DISTRIBUTED SYSTEMS

ASSIGNMENT 2

Final Report for Fire Alarm Monitoring System

TABLE OF CONTENTS

Contents

ASSIGNMENT 2

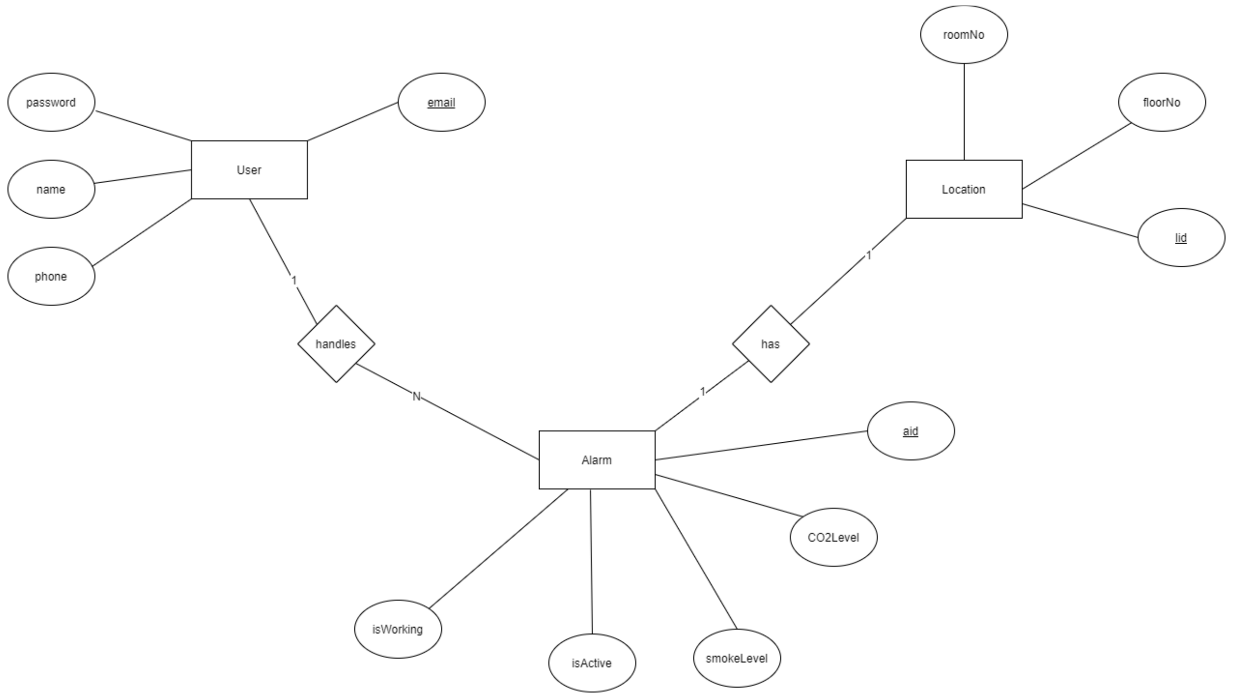
Introduction

The fire alarm monitoring system, has multiple components working together to make the monitoring real-time.

These components include,

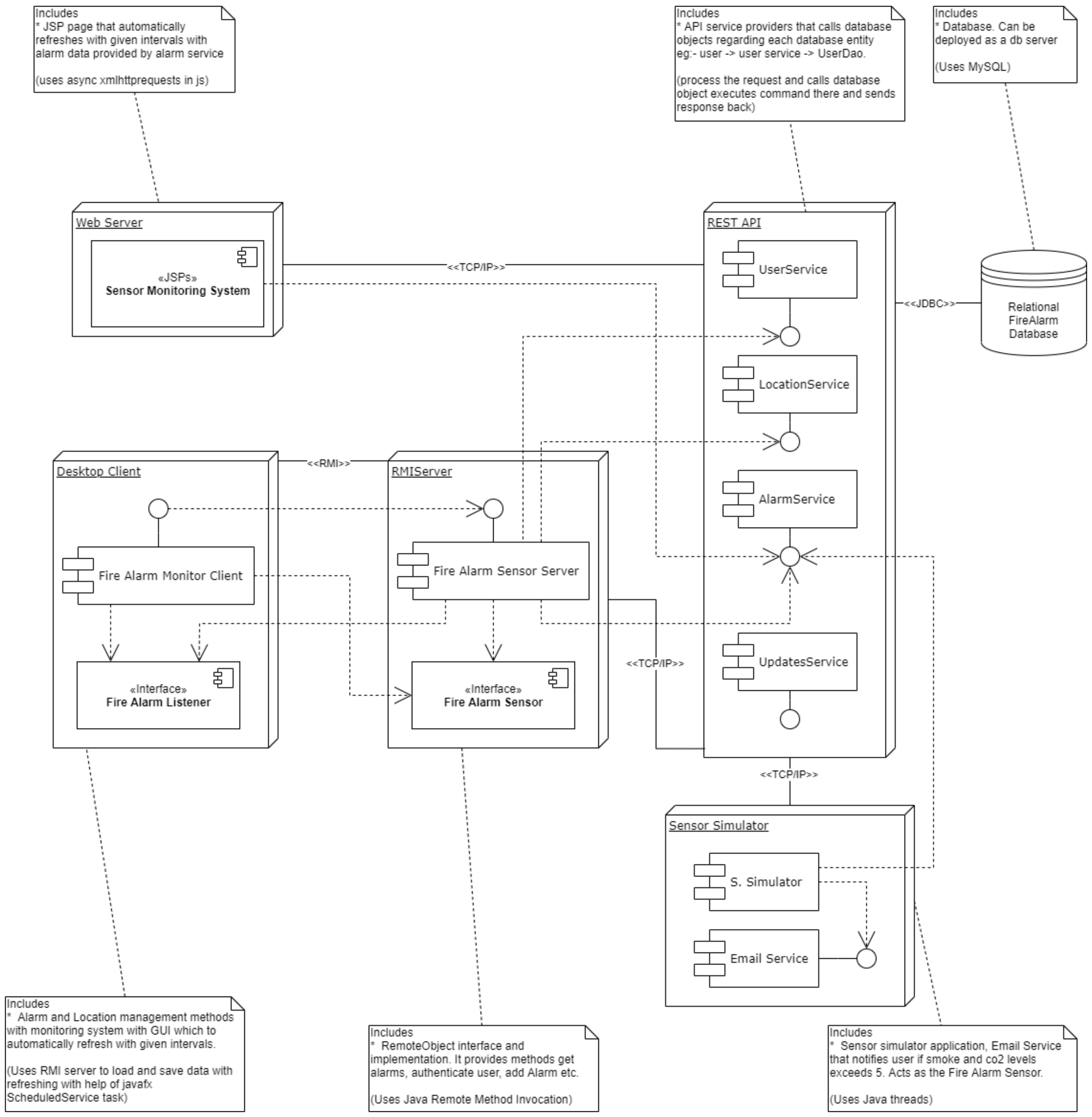
- Fire Alarm Database which stores fire alarm sensor, location and admin data,
- REST API to handle client and sensor requests,
- Fire alarm web client,
- Fire alarm desktop client which powered by a RMI (Remote Method Invocation) Server,
- Fire alarm Simulator application which can be applied to a real life fire alarm sensor,
- Method to notify the desktop user admin, which handles the alarms that by Via email or SMS.

The ER diagram



Layout of the System

The basic Layout of the Fire Alarm Monitoring System described in the diagram below.



Includes
* JSP page that automatically refreshes with given intervals with alarm data provided by alarm service

(uses async xmlhttprequests in js)

Includes
* API service providers that calls database objects regarding each database entity eg:- user -> user service -> UserDao.

(process the request and calls database object executes command there and sends response back)

Includes
* Database. Can be deployed as a db server

(Uses MySQL)

Includes
* Alarm and Location management methods with monitoring system with GUI which to automatically refresh with given intervals.

(Uses RMI server to load and save data with refreshing with help of javafx ScheduledService task)

Includes
* RemoteObject interface and implementation. It provides methods get alarms, authenticate user, add Alarm etc.

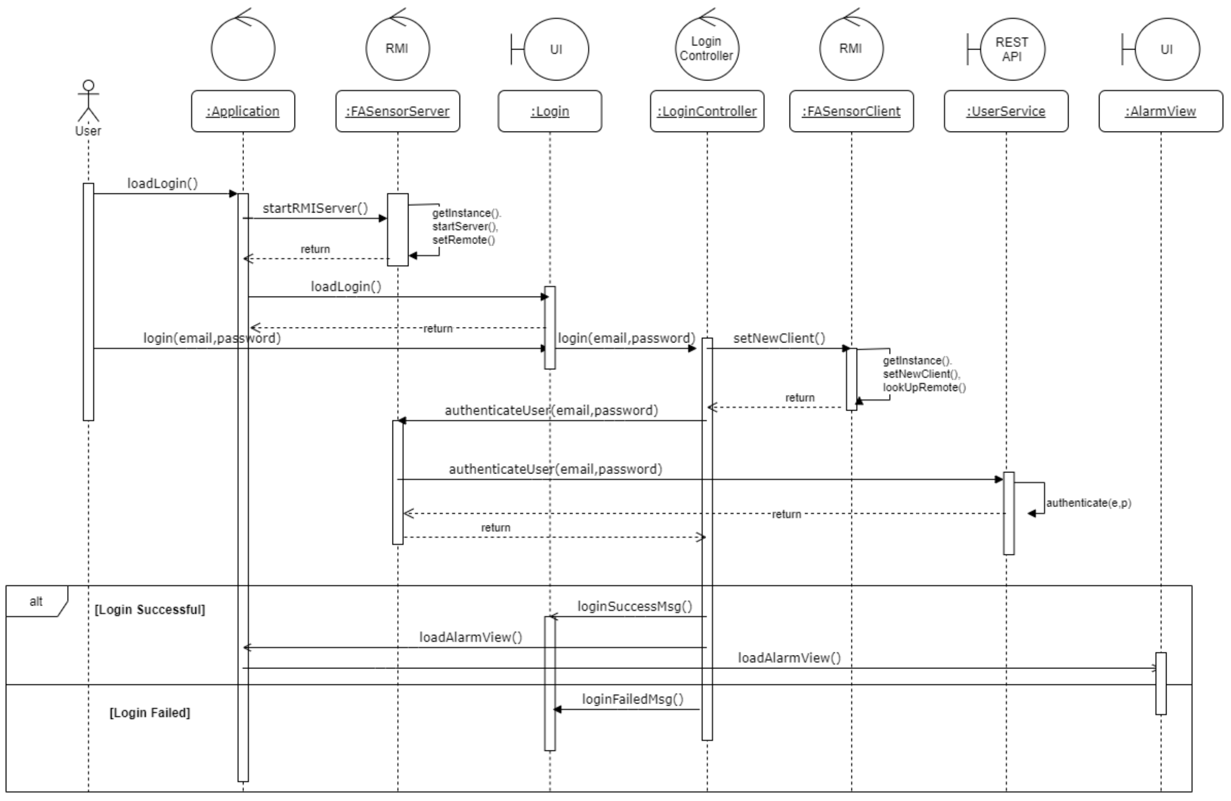
(Uses Java Remote Method Invocation)

Includes
* Sensor simulator application, Email Service that notifies user if smoke and co2 levels exceeds 5. Acts as the Fire Alarm Sensor.

(Uses Java threads)

Desktop Client

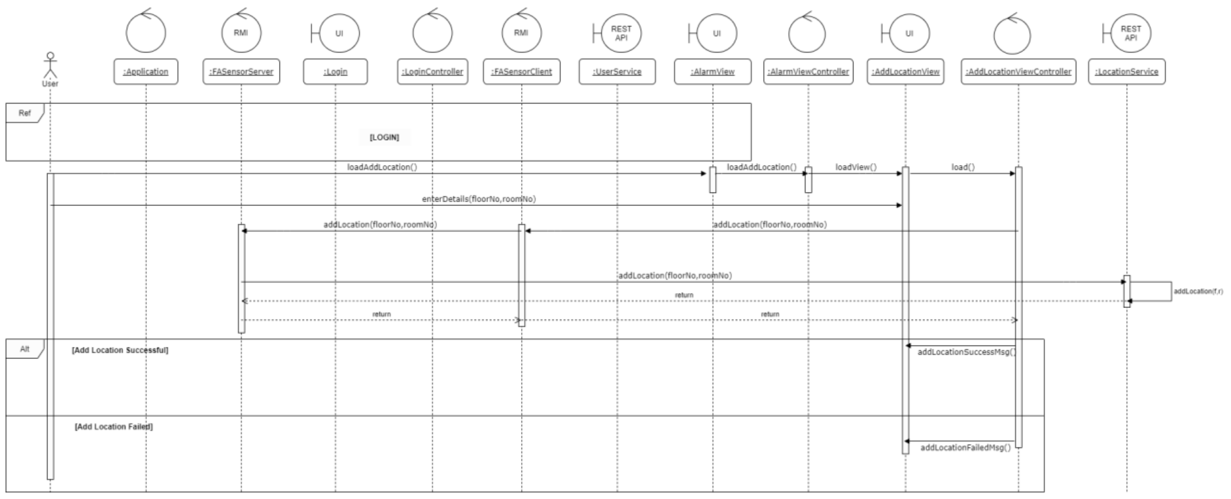
User Login



User logs in with giving email and password. RMI client calls Remote object's authenticate method and validates the authentication using REST.

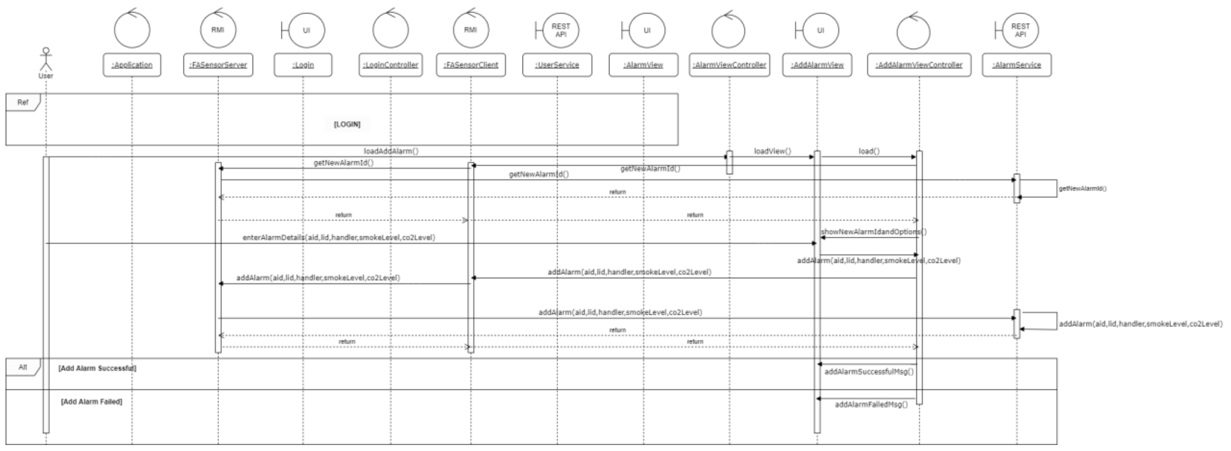
Desktop Client

Add Location



Logged-in user clicks add location then sets floorNo and roomNo then clicks ‘Add’. Client calls Remote object’s addLocation (floor, room) method and validates the new location’s content inside Location Service of REST.

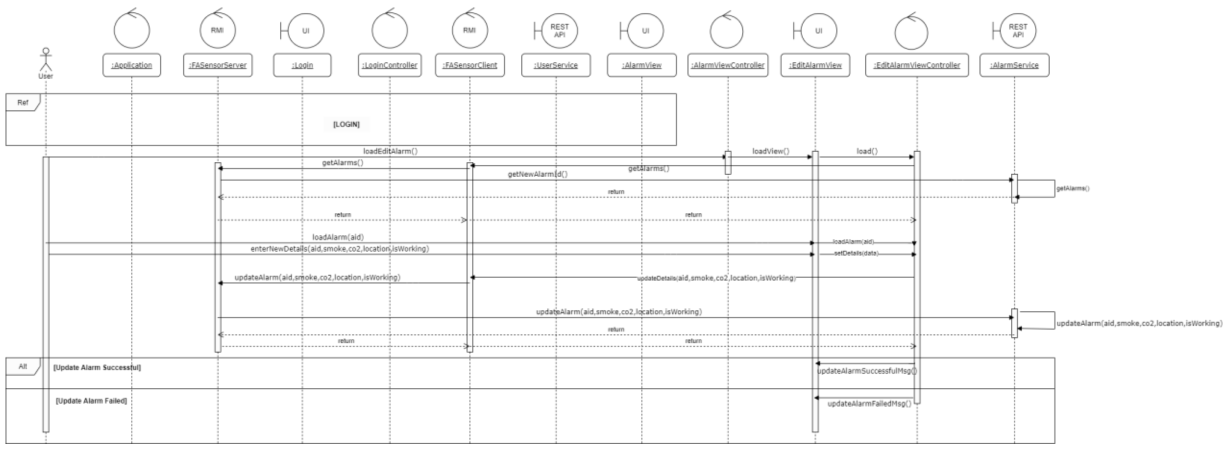
Add Alarm



Then user clicks add alarm. The controller gets a new alarm id from the system; user gives the inputs and add. Client calls Remote object’s addAlarm (aid, smoke, co2, handler…) method and validates the new alarm’s content inside Alarm Service of REST.

Desktop Client

Edit Alarm



User clicks edit alarm button and controller gets alarms details from the system; user gives new inputs and clicks edit. Client calls Remote object’s updateAlarm (aid, smoke, co2, handler…) method and validates the updated alarm’s content inside Alarm Service of REST.

Refresh Alarms

- The alarm view UI consists of alarm details, which auto, refreshed each 15 seconds. The UI uses Scheduled Service class of concurrent javafx package to do so. It also has the manual refresh option included for the user.

ASSIGNMENT 2

Web Client

- Web Client uses java's JSP pages to view alarms in the table. Initially it uses a static alarm objects in loading time.
- Then the web client waits for 30 seconds while our javascript program calls rest API directly and get Alarms as JSON objects as response text and DOM manipulate it to the Sensor jsp view also updates last updated time.
- This is done using setInterval method and XMLHttpRequest method and Class of javascript.

Look at codes codes -> Web

```
const getAlarmsList = function() {
	const tableBody = table.querySelector('#tableBody');
	const req = new XMLHttpRequest();
	req.open('GET',
'http://localhost:8080/FireAlarmRest/rest/AlarmService/getAlarms', true);
	req.onreadystatechange = function() {
		if(req.readyState == 4 && req.status == 200) {
			var alarmsArray = req.responseText;
			let jso = JSON.parse(alarmsArray);
			var newBody = document.createElement('tbody');
			newBody.setAttribute('id','tableBody');
			for (var i = 0; i < jso.length; i++) {
				let obj = JSON.parse(JSON.stringify(jso[i]));
				var a = document.createElement('tr');
				a.setAttribute('class','row');
				…
			}
		}
	}

const repeat = function() {
    setInterval(function() {
	getAlarmsList();
	lastupdated.innerHTML = new Date().toUTCString();
	}, 30000);
};

repeat();
```

ASSIGNMENT 2

REST API

The REST API made with Java Jersey and JSON make the system easy to serialize data and de-serialize data very fast & the RMI does not work with Objects so we had to transfer data as JSON object / array as String. The API consist of User to handle user entity and Location to location and Alarm to handle alarms and Updates for later expansions of system or to load event history.

AlarmService.java

```
@Path("/AlarmService")
public class AlarmService {
	@GET
	@Path("/getAlarms")
	@Produces(MediaType.APPLICATION_JSON)
	public List<Alarm> getAlarms() {
		List<Alarm> list = AlarmDao.getAlarms();

		return list;
	}
	…
}
```

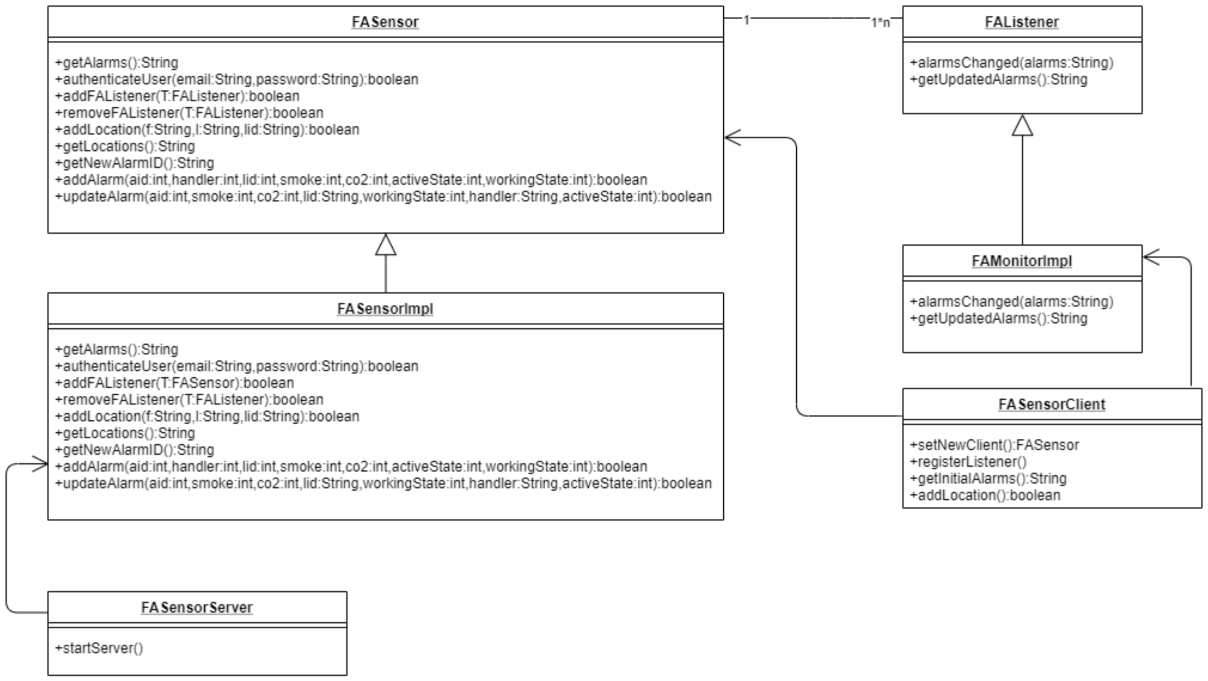
AlarmDao.java

```
public static List<Alarm> getAlarms() {
	List<Alarm> list = new ArrayList<>();

	con = ConnectFA.connect();
	String queryA = "SELECT * FROM Alarm ;";
	ps = con.prepareStatement(queryA);
	rs = ps.executeQuery();
	while(rs.next()) {
		Alarm a = new Alarm();
		a.setAid(rs.getString(1));
		a.setSmokeLevel(rs.getInt(2));
		a.setCo2Level(rs.getInt(3));
		list.add(a); }
		return list;
	}
	return null;
}
```

RMI Server

RMI server, which developed using java and is constantly updating the alarms for the RMI client to use or desktop client to use. Then the clients use the updated variable each given interval in seconds.


FASensor
+getAlarms():String
+authenticateUser(email:String,password:String):boolean
+addFAListener(T:FAListener):boolean
+removeFAListener(T:FAListener):boolean
+addLocation(f:String,l:String,lid:String):boolean
+getLocations():String
+getNewAlarmID():String
+addAlarm(aid:int,handler:int,lid:int,smoke:int,co2:int,activeState:int,workingState:int):boolean
+updateAlarm(aid:int,smoke:int,co2:int,lid:String,workingState:int,handler:String,activeState:int):boolean
1
1*n
FAListener
+alarmsChanged(alarms:String)
+getUpdatedAlarms():String
FAMonitorImpl
+alarmsChanged(alarms:String)
+getUpdatedAlarms():String
FASensorImpl
+getAlarms():String
+authenticateUser(email:String,password:String):boolean
+addFAListener(T:FASensor):boolean
+removeFAListener(T:FAListener):boolean
+addLocation(f:String,l:String,lid:String):boolean
+getLocations():String
+getNewAlarmID():String
+addAlarm(aid:int,handler:int,lid:int,smoke:int,co2:int,activeState:int,workingState:int):boolean
+updateAlarm(aid:int,smoke:int,co2:int,lid:String,workingState:int,handler:String,activeState:int):boolean
FASensorClient
+setNewClient():FASensor
+registerListener()
+getInitialAlarms():String
+addLocation():boolean
FASensorServer
+startServer()

If a sensor, already running on rmiregistry it will rebound for new.

Simulator

Simulator, which simulates sensors, works as the sensor of the system. For every 30 seconds, gets random numbers from the method in given range and sets smoke and co2 levels of an alarm. Therefore, Web Client and Desktop client via RMI gets the alerts with their time intervals.

A dummy email service also included for the sensor and by embedding it with the sensor, we can notify the admin of the system even if the user is not watching in the desktop client application.

Randomizing method in use and Dummy Email Service

```
public static int getRandomInteger(int max) {
	int i = new Random().nextInt(max);
	if(i != 0)
		return i;
	else
		return getRandomInteger(max);
}
```

```
private void sendEmail(Alarm a) {
	if(a.getSmokeLevel() > 5) {
	 System.out.println("\n---------------------------------------\n");
	 System.out.println("Notification by FireAlarm---------------\n");
	 System.out.println("Fire Alert due to smoke levels in floor of 
"+a.getLid().substring(0, 5)+" and room "+a.getLid().substring(5, 10)+". Please 
go to a safer area.");
	 System.out.println("-----------------------------------------\n");
	 System.out.println("Sending email to "+a.getEmail());
	}
	if(a.getCo2Level() > 5) {
	 System.out.println("\n---------------------------------------\n");
	 System.out.println("Notification by FireAlarm-----------------\n");
	 System.out.println("High CO2 levels in floor of 
"+a.getLid().substring(0, 5)+" and room "+a.getLid().substring(5, 10)+". Please 
go to a safer area.");
	 System.out.println("------------------------------------------\n");
	System.out.println("Sending email to "+a.getEmail());}
}
```

Appendix

Desktop Client

1. Alarm

- AddAlarmUI

```
******************************************************************************************
package alarm;

import java.io.IOException;

import animatefx.animation.Pulse;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class AddAlarmUI {
	private static AddAlarmUI addAlarmUIInstance;

	private AddAlarmUI() {}

	public static AddAlarmUI getInstance() {
		if(addAlarmUIInstance == null) {
			synchronized (AddAlarmUI.class) {
				addAlarmUIInstance = new AddAlarmUI();
			}
		}

		return addAlarmUIInstance;
	}

	public void display(Stage addAlarmStage) throws IOException {
		Parent root = FXMLLoader.load(getClass().getResource("AddAlarmView.fxml"));

    Scene scene = new Scene(root);

    addAlarmStage.setTitle("Add Alarm");
    addAlarmStage.setScene(scene);
    addAlarmStage.show();
    new Pulse(root).play();
            }

}
```
****

- AddAlarmViewController

****

```
package alarm;

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

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import animatefx.animation.Pulse;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import location.Location;
import rmi.FASensorClient;
import user.User;

public class AddAlarmViewController implements Initializable {
        @FXML
  public TextField fxAlarmId;
  public ComboBox<Integer> fxSmokeLevel;
  public ComboBox<Integer> fxCo2Level;
  public ComboBox<String> fxLocation;
  public TextField fxHandledBy;
  public Label fxStatus;
```

```
    public Button fxAdd;

    @FXML
    public void addAlarm(ActionEvent event) throws RemoteException, IOException {
            String location = fxLocation.getValue();

            if(location == null)
                    fxStatus.setText("Set values first");
            else {
                    location = fxLocation.getValue();
            String aid = fxAlarmId.getText();
            Integer smoke = fxSmokeLevel.getValue();
            Integer co2 = fxCo2Level.getValue();
                    String handler = fxHandledBy.getText();
            String lid = location.trim().replaceFirst(" ", "");
            int activeState = (smoke > 5 || co2 > 5) ? 1 : 0;

                    boolean success =
FASensorClient.addAlarm(aid,handler,lid,smoke,co2,activeState,1);

                    if(success)
                            fxStatus.setText("Add alarm successful");
                    else
                            fxStatus.setText("Alarm add failed");
            }

            // animate
            new Pulse(fxStatus).play();
    }

            @Override
            public void initialize(URL url, ResourceBundle rb) {
                    // get new alarm id
                    try {
                            String alarm = FASensorClient.getNewAlarmID();
                            JSONObject jsonAid = new JSONObject(alarm);
                            String aid = jsonAid.get("aid").toString();
                            fxAlarmId.setText(aid);
                    }
                    catch (IOException | JSONException e) {
                            e.printStackTrace();
```

```
		}

		// set smoke Levels and default
		ObservableList<Integer> smokelist = FXCollections.observableArrayList();
		smokelist.addAll(1,2,3,4,5,6,7,8,9,10);
		fxSmokeLevel.setItems(smokelist);
		fxSmokeLevel.getSelectionModel().selectFirst();

		// set co2 Levels and default
		ObservableList<Integer> co2list = FXCollections.observableArrayList();
		co2list.addAll(1,2,3,4,5,6,7,8,9,10);
		fxCo2Level.setItems(co2list);
		fxCo2Level.getSelectionModel().selectFirst();

		// set locations in combo box
		ObservableList<String> locations = FXCollections.observableArrayList();

		JSONArray jsonArr;
		try {
			jsonArr = new JSONArray(FASensorClient.getLocations());

			for (int i=0;i<jsonArr.length();i++) {
				JSONObject o = jsonArr.getJSONObject(i);
				Location l = new Location();

				l.setLid(o.getString("lid"));
				l.setFloorNo(o.getString("floorNo"));
				l.setRoomNo(o.getString("roomNo"));

				locations.add(l.getFloorNo()+" "+l.getRoomNo());
			}

	    fxLocation.setItems(locations);
		}
		catch (JSONException | IOException e1) {
			e1.printStackTrace();
		}

		// set email as handler
		fxHandledBy.setText(User.getInstance().getEmail());
	}
```

```
}
*********************************************************************************************
```

- AlarmUI

```
*********************************************************************************************
package alarm;

import java.io.IOException;

import animatefx.animation.Pulse;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class AlarmUI {
	private static AlarmUI alarmUIInstance;

	private AlarmUI() {}

	public static AlarmUI getInstance() {
		if(alarmUIInstance == null) {
			synchronized (AlarmUI.class) {
				alarmUIInstance = new AlarmUI();
			}
		}

		return alarmUIInstance;
	}

	public void display(Stage alarmStage) throws IOException {
		Parent root = FXMLLoader.load(getClass().getResource("AlarmView.fxml"));

    Scene scene = new Scene(root);

    alarmStage.setTitle("Welcome");
    alarmStage.setScene(scene);
    alarmStage.show();

    new Pulse(root).play();
```

```
        }
}
*********************************************************************************************
```

- AlarmViewController

```
*********************************************************************************************
package alarm;

import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.util.Date;
import java.util.ResourceBundle;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import image.MyImage;
import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.concurrent.ScheduledService;
import javafx.concurrent.Task;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;
import javafx.util.Duration;
import location.AddLocationUI;
import rmi.FASensorClient;

public class AlarmViewController implements Initializable {
    ObservableList<TableAlarm> alarmslist = FXCollections.observableArrayList();

            @FXML
    public Button fxAddLocation;
    public Button fxAddAlarm;
    public Button fxRefresh;
    public Button fxEditAlarm;
    public Label fxTime;

    @FXML
    public TableView<TableAlarm> fxAlarmsTable;
    public TableColumn<TableAlarm, String> fxAlarmId;
    public TableColumn<TableAlarm, Integer> fxFloorNo;
    public TableColumn<TableAlarm, Integer> fxRoomNo;
    public TableColumn<TableAlarm, Integer> fxSmokeLevel;
    public TableColumn<TableAlarm, Integer> fxCo2Level;
    public TableColumn<TableAlarm, Integer> fxActiveState;
    public TableColumn<TableAlarm, Integer> fxWorkingState;
    public TableColumn<TableAlarm, ImageView> fxSignal;

    @FXML
    public void loadAddLocation(ActionEvent event) throws IOException {
            AddLocationUI addlocationinstance = AddLocationUI.getInstance();
      Stage addlocationstage = new Stage();
      addlocationinstance.display(addlocationstage);
    }

    @FXML
    public void loadAddAlarm(ActionEvent event) throws IOException, JSONException,
  NotBoundException {
            AddAlarmUI addalarminstance = AddAlarmUI.getInstance();
      Stage addalarmstage = new Stage();
      addalarminstance.display(addalarmstage);
    }

    @FXML
    public void loadEditAlarm(ActionEvent event) throws IOException {
            EditAlarmUI editalarminstance = EditAlarmUI.getInstance();
            Stage editalarmstage = new Stage();
            editalarminstance.display(editalarmstage);
    }
```

```
   @FXML
   public void refreshTable() throws MalformedURLException, RemoteException, IOException,
JSONException, NotBoundException {
           if(!alarmslist.isEmpty())
                   alarmslist.clear();
           String alarms = FASensorClient.getUpdatedAlarms();

           JSONArray alarmsJSONObj = new JSONArray(alarms);
           alarmslist = deserializeAndGet(alarmsJSONObj);

           fxAlarmsTable.setItems(alarmslist);
   }

   public void loadAlarmsFirst() throws MalformedURLException, RemoteException, IOException,
JSONException, NotBoundException {
           if(!alarmslist.isEmpty())
                   alarmslist.clear();
                   String alarms = FASensorClient.getInitialAlarms();
           System.out.println(alarms);

           JSONArray alarmsJSONObj = new JSONArray(alarms);
           alarmslist = deserializeAndGet(alarmsJSONObj);

           fxAlarmsTable.setItems(alarmslist);
   }

   public ObservableList<TableAlarm> deserializeAndGet(JSONArray alarmsJsArr) throws
JSONException {
           ObservableList<TableAlarm> oblist = FXCollections.observableArrayList();

           for (int i=0;i<alarmsJsArr.length();i++) {
                           JSONObject o = alarmsJsArr.getJSONObject(i);
                           TableAlarm a = new TableAlarm();

                           a.setAid(o.getString("aid"));
                           a.setCo2Level(o.getInt("co2Level"));
                           a.setEmail(o.getString("email"));
                           a.setIsActive(o.getInt("isActive"));
                           a.setIsWorking(o.getInt("isWorking"));
                           a.setLid(o.getString("lid"));
```

```
                                        a.setSmokeLevel(o.getInt("smokeLevel"));
                                        a.setFloorNo(o.getString("lid"));
                                        a.setRoomNo(o.getString("lid"));
                                        a.setSignal(getIcon(o.getInt("isWorking"), o.getInt("smokeLevel"),
o.getInt("co2Level")));

                                        oblist.add(a);
                                }

                fxTime.setText(new Date().toString());

                return oblist;
        }

        public ImageView getIcon(int isWorking, int smoke, int co2) {
                if(isWorking == 0)
                        return MyImage.getNW();
                else {
                        if(smoke > 5 || co2 > 5)
                                return MyImage.getF();
                }

                return null;
        }

        public void refreshService() {
                ScheduledService<Boolean> ssv = new ScheduledService<Boolean>() {
                                @Override
                                protected Task<Boolean> createTask() {
                                        return new Task<Boolean>() {
                                                protected Boolean call() {
                                                        Platform.runLater( () -> {
                                                                try {
                                                                        refreshTable();
                                                                }
                                                                catch (IOException | JSONException |
NotBoundException e) {
                                                                        e.printStackTrace();
                                                                }
                                                        });
                                                        return true;
                                                }
                                        };
                                }
                };
                ssv.setPeriod(Duration.millis(15000));
                ssv.start();
        }

        @Override
        public void initialize(URL url, ResourceBundle rb) {
                // set table placeholder
                fxAlarmsTable.setPlaceholder(new Label("Please wait... Connecting..."));

                try {
                                FASensorClient.registerListener();
                        }
                catch (RemoteException e) {
                                e.printStackTrace();
                        }

                // set table columns
                fxAlarmId.setCellValueFactory(new PropertyValueFactory<>("aid"));
                fxFloorNo.setCellValueFactory(new PropertyValueFactory<>("floorNo"));
                fxRoomNo.setCellValueFactory(new PropertyValueFactory<>("roomNo"));
                fxSmokeLevel.setCellValueFactory(new PropertyValueFactory<>("smokeLevel"));
                fxCo2Level.setCellValueFactory(new PropertyValueFactory<>("co2Level"));
                fxActiveState.setCellValueFactory(new PropertyValueFactory<>("isActive"));
                fxWorkingState.setCellValueFactory(new PropertyValueFactory<>("isWorking"));
                fxSignal.setCellValueFactory(new PropertyValueFactory<>("signal"));

                // set time
                fxTime.setText(new Date().toString());

                // calls to refresh every 30 seconds
                refreshService();
        }
}
****************************************************************************************************
```

- EditAlarmUI

```
****************************************************************************************************
package alarm;

import java.io.IOException;

import animatefx.animation.Pulse;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class EditAlarmUI {
	private static EditAlarmUI editAlarmInstance;

	private EditAlarmUI() {}

	public static EditAlarmUI getInstance() {
		if(editAlarmInstance == null) {
			synchronized (EditAlarmUI.class) {
				editAlarmInstance = new EditAlarmUI();
			}
		}

		return editAlarmInstance;
	}

	public void display(Stage editAlarmStage) throws IOException {
		Parent root = FXMLLoader.load(getClass().getResource("EditAlarmView.fxml"));

		Scene scene = new Scene(root);

		editAlarmStage.setTitle("Edit Alarm");
    editAlarmStage.setScene(scene);
    editAlarmStage.show();

    editAlarmStage.show();

		// animate
		new Pulse(root).play();
```

```
        }
}
**********************************************************************************************
```

- EditAlarmViewController

```
**********************************************************************************************
package alarm;

import java.io.IOException;
import java.net.URL;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.util.HashMap;
import java.util.Map;
import java.util.ResourceBundle;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import animatefx.animation.Pulse;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import location.Location;
import rmi.FASensorClient;
import user.User;

public class EditAlarmViewController implements Initializable {
        Map<String,Alarm> alarmsmap = new HashMap<>();

        @FXML
  public ComboBox<String> fxAlarmId;
  public ComboBox<Integer> fxSmokeLevel;
  public ComboBox<Integer> fxCo2Level;
```

```
    public ComboBox<String> fxLocation;
    public ComboBox<Integer> fxWorkingState;
    public Label fxHandledBy;
    public Label fxStatus;
    public Button fxEdit;

    @FXML
    public void editAlarm(ActionEvent event) throws RemoteException, IOException {
        String aid = fxAlarmId.getValue();

        if(aid != null) {
            int smoke = fxSmokeLevel.getValue();
            int co2 = fxCo2Level.getValue();
            String lid = fxLocation.getValue().trim().replaceFirst(" ", "");
            int workingState = fxWorkingState.getValue();
            String handler = fxHandledBy.getText();
            int activeState = (smoke > 5 || co2 > 5) ? 1 : 0;

            boolean success =
  FASensorClient.updateAlarm(aid,smoke,co2,lid,workingState,handler,activeState);

            if(success)
                fxStatus.setText("Update alarm successful");
            else
                fxStatus.setText("Alarm update failed");
        }
        else
            fxStatus.setText("Set details first");

        // animate
        new Pulse(fxStatus).play();
    }

    @FXML
    public void loadAlarm(ActionEvent event) {
        String key = fxAlarmId.getValue();

        Alarm a = alarmsmap.get(key);

        fxSmokeLevel.setValue(a.getSmokeLevel());
        fxCo2Level.setValue(a.getCo2Level());
		fxLocation.setValue(a.getLid().substring(0, 5)+" "+a.getLid().substring(5,10));
		fxWorkingState.setValue(a.getIsWorking());
	}

		@Override
		public void initialize(URL url, ResourceBundle rb) {
			// get alarms id s
			try {
				String alarms = FASensorClient.getInitialAlarms();

				JSONArray alarmsJsArr = new JSONArray(alarms);

				for (int i=0;i<alarmsJsArr.length();i++) {
					JSONObject o = alarmsJsArr.getJSONObject(i);
					Alarm a = new Alarm();

					a.setAid(o.getString("aid"));
					a.setCo2Level(o.getInt("co2Level"));
					a.setEmail(o.getString("email"));
					a.setIsActive(o.getInt("isActive"));
					a.setIsWorking(o.getInt("isWorking"));
					a.setLid(o.getString("lid"));
					a.setSmokeLevel(o.getInt("smokeLevel"));

					alarmsmap.put(a.getAid(), a);
				}

				ObservableList<String> aids = FXCollections.observableArrayList();
				aids.addAll(alarmsmap.keySet());

				fxAlarmId.setItems(aids);
			}
			catch (IOException | JSONException | NotBoundException e) {
				e.printStackTrace();
			}

			// set smoke Levels
			ObservableList<Integer> smokelist = FXCollections.observableArrayList();
			smokelist.addAll(1,2,3,4,5,6,7,8,9,10);
			fxSmokeLevel.setItems(smokelist);
			// set co2 Levels
			ObservableList<Integer> co2list = FXCollections.observableArrayList();
			co2list.addAll(1,2,3,4,5,6,7,8,9,10);
			fxCo2Level.setItems(co2list);

			// set locations in combo box
			ObservableList<String> locations = FXCollections.observableArrayList();

			JSONArray jsonArr;
			try {
				jsonArr = new JSONArray(FASensorClient.getLocations());

				for (int i=0;i<jsonArr.length();i++) {
					JSONObject o = jsonArr.getJSONObject(i);
					Location l = new Location();

					l.setLid(o.getString("lid"));
					l.setFloorNo(o.getString("floorNo"));
					l.setRoomNo(o.getString("roomNo"));

					locations.add(l.getFloorNo()+" "+l.getRoomNo());
				}

		fxLocation.setItems(locations);
			}
			catch (JSONException | IOException e1) {
				e1.printStackTrace();
			}

			// set working state
			ObservableList<Integer> workingstates = FXCollections.observableArrayList();
			workingstates.addAll(0,1);
			fxWorkingState.setItems(workingstates);

			// set email as handler
			fxHandledBy.setText(User.getInstance().getEmail());
		}
	}
*****************************************************************************************************
```

2. Location

- AddLocationUI

```
*************************************************************************************************
package location;

import java.io.IOException;

import animatefx.animation.Pulse;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class AddLocationUI {
	private static AddLocationUI addLocationUIInstance;

	private AddLocationUI() {}

	public static AddLocationUI getInstance() {
		if(addLocationUIInstance == null) {
			synchronized (AddLocationUI.class) {
				addLocationUIInstance = new AddLocationUI();
			}
		}

		return addLocationUIInstance;
	}

	public void display(Stage addlocationstage) throws IOException {
		Parent root =
FXMLLoader.load(getClass().getResource("AddLocationView.fxml"));

		Scene scene = new Scene(root);

		addlocationstage.setTitle("Add Location");
		addlocationstage.setScene(scene);
		addlocationstage.show();

    new Pulse(root).play();
	}
}
***********************************************************************************************
```

- AddLocationViewController

```
***********************************************************************************************
package location;

import java.io.IOException;

import animatefx.animation.Pulse;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import rmi.FASensorClient;

public class AddLocationViewController {
        @FXML
  public TextField fxFloorNo;
  public TextField fxRoomNo;
  public Button fxAddButton;
  public Label fxStatus;

  @FXML
  public void addLocation(ActionEvent event) throws IOException {
        String f = fxFloorNo.getText();
        String r = fxRoomNo.getText();
        String lid = (f+r);

        if(f.length() == 0 || r.length() == 0)
                        fxStatus.setText("Set values first");
        else {
                boolean success = FASensorClient.addLocation(f,r,lid);

                if(success)
                        fxStatus.setText("Location added");
                else
                        fxStatus.setText("Add location failed. Check your inputs.");
        }
```

```
            // animate
            new Pulse(fxStatus).play();
    }
}
```
****

3. rmi

- FAListener

****

```
package rmi;

import java.rmi.Remote;
import java.rmi.RemoteException;

import org.json.JSONException;

public interface FAListener extends Remote {
            public void alarmsChanged(String alarms) throws RemoteException, JSONException;
            public String getUpdatedAlarms() throws RemoteException, JSONException;
}
```
****

- FAMonitorImpl

****

```
package rmi;

import java.io.Serializable;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.util.List;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import alarm.Alarm;

public class FAMonitorImpl extends UnicastRemoteObject implements
FAListener,Serializable,Runnable {
```

```
private static final long serialVersionUID = 1L;
volatile List<Alarm> alarmsList = new ArrayList<>();

protected FAMonitorImpl() throws RemoteException {
	super();
}

@Override
public void alarmsChanged(String alarms) throws RemoteException, JSONException {
	JSONArray alarmsJsArr = new JSONArray(alarms);
	if(!alarmsList.isEmpty())
		alarmsList.clear();

	for (int i=0;i<alarmsJsArr.length();i++) {
		JSONObject o = alarmsJsArr.getJSONObject(i);
		Alarm a = new Alarm();

		a.setAid(o.getString("aid"));
		a.setCo2Level(o.getInt("co2Level"));
		a.setEmail(o.getString("email"));
		a.setIsActive(o.getInt("isActive"));
		a.setIsWorking(o.getInt("isWorking"));
		a.setLid(o.getString("lid"));
		a.setSmokeLevel(o.getInt("smokeLevel"));

		alarmsList.add(a);
	}
}

@Override
public void run() {
	for(;;) {
		try {
			Thread.sleep(10000);
		}
		catch (InterruptedException ie) {
			System.out.println(ie);
		}
	}
}
```

```
		@Override
		public String getUpdatedAlarms() throws RemoteException, JSONException {
			return new JSONArray(alarmsList).toString();
		}
}
```

****

- FASensor

****

```
package rmi;

import java.io.IOException;
import java.rmi.Remote;
import java.rmi.RemoteException;

import org.json.JSONException;

public interface FASensor extends Remote {
		public String getAlarms() throws IOException,JSONException;
		public boolean authenticateUser(String email,String password) throws
IOException,RemoteException;
		public boolean addFAListener(FAListener faListener) throws RemoteException;
		public boolean removeFAListener(FAListener faListener) throws RemoteException;
		public boolean addLocation(String f, String l, String lid) throws
IOException,RemoteException;
		public String getLocations() throws IOException,JSONException;
		public String getNewAlarmID() throws IOException,JSONException;
		public boolean addAlarm(String aid, String handler, String lid, int smoke, int co2, int
activeState, int workingState) throws IOException,RemoteException;
		public boolean updateAlarm(String aid, int smoke, int co2, String lid, int workingState,
String handler, int activeState) throws IOException,RemoteException;
}
```

****

- FASensor

****

```
package rmi;

import java.io.IOException;
import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.Remote;
import java.rmi.RemoteException;

import org.json.JSONException;

public class FASensorClient {
	private static FASensorClient faSensorClientInstance = null;
	private FASensorClient() {}
	static FASensor faSensor = null;
	static FAListener faListener = null;

	public static FASensorClient getInstance() {
		if(faSensorClientInstance == null) {
			synchronized (FASensorClient.class) {
				faSensorClientInstance = new FASensorClient();
			}
		}

		return faSensorClientInstance;
	}

	public FASensor setNewClient() throws MalformedURLException, RemoteException,
NotBoundException {
		System.setProperty("java.security.policy", "file:allowall.policy");

		Remote remoteSensor = Naming.lookup("//localhost:5500/faSensor");
		FASensor fas = (FASensor) remoteSensor;

		faSensor = fas;

		return faSensor;
	}

	public static void registerListener() throws RemoteException {
		FAListener faLi = new FAMonitorImpl();
		faListener = faLi;
		faSensor.addFAListener(faLi);
	}
```

```
		public static String getInitialAlarms() throws MalformedURLException, RemoteException,
IOException, JSONException, NotBoundException {
				return faSensor.getAlarms();
		}

		public static boolean addLocation(String f, String r, String lid) throws IOException {
				return faSensor.addLocation(f,r,lid);
		}

		public static FASensor getFaSensor() {
				return faSensor;
		}

		public static FAListener getFaListener() {
				return faListener;
		}

		public static String getLocations() throws IOException, JSONException {
				return faSensor.getLocations();
		}

		public static String getNewAlarmID() throws IOException, JSONException {
				return faSensor.getNewAlarmID();
		}

		public static boolean addAlarm(String aid, String handler, String lid, int smoke, int co2, int
activeState, int workingState) throws RemoteException, IOException {
				return faSensor.addAlarm(aid,handler,lid,smoke,co2,activeState,workingState);
		}

		public static String getUpdatedAlarms() throws RemoteException, JSONException {
				return faListener.getUpdatedAlarms();
		}

		public static boolean updateAlarm(String aid, int smoke, int co2, String lid, int
workingState, String handler, int activeState) throws RemoteException, IOException {
				return
faSensor.updateAlarm(aid,smoke,co2,lid,workingState,handler,activeState);
		}

}
```

****

- FASensorImpl

****

```
package rmi;

import java.io.IOException;
import java.io.Serializable;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class FASensorImpl extends UnicastRemoteObject implements
FASensor,Serializable,Runnable {
	private static final long serialVersionUID = 1L;
	private OkHttpClient httpClient = new OkHttpClient();
	private ArrayList<FAListener> listeners = new ArrayList<>();
	volatile JSONArray alarms = null;

	protected FASensorImpl() throws RemoteException {
		super();
	}

	@Override
	public boolean authenticateUser(String email, String password) throws
IOException,RemoteException {
		String json = new StringBuilder()
		.append("{"
			+ "\"email\":\""+email+"\","
			+ "\"password\":\""+password+"\""
```

```
                        + "}").toString();

                RequestBody requestBody = RequestBody.create (
                        MediaType.parse("application/json; charset=UTF-8"), json
                );

                Request request = new Request.Builder()
                .url("http://localhost:8080/FireAlarmRest/rest/UserService/authenticateUser")
                .post(requestBody)
                .build();

                try (Response response = httpClient.newCall(request).execute()) {
                        int code = response.code();

                        if(code == 201)
                                return true;
                        else
                                return false;
                }
        }

        @Override
        public String getAlarms() throws IOException, JSONException {
                Request request = new Request.Builder()
.url("http://localhost:8080/FireAlarmRest/rest/AlarmService/getAlarms")
.build();

try (Response response = httpClient.newCall(request).execute()) {
  if (!response.isSuccessful())
    throw new IOException("AlarmService not responding" + response);

  String resBody = response.body().string();
  alarms = new JSONArray(resBody);
}

                return alarms.toString();
        }

        @Override
        public boolean addFAListener(FAListener faListener) {
                if(listeners.add(faListener))
				return true;
			else
				return false;
		}

		@Override
		public boolean removeFAListener(FAListener faListener) {
			if(listeners.remove(faListener))
				return true;
			else
				return false;
		}

		@Override
		public boolean addLocation(String f, String r, String lid) throws
	IOException,RemoteException {
			String json = new StringBuilder()
			.append("{"
				+ "\"lid\":\""+lid+"\","
				+ "\"floorNo\":\""+f+"\","
				+ "\"roomNo\":\""+r+"\""
				+ "}").toString();

			RequestBody requestBody = RequestBody.create (
				MediaType.parse("application/json; charset=UTF-8"), json
			);

			Request request = new Request.Builder()
			.url("http://localhost:8080/FireAlarmRest/rest/LocationService/addLocation")
			.post(requestBody)
			.build();

			try (Response response = httpClient.newCall(request).execute()) {
				int code = response.code();

				if(code == 201)
					return true;
				else
					return false;
			}
		}
	@Override
	public String getLocations() throws IOException, JSONException {
		JSONArray locations = null;

		Request request = new Request.Builder()
.url("http://localhost:8080/FireAlarmRest/rest/LocationService/getLocations")
.build();

try (Response response = httpClient.newCall(request).execute()) {
  if (!response.isSuccessful())
    throw new IOException("LocationService not responding" + response);

  String resBody = response.body().string();
  locations = new JSONArray(resBody);
}

		return locations.toString();
	}

	@Override
	public String getNewAlarmID() throws IOException, JSONException {
		JSONObject newAid = null;

		Request request = new Request.Builder()
.url("http://localhost:8080/FireAlarmRest/rest/AlarmService/getNewAlarmId")
.build();

try (Response response = httpClient.newCall(request).execute()) {
  if (!response.isSuccessful())
    throw new IOException("AlarmService not responding" + response);

  String resBody = response.body().string();
  newAid = new JSONObject(resBody);
}

		return newAid.toString();
	}

	@Override
```

```
        public boolean addAlarm(String aid, String handler, String lid, int smoke, int co2, int
activeState, int workingState) throws IOException, RemoteException {
                String json = new StringBuilder()
                .append("{"
                        + "\"aid\":\""+aid+"\","
                        + "\"email\":\""+handler+"\","
                        + "\"lid\":\""+lid+"\","
                        + "\"smokeLevel\":\""+smoke+"\","
                        + "\"co2Level\":\""+co2+"\","
                        + "\"isActive\":\""+activeState+"\","
                        + "\"isWorking\":\""+workingState+"\""
                        + "}").toString();

                RequestBody requestBody = RequestBody.create (
                        MediaType.parse("application/json; charset=UTF-8"), json
                );

                Request request = new Request.Builder()
                .url("http://localhost:8080/FireAlarmRest/rest/AlarmService/addAlarm")
                .post(requestBody)
                .build();

                try (Response response = httpClient.newCall(request).execute()) {
                        int code = response.code();

                        if(code == 201)
                                return true;
                        else
                                return false;
                }
        }

        @Override
        public boolean updateAlarm(String aid, int smoke, int co2, String lid, int workingState,
String handler, int activeState) throws IOException, RemoteException {
                String json = new StringBuilder()
                .append("{"
                        + "\"aid\":\""+aid+"\","
                        + "\"email\":\""+handler+"\","
                        + "\"lid\":\""+lid+"\","
                        + "\"smokeLevel\":\""+smoke+"\","
                + "\"co2Level\":\""+co2+"\","
                + "\"isActive\":\""+activeState+"\","
                + "\"isWorking\":\""+workingState+"\""
                + "}").toString();

        RequestBody requestBody = RequestBody.create (
                MediaType.parse("application/json; charset=UTF-8"), json
        );

        Request request = new Request.Builder()
        .url("http://localhost:8080/FireAlarmRest/rest/AlarmService/updateAlarm")
        .put(requestBody)
        .build();

        try (Response response = httpClient.newCall(request).execute()) {
                int code = response.code();

                if(code == 201)
                        return true;
                else
                        return false;
        }
}

public void run() {
        for(;;) {
                try {
                        Thread.sleep(15000);
                }
                catch (InterruptedException ie) {
                        System.out.println(ie);
                }

                try {
                        getAlarms();
                } catch (IOException | JSONException e1) {
                        e1.printStackTrace();
                }

                try {
                        notifyOthers();
                        } catch (IOException | JSONException e2) {
                            e2.printStackTrace();
                        }
                    }
                }

                public void notifyOthers() throws IOException, JSONException {
                    for(FAListener listener: listeners) {
                        try {
                            listener.alarmsChanged(alarms.toString());
                        }
                        catch(RemoteException re) {
                            System.out.println(re);
                        }
                    }
                }

            }
```

****

- FASensorServer

****

```
package rmi;

import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;

public class FASensorServer {
    private static FASensorServer faSensorServerInstance = null;
    private FASensorServer() {}

    public static FASensorServer getInstance() {
        if(faSensorServerInstance == null) {
            synchronized (FASensorServer.class) {
                faSensorServerInstance = new FASensorServer();
            }
        }

        return faSensorServerInstance;
```

```
		}

		//public static void main(String[] args) throws RemoteException {
		public void startServer() throws RemoteException {
			System.setProperty("java.security.policy", "file:allowall.policy");

			FASensor faSensor = (FASensor) new FASensorImpl();

			try {
				Naming.rebind("rmi://localhost:5500/faSensor", faSensor);

				Thread thread = new Thread((Runnable) faSensor);
				thread.start();

				System.out.println("FireAlarmSensorServer ONLINE...");
			}
			catch (RemoteException | MalformedURLException e) {
				System.out.println("FireAlarmSensorServer Failed");
				System.out.println(e);
			}
		//}
		}
	}
```

****

4. User

- Login

****

```
package login;

import java.rmi.RemoteException;

import animatefx.animation.Pulse;
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.stage.StageStyle;
import rmi.FASensorServer;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.fxml.FXMLLoader;
```

```
public class Login extends Application {
        @Override
        public void start(Stage loginStage) {
                try {
                        Parent root =
FXMLLoader.load(getClass().getResource("LoginView.fxml"));

            Scene scene = new Scene(root);

            loginStage.setTitle("RedCare Login");
            loginStage.resizableProperty().setValue(Boolean.TRUE);
            loginStage.initStyle(StageStyle.DECORATED);
            loginStage.setScene(scene);
            loginStage.show();

            //animate the stage
            new Pulse(root).play();
                }
                catch(Exception e) {
                        e.printStackTrace();
                }
        }

        public static void main(String[] args) throws RemoteException {
                FASensorServer.getInstance().startServer();
                launch(args);
        }
}
```
****

- LoginController

****

```
package login;

import alarm.AlarmUI;
import animatefx.animation.Pulse;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
import rmi.FASensor;
import rmi.FASensorClient;
import user.User;

public class LoginController {
        @FXML
        public Label status;
        public TextField email;
        public PasswordField pass;
        public Button loginBtn;

        @FXML
        public void Login(ActionEvent event) throws Exception {
                String e = email.getText();
    String p = pass.getText();

    try {
        FASensor fas = (FASensor) FASensorClient.getInstance().setNewClient();
            boolean isValid = fas.authenticateUser(e, p);
            System.out.println(isValid);

                        if(isValid) {
                                User.getInstance().setEmail(e);
                                User.getInstance().setPassword(p);

                                status.setText("Login Successful");

                                Stage stage = (Stage) loginBtn.getScene().getWindow();
              stage.close();

              AlarmUI alarmUIInstance = AlarmUI.getInstance();
              Stage alarmStage = new Stage();
              alarmUIInstance.display(alarmStage);
                        }
                        else
                                status.setText("Login Failed");
```

```
                                        // animate
                                        new Pulse(status).play();
        }
        catch(Exception e1) {
                System.out.println(e1);
        }
                }

}
```
****

ASSIGNMENT 2

Entities

- Alarm

****

```
package com;

public class Alarm {
	private String aid,email,lid;
	private int smokeLevel,co2Level,isActive,isWorking;

	public Alarm() {
		super();
	}

	public Alarm(String aid, String email, String lid, int smokeLevel, int
co2Level, int isActive, int isWorking) {
		super();
		this.aid = aid;
		this.email = email;
		this.lid = lid;
		this.smokeLevel = smokeLevel;
		this.co2Level = co2Level;
		this.isActive = isActive;
		this.isWorking = isWorking;
	}

	public String getAid() {
		return aid;
	}

	public void setAid(String aid) {
		this.aid = aid;
	}
```

```
public String getEmail() {
		return email;
}

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

public String getLid() {
		return lid;
}

public void setLid(String lid) {
		this.lid = lid;
}

public int getSmokeLevel() {
		return smokeLevel;
}

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

public int getCo2Level() {
		return co2Level;
}

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

public int getIsActive() {
		return isActive;
```

```
	}

	public void setIsActive(int isActive) {
		this.isActive = isActive;
	}

	public int getIsWorking() {
		return isWorking;
	}

	public void setIsWorking(int isWorking) {
		this.isWorking = isWorking;
	}

	@Override
	public String toString() {
		return "Alarm [aid=" + aid + ", email=" + email + ", lid=" + lid + ",
smokeLevel=" + smokeLevel + ", co2Level="
				+ co2Level + ", isActive=" + isActive + ", isWorking=" +
isWorking + "]";
	}

}
********************************************************************************
****************

* Location
********************************************************************************
****************
package com;

public class Location {
	private String lid,floorNo,roomNo;
```

```
public Location() {
        super();
}

public Location(String lid, String floorNo, String roomNo) {
        super();
        this.lid = lid;
        this.floorNo = floorNo;
        this.roomNo = roomNo;
}

public String getLid() {
        return lid;
}

public void setLid(String lid) {
        this.lid = lid;
}

public String getFloorNo() {
        return floorNo;
}

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

public String getRoomNo() {
        return roomNo;
}

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

```
	@Override
	public String toString() {
		return "Location [lid=" + lid + ", floorNo=" + floorNo + ", roomNo="
+ roomNo + "]";
	}

}
********************************************************************************
****************

* TableAlarm
********************************************************************************
****************
package alarm;

import javafx.scene.image.ImageView;

public class TableAlarm extends Alarm {
	private String floorNo,roomNo;
	private ImageView signal;

	public TableAlarm() {
		super();
	}

	public TableAlarm(String aid, String email, String lid, int smokeLevel, int
co2Level, int isActive, int isWorking) {
		super(aid, email, lid, smokeLevel, co2Level, isActive, isWorking);
		this.floorNo = lid.substring(0, 5);
		this.roomNo = lid.substring(5, 10);
	}

	public String getFloorNo() {
		return floorNo;
	}

	public void setFloorNo(String lid) {
		this.floorNo = lid.substring(0, 5);
	}

	public String getRoomNo() {
		return roomNo;
	}

	public void setRoomNo(String lid) {
		this.roomNo = lid.substring(5, 10);
	}

	public ImageView getSignal() {
		return signal;
	}

	public void setSignal(ImageView signal) {
		this.signal = signal;
	}

}
*********************************************************************************
****************

* Updates
*********************************************************************************
****************
package com;

public class Updates {
	private String aid,occured;
	private int smokeLevel,co2Level,isActive,isWorking;

	public Updates() {
		super();
	}

	public Updates(String aid, String occured, int smokeLevel, int co2Level, int
isActive, int isWorking) {
		super();
		this.aid = aid;
		this.occured = occured;
		this.smokeLevel = smokeLevel;
		this.co2Level = co2Level;
		this.isActive = isActive;
		this.isWorking = isWorking;
	}

	public String getAid() {
		return aid;
	}

	public void setAid(String aid) {
		this.aid = aid;
	}

	public String getOccured() {
		return occured;
	}

	public void setOccured(String occured) {
		this.occured = occured;
	}

	public int getSmokeLevel() {
		return smokeLevel;
	}

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

	public int getCo2Level() {
		return co2Level;
	}

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

	public int getIsActive() {
		return isActive;
	}

	public void setIsActive(int isActive) {
		this.isActive = isActive;
	}

	public int getIsWorking() {
		return isWorking;
	}

	public void setIsWorking(int isWorking) {
		this.isWorking = isWorking;
	}

	@Override
	public String toString() {
			return "Updates [aid=" + aid + ", occured=" + occured + ",
smokeLevel=" + smokeLevel + ", co2Level=" + co2Level
					+ ", isActive=" + isActive + ", isWorking=" + isWorking
+ "]";
		}

}
****************************************************************************************
*****************

* User
****************************************************************************************
*****************
package com;

public class User {
	private String email,password,name,phone,address,bio,img;

	public User() {
		super();
	}

	public User(String email, String password, String name, String phone,
String address, String bio, String img) {
		super();
		this.email = email;
		this.password = password;
		this.name = name;
		this.phone = phone;
		this.address = address;
		this.bio = bio;
		this.img = img;
	}
```

```
public String getEmail() {
        return email;
}

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

public String getPassword() {
        return password;
}

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

public String getName() {
        return name;
}

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

public String getPhone() {
        return phone;
}

public void setPhone(String phone) {
        this.phone = phone;
}

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

```
	}

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

	public String getBio() {
		return bio;
	}

	public void setBio(String bio) {
		this.bio = bio;
	}

	public String getImg() {
		return img;
	}

	public void setImg(String img) {
		this.img = img;
	}

	@Override
	public String toString() {
		return "User [email=" + email + ", password=" + password + ",
name=" + name + ", phone=" + phone + ", address="
				+ address + ", bio=" + bio + ", img=" + img + "]";
	}

}
```

****

REST API

```
* AlarmDao
********************************************************************************
****************
package com;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class AlarmDao {
	static Connection con;
	static PreparedStatement ps;
	static ResultSet rs;

	public static List<Alarm> getAlarms() {
		List<Alarm> list = new ArrayList<>();

		try {
			con = ConnectFA.connect();
			String queryA = "SELECT * FROM Alarm ;";
			ps = con.prepareStatement(queryA);
			rs = ps.executeQuery();

			while(rs.next()) {
				Alarm a = new Alarm();

				a.setAid(rs.getString(1));
				a.setSmokeLevel(rs.getInt(2));
				a.setCo2Level(rs.getInt(3));
				a.setIsActive(rs.getInt(4));
```

```
                        a.setIsWorking(rs.getInt(5));
                        a.setEmail(rs.getString(6));
                        a.setLid(rs.getString(7));

                        list.add(a);
                }

                return list;
        }
        catch(SQLException sqle) {
                System.out.println(sqle);
        }
        finally {
                if(rs != null) try { rs.close(); } catch(Exception e) {}
if(ps != null) try { ps.close(); } catch(Exception e) {}
if(con != null) try { con.close(); } catch(Exception e) {}
        }

        return null;
 }

 public static Alarm getAlarm(String aid) {
        Alarm a = null;

        try {
                con = ConnectFA.connect();
                String queryA = "SELECT * FROM Alarm WHERE aid = ? ;";
                ps = con.prepareStatement(queryA);
                ps.setString(1, aid);
                rs = ps.executeQuery();

                while (rs.next()) {
                        a = new Alarm();
                        a.setAid(rs.getString(1));
                        a.setSmokeLevel(rs.getInt(2));
                        a.setCo2Level(rs.getInt(3));
                        a.setIsActive(rs.getInt(4));
                        a.setIsWorking(rs.getInt(5));
                        a.setEmail(rs.getString(6));
                        a.setLid(rs.getString(7));
                }

                return a;
        }
        catch(SQLException sqle) {
                System.out.println(sqle);
        }
        finally {
                if(rs != null) try { rs.close(); } catch(Exception e) {}
if(ps != null) try { ps.close(); } catch(Exception e) {}
if(con != null) try { con.close(); } catch(Exception e) {}
        }

        return null;
 }

 public static String getLastAlarmId() {
        String a = null;

        try {
                con = ConnectFA.connect();
                String queryA = "SELECT MAX(aid) FROM Alarm ;";
                ps = con.prepareStatement(queryA);
                rs = ps.executeQuery();

                while (rs.next()) {
                        a = rs.getString(1);
```

```
				}

				return a;
			}
			catch(SQLException sqle) {
				System.out.println(sqle);
			}
			finally {
				if(rs != null) try { rs.close(); } catch(Exception e) {}
	if(ps != null) try { ps.close(); } catch(Exception e) {}
	if(con != null) try { con.close(); } catch(Exception e) {}
			}

			return null;
		}

		public static boolean addAlarm(Alarm a) {
			try {
				con = ConnectFA.connect();
				String queryA = "INSERT INTO Alarm
(aid,smokeLevel,CO2Level,isActive,isWorking,email,lid) VALUES (?,?,?,?,?,?,?) ;";
				ps = con.prepareStatement(queryA);
				ps.setString(1, a.getAid());
				ps.setInt(2, a.getSmokeLevel());
				ps.setInt(3, a.getCo2Level());
				ps.setInt(4, a.getIsActive());
				ps.setInt(5, a.getIsWorking());
				ps.setString(6, a.getEmail());
				ps.setString(7, a.getLid());
				int count = ps.executeUpdate();
				if(count > 0)
					return true;
				else
					return false;
		}
		catch(SQLException sqle) {
			System.out.println(sqle);
		}
		finally {
			if(rs != null) try { rs.close(); } catch(Exception e) {}
	if(ps != null) try { ps.close(); } catch(Exception e) {}
	if(con != null) try { con.close(); } catch(Exception e) {}
		}

		return false;
	}

	public static boolean updateAlarm(Alarm a) {
		try {
			con = ConnectFA.connect();
			String queryA = "UPDATE Alarm SET smokeLevel
= ?,CO2Level = ?,isActive = ?,isWorking = ?,email = ?,lid = ? WHERE aid = ? ;";
			ps = con.prepareStatement(queryA);
			ps.setInt(1, a.getSmokeLevel());
			ps.setInt(2, a.getCo2Level());
			ps.setInt(3, a.getIsActive());
			ps.setInt(4, a.getIsWorking());
			ps.setString(5, a.getEmail());
			ps.setString(6, a.getLid());
			ps.setString(7, a.getAid());
			int count = ps.executeUpdate();

			if(count > 0)
				return true;
			else
				return false;
		}
		catch(SQLException sqle) {
```

```
                        System.out.println(sqle);
                }
                finally {
                        if(rs != null) try { rs.close(); } catch(Exception e) {}
        if(ps != null) try { ps.close(); } catch(Exception e) {}
        if(con != null) try { con.close(); } catch(Exception e) {}
                }

                return false;
        }

        public static boolean deleteAlarm(String aid) {
                try {
                        con = ConnectFA.connect();
                        String queryA = "UPDATE Alarm SET isWorking = ? WHERE
aid = ? ;";
                        ps = con.prepareStatement(queryA);
                        ps.setInt(1, 0);
                        ps.setString(2, aid);
                        int count = ps.executeUpdate();

                        if(count > 0)
                                return true;
                        else
                                return false;
                }
                catch(SQLException sqle) {
                        System.out.println(sqle);
                }
                finally {
                        if(rs != null) try { rs.close(); } catch(Exception e) {}
        if(ps != null) try { ps.close(); } catch(Exception e) {}
        if(con != null) try { con.close(); } catch(Exception e) {}
                }
```

```
			return false;
		}
}
*********************************************************************************
****************

* AlarmService
*********************************************************************************
****************
package com;

import java.util.List;

import javax.ws.rs.DELETE;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/AlarmService")
public class AlarmService {
		@GET
		@Path("/")
		@Produces(MediaType.TEXT_PLAIN)
		public String Hello() {
			return "AlarmService ONLINE...";
		}

		@GET
```

```
@Path("/getAlarms")
@Produces(MediaType.APPLICATION_JSON)
public List<Alarm> getAlarms() {
        List<Alarm> list = AlarmDao.getAlarms();

        return list;
}

@GET
@Path("/getAlarm/{aid}")
@Produces(MediaType.APPLICATION_JSON)
public Alarm getAlarm(@PathParam("aid") String aid) {
        Alarm alarm = AlarmDao.getAlarm(aid);

        return alarm;
}

@GET
@Path("/getNewAlarmId")
@Produces(MediaType.APPLICATION_JSON)
public Alarm getNewAlarmId() {
        Alarm a = new Alarm();
        String lastId = AlarmDao.getLastAlarmId();
        String newId = IdentityFA.getID("Alarm", lastId);
        a.setAid(newId);

        return a;
}

@POST
@Path("/addAlarm")
@Produces(MediaType.APPLICATION_JSON)
public Response addAlarm(Alarm a) {
        boolean state = AlarmDao.addAlarm(a);
```

```
		if(state) {
			Alarm newAlarm = AlarmDao.getAlarm(a.getAid());
			String success = "Alarm added successfully \n"+newAlarm;
			return Response.status(201).entity(success).build();
		}
		else {
			Alarm newAlarm = null;
			String failure= "Add alarm failed \nCheck your aid,lid and 
email \n"+newAlarm;
			return Response.status(400).entity(failure).build();
		}
	}

	@PUT
	@Path("/updateAlarm")
	@Produces(MediaType.APPLICATION_JSON)
	public Response updateAlarm(Alarm a) {
		boolean state = AlarmDao.updateAlarm(a);
		if(state) {
			Alarm newAlarm = AlarmDao.getAlarm(a.getAid());
			String success = "Alarm updated successfully \n"+newAlarm;
			return Response.status(201).entity(success).build();
		}
		else {
			Alarm newAlarm = null;
			String failure= "Update alarm failed \nCheck your email and 
lid \n"+newAlarm;
			return Response.status(400).entity(failure).build();
		}
	}

	@DELETE
	@Path("/deleteAlarm/{aid}")
	@Produces(MediaType.APPLICATION_JSON)
	public Response deleteAlarm(@PathParam("aid") String aid) {
		boolean state = AlarmDao.deleteAlarm(aid);
		if(state) {
			String success = "Alarm "+aid+" deactivated successfully";
			return Response.status(201).entity(success).build();
		}
		else {
			String failure= "Deactivate alarm "+aid+" failed \nCheck your 
aid";
			return Response.status(400).entity(failure).build();
		}
	}
}
**********************************************************************************
****************

* ConnectFA
**********************************************************************************
****************
package com;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class ConnectFA {
	private static Connection con;
	private static String connectionURL = 
"jdbc:mysql://localhost:3306/FireAlarmDB?useUnicode=true&useJDBCComplian
tTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";
	private static String user = "root";
	private static String password = "";
	static PreparedStatement stmt;
```

```
        public static Connection connect() {
                try {
                        Class.forName("com.mysql.cj.jdbc.Driver");
                        con = DriverManager.getConnection(connectionURL, user,
password);
                }
                catch(ClassNotFoundException | SQLException e) {
                        System.out.println(e);
                }

                return con;
        }

}
*******************************************************************************
****************

* IdentityFA
*******************************************************************************
****************
package com;

public class IdentityFA {
        public static String getID(String tableName, String lastID) {
                try {
       String prevID = "",oneSubID = "",newSubID = "",newID = "";
       char letter = '0';
       int newOneSubID;

       switch (tableName) {
         case "Alarm":
           letter = 'A';
           prevID = lastID;
```

```
                break;
              default:
                    break;
            }

            if(prevID != null) {
              oneSubID = 1 + prevID.substring(1);
              newOneSubID = Integer.parseInt(oneSubID)+1;
              newSubID = Integer.toString(newOneSubID);
              newID = letter+newSubID.substring(1);
              System.out.println(newID);
            }
            else
              return letter+"000000001";

            return newID;
          }
          catch(Exception e) {
            System.out.println(e);
          }

          return null;
             }
}
********************************************************************************
****************

* LocationDao
********************************************************************************
****************
package com;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class LocationDao {
        static Connection con;
        static PreparedStatement ps;
        static ResultSet rs;

        public static List<Location> getLocations() {
                List<Location> list = new ArrayList<>();

                try {
                        con = ConnectFA.connect();
                        String queryA = "SELECT * FROM Location;";
                        ps = con.prepareStatement(queryA);
                        rs = ps.executeQuery();

                        while(rs.next()) {
                                Location a = new Location();

                                a.setLid(rs.getString(1));
                                a.setFloorNo(rs.getString(2));
                                a.setRoomNo(rs.getString(3));

                                list.add(a);
                        }

                        return list;
                }
                catch(SQLException sqle) {
                        System.out.println(sqle);
                }
```

```
			finally {
				if(rs != null) try { rs.close(); } catch(Exception e) {}
if(ps != null) try { ps.close(); } catch(Exception e) {}
if(con != null) try { con.close(); } catch(Exception e) {}
			}

			return null;
		}

		public static Location getLocation(String lid) {
			Location l = null;

			try {
				con = ConnectFA.connect();
				String queryA = "SELECT * FROM Location WHERE lid = ? ;";
				ps = con.prepareStatement(queryA);
				ps.setString(1, lid);
				rs = ps.executeQuery();

				while (rs.next()) {
					l = new Location();

					l.setLid(rs.getString(1));
					l.setFloorNo(rs.getString(2));
					l.setRoomNo(rs.getString(3));
				}

				return l;
			}
			catch(SQLException sqle) {
				System.out.println(sqle);
			}
			finally {
				if(rs != null) try { rs.close(); } catch(Exception e) {}
```

```
        if(ps != null) try { ps.close(); } catch(Exception e) {}
        if(con != null) try { con.close(); } catch(Exception e) {}
                }

                return null;
         }

         public static boolean addLocation(Location l) {
                try {
                        con = ConnectFA.connect();
                        String queryA = "INSERT INTO Location
(lid,floorNo,roomNo) VALUES (?,?,?) ;";
                        ps = con.prepareStatement(queryA);
                        ps.setString(1, l.getLid());
                        ps.setString(2, l.getFloorNo());
                        ps.setString(3, l.getRoomNo());
                        int count = ps.executeUpdate();
                        if(count > 0)
                                return true;
                        else
                                return false;
                }
                catch(SQLException sqle) {
                        System.out.println(sqle);
                }
                finally {
                        if(rs != null) try { rs.close(); } catch(Exception e) {}
        if(ps != null) try { ps.close(); } catch(Exception e) {}
        if(con != null) try { con.close(); } catch(Exception e) {}
                }

                return false;
         }
        public static boolean updateLocation(Location l) {
                String f = l.getFloorNo();
                String r = l.getRoomNo();
                String fr = f+r;

                try {
                        con = ConnectFA.connect();
                        String queryA = "UPDATE Location SET lid = ?,floorNo
= ?,roomNo = ? WHERE lid = ? ;";
                        ps = con.prepareStatement(queryA);
                        ps.setString(1, fr);
                        ps.setString(2, f);
                        ps.setString(3, r);
                        ps.setString(4, l.getLid());
                        int count = ps.executeUpdate();

                        if(count > 0)
                                return true;
                        else
                                return false;
                }
                catch(SQLException sqle) {
                        System.out.println(sqle);
                }
                finally {
                        if(rs != null) try { rs.close(); } catch(Exception e) {}
       if(ps != null) try { ps.close(); } catch(Exception e) {}
       if(con != null) try { con.close(); } catch(Exception e) {}
                }

                return false;
        }
}
```

```
*********************************************************************************
****************

* LocationService
*********************************************************************************
****************
package com;

import java.util.List;

import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.PUT;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/LocationService")
public class LocationService {
	@GET
	@Path("/")
	@Produces(MediaType.TEXT_PLAIN)
	public String Hello() {
		return "LocationService ONLINE...";
	}

	@GET
	@Path("/getLocations")
	@Produces(MediaType.APPLICATION_JSON)
	public List<Location> getLocations() {
		List<Location> list = LocationDao.getLocations();
		return list;
	}

	@GET
	@Path("/getLocation/{lid}")
	@Produces(MediaType.APPLICATION_JSON)
	public Location getLocation(@PathParam("lid") String lid) {
		Location location = LocationDao.getLocation(lid);

		return location;
	}

	@POST
	@Path("/addLocation")
	@Produces(MediaType.APPLICATION_JSON)
	public Response addLocation(Location l) {
		boolean state = LocationDao.addLocation(l);
		if(state) {
			Location newLocation = LocationDao.getLocation(l.getLid());
			String success = "Location added successfully
\n"+newLocation;
			return Response.status(201).entity(success).build();
		}
		else {
			Alarm newLocation = null;
			String failure= "Add location failed \nCheck your lid
\n"+newLocation;
			return Response.status(400).entity(failure).build();
		}
	}

	@PUT
	@Path("/updateLocation")
	@Produces(MediaType.APPLICATION_JSON)
	public Response updateLocation(Location l) {
		boolean state = LocationDao.updateLocation(l);
		if(state) {
			Location newLocation = 
LocationDao.getLocation(l.getFloorNo()+l.getRoomNo());
			String success = "Location updated successfully 
\n"+newLocation;
			return Response.status(201).entity(success).build();
		}
		else {
			Alarm newLocation = null;
			String failure= "Update location failed \nCheck your lid 
\n"+newLocation;
			return Response.status(400).entity(failure).build();
		}
	}
}
**********************************************************************************
****************

* UpdatesDao
**********************************************************************************
****************
package com;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class UpdatesDao {
	static Connection con;
static PreparedStatement ps;
static ResultSet rs;

public static List<Updates> getUpdates() {
        List<Updates> list = new ArrayList<>();

        try {
                con = ConnectFA.connect();
                String queryA = "SELECT * FROM Updates;";
                ps = con.prepareStatement(queryA);
                rs = ps.executeQuery();

                while(rs.next()) {
                        Updates a = new Updates();

                        a.setAid(rs.getString(1));
                        a.setOccured(rs.getString(2));
                        a.setSmokeLevel(rs.getInt(3));
                        a.setCo2Level(rs.getInt(4));
                        a.setIsActive(rs.getInt(5));
                        a.setIsWorking(rs.getInt(6));

                        list.add(a);
                }

                return list;
        }
        catch(SQLException sqle) {
                System.out.println(sqle);
        }
        finally {
                if(rs != null) try { rs.close(); } catch(Exception e) {}
if(ps != null) try { ps.close(); } catch(Exception e) {}
if(con != null) try { con.close(); } catch(Exception e) {}
```

```
		}

		return null;
	}
}
*******************************************************************************
****************

* UpdatesService
*******************************************************************************
****************
package com;

import java.util.List;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/UpdatesService")
public class UpdatesService {
	@GET
	@Path("/")
	@Produces(MediaType.TEXT_PLAIN)
	public String Hello() {
		return "UpdatesService ONLINE...";
	}

	@GET
	@Path("/getUpdates")
	@Produces(MediaType.APPLICATION_JSON)
	public List<Updates> getUpdates() {
		List<Updates> list = UpdatesDao.getUpdates();
```

```
                    return list;
            }
}
********************************************************************************
****************

* UserDao
********************************************************************************
****************
package com;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class UserDao {
            static Connection con;
    static PreparedStatement ps;
    static ResultSet rs;

            public static User authenticateUser(String email, String password) {
                    User dbUser = null;

                    try {
                            con = ConnectFA.connect();
                            String queryA = "SELECT * FROM UserFA WHERE email = ?
AND password = ?;";
                            ps = con.prepareStatement(queryA);
                            ps.setString(1, email);
                            ps.setString(2, password);
                            rs = ps.executeQuery();

                            while(rs.next()) {
```

```
                        dbUser = new User();

                        dbUser.setEmail(rs.getString(1));
                        dbUser.setPassword(rs.getString(2));
                        dbUser.setName(rs.getString(3));
                        dbUser.setPhone(rs.getString(4));
                        dbUser.setAddress(rs.getString(5));
                        dbUser.setBio(rs.getString(6));
                        dbUser.setImg(rs.getString(7));
                    }

                    return dbUser;
                }
                catch(Exception e) {
                    System.out.println(e);
                }
                finally {
                    if(rs != null) try { rs.close(); } catch(Exception e) {}
            if(ps != null) try { ps.close(); } catch(Exception e) {}
            if(con != null) try { con.close(); } catch(Exception e) {}
                }

                return dbUser;
            }

        }
        ***********************************************************************************
        ****************

        * UserService
        ***********************************************************************************
        ****************
        package com;
```

```
import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

@Path("/UserService")
public class UserService {
	@GET
	@Path("/")
	@Produces(MediaType.TEXT_PLAIN)
	public String Hello() {
		return "UserService ONLINE...";
	}

	@POST
	@Path("/authenticateUser")
	@Consumes(MediaType.APPLICATION_JSON)
	public Response authenticateUser(User postUser) {
		User user = UserDao.authenticateUser(postUser.getEmail(), 
postUser.getPassword());
		if(user != null) {
			String success = "User authenticated \n"+user;
			return Response.status(201).entity(success).build();
		}
		else {
			String failure = "User authentication failed \n"+user;
			return Response.status(403).entity(failure).build();
		}
	}
}
```

****
****

RMI Server

* FAListener
****
****
```
package com;

import java.rmi.Remote;
import java.rmi.RemoteException;

import org.json.JSONException;

public interface FAListener extends Remote {
	public void alarmsChanged(String alarms) throws RemoteException,
JSONException;
	public String getUpdatedAlarms() throws RemoteException,
JSONException;
}
```
****
****

*FASensor
****
****
```
package com;

import java.io.IOException;
import java.rmi.Remote;
import java.rmi.RemoteException;

import org.json.JSONException;
```

```
public interface FASensor extends Remote {
	public String getAlarms() throws IOException,JSONException;
	public boolean authenticateUser(String email,String password) throws
IOException,RemoteException;
	public boolean addFAListener(FAListener faListener) throws
RemoteException;
	public boolean removeFAListener(FAListener faListener) throws
RemoteException;
	public boolean addLocation(String f, String l, String lid) throws
IOException,RemoteException;
	public String getLocations() throws IOException,JSONException;
	public String getNewAlarmID() throws IOException,JSONException;
	public boolean addAlarm(String aid, String handler, String lid, int smoke,
int co2, int activeState, int workingState) throws IOException,RemoteException;
	public boolean updateAlarm(String aid, int smoke, int co2, String lid, int
workingState, String handler, int activeState) throws
IOException,RemoteException;
}
*********************************************************************************
****************

* FASensorImpl
*********************************************************************************
****************
package com;

import java.io.IOException;
import java.io.Serializable;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class FASensorImpl extends UnicastRemoteObject implements
FASensor,Serializable,Runnable {
	private static final long serialVersionUID = 1L;
	private OkHttpClient httpClient = new OkHttpClient();
	private ArrayList<FAListener> listeners = new ArrayList<>();
	volatile JSONArray alarms = null;

	protected FASensorImpl() throws RemoteException {
		super();
	}

	@Override
	public boolean authenticateUser(String email, String password) throws
IOException,RemoteException {
		String json = new StringBuilder()
		.append("{"
			+ "\"email\":\""+email+"\","
			+ "\"password\":\""+password+"\""
			+ "}").toString();

		RequestBody requestBody = RequestBody.create (
			MediaType.parse("application/json; charset=UTF-8"), json
		);

		Request request = new Request.Builder()
```

```
        .url("http://localhost:8080/FireAlarmRest/rest/UserService/authenticate
User")
                .post(requestBody)
                .build();

                try (Response response = httpClient.newCall(request).execute()) {
                        int code = response.code();

                        if(code == 201)
                                return true;
                        else
                                return false;
                }
        }

        @Override
        public String getAlarms() throws IOException, JSONException {
                Request request = new Request.Builder()
     .url("http://localhost:8080/FireAlarmRest/rest/AlarmService/getAlarms")
     .build();

     try (Response response = httpClient.newCall(request).execute()) {
       if (!response.isSuccessful())
        throw new IOException("AlarmService not responding" + response);

       String resBody = response.body().string();
       alarms = new JSONArray(resBody);
     }

                return alarms.toString();
        }

        @Override
```

```
		public boolean addFAListener(FAListener faListener) {
			if(listeners.add(faListener))
				return true;
			else
				return false;
		}

		@Override
		public boolean removeFAListener(FAListener faListener) {
			if(listeners.remove(faListener))
				return true;
			else
				return false;
		}

		@Override
		public boolean addLocation(String f, String r, String lid) throws 
	IOException,RemoteException {
			String json = new StringBuilder()
			.append("{"
				+ "\"lid\":\""+lid+"\","
				+ "\"floorNo\":\""+f+"\","
				+ "\"roomNo\":\""+r+"\""
				+ "}").toString();

			RequestBody requestBody = RequestBody.create (
				MediaType.parse("application/json; charset=UTF-8"), json
			);

			Request request = new Request.Builder()

		.url("http://localhost:8080/FireAlarmRest/rest/LocationService/addLoca
	tion")
			.post(requestBody)
```

```
                    .build();

                    try (Response response = httpClient.newCall(request).execute()) {
                              int code = response.code();

                              if(code == 201)
                                        return true;
                              else
                                        return false;
                    }
          }

          @Override
          public String getLocations() throws IOException, JSONException {
                    JSONArray locations = null;

                    Request request = new Request.Builder()
      .url("http://localhost:8080/FireAlarmRest/rest/LocationService/getLocation
s")
      .build();

      try (Response response = httpClient.newCall(request).execute()) {
         if (!response.isSuccessful())
          throw new IOException("LocationService not responding" + response);

         String resBody = response.body().string();
         locations = new JSONArray(resBody);
      }

                    return locations.toString();
          }

          @Override
          public String getNewAlarmID() throws IOException, JSONException {
```

```
                    JSONObject newAid = null;

                    Request request = new Request.Builder()
        .url("http://localhost:8080/FireAlarmRest/rest/AlarmService/getNewAlarm
Id")
        .build();

        try (Response response = httpClient.newCall(request).execute()) {
            if (!response.isSuccessful())
             throw new IOException("AlarmService not responding" + response);

            String resBody = response.body().string();
            newAid = new JSONObject(resBody);
        }

                    return newAid.toString();
             }

             @Override
             public boolean addAlarm(String aid, String handler, String lid, int smoke,
int co2, int activeState, int workingState) throws IOException, RemoteException {
                    String json = new StringBuilder()
                    .append("{"
                            + "\"aid\":\""+aid+"\","
                            + "\"email\":\""+handler+"\","
                            + "\"lid\":\""+lid+"\","
                            + "\"smokeLevel\":\""+smoke+"\","
                            + "\"co2Level\":\""+co2+"\","
                            + "\"isActive\":\""+activeState+"\","
                            + "\"isWorking\":\""+workingState+"\""
                            + "}").toString();

                    RequestBody requestBody = RequestBody.create (
                            MediaType.parse("application/json; charset=UTF-8"), json
                    );

                    Request request = new Request.Builder()

          .url("http://localhost:8080/FireAlarmRest/rest/AlarmService/addAlarm"
)
                    .post(requestBody)
                    .build();

                    try (Response response = httpClient.newCall(request).execute()) {
                              int code = response.code();

                              if(code == 201)
                                        return true;
                              else
                                        return false;
                    }
          }

          @Override
          public boolean updateAlarm(String aid, int smoke, int co2, String lid, int
workingState, String handler, int activeState) throws IOException,
RemoteException {
                    String json = new StringBuilder()
                    .append("{"
                              + "\"aid\":\""+aid+"\","
                              + "\"email\":\""+handler+"\","
                              + "\"lid\":\""+lid+"\","
                              + "\"smokeLevel\":\""+smoke+"\","
                              + "\"co2Level\":\""+co2+"\","
                              + "\"isActive\":\""+activeState+"\","
                              + "\"isWorking\":\""+workingState+"\""
                              + "}").toString();
```

```
				RequestBody requestBody = RequestBody.create (
					MediaType.parse("application/json; charset=UTF-8"), json
				);
				
				Request request = new Request.Builder()

			.url("http://localhost:8080/FireAlarmRest/rest/AlarmService/updateAlar
		m")
				.put(requestBody)
				.build();
				
				try (Response response = httpClient.newCall(request).execute()) {
					int code = response.code();
					
					if(code == 201)
						return true;
					else
						return false;
				}
			}
			
			public void run() {
				for(;;) {
					try {
						Thread.sleep(15000);
					}
					catch (InterruptedException ie) {
						System.out.println(ie);
					}
					
					try {
						getAlarms();
					} catch (IOException | JSONException e1) {
						e1.printStackTrace();
```

```
                    }

                    try {
                              notifyOthers();
                    } catch (IOException | JSONException e2) {
                              e2.printStackTrace();
                    }
          }
}

public void notifyOthers() throws IOException, JSONException {
          for(FAListener listener: listeners) {
                    try {
                              listener.alarmsChanged(alarms.toString());
                    }
                    catch(RemoteException re) {
                              System.out.println(re);
                    }
          }
}
}
```

```
********************************************************************************
****************

* FASensorServer
********************************************************************************
****************
```

```
package com;

import java.net.MalformedURLException;
import java.rmi.Naming;
import java.rmi.RemoteException;

public class FASensorServer {
	private static FASensorServer faSensorServerInstance = null;
	private FASensorServer() {}

	public static FASensorServer getInstance() {
		if(faSensorServerInstance == null) {
			synchronized (FASensorServer.class) {
				faSensorServerInstance = new FASensorServer();
			}
		}

		return faSensorServerInstance;
	}

	public static void main(String[] args) throws RemoteException {
//	public void startServer() throws RemoteException {
		System.setProperty("java.security.policy", "file:allowall.policy");

		FASensor faSensor = (FASensor) new FASensorImpl();

		try {
			Naming.rebind("rmi://localhost:5500/faSensor", faSensor);

			Thread thread = new Thread((Runnable) faSensor);
			thread.start();

			System.out.println("FireAlarmSensorServer ONLINE...");
		}
		catch (RemoteException | MalformedURLException e) {
			System.out.println("FireAlarmSensorServer Failed");
			System.out.println(e);
		}
	//}
	}
}
```

ASSIGNMENT 2

```
*************************************************************************************
****************

Simulator

* Simulator
*************************************************************************************
****************
package com;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import java.util.Scanner;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request;
import okhttp3.RequestBody;
import okhttp3.Response;

public class Simulator implements Runnable {
	private static OkHttpClient httpClient = new OkHttpClient();
	volatile static List<Alarm> alarmslist = new ArrayList<>();
	static Scanner sc = new Scanner(System.in);

	public static void main(String[] args) {
		Simulator sm = new Simulator();
```

```
				System.out.println("FireAlarm SIMULATION----------------");
				System.out.println("Press y and enter to start----------");
				String choice = sc.nextLine();

				if(choice.equalsIgnoreCase("y")) {
					Thread thread = new Thread(sm);
					thread.start();
				}
			}

			@Override
			public void run() {
				for(;;) {
					try {
						System.out.println("Loading alarms...");
						Thread.sleep(5000);
						List<Alarm> list = new ArrayList<>();
						try {
							list.addAll(setAlarmsList());

							System.out.println("Setting values...");
							Thread.sleep(5000);

							int smokeRandom = getRandomInteger(10);
							int co2Random = getRandomInteger(10);
							int index = 
		getRandomInteger(alarmslist.size());

							Alarm a = alarmslist.get(index);

							System.out.println("Random Smoke value = 
		"+smokeRandom);
							System.out.println("Random CO2 value   = 
		"+co2Random);
```

```
					System.out.println("Selected alarm	=
"+a.getAid());

					if(smokeRandom > 5 || co2Random > 5)
a.setIsActive(1);
					else a.setIsActive(0);

					Thread.sleep(5000);

					a.setSmokeLevel(smokeRandom);
					a.setCo2Level(co2Random);

					Thread.sleep(5000);

					System.out.println("Saving the data...");
					updateAlarm(a);
					sendEmail(a);

					Thread.sleep(5000);

					System.out.println("\nPlease Wait for
next...\n");
				}
				catch (JSONException | IOException e) {
					e.printStackTrace();
				}
				Thread.sleep(5000);
			}
			catch (InterruptedException e) {
				e.printStackTrace();
			}
		}
	}
```

```
		private void sendEmail(Alarm a) {
			if(a.getSmokeLevel() > 5) {
				System.out.println("\n------------------------------------------------
------\n");
				System.out.println("Notification by FireAlarm-------------------
-----------\n");
				System.out.println("Fire Alert due to smoke levels in floor of
"+a.getLid().substring(0, 5)+" and room "+a.getLid().substring(5, 10)+". Please go
to a safer area.");
				System.out.println("---------------------------------------------------
---\n");
				System.out.println("Sending email to "+a.getEmail());

			}
			if(a.getCo2Level() > 5) {
				System.out.println("\n------------------------------------------------
------\n");
				System.out.println("Notification by FireAlarm-------------------
-----------\n");
				System.out.println("High CO2 levels in floor of
"+a.getLid().substring(0, 5)+" and room "+a.getLid().substring(5, 10)+". Please go
to a safer area.");
				System.out.println("---------------------------------------------------
---\n");
				System.out.println("Sending email to "+a.getEmail());
			}
		}

		public static int getRandomInteger(int max) {
			int i = new Random().nextInt(max);
			if(i != 0)
				return i;
			else
				return getRandomInteger(max);
```

```
        }

        public static JSONArray getAlarms() throws IOException, JSONException {
                JSONArray alarms = null;
                Request request = new Request.Builder()
      .url("http://localhost:8080/FireAlarmRest/rest/AlarmService/getAlarms")
      .build();

      try (Response response = httpClient.newCall(request).execute()) {
         if (!response.isSuccessful())
          throw new IOException("AlarmService not responding" + response);

         String resBody = response.body().string();
         alarms = new JSONArray(resBody);
      }

      return alarms;
        }

        public static List<Alarm> setAlarmsList() throws JSONException,
IOException {
                if(!alarmslist.isEmpty())
                        alarmslist.clear();

                JSONArray alarmsJsArr = getAlarms();

                for (int i=0;i<alarmsJsArr.length();i++) {
                        JSONObject o = alarmsJsArr.getJSONObject(i);
                        Alarm a = new Alarm();

                        a.setAid(o.getString("aid"));
                        a.setCo2Level(o.getInt("co2Level"));
                        a.setEmail(o.getString("email"));
                        a.setIsActive(o.getInt("isActive"));
                a.setIsWorking(o.getInt("isWorking"));
                a.setLid(o.getString("lid"));
                a.setSmokeLevel(o.getInt("smokeLevel"));

                if(a.getIsWorking() == 1)
                    alarmslist.add(a);
            }

            return alarmslist;
        }

        public boolean updateAlarm(Alarm a) throws IOException {
            String json = new StringBuilder()
            .append("{"
                + "\"aid\":\""+a.getAid()+"\","
                + "\"email\":\""+a.getEmail()+"\","
                + "\"lid\":\""+a.getLid()+"\","
                + "\"smokeLevel\":\""+a.getSmokeLevel()+"\","
                + "\"co2Level\":\""+a.getCo2Level()+"\","
                + "\"isActive\":\""+a.getIsActive()+"\","
                + "\"isWorking\":\""+a.getIsWorking()+"\""
                + "}").toString();

            RequestBody requestBody = RequestBody.create (
                MediaType.parse("application/json; charset=UTF-8"), json
            );

            Request request = new Request.Builder()

        .url("http://localhost:8080/FireAlarmRest/rest/AlarmService/updateAlar
    m")
            .put(requestBody)
            .build();
```

```
                    try (Response response = httpClient.newCall(request).execute()) {
                              int code = response.code();

                              if(code == 201)
                                        return true;
                              else
                                        return false;
                    }
          }
}
```

****

Web

* main.js
****

```
const table = document.querySelector('#table');
const lastupdated = document.querySelector('#lastupdated');
lastupdated.innerHTML = new Date().toUTCString();

const getAlarmsList = function() {
          const tableBody = table.querySelector('#tableBody');
          const req = new XMLHttpRequest();
          req.open('GET',
'http://localhost:8080/FireAlarmRest/rest/AlarmService/getAlarms', true);
          req.onreadystatechange = function() {
                    if(req.readyState == 4 && req.status == 200) {
                              var alarmsArray = req.responseText;
                              let jso = JSON.parse(alarmsArray);

                              var newBody = document.createElement('tbody');
                              newBody.setAttribute('id','tableBody');
```

```
for (var i = 0; i < jso.length; i++) {
        let obj = JSON.parse(JSON.stringify(jso[i]));

        var a = document.createElement('tr');
        a.setAttribute('class','row');

        var b1 = document.createElement('td');
        b1.setAttribute('class','rowCell');
        b1.append(obj.aid.toString());

        var b2A = document.createElement('td');
        b2A.setAttribute('class','rowCell');
        b2A.append(obj.lid.toString().substring(0,5));

        var b2B = document.createElement('td');
        b2B.setAttribute('class','rowCell');
        b2B.append(obj.lid.toString().substring(5,10));

        var b3 = document.createElement('td');
        if(parseInt(obj.smokeLevel.toString()) > 5)
                b3.setAttribute('class','rowCell danger');
        else
                b3.setAttribute('class','rowCell');
        b3.append(obj.smokeLevel.toString());

        var b4 = document.createElement('td');
        if(parseInt(obj.co2Level.toString()) > 5)
                b4.setAttribute('class','rowCell danger');
        else
                b4.setAttribute('class','rowCell');
        b4.append(obj.co2Level.toString());

        var b5 = document.createElement('td');
```

```
                    b5.setAttribute('class','rowCell');
                    b5.append(obj.isActive.toString());

                    var b6 = document.createElement('td');
                    if(parseInt(obj.isWorking.toString()) == 0)
                        b6.setAttribute('class','rowCell broke');
                    else
                        b6.setAttribute('class','rowCell');
                    b6.append(obj.isWorking.toString());

                    a.append(b1,b2A,b2B,b3,b4,b5,b6);

                    newBody.appendChild(a);
                }

                clearTable();
                tableBody.parentNode.replaceChild(newBody, tableBody);
            }
            else
                return null;
        };

        req.send();
}

const clearTable = function() {
        tableBody.innerHTML = '';
}

const repeat = function() {
   setInterval(function() {
        getAlarmsList();
        lastupdated.innerHTML = new Date().toUTCString();
        }, 30000);
```

```
};

repeat();
*********************************************************************************
****************

* Sensor.jsp
*********************************************************************************
****************
<%@page import="org.apache.jasper.tagplugins.jstl.core.ForEach"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1" import="java.util.*, com.sensor.*"%>
<!DOCTYPE html>
<html>

<head>
<meta charset="ISO-8859-1">
<title>Fire Alarm Monitoring System</title>
<link type="text/css" rel="stylesheet" href="css/style.css">
</head>

<%
	List<Alarm> theAlarms = (List<Alarm>) request.getAttribute("alarm_list");
%>

<body>
	<div id="wrapper">
		<div id="header">
			<h1>Fire Alarm Monitoring System</h1>
		</div>
	</div>
	<div id="container">
		<div id ="legend">
			<div class="legendIn">
```

```
                    <div class="circle danger"></div>
                    <small> Smoke/CO2 is high levels</small>
                </div>
                <div class="legendIn">
                    <div class="circle broke"></div>
                    <small> Alarm broke </small>
                </div>
                <div class="legendIn small">
                    <small>Last updated : </small>
                    <small id="lastupdated"></small>
                </div>
            </div>
            <div id="content">
                <table id="table">
                    <tr>
                        <th>Alarm ID</th>
                        <th>Floor No</th>
                        <th>Room No</th>
                        <th>Smoke Level</th>
                        <th>CO2 Level</th>
                        <th>Active state</th>
                        <th>Working state</th>
                    </tr>
                    <!-- -->
                    <tbody id="tableBody">
                    <%
                        for(Alarm a : theAlarms) {
                                                    if(a.getSmokeLevel()>5)
                                                        out.print("<tr
    class=\"row danger\">");
                                                    else if(a.getIsWorking()
    == 0)
                                                        out.print("<tr
    class=\"row danger\">");
```

```
                                                            else
                                                                out.print("<tr
class=\"row danger\">");

                                                                out.print("<td
class=\"rowCell\">"+a.getAid()+"</td>");
                                                                out.print("<td
class=\"rowCell\">"+a.getLid().substring(0, 5)+"</td>");
                                                                out.print("<td
class=\"rowCell\">"+a.getLid().substring(5, 10)+"</td>");
                                                                out.print("<td
class=\"rowCell\">"+a.getSmokeLevel()+"</td>");
                                                                out.print("<td
class=\"rowCell\">"+a.getCo2Level()+"</td>");
                                                                out.print("<td
class=\"rowCell\">"+a.getIsActive()+"</td>");
                                                                out.print("<td
class=\"rowCell\">"+a.getIsWorking()+"</td>");
                                                            out.print("<tr/>");
                                                        }
                        %>
                        </tbody>
                    </table>
                </div>
            </div>

            <script type="text/javascript" src="js/main.js"></script>
</body>

</html>
```

****

* AlarmDb (dummy)

```
**********************************************************************************************
package com.sensor;

import java.util.ArrayList;
import java.util.List;

public class AlarmDb {
	public static List<Alarm> getAlarms(){
		List<Alarm> alarms = new ArrayList<>();

		alarms.add(new
Alarm("A000000001","a@a.lk","F0001R0001",3,2,0,1));
		alarms.add(new
Alarm("A000000002","b@a.lk","F0001R0002",7,6,1,1));
		alarms.add(new
Alarm("A000000003","c@a.lk","F0001R0003",8,7,1,1));
		alarms.add(new
Alarm("A000000004","d@a.lk","F0001R0004",4,9,0,1));

		return alarms;
	}
}
**********************************************************************************************

* SensorControllerServlet
**********************************************************************************************
package com.sensor;

import java.io.IOException;
import java.util.List;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/SensorControllerServlet")
public class SensorControllerServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

   public SensorControllerServlet() {
      super();
   }

   protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		List<Alarm> alarms = AlarmDb.getAlarms();
		request.setAttribute("alarm_list", alarms);
		RequestDispatcher dispatcher =
request.getRequestDispatcher("Sensor.jsp");
		dispatcher.forward(request, response);
	}

	protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		doGet(request, response);
	}
}
```

****