) {
    this.sensorDataID = sensorDataID;
}

public String getSensorID() {
    return sensorID;
}

public void setSensorID(String sensorID) {
    this.sensorID = sensorID;
}

public int getCo2Level() {
    return co2Level;
}
```

```
    public void setCo2Level(int co2Level) {
        this.co2Level = co2Level;
    }

    public int getSmokeLevel() {
        return smokeLevel;
    }

    public void setSmokeLevel(int smokeLevel) {
        this.smokeLevel = smokeLevel;
    }

    public String getDate() {
        return date;
    }

    public void setDate(String date) {
        this.date = date;
    }

}
```

SensorDTO.java

```
module com.fantastic4 {
    requires javafx.controls;
    requires java.net.http;
    requires org.jvnet.staxex;
//    requires org.json;
    exports com.fantastic4;
}
```

Module-info.java

```
module com.fantastic4 {
    requires javafx.controls;
    requires java.net.http;
    requires org.jvnet.staxex;
//    requires org.json;
    exports com.fantastic4;
}
```

Email Service

EmailAlert.java

```
package com.emailserver.emailserver;

import java.io.Serializable;
import javax.mail.MessagingException;
import javax.mail.internet.MimeMessage;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Component;

@Component

public class EmailAlert implements Serializable{

   @Autowired
```

```
    private JavaMailSender javaMailSender;

    public void send(String to, String subject, String body) throws MessagingException {

        MimeMessage message = javaMailSender.createMimeMessage();
        MimeMessageHelper helper;

        helper = new MimeMessageHelper(message, true);
        helper.setSubject(subject);
        helper.setTo(to);
        helper.setText(body, true);

        javaMailSender.send(message);
    }

}
```

EmailController.java

```
package com.emailserver.emailserver;
import javax.mail.MessagingException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class EmailController {
```

```
    @Autowired
    private EmailAlert emailAlert;

    @RequestMapping("/send-mail")
    public void sendMail() throws MessagingException {

        emailAlert.send("afirealert@gmail.com", "Fire Alert !", "This is an emergency fire alert!!!");

    }
}
```

EmailserverApplication.java

```
package com.emailserver.emailserver;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan
@EnableAutoConfiguration
@SpringBootApplication
```

```
public class EmailserverApplication {

    public static void main(String[] args) {
        SpringApplication.run(EmailserverApplication.class, args);
    }

}
```

SMS Service

SMSAlert.java

```
package com.SMSServer.Server;

import java.io.Serializable;

import com.twilio.Twilio;
import com.twilio.rest.api.v2010.account.Message;
import com.twilio.type.PhoneNumber;
import java.io.Serializable;
import org.springframework.stereotype.Component;

@Component
public class SMSAlert implements Serializable {

    private static final String ACCOUNT_SID = "AC591f0ae975834806acebdeee5239c449";
    private static final String AUTH_TOKEN = "29cad427d6840ee62626fa0b437ddfa1";
```

```
    public SMSAlert() {
    }

    public void sendsms(String... args) throws Exception {
        Twilio.init("AC591f0ae975834806acebdeee5239c449",
"29cad427d6840ee62626fa0b437ddfa1");
        Message.creator(new PhoneNumber("+94714331418"), new
PhoneNumber("+12183963759"), "This is an emergency fire alert!!!").create();
}
}
```

SMSController.java

```
package com.SMSServer.Server;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class SMSController {
    @Autowired
    private SMSAlert smsalert;
    public SMSController() {
    }
    @RequestMapping({"/send-sms"})
    public void send() throws Exception {
        this.smsalert.sendsms(new String[0]);
```

```
    }
}
```

ServerApplication.java

```
package com.SMSServer.Server;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan
@EnableAutoConfiguration
@SpringBootApplication
public class ServerApplication {

    public ServerApplication() {
    }

    public static void main(String[] args) {
        SpringApplication.run(ServerApplication.class, args);
    }

}
```