



Sri Lanka Institute of Information Technology

**Distributed System
(SE3020)**

Assignment 02 – REST API

**Fire Alarm Monitoring System
Assignment Report**

Group Details:

IT18059878	Abdurrahmaan A.N
IT18037920	Thenuwara T.B.K.P
IT18019278	Hareeni Vimalaraj
IT18138382	S.D Fernando

Table of Contents

1.Introduction	3
2.High Level Architectural Diagram	4
3.System Workflow Diagram	5
4.System Workflow Scenario Execution	6
5.Appendix (Source Codes & Binaries).....	12
5.1 Web Client	13
5.2 REST API.....	18
5.3 RMI Server/Client	34

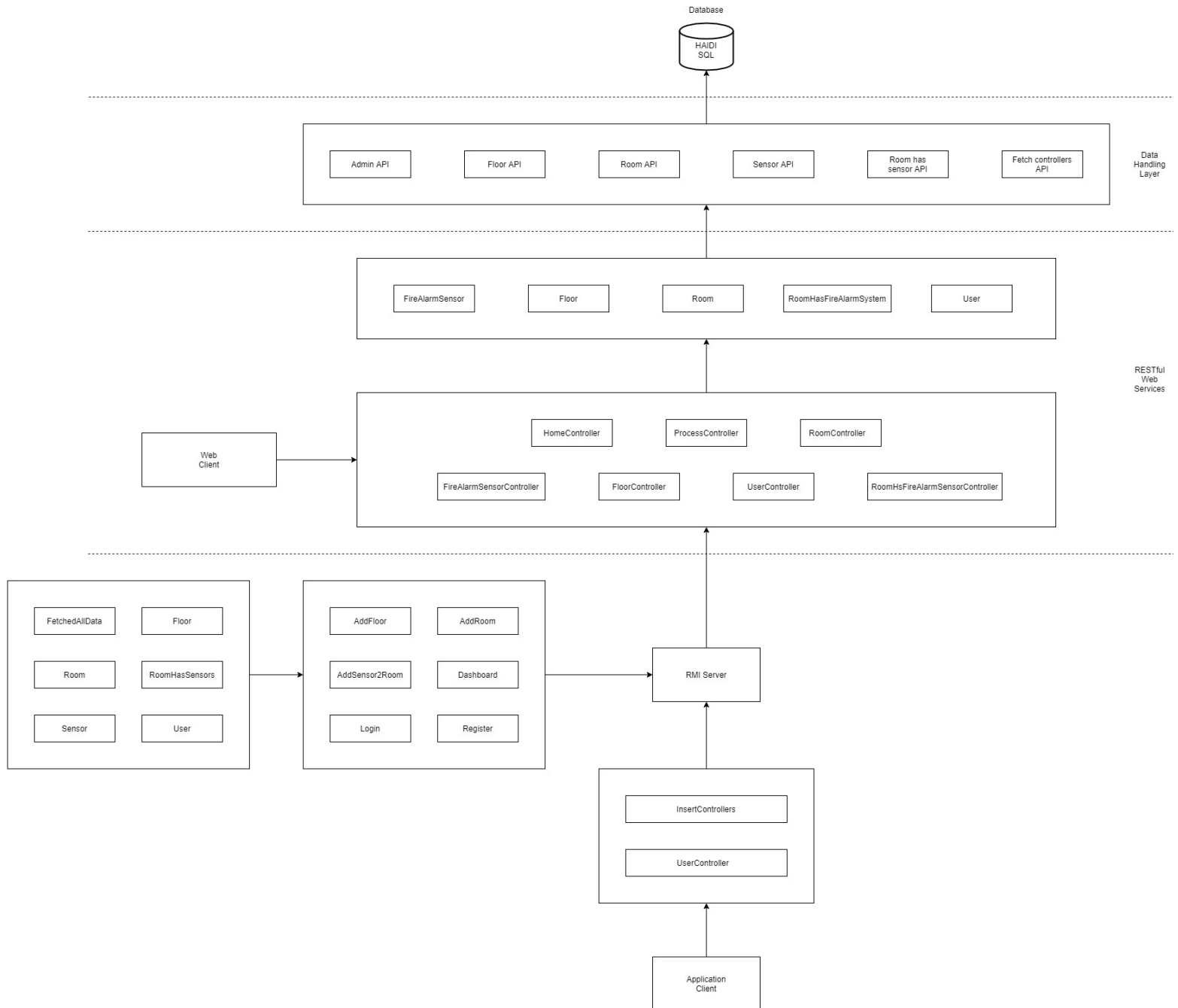
1.Introduction

The fire alarm monitoring system is implemented using the technologies such as Laravel PHP with REST architecture and JAVA for the RMI Client/Server and MySQL as the database.

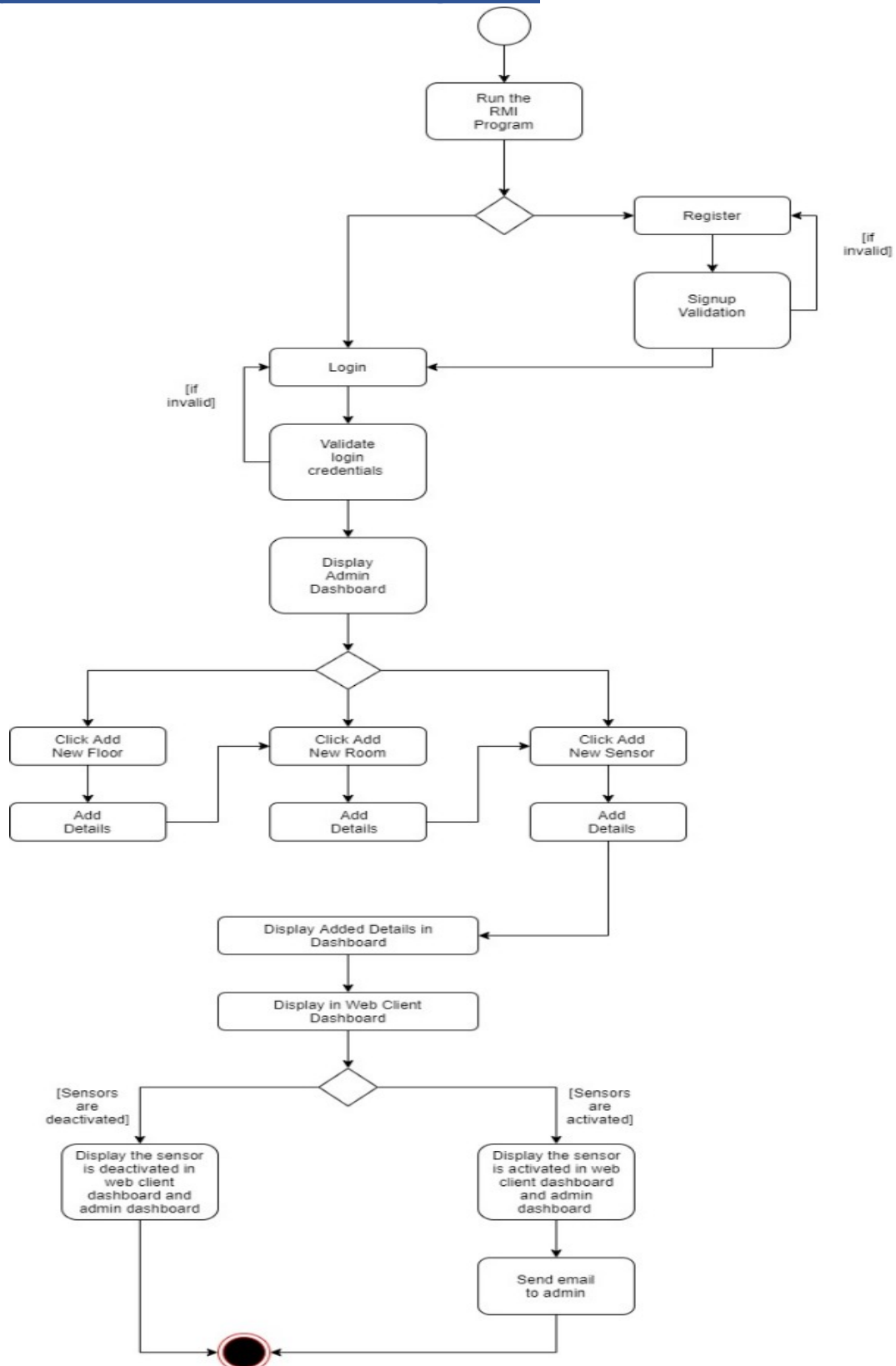
This is a system where an admin can monetarize the fire alarm system of a building. When the smoke level or sensor level goes more than 5 then the administrator will be notified that fire alarm sensor has been activated. He will be notified in his administration dashboard and he will even receive an email alert notification for his email address notifying which Sensor has been activated. He can even look it from his Administration dashboard which sensor level has gone high with numbers of level increased. We have explained this scenario in this report with screenshots.

The System User should register into the system first using his valid email and phone number and other information such as username and passwords. And then he has to login to the system using his Username and Password and he will lead into his Dashboard. In that dashboard he can see all the registered sensor details and the sensor levels with the floor and room numbers including sensor details. The user even can add floor, add room and assign sensors through the dashboard.

2. High Level Architectural Diagram

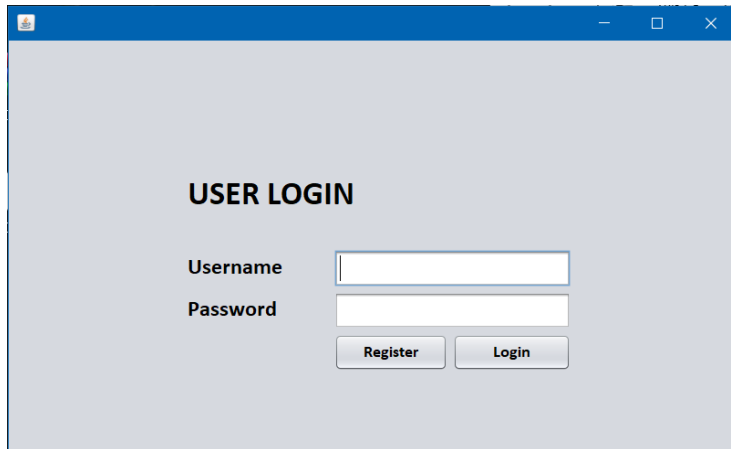


3. System Workflow Diagram



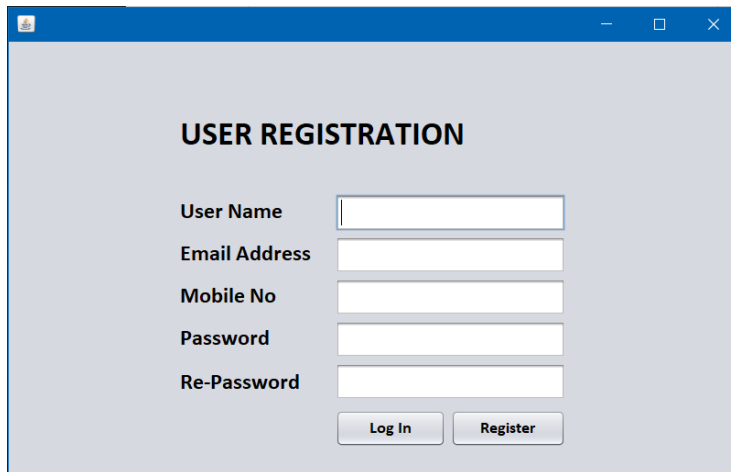
4. System Workflow Scenario Execution

4.1 This will be the first user interface you will get when you run the RMI Program. The System User should register him/her self to login in to the system. They should press **the Register button**.



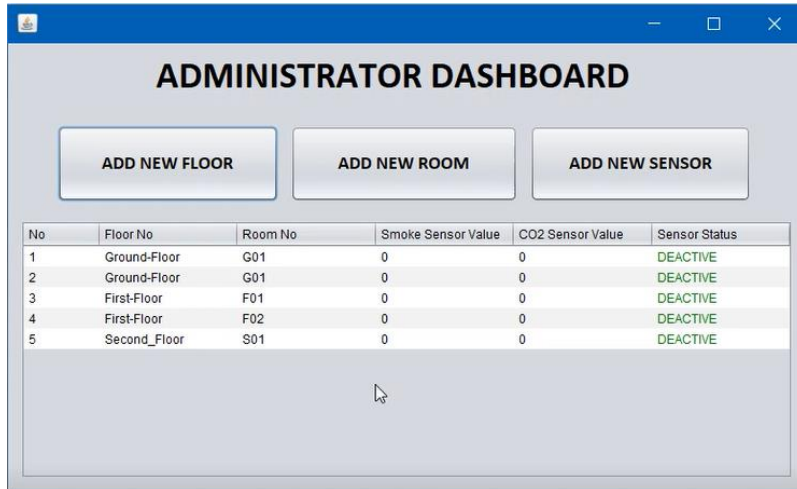
A screenshot of a web application window titled "USER LOGIN". The window has a blue header bar with standard window controls (minimize, maximize, close). The main content area is light gray. In the center, the text "USER LOGIN" is displayed in bold. Below this, there are two input fields: "Username" and "Password". To the right of each label is a white text box with a blue border. Below the input fields are two buttons: "Register" and "Login".

4.2 Once you click the **Register button** you will get a register form to register yourself into the system. you should fill the following form with correct credentials. Or else you get an error message.



A screenshot of a web application window titled "USER REGISTRATION". The window has a blue header bar with standard window controls (minimize, maximize, close). The main content area is light gray. In the center, the text "USER REGISTRATION" is displayed in bold. Below this, there are five input fields: "User Name", "Email Address", "Mobile No", "Password", and "Re-Password". To the right of each label is a white text box with a blue border. Below the input fields are two buttons: "Log In" and "Register".

4.3 Once you login to the system with your correct username and password, it will take you to the **Admin Dashboard**.



No	Floor No	Room No	Smoke Sensor Value	CO2 Sensor Value	Sensor Status
1	Ground-Floor	G01	0	0	DEACTIVE
2	Ground-Floor	G01	0	0	DEACTIVE
3	First-Floor	F01	0	0	DEACTIVE
4	First-Floor	F02	0	0	DEACTIVE
5	Second_Floor	S01	0	0	DEACTIVE

4.4 You can Add new Floor by clicking the **Add New Floor Button** and insert the floor number and save by pressing the save button.



ADD NEW FLOOR

Floor No

4.5 You can Add new Room by clicking the **Add New Room Button** and insert the Room number and save by pressing the save button.

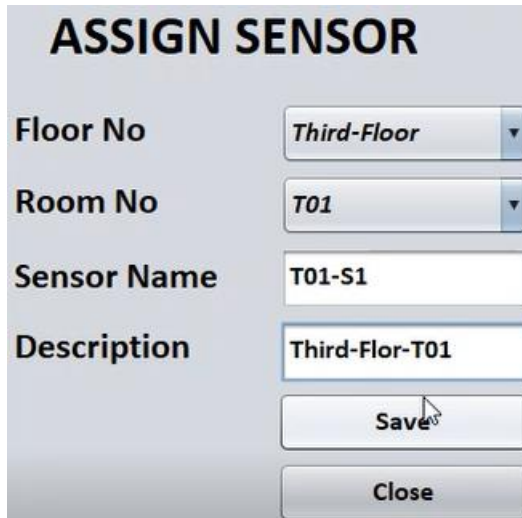


ADD NEW ROOM

Room No

Floor No

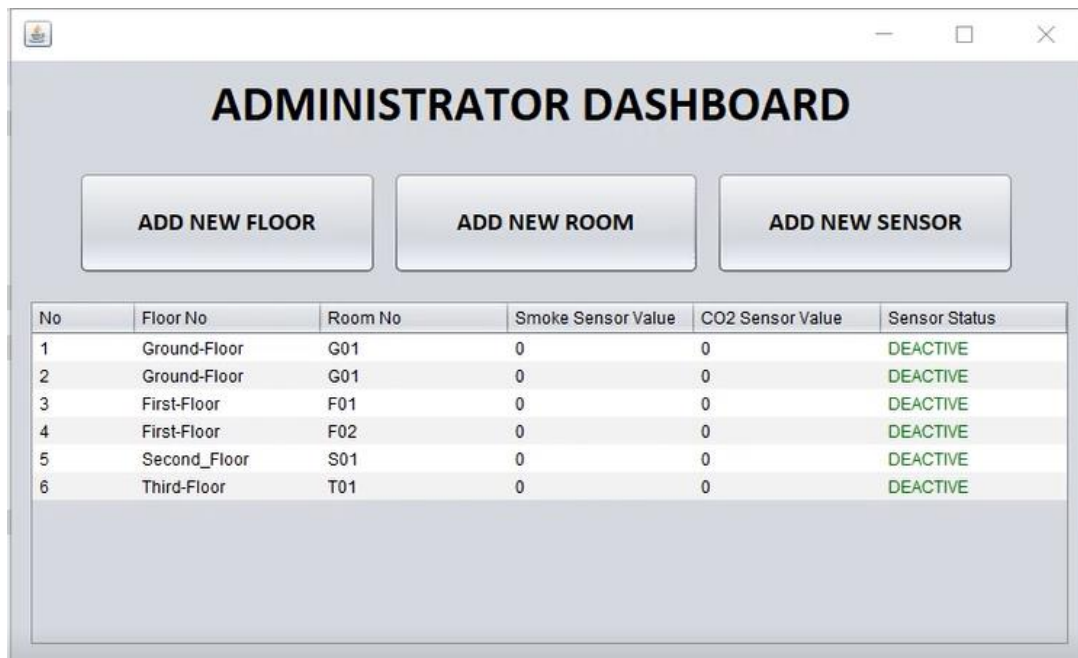
4.5 You can assign new Sensor by clicking the **Add New Sensor Button** and insert the Sensor details and save by pressing the save button.



The 'ASSIGN SENSOR' dialog box contains the following fields and buttons:

- Floor No:** A dropdown menu with 'Third-Floor' selected.
- Room No:** A dropdown menu with 'T01' selected.
- Sensor Name:** A text input field containing 'T01-S1'.
- Description:** A text input field containing 'Third-Flor-T01'.
- Buttons:** 'Save' and 'Close' buttons at the bottom.

4.6 And now you can see the following details have been added to the list.(Row Number 6)



The 'ADMINISTRATOR DASHBOARD' window features three buttons at the top: 'ADD NEW FLOOR', 'ADD NEW ROOM', and 'ADD NEW SENSOR'. Below these is a table with 6 rows of sensor data.

No	Floor No	Room No	Smoke Sensor Value	CO2 Sensor Value	Sensor Status
1	Ground-Floor	G01	0	0	DEACTIVE
2	Ground-Floor	G01	0	0	DEACTIVE
3	First-Floor	F01	0	0	DEACTIVE
4	First-Floor	F02	0	0	DEACTIVE
5	Second_Floor	S01	0	0	DEACTIVE
6	Third-Floor	T01	0	0	DEACTIVE

4.6 And this is the web client dashboard before adding the details

Fire Alarm Sensors

Fire alarm sensor status

#	Floor	Room	Sensor	Status
1	Ground-Floor	G01	1 G01-S1	●
2	Ground-Floor	G01	2 G02-S2	●
3	First-Floor	F01	3 F01-S1	●
4	First-Floor	F02	4 F02-S2	●
5	Second_Floor	S01	9 S01-S1	●

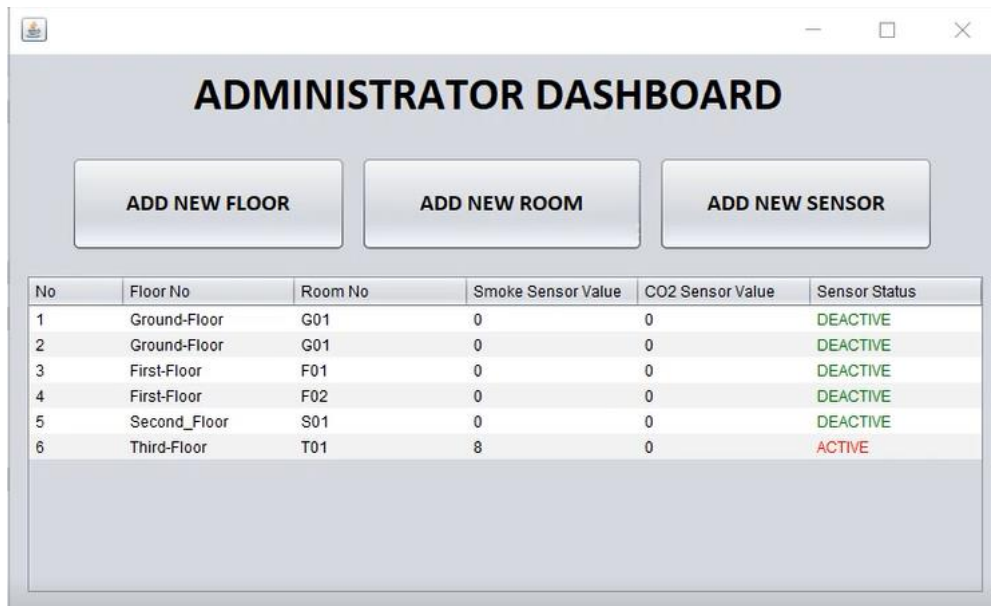
4.6 And this is the web client dashboard after adding the details

Fire Alarm Sensors

Fire alarm sensor status

#	Floor	Room	Sensor	Status
1	Ground-Floor	G01	1 G01-S1	●
2	Ground-Floor	G01	2 G02-S2	●
3	First-Floor	F01	3 F01-S1	●
4	First-Floor	F02	4 F02-S2	●
5	Second_Floor	S01	9 S01-S1	●
6	Third-Floor	T01	10 T01-S1	●

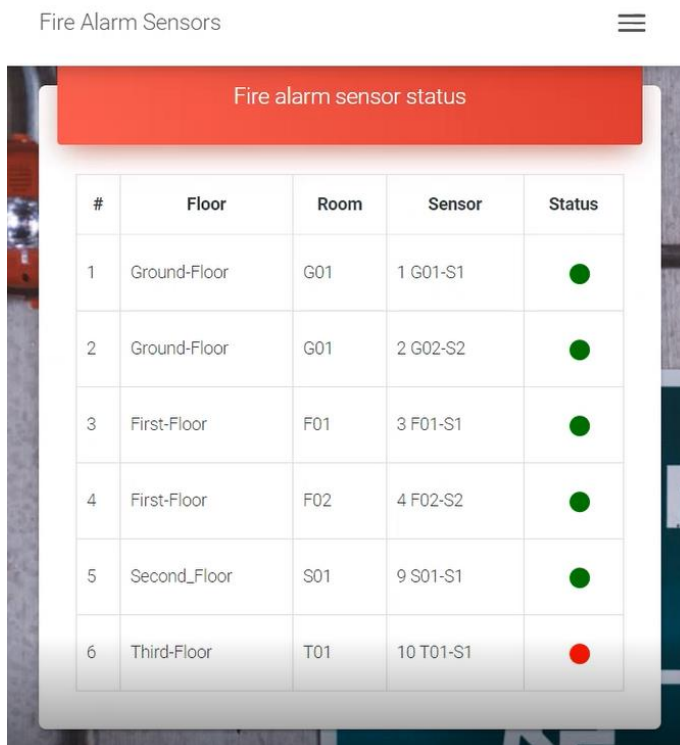
4.7 When the Smoke or Sensor value goes more than 5 the sensor gets Active and it will show in the Admin Dashboard (15secs)



The screenshot shows a web application window titled "ADMINISTRATOR DASHBOARD". At the top, there are three buttons: "ADD NEW FLOOR", "ADD NEW ROOM", and "ADD NEW SENSOR". Below these buttons is a table with the following data:

No	Floor No	Room No	Smoke Sensor Value	CO2 Sensor Value	Sensor Status
1	Ground-Floor	G01	0	0	DEACTIVE
2	Ground-Floor	G01	0	0	DEACTIVE
3	First-Floor	F01	0	0	DEACTIVE
4	First-Floor	F02	0	0	DEACTIVE
5	Second_Floor	S01	0	0	DEACTIVE
6	Third-Floor	T01	8	0	ACTIVE

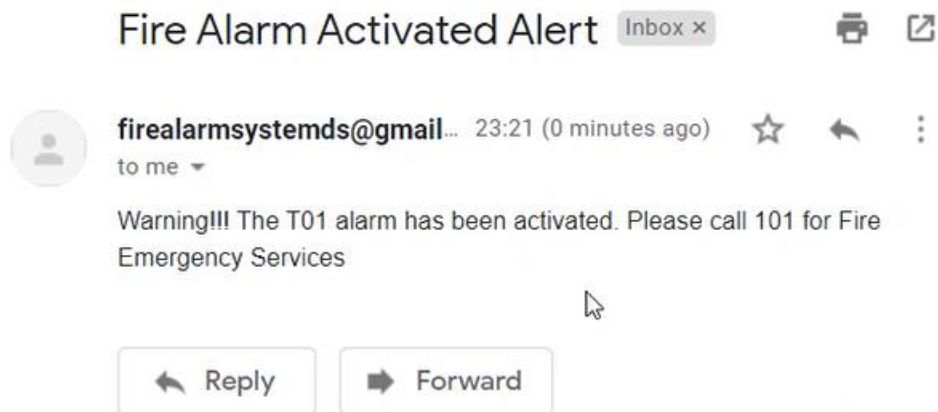
4.8 It will also display in the web client dashboard. (10secs)



The screenshot shows a web application window titled "Fire Alarm Sensors". Below the title bar, there is a red header with the text "Fire alarm sensor status". Below this header is a table with the following data:

#	Floor	Room	Sensor	Status
1	Ground-Floor	G01	1 G01-S1	●
2	Ground-Floor	G01	2 G02-S2	●
3	First-Floor	F01	3 F01-S1	●
4	First-Floor	F02	4 F02-S2	●
5	Second_Floor	S01	9 S01-S1	●
6	Third-Floor	T01	10 T01-S1	●

4.8 And the Admin will also get an email when the sensor gets active.



5.Appendix

Source Codes & Binaries

5.1 Web Client

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-
scalable=0" name="viewport" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
  <title>
    Fire Alarm Sensors
  </title>
  <!-- Fonts and icons -->
  <link rel="stylesheet" type="text/css" href="https://fonts.googleapis.com/css?family=Roboto:300,400,50
0,700|RobotoSlab:400,700|Material+Icons" />
  <link rel="stylesheet" href="assets/css/font-awesome.min.css" />
  <link rel="stylesheet" href="assets/css/material-kit.min40a0.css?v=2.0.2">
  <!-- Documentation extras -->
  <!-- CSS Just for demo purpose, don't include it in your project -->
  <link href="assets/assets-for-demo/demo.css" rel="stylesheet" />
  <link href="assets/assets-for-demo/vertical-nav.css" rel="stylesheet" />
  <style>
    .cat-discount-wrapper {
      position: absolute;
      top: 5px;
      width: 60px;
      height: 51px;
      margin-left: -6px;
      padding: 8px 5px;
      z-index: 500;
    }
    .brand-discount-wrapper {
      position: absolute;
      top: 5px;
      width: 60px;
      height: 51px;
      margin-left: -11px;
      padding: 5px;
      background: url(assets/img/discountPic/discount-wrapper2.png) no-repeat;
      z-index: 500;
    }
    .brand-discount-wrapper .pers {
      color: #ffffff;
      float: left;
```

```

        font-size: 1.1em;
        font-weight: 600;
        line-height: 1.2em;
        margin: 0;
        padding: 0;
        width: 100%;
    }
    .div2 {
        background-color: #ffffff;
        border-radius: 5px;
        border: 1px solid #e0e0e0;
    }
    .margintop0p {
        margin-top: 0px;
    }
    .margintop5p {
        margin-top: 5%;
    }
    .margintop10p {
        margin-top: 10%;
    }
    .card_border {
        border: 5px solid #1565c0;
    }
    .paragraph {
        font-weight: 400;
    }
    .dot {
        height: 25px;
        width: 25px;
        background-color: #bbb;
        border-radius: 50%;
        display: inline-block;
    }
</style>
</head>
<body class="blog-posts sidebar-collapse" style="background-
image: url('https://images.pexels.com/photos/103592/pexels-photo-
103592.jpeg?auto=compress&cs=tinysrgb&dpr=2&h=650&w=940')">
    <nav class="navbar fixed-top navbar-expand-lg " id="sectionsNav">
        <div class="container">
            <div class="navbar-translate">
                <a class="navbar-brand" href=" ../index.html">
                    Fire Alarm Sensors
                </a>
            </div>
        </div>
    </nav>

```

```

        <button class="navbar-toggler" type="button" data-toggle="collapse" aria-
expanded="false" aria-label="Toggle navigation">
            <span class="sr-only">Toggle navigation</span>
            <span class="navbar-toggler-icon"></span>
            <span class="navbar-toggler-icon"></span>
            <span class="navbar-toggler-icon"></span>
        </button>
    </div>
    <div class="collapse navbar-collapse">
        <ul class="navbar-nav ml-auto">
        </ul>
    </div>
</nav>
<div class="page-header" id="databale">
    <div class="container">
        <div class="card">
            <div class="card-header card-header-danger text-center">
                <h4>Fire alarm sensor status</h4>
            </div>
            <div class="card-body pt-4">
                <table class="table table-bordered text-dark" style="overflow: scroll; max-
height:100px;">
                    <tr class="text-center">
                        <th>#</th>
                        <th>Floor</th>
                        <th>Room</th>
                        <th>Sensor</th>
                        <th>Status</th>
                    </tr>
                    <tbody id="databody">
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
<script src="../../assets/js/core/jquery.min.js "></script>
<script src="../../assets/js/core/popper.min.js "></script>
<script src="../../assets/js/bootstrap-material-design.min.js "></script>
<!-- Google Maps Plugin -->
<script type="text/javascript" src="https://maps.googleapis.com/maps/api/js?key=AIzaSyB2Yno10-
YTnLjjn_Vtk0V8cdcY5lC4plU "></script>
<!-- Plugin for Date Time Picker and Full Calendar Plugin -->
<script src="../../assets/js/plugins/moment.min.js "></script>

```

```

    <!-- Plugin for the Datpicker, full documentation here: https://github.com/Eonasdan/bootstrap-
    datetimerpicker -->
    <script src="../../assets/js/plugins/bootstrap-datetimepicker.min.js "></script>
    <!-- Plugin for the Sliders, full documentation here: http://refreshless.com/nouislider/ -->
    <script src="../../assets/js/plugins/nouislider.min.js "></script>
    <!-- Plugin for Select, full documentation here: http://silviomoreto.github.io/bootstrap-select -->
    <script src="../../assets/js/plugins/bootstrap-selectpicker.js "></script>
    <!-- Plugin for Tags, full documentation here: http://xoxco.com/projects/code/tagsinput/ -->
    <script src="../../assets/js/plugins/bootstrap-tagsinput.js "></script>
    <!--
-   Plugin for Fileupload, full documentation here: http://www.jasny.net/bootstrap/javascript/#fileinput
-->
    <script src="../../assets/js/plugins/jasny-bootstrap.min.js "></script>
    <!-- Plugin for Small Gallery in Product Page -->
    <script src="../../assets/js/plugins/jquery.flexisel.js "></script>
    <!-- Plugins for presentation and navigation -->
    <script src="../../assets/assets-for-demo/js/modernizr.js "></script>
    <script src="../../assets/assets-for-demo/js/vertical-nav.js "></script>
    <!-- Material Kit Core initialisations of plugins and Bootstrap Material Design Library -->
    <script src="../../assets/js/material-kit.min40a0.js?v=2.0.2 "></script>
    <!-- Fixed Sidebar Nav - js With initialisations For Demo Purpose, Don't Include it in your project --
>
    <script src="../../assets/assets-for-demo/js/material-kit-demo.js "></script>
    <script src="https://code.jquery.com/jquery-3.5.0.min.js" integrity="sha256-
xNzN2a4ltkB44Mc/Jz3pT4iU1cmeR0FkXs4pru/JxaQ=" crossorigin="anonymous"></script>
    <script>
        doprocess();
        function doprocess() {
            $.ajax({
                url: "http://127.0.0.1:8000/api/fetchSensors",
                cache: true,
                success: function(html) {
                    if (html.length > 0) {
                        var resp = JSON.parse(html);
                        $('#databody').html('');
                        resp.forEach(function(oneData) {
                            var tdObj = $('<td></td>');
                            tdObj.attr('class', 'text-center');
                            var dotCircle = $('<span></span>');
                            dotCircle.attr('class', 'dot');
                            dotCircle.attr('style', 'background-color:' + oneData['status']);
                            tdObj.append(dotCircle);
                            $('#databody').append($('<tr></tr>').append('<td>' + oneData['index'] + '</td>' +
                            ').append('<td>' + oneData['floor'] + '</td>').append('<td>' + oneData['room'] + '</td>').append('<td>' +
                            oneData['sensor'] + '</td>').append(tdObj));

```



```
        });  
        setTimeout("doprocess()", 5000);  
    } else {  
        alert('Something wrong. No reponse found');  
    }  
}  
});  
}  
</script>  
</body>  
</html>
```

5.2 REST API

Models

User.php

```
<?php
namespace App;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
class User extends Authenticatable
{
    use Notifiable;
    protected $fillable = [
        'username', 'email', 'password','mobilen','usertype'
    ];
    protected $hidden = [
        'password',
    ];
}
```

FireAlarmSensor.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class FireAlarmSensor extends Model
{
    protected $fillable = [
        'user_id','smoke','co2','name','detail'
    ];
}
```

Floor.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;

class Floor extends Model
{
    protected $fillable = [
        'no', 'user_id',
    ];
}
```

Room.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Room extends Model
{
    protected $fillable = [
        'no', 'user_id', 'floor_id'
    ];
}
```

RoomHasFireAlarmSystem.php

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class RoomHasFireAlarmSystem extends Model
{
    protected $fillable = [
        'room_id', 'fire_alarm_sensor_id'
    ];
}
```

Controllers

UserController.php

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\User;
use Illuminate\Support\Facades\Hash;
class UserController extends Controller
{
    public function adminAuth(Request $request)
    {
        $request->validate([
            'un' => ['required', 'string'],
            'pw' => ['required', 'string'],
        ]);
        $userObj = User::where('username', $request->un)->first();
        $responseJson = [];
        if (!empty($userObj)) {
            if (Hash::check($request->pw, $userObj->password) && $userObj->usertype == "1") {
                $responseJson = [
                    "code" => 1,

```

```

        "msg" => "Welcome",
        "user" => $userObj
    ];
} else {
    $responseJson = [
        "code" => 2,
        "msg" => "Incorrect password or access denied",
    ];
}
} else {
    $responseJson = [
        "code" => 2,
        "msg" => "No username or password found. please recheck"
    ];
}
return json_encode($responseJson);
}
public function registerAdmin(Request $request)
{
    $request->validate([
        'un' => ['required', 'string'],
        'pw' => ['required', 'string'],
        'email' => ['required', 'string'],
        'mno' => ['required', 'string'],
    ]);
    $userObj=User::create([
        'username'=>$request->un,
        'password'=>Hash::make($request->pw),
        'email'=>$request->email,
        'mobileno'=>$request->mno,
        'usertype'=>1,
    ]);
    $responseJson = [
        "code" => 1,
        "msg" => "Welcome",
        "user" => $userObj
    ];
    return json_encode($responseJson);
}
}

```

FireAlarmSensorController.php

```
<?php
namespace App\Http\Controllers;
use App\FireAlarmSensor;
use App\User;
use Illuminate\Http\Request;
class FireAlarmSensorController extends Controller
{
    public function newSensor(Request $request)
    {
        $request->validate([
            'uid' => ['required', 'integer'],
            'param1' => ['required', 'string'],
            'param2' => ['required', 'string'],
            'name' => ['required', 'string'],
            'detail' => ['required', 'string'],
        ]);
        $userObj = User::where('id', $request->uid)->first();
        $responseJson = [];
        if (!empty($userObj)) {
            $reged_sensor = FireAlarmSensor::create([
                "name" => $request->name,
                "detail" => $request->detail,
                "user_id" => $request->uid,
                "smoke" => $request->param1,
                "co2" => $request->param2,
            ]);
            $responseJson = [
                "code" => 1,
                "msg" => "Success",
                "floor" => $reged_sensor
            ];
        } else {
            $responseJson = [
                "code" => 2,
                "msg" => "User invalid"
            ];
        }
        return json_encode($responseJson);
    }
    public function sensorFindById(Request $request)
    {
        $request->validate([
            'id' => ['required', 'integer'],
```

```

]);
$data = FireAlarmSensor::where('id', $request->id)->first();
$responseJson = [];
if (!empty($data)) {
    $responseJson = [
        "code" => 1,
        "msg" => "Success",
        "floor"=>$data
    ];
} else {
    $responseJson = [
        "code" => 2,
        "msg" => "Not Found"
    ];
}
return json_encode($responseJson);
}
public function sensorAllFind(Request $request)
{
    $Obj = FireAlarmSensor::get();
    $responseJson = []
    $responseJson = [
        "code" => 1,
        "msg" => "Success",
        "floors"=>$Obj
    ];
    return json_encode($responseJson);
}
}

```

FloorController.php

```

<?php
namespace App\Http\Controllers;
use App\Floor;
use App\User;
use Illuminate\Http\Request;
class FloorController extends Controller
{
    public function newFloor(Request $request)
    {
        $request->validate([
            'uid' => ['required', 'integer'],
            'no' => ['required', 'string'],
        ]);
    }
}

```

```

$userObj = User::where('id', $request->uid)->first();
$responseJson = [];

if (!empty($userObj)) {
    $reged_floor = Floor::create([
        'user_id' => $request->uid,
        'no' => $request->no
    ]);
    $responseJson = [
        "code" => 1,
        "msg" => "Success",
        "floor" => $reged_floor
    ];
} else {
    $responseJson = [
        "code" => 2,
        "msg" => "User invalid"
    ];
}
return json_encode($responseJson);
}

public function floorFindById(Request $request)
{
    $request->validate([
        'fid' => ['required', 'integer'],
    ]);
    $floorObj = Floor::where('id', $request->fid)->first();
    $responseJson = [];
    if (!empty($floorObj)) {
        $responseJson = [
            "code" => 1,
            "msg" => "Success",
            "floor"=>$floorObj
        ];
    } else {
        $responseJson = [
            "code" => 2,
            "msg" => "Not Found"
        ];
    }
    return json_encode($responseJson);
}

public function floorAllFind(Request $request)
{
    $floorsObj = Floor::get();

```

```

        $responseJson = [];
        $responseJson = [
            "code" => 1,
            "msg" => "Success",
            "floors"=>$floorsObj
        ];
        return json_encode($responseJson);
    }
}

```

RoomController.php

```

<?php
namespace App\Http\Controllers;
use App\Floor;
use App\Room;
use App\User;
use Illuminate\Http\Request;
class RoomController extends Controller
{
    public function newRoom(Request $request)
    {
        $request->validate([
            'uid' => ['required', 'integer'],
            'fid' => ['required', 'integer'],
            'no' => ['required', 'string'],
        ]);
        $checkFloor = false;
        $checkUser = false;
        $floorObj = Floor::where('id', $request->fid)->first();
        $userObj = User::where('id', $request->uid)->first();
        if (!empty($floorObj)) {
            $checkFloor = true;
        }
        if (!empty($userObj)) {
            $checkUser = true;
        }
        $responseJson = [];
        if ($checkFloor && $checkUser) {
            $reged_room = Room::create([
                'user_id' => $request->uid,
                'floor_id' => $request->fid,
                'no' => $request->no
            ]);
            $responseJson = [
                "code" => 1,

```



```

        "msg" => "Success",
        "floor" => $reged_room
    ];
} else {
    $responseJson = [
        "code" => 2,
        "msg" => "Invalid data"
    ];
}
return json_encode($responseJson);
}

public function roomFindById(Request $request)
{
    $request->validate([
        'rid' => ['required', 'integer'],
    ]);
    $roomObj = Room::where('id', $request->rid)->first();
    $responseJson = [];
    if (!empty($roomObj)) {
        $responseJson = [
            "code" => 1,
            "msg" => "Success",
            "room" => $roomObj
        ];
    } else {
        $responseJson = [
            "code" => 2,
            "msg" => "Not Found"
        ];
    }
    return json_encode($responseJson);
}

public function roomAllFind(Request $request)
{
    $roomsObj = Room::get();
    $responseJson = [];
    $responseJson = [
        "code" => 1,
        "msg" => "Success",
        "rooms" => $roomsObj
    ];
    return json_encode($responseJson);
}
}

```

RoomHasFireAlarmSensorController.php

```
<?php
namespace App\Http\Controllers;
use App\FireAlarmSensor;
use App\Floor;
use App\Room;
use App\RoomHasFireAlarmSystem;
use Illuminate\Http\Request;
class RoomHasFireAlarmSensorController extends Controller
{
    public function findRoomHasSensorById(Request $request)
    {
        $request->validate([
            'id' => ['required', 'integer'],
        ]);
        $Obj = RoomHasFireAlarmSystem::where('id', $request->id)->first();
        $responseJson = [];
        if (!empty($Obj)) {
            $responseJson = [
                "code" => 1,
                "msg" => "Success",
                "data" => $Obj
            ];
        } else {
            $responseJson = [
                "code" => 2,
                "msg" => "No data found"
            ];
        }
        return json_encode($responseJson);
    }
    public function findRoomHasSensors(Request $request)
    {
        $Obj = RoomHasFireAlarmSystem::get();
        $responseJson = [];
        if (!empty($Obj)) {
            $responseJson = [
                "code" => 1,
                "msg" => "Success",
                "data" => $Obj
            ];
        } else {
            $responseJson = [
                "code" => 2,
```

```

        "msg" => "No Data"
    ];
}
return json_encode($responseJson);
}
public function saveRoomHasSensor(Request $request)
{
    $request->validate([
        'rid' => ['required', 'integer'],
        'fid' => ['required', 'integer'],
    ]);
    $responseJson = [];
    $checkRoom = false;
    $checkSensor = false;
    $checkRoom = Room::where('id', $request->rid)->first();
    $checkSensor = RoomHasFireAlarmSystem::where('id', $request->fid)->first();
    if (!empty($checkRoom)) {
        $checkFloor = true;
    }
    if (!empty($checkSensor)) {
        $checkUser = true;
    }
    $obj = RoomHasFireAlarmSystem::create([
        'room_id' => $request->rid,
        'fire_alarm_sensor_id' => $request->fid,
    ]);
    $responseJson = [
        "code" => 1,
        "msg" => "Success",
        "floor" => $obj
    ];
    return json_encode($responseJson);
}
public function findAllDataByFireAlarmSensorId(Request $request)
{
    $request->validate([
        'fid' => ['required', 'integer'],
    ]);
    $Obj = RoomHasFireAlarmSystem::where('fire_alarm_sensor_id', $request->fid)->first();
    echo $Obj;
    exit;
    $responseJson = [];
    if (!empty($Obj)) {
        $responseJson = [
            "code" => 1,

```

```

        "msg" => "Success",
        "data" => $Obj
    ];
} else {
    $responseJson = [
        "code" => 2,
        "msg" => "No data found"
    ];
}
return json_encode($responseJson);
}
}

```

ProcessController.php

```

<?php
namespace App\Http\Controllers;
use App\FireAlarmSensor;
use App\Floor;
use App\Room;
use App\RoomHasFireAlarmSystem;
use Illuminate\Http\Request;
class ProcessController extends Controller
{
    public function fetchData(Request $request)
    {
        $sensorsmapping=RoomHasFireAlarmSystem::get();
        $index=0;
        foreach ($sensorsmapping as $mapRecord) {
            $sensorData=FireAlarmSensor::where('id', $mapRecord->fire_alarm_sensor_id)->first();
            $roomData=Room::where('id', $mapRecord->room_id)->first();
            $floorData=Floor::where('id', $roomData->floor_id)->first();
            $index++;
            $records[]=[
                "index"=>$index,
                "floor"=>$floorData->no,
                "sensor"=>$sensorData->id." ".$sensorData->name,
                "smoke"=>$sensorData->smoke,
                "co2"=>$sensorData->co2,
                "room"=>$roomData->no,
                "statusval"=>($sensorData->smoke>=5 || $sensorData->co2>=5)?1:0,
                "status"=>($sensorData->smoke>=5 || $sensorData->co2>=5)?"red":"green",
            ];
        }
        return json_encode($records);
    }
}

```

```
}  
}
```

HomeController.php

```
<?php  
namespace App\Http\Controllers;  
use App\FireAlarmSensor;  
use App\Floor;  
use App\Room;  
use App\RoomHasFireAlarmSystem;  
use Illuminate\Http\Request;  
class HomeController extends Controller  
{  
    public function welcome(Request $request)  
    {  
        $sensorsmapping=RoomHasFireAlarmSystem::get();  
        $index=0;  
        foreach($sensorsmapping as $mapRecord){  
            $sensorData=FireAlarmSensor::where('id',$mapRecord->fire_alarm_sensor_id)->first();  
            $roomData=Room::where('id',$mapRecord->room_id )->first();  
            $floorData=Floor::where('id',$roomData->floor_id )->first();  
            $index++;  
            $records[]=[  
                "index"=>$index,  
                "floor"=>$floorData->no,  
                "sensor"=>$sensorData->id." ".$sensorData->name,  
                "smoke"=>$sensorData->smoke,  
                "co2"=>$sensorData->co2,  
                "room"=>$roomData->no,  
                "status"=>($sensorData->smoke>=5 || $sensorData->co2>=5)?"red":"green",  
            ];  
        }  
        $data = array(  
            'sensors'=>$records,  
            'title' => "Fire Alarm System",  
            'navtitle' => "Fire Alarm Sensors"  
        );  
        return view('welcome', $data)->render();  
    }  
}
```

Database Migrations

Create_users_table.php

```

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateUsersTable extends Migration
{
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id');
            $table->string('username')->unique();
            $table->string('email')->unique();
            $table->string('password');
            $table->string('mobilen0');
            $table->integer('usertype');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('users');
    }
}

```

Create_fire_alarm_sensors_table.php

```

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateFireAlarmSensorsTable extends Migration
{
    public function up()
    {
        Schema::create('fire_alarm_sensors', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name')->nullable();
            $table->string('detail')->nullable();
            $table->integer('user_id')->unsigned();
            $table->foreign('user_id')->references('id')->on('users');
            $table->integer('smoke');
            $table->integer('co2');
            $table->timestamps();
        });
    }
    public function down()

```

```

    {
        Schema::dropIfExists('fire_alarm_sensors');
    }
}

```

Create_floors_table.php

```

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateFloorsTable extends Migration
{
    function up()
    {
        Schema::create('floors', function (Blueprint $table) {
            $table->increments('id');
            $table->string('no');
            $table->integer('user_id')->unsigned();
            $table->foreign('user_id')->references('id')->on('users');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('floors');
    }
}

```

Create_rooms_table.php

```

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateRoomsTable extends Migration
{
    public function up()
    {
        Schema::create('rooms', function (Blueprint $table) {
            $table->increments('id');
            $table->string('no');
            $table->integer('user_id')->unsigned();
            $table->integer('floor_id')->unsigned();
            $table->foreign('user_id')->references('id')->on('users');
            $table->foreign('floor_id')->references('id')->on('floors');
            $table->timestamps();
        });
    }
}

```

```

        });
    }
    public function down()
    {
        Schema::dropIfExists('rooms');
    }
}

```

Create_room_has_fire_alarm_systems_table.php

```

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateRoomHasFireAlarmSystemsTable extends Migration
{
    public function up()
    {
        Schema::create('room_has_fire_alarm_systems', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('room_id')->unsigned();
            $table->foreign('room_id')->references('id')->on('rooms');
            $table->integer('fire_alarm_sensor_id')->unsigned();
            $table->foreign('fire_alarm_sensor_id')->references('id')->on('fire_alarm_sensors');
            $table->timestamps();
        });
    }
    public function down()
    {
        Schema::dropIfExists('room_has_fire_alarm_systems');
    }
}

```

Create_failed_jobs_table.php

```

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;
class CreateFailedJobsTable extends Migration
{
    public function up()
    {
        Schema::create('failed_jobs', function (Blueprint $table) {
            $table->increments('id');
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');

```



```
        $table->longText('exception');
        $table->timestamp('failed_at')->useCurrent();
    });
}
public function down()
{
    Schema::dropIfExists('failed_jobs');
}
}
```

5.3 RMI Server/Client

Models

User.java

```
package Models;

import java.io.Serializable;

public class User implements Serializable {

    private int id;

    private String username;

    private String email;

    private String password;

    private String mobileno;

    private int userType;

    public User() {

    }

    public User(int id, String username, String email, String password, String mobileno, int userType) {

        this.id = id;

        this.username = username;

        this.email = email;

        this.password = password;

        this.mobileno = mobileno;

        this.userType = userType;

    }

    public int getId() {

        return id;

    }

    public String getUsername() {

        return username;

    }

    public void setUsername(String username) {

        this.username = username;

    }

    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 getMobilen() {
        return mobilen;
    }

    public void setMobilen(String mobilen) {
        this.mobilen = mobilen;
    }

    public int getUserType() {
        return userType;
    }

    public void setUserType(int userType) {
        this.userType = userType;
    }
}

```

Sensor.java

```

package Models;

import java.io.Serializable;

public class Sensor implements Serializable {

    public int sensor_id;

    public String sensor_name;

    public String sensor_description;

    public int user_id;

    public int smoke_value;

    public int co2_value;
}

```

```

public Sensor() {
}

public Sensor(int sensor_id, String sensor_name, String sensor_description, int user_id, int smoke_value, int co2_value) {
    this.sensor_id = sensor_id;
    this.sensor_name = sensor_name;
    this.sensor_description = sensor_description;
    this.user_id = user_id;
    this.smoke_value = smoke_value;
    this.co2_value = co2_value;
}

public int getSensor_id() {
    return sensor_id;
}

public void setSensor_id(int sensor_id) {
    this.sensor_id = sensor_id;
}

public String getSensor_name() {
    return sensor_name;
}

public void setSensor_name(String sensor_name) {
    this.sensor_name = sensor_name;
}

public String getSensor_description() {
    return sensor_description;
}

public void setSensor_description(String sensor_description) {
    this.sensor_description = sensor_description;
}

public int getUser_id() {
    return user_id;
}

public void setUser_id(int user_id) {
    this.user_id = user_id;
}

```

```

public int getSmoke_value() {
    return smoke_value;
}

public void setSmoke_value(int smoke_value) {
    this.smoke_value = smoke_value;
}

public int getCo2_value() {
    return co2_value;
}

public void setCo2_value(int co2_value) {
    this.co2_value = co2_value;
}
}

```

Floor.java

```

package Models;

import java.io.Serializable;

public class Floor implements Serializable{

    public int id;

    public String no;

    public int user;

    public Floor() {
    }

    public Floor(int id, String no, int user) {
        this.id = id;
        this.no = no;
        this.user = user;
    }

    public int getId() {
        return id;
    }

    public String getNo() {
        return no;
    }

    public int getUser() {

```

```

        return user;
    }
    public void setNo(String no) {
        this.no = no;
    }
    public void setUser(int user) {
        this.user = user;
    }
}

```

Room.java

```

package Models;

import java.io.Serializable;

public class Room implements Serializable{

    public int room_id;
    public String room_no;
    public int user_id;
    public int floor_id;
    public Room() {
    }
    public Room(int room_id, String room_no, int user_id, int floor_id) {
        this.room_id = room_id;
        this.room_no = room_no;
        this.user_id = user_id;
        this.floor_id = floor_id;
    }
    public int getRoom_id() {
        return room_id;
    }
    public String getRoom_no() {
        return room_no;
    }
    public int getUser_id() {
        return user_id;
    }
}

```

```

public int getFloor_id() {
    return floor_id;
}

public void setRoom_id(int room_id) {
    this.room_id = room_id;
}

public void setRoom_no(String room_no) {
    this.room_no = room_no;
}

public void setUser_id(int user_id) {
    this.user_id = user_id;
}

public void setFloor_id(int floor_id) {
    this.floor_id = floor_id;
}
}

```

RoomHasSensors.java

```

package Models;

import java.io.Serializable;

public class RoomHasSensors implements Serializable{

    private int rhs_id;

    private int room_id;

    private int sensor_id;

    public RoomHasSensors() {}

    public RoomHasSensors(int rhs_id, int room_id, int sensor_id) {

        this.rhs_id = rhs_id;

        this.room_id = room_id;

        this.sensor_id = sensor_id;

    }

    public int getRhs_id() {

        return rhs_id;

    }

    public void setRhs_id(int rhs_id) {

        this.rhs_id = rhs_id;
    }
}

```

```

    }

    public int getRoom_id() {
        return room_id;
    }

    public void setRoom_id(int room_id) {
        this.room_id = room_id;
    }

    public int getSensor_id() {
        return sensor_id;
    }

    public void setSensor_id(int sensor_id) {
        this.sensor_id = sensor_id;
    }
}

```

FethcehdAllData.java

```

package Models;

import java.io.Serializable;

public class FetchedAllData implements Serializable{

    private int index;

    private String floor;

    private String sensor;

    private String smoke;

    private String co2;

    private String room;

    private String status_val;

    private String status_color;

    public FetchedAllData() {

    }

    public FetchedAllData(int index, String floor, String sensor, String smoke, String co2, String room, String status_val, String status_color) {

        this.index = index;

        this.floor = floor;

        this.sensor = sensor;

        this.smoke = smoke;
    }
}

```



```

    this.co2 = co2;

    this.room = room;

    this.status_val = status_val;

    this.status_color = status_color;
}

public int getIndex() {
    return index;
}

public void setIndex(int index) {
    this.index = index;
}

public String getFloor() {
    return floor;
}

public void setFloor(String floor) {
    this.floor = floor;
}

public String getSensor() {
    return sensor;
}

public void setSensor(String sensor) {
    this.sensor = sensor;
}

public String getSmoke() {
    return smoke;
}

public void setSmoke(String smoke) {
    this.smoke = smoke;
}

public String getCo2() {
    return co2;
}

public void setCo2(String co2) {
    this.co2 = co2;
}

```

```

    }

    public String getRoom() {
        return room;
    }

    public void setRoom(String room) {
        this.room = room;
    }

    public String getStatus_val() {
        return status_val;
    }

    public void setStatus_val(String status_val) {
        this.status_val = status_val;
    }

    public String getStatus_color() {
        return status_color;
    }

    public void setStatus_color(String status_color) {
        this.status_color = status_color;
    }
}

```

Controllers

InsertControllers.java

```

package Controllers;

import Common.httpJson;

import Models.Floor;

import Models.Room;

import Models.RoomHasSensors;

import Models.Sensor;

import firesensor.RMIInterface;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import org.json.simple.JSONArray;

```

```

import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
public class InsertControllers {

    private static RMIIInterface look_up;

    public InsertControllers() throws RemoteException, NotBoundException {

        Registry reg = LocateRegistry.getRegistry("localhost", 9999);

        look_up = (RMIIInterface) reg.lookup("hi_server");

    }

    //insert floor

    public int insertFloor(Floor floor) {

        String resp = new httpJson().jsonRequest("http://127.0.0.1:8000/api/newfloor?uid=" + floor.getUser() + "&no=" +
floor.getNo());

        System.out.println(resp);

        System.out.println("-----");

        if (resp != null) {

            try {

                JSONParser jsonParser = new JSONParser();

                JSONObject responseObj = (JSONObject) jsonParser.parse(resp);

                System.out.println(responseObj.get("code"));

                if (Integer.parseInt(responseObj.get("code").toString()) == 1) {

                    JSONObject newsensor = (JSONObject) responseObj.get("floor");

                    return Integer.parseInt(newsensor.get("id").toString());

                } else {

                    System.out.println("Something wrong");

                }

            } catch (Exception e) {

                System.out.println("Server Exception: " + e);

            }

        }

        return 0;

    }

    //insert room

    public int insertRoom(Room room) {

```

```

        String resp = new
        httpJson().jsonRequest("http://127.0.0.1:8000/api/newroom?uid="+room.getUser_id()+"&fid="+room.getFloor_id()+"&no="+r
        oom.getRoom_no());

        System.out.println(resp);

        System.out.println("-----");

        if (resp != null) {
            try {
                JSONParser jsonParser = new JSONParser();

                JSONObject responseObj = (JSONObject) jsonParser.parse(resp);

                System.out.println(responseObj.get("code"));

                if (Integer.parseInt(responseObj.get("code").toString()) == 1) {
                    JSONObject newsensor = (JSONObject) responseObj.get("floor");

                    return Integer.parseInt(newsensor.get("id").toString());
                } else {
                    System.out.println("Something wrong");
                }
            } catch (Exception e) {
                System.out.println("Server Exception: " + e);
            }
        }

        return 0;
    }

    //insert sensor

    public int insertSensor(Sensor sensor) {

        String resp = new httpJson().jsonRequest("http://127.0.0.1:8000/api/newsensor?uid=" + sensor.getUser_id() +
        "&param1=0&param2=0&name=" + sensor.getSensor_name() + "&detail=" + sensor.getSensor_description());

        System.out.println(resp);

        System.out.println("-----");

        if (resp != null) {
            try {
                JSONParser jsonParser = new JSONParser();

                JSONObject responseObj = (JSONObject) jsonParser.parse(resp);

                System.out.println(responseObj.get("code"));

                if (Integer.parseInt(responseObj.get("code").toString()) == 1) {
                    JSONObject newsensor = (JSONObject) responseObj.get("floor");

```

```

        return Integer.parseInt(newsensor.get("id").toString());
    } else {
        System.out.println("Something wrong");
    }
} catch (Exception e) {
    System.out.println("Server Exception: " + e);
}
}
return 0;
}

//insert room has sensor
public int insertRoomHasSensor(RoomHasSensors rhs) {
    String resp = new httpJson().jsonRequest("http://127.0.0.1:8000/api/newroomhassensor?rid=" + rhs.getRoom_id() +
"&fid=" + rhs.getSensor_id());
    System.out.println(resp);
    if (resp != null) {
        try {
            JSONParser jsonParser = new JSONParser();
            JSONObject responseObj = (JSONObject) jsonParser.parse(resp);
            System.out.println(responseObj.get("code"));
            if (Integer.parseInt(responseObj.get("code").toString()) == 1) {
                JSONObject newsensor = (JSONObject) responseObj.get("floor");
                System.out.println(newsensor.get("id").toString());
                return Integer.parseInt(newsensor.get("id").toString());
            } else {
                System.out.println("Something wrong");
            }
        } catch (Exception e) {
            System.out.println("Server Exception: " + e);
        }
    }
    return 0;
}
}

```

UserController.java

```
package Controllers;

import Common.httpJson;

import Models.User;

import firesensor.RMIInterface;

import firesensor.Server;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import org.json.simple.JSONObject;

import org.json.simple.parser.JSONParser;

public class UserController {

    private static RMIInterface look_up;

    public static int user_id;

    public static String user_email;

    public UserController() throws RemoteException, NotBoundException {

        Registry reg = LocateRegistry.getRegistry("localhost", 9999);

        look_up = (RMIInterface) reg.lookup("hi_server");

    }

    //login user

    public int loginUser(String username, String password) {

        String resp = new httpJson().jsonRequest("http://127.0.0.1:8000/api/authadmin?un=" + username + "&pw=" + password);

        System.out.println("-----");

        if (resp != null) {

            try {

                JSONParser jsonParser = new JSONParser();

                JSONObject responseObj = (JSONObject) jsonParser.parse(resp);

                System.out.println(responseObj.get("code"));

                if (Integer.parseInt(responseObj.get("code").toString()) == 1) {

                    JSONObject newsensor = (JSONObject) responseObj.get("user");

                    user_id = Integer.parseInt(newsensor.get("id").toString());

                    user_email = newsensor.get("email").toString();

                    return 1;

                }

            }

        }

    }

}
```

```

        } else {

            System.out.println(responseObj.get("msg"));

        }
    } catch (Exception e) {

        System.out.println("Server Exception: " + e);

    }

}

return 0;

}

//register user

public int registerUser(String username, String email, String password, String mobileNo, int userType) {

    String resp = new
    httpJson().jsonRequest("http://127.0.0.1:8000/api/regadmin?un="+username+"&pw="+password+"&email="+email+"&mno="+
    +mobileNo);

    System.out.println("http://127.0.0.1:8000/api/regadmin?un="+username+"&pw="+password+"&email="+email+"&mno="+mo
    bileNo);

    System.out.println("-----");

    if (resp != null) {

        try {

            JSONParser jsonParser = new JSONParser();

            JSONObject responseObj = (JSONObject) jsonParser.parse(resp);

            System.out.println(responseObj.get("code"));

            if (Integer.parseInt(responseObj.get("code").toString()) == 1) {

                JSONObject newsensor = (JSONObject) responseObj.get("user");

                return 1;

            } else {

                System.out.println("");

            }

        } catch (Exception e) {

            System.out.println("Server Exception: " + e);

        }

    }

    return 0;

}

}

```

Views

Dashboard.java

```
package Views;

import Common.Email_Send;

import Controllers.UserController;

import Models.FetchedAllData;

import firesensor.RMIInterface;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.ArrayList;

import java.util.List;

import java.util.Vector;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.table.DefaultTableModel;

public class Dashboard extends javax.swing.JFrame {

    private static RMIInterface look_up;

    public Dashboard() throws RemoteException, NotBoundException {

        if (UserController.user_id != 0) {

            initComponents();

            this.setAlwaysOnTop(true);

            loadDataToTable();

        } else {

            Login login = new Login();

            login.setVisible(true);

            this.dispose();

        }

    }

    @SuppressWarnings("unchecked")

    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
```



```

jPanel2 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();
jPanel3 = new javax.swing.JPanel();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
jButton3 = new javax.swing.JButton();
jPanel4 = new javax.swing.JPanel();
jScrollPane1 = new javax.swing.JScrollPane();
fetchTable = new javax.swing.JTable();
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
jLabel1.setFont(new java.awt.Font("Calibri", 1, 36));
jLabel1.setText("ADMINISTRATOR DASHBOARD");
javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addGap(138, 138, 138)
            .addComponent(jLabel1)
            .addGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );
jPanel2Layout.setVerticalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addGap(138, 138, 138)
            .addComponent(jLabel1)
            .addGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        );

jButton1.setFont(new java.awt.Font("Calibri", 1, 18));
jButton1.setText("ADD NEW FLOOR");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

```

```

    }
});
jButton2.setFont(new java.awt.Font("Calibri", 1, 18));
jButton2.setText("ADD NEW ROOM");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});
jButton3.setFont(new java.awt.Font("Calibri", 1, 18));
jButton3.setText("ADD NEW SENSOR");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});
javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(
    jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel3Layout.createSequentialGroup()
            .addGap(12, 12, 12)
            .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 215, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 221, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 215, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(12, true))
);
jPanel3Layout.setVerticalGroup(
    jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel3Layout.createSequentialGroup()
            .addGap(12, 12, 12)
            .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 40, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addContainerGap(12, true))
);

```

```

        .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

        .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 74,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 74,
javax.swing.GroupLayout.PREFERRED_SIZE)))

        .addContainerGap())
);

fetchTable.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        },
    new String [] {
        "No", "Floor No", "Room No", "Smoke Sensor Value", "CO2 Sensor Value", "Sensor Status"
    }
));

fetchTable.setRowHeight(20);
jScrollPane1.setViewportView(fetchTable);
if (fetchTable.getColumnModel().getColumnCount() > 0) {
    fetchTable.getColumnModel().getColumn(0).setPreferredWidth(5);
    fetchTable.getColumnModel().getColumn(0).setHeaderValue("No");
    fetchTable.getColumnModel().getColumn(1).setHeaderValue("Floor No");
    fetchTable.getColumnModel().getColumn(2).setHeaderValue("Room No");
    fetchTable.getColumnModel().getColumn(3).setHeaderValue("Smoke Sensor Value");
    fetchTable.getColumnModel().getColumn(4).setHeaderValue("CO2 Sensor Value");
    fetchTable.getColumnModel().getColumn(5).setHeaderValue("Sensor Status");
}

javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
jPanel4.setLayout(jPanel4Layout);
jPanel4Layout.setHorizontalGroup(
    jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel4Layout.createSequentialGroup()
            .addContainerGap()

            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 752, Short.MAX_VALUE)

            .addContainerGap())

```

```

);
jPanel4Layout.setVerticalGroup(
    jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel4Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 251, Short.MAX_VALUE)
            .addContainerGap())
        );
javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(36, 36, 36)
                    .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(0, 0, Short.MAX_VALUE)))
            .addContainerGap())
        );
jPanel1Layout.setVerticalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        );

```

```

        .addComponent(jPanel4, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(0, 0, 0))
    );

    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
    );

    pack();

    setLocationRelativeTo(null);
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    AddFloor addFloor = new AddFloor();
    addFloor.setVisible(true);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        AddRoom addRoom = new AddRoom();
        addRoom.setVisible(true);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    try {

```

```

        AddSensor2Room addSensor2Room = new AddSensor2Room();
        addSensor2Room.setVisible(true);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public static void main(String args[]) {
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Dashboard.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Dashboard.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Dashboard.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(Dashboard.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }

    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                new Dashboard().setVisible(true);
            } catch (RemoteException ex) {
                Logger.getLogger(Dashboard.class.getName()).log(Level.SEVERE, null, ex);
            } catch (NotBoundException ex) {
                Logger.getLogger(Dashboard.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });
}

```

```

}

private javax.swing.JTable fetchTable;

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JLabel jLabel1;

private javax.swing.JPanel jPanel1;

private javax.swing.JPanel jPanel2;

private javax.swing.JPanel jPanel3;

private javax.swing.JPanel jPanel4;

private javax.swing.JScrollPane jScrollPane1;

List<FetchedAllData> allDatasExists;

private void loadDataToTable() throws RemoteException, NotBoundException {

    allDatasExists = new ArrayList<>();

    new Thread(new Runnable() {

        @Override

        public void run() {

            try {

                Registry reg = LocateRegistry.getRegistry("localhost", 9999);

                look_up = (RMIIInterface) reg.lookup("hi_server");

                while (true) {

                    List<FetchedAllData> allDatas = look_up.allFetchData();

                    DefaultTableModel dtm = (DefaultTableModel) fetchTable.getModel();

                    dtm.setRowCount(0);

                    int i = 0;

                    for (FetchedAllData sensor : allDatas) {

                        Vector v = new Vector();

                        v.add(sensor.getIndex());

                        v.add(sensor.getFloor());

                        v.add(sensor.getRoom());

                        v.add(sensor.getSmoke());

                        v.add(sensor.getCo2());

                        v.add((sensor.getStatus_val().equals("1")) ? "<html><p style='color:red'>ACTIVE</p></html>" : "<html><p style='color:green'>DEACTIVE</p></html>");

```

```

dtm.addRow(v);
if (sensor.getStatus_val().equals("1")) {
    boolean checkA = true;
    for (FetchedAllData allDataa : allDatasExists) {
        if (allDataa.getIndex() == sensor.getIndex()) {
            checkA = false;
        }
    }
    if (checkA) {
        allDatasExists.add(sensor);
        new Email_Send().sendEmail("Fire Alarm Activated Alert", UserController.user_email, "Warning!!! The " +
sensor.getRoom() + " alarm has been activated. Please call 101 for Fire Emergency Services");
        System.out.println("Email sent");
        System.out.println("SMS sent");
    }
}

//notification time
Thread.sleep(15000);
boolean checkb = true;
for (FetchedAllData allDataa : allDatasExists) {
    if (allDataa.getStatus_val().equals("1")) {
        checkb = false;
    }
}
if (checkb) {
    allDatasExists.clear();
}
} catch (Exception e) {
    System.out.println(e);
}
}
}).start();
}
}

```


Common

Email_Data.java

```
package Common;

public class Email_Data {

    String email=null;

    String password=null;

    //Email and Password of the System email

    public Email_Data getdata(){

        Email_Data ed=new Email_Data();

        ed.email="firealarmsystemds@gmail.com";

        ed.password="sliitds12345";

        return ed;

    }

}
```

Email_Send.java

```
package Common;

import java.util.Properties;

import javax.mail.Authenticator;

import javax.mail.Message;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

public class Email_Send {

    public void sendEmail(String subject, String reciever, String htmlcontent) {

        new Thread(new Runnable() {

            public void run() {

                try {

                    final String sender_mail = new Email_Data().getdata().email;

                    final String sender_password = new Email_Data().getdata().password;

                    Properties props = new Properties();

                    props.put("mail.smtp.auth", "true");

                }

            }

        }).start();

    }

}
```

```

        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", "smtp.gmail.com");
        props.put("mail.smtp.port", "587");

        Session session = Session.getDefaultInstance(props,
            new Authenticator() {
                protected PasswordAuthentication getPasswordAuthentication() {
                    return new PasswordAuthentication(
                        new Email_Data().getdata().email, new Email_Data().getdata().password);
                }
            });

        Message message = new MimeMessage(session);
        message.setFrom(new InternetAddress(sender_mail));
        message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(reciever));
        message.setSubject(subject);
        // message.setText(msg_content);
        message.setContent(htmlcontent, "text/html;charset=utf-8");
        Transport.send(message);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

}).start();
}
}

```

httpJson.java

```

package Common;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import org.json.simple.JSONArray;

public class httpJson {

```

```

public String jsonRequest(String url) {
    String response = null;
    try {
        URL u = new URL(url);
        HttpURLConnection hr = (HttpURLConnection) u.openConnection();
        if (hr.getResponseCode() == 200) {
            InputStream im = hr.getInputStream();
            StringBuffer sb = new StringBuffer();
            BufferedReader br = new BufferedReader(new InputStreamReader(im));
            String line = br.readLine();
            response = line;
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return response;
}
}

```

[Main](#)

RMInterface.java

```

package firesensor;

import Models.FetchedAllData;
import Models.Floor;
import Models.Room;
import Models.Sensor;
import Models.User;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
import sun.security.util.PropertyExpander;

public interface RMInterface extends Remote{
    User findUser(User u) throws RemoteException;
}

```

```

List<User> allUser() throws RemoteException;

Floor findFloor(Floor f) throws RemoteException;

List<Floor> allFloor() throws RemoteException;

Room findRoom(Room r) throws RemoteException;

List<Room> allRoom()throws RemoteException;

Sensor findSensor(Sensor s)throws RemoteException;

List<Sensor> allSensors() throws RemoteException;

List<FetchedAllData> allFetchData() throws RemoteException;

}

```

Client.java

```

package firesensor;

import Models.Floor;

import Models.User;

import java.net.MalformedURLException;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.List;

public class Client {

    private static RMIInterface look_up;

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

        Client client = new Client();

        client.connectRemote();

    }

    private void connectRemote() throws RemoteException, NotBoundException {

        Registry reg = LocateRegistry.getRegistry("localhost", 9999);

        look_up =(RMIInterface) reg.lookup("hi_server");

    }

}

```

Server.java

```

package firesensor;

```

```

import Common.httpJson;
import Models.FetchedAllData;
import Models.Floor;
import Models.Room;
import Models.Sensor;
import Models.User;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;

public class Server extends UnicastRemoteObject implements RMIInterface {

    private List<User> userList;

    private List<Floor> floorList;

    private List<Room> roomList;

    private List<Sensor> sensorList;

    private List<FetchedAllData> fetchAllDataList;

    public Server() throws RemoteException {

        super();

        initializeData();

    }

    public static void main(String[] args) {

        try {

```

```

        Registry reg = LocateRegistry.createRegistry(9999);

        reg.rebind("hi_server", new Server());

        System.err.println("Server ready");
    } catch (Exception e) {

        System.err.println("Server exception: " + e.getMessage());
    }
}

private void initializeData() {
    userList = new ArrayList<>();
    floorList = new ArrayList<>();
    roomList = new ArrayList<>();
    sensorList = new ArrayList<>();
    fetchAIDataList = new ArrayList<>();
    serverUpdate();
}

public User findUser(User user) throws RemoteException {
    Predicate<User> predicate = x -> x.getEmail().equals(user.getEmail());
    return userList.stream().filter(predicate).findFirst().get();
}

public List<User> allUser() throws RemoteException {
    return userList;
}

public Floor findFloor(Floor f) throws RemoteException {
    Predicate<Floor> predicate = x -> x.getNo().equals(f.getId());
    return floorList.stream().filter(predicate).findFirst().get();
}

public List<Floor> allFloor() throws RemoteException {
    return floorList;
}

public Room findRoom(Room r) throws RemoteException {
    Predicate<Room> predicate = x -> x.getRoom_id() == r.getRoom_id();
    return roomList.stream().filter(predicate).findFirst().get();
}

public List<Room> allRoom() throws RemoteException {

```

```

        return roomList;
    }

    public Sensor findSensor(Sensor s) throws RemoteException {
        Predicate<Sensor> predicate = x -> x.getSensor_id() == s.getSensor_id();
        return sensorList.stream().filter(predicate).findFirst().get();
    }

    public List<Sensor> allSensors() throws RemoteException {
        return sensorList;
    }

    public List<FetchedAllData> allFetchData() throws RemoteException {
        return fetchAllDataList;
    }

    private void serverUpdate() {
        System.out.println("Server Update Function Execute-----");
        Thread thread = new Thread(new Runnable() {
            public void run() {
                try {
                    System.out.println("Server Thread Start-----");
                    apiRequests();
                    Thread.sleep(5000);
                    serverUpdate();
                } catch (Exception e) {
                    System.out.println("Thread Exception : " + e);
                }
            }
        });
        thread.start();
    }

    private void apiRequests() {
        System.out.println("Send API Requests");
        //-----Floor JSON
        //-----
        String floor_response = new httpJson().jsonRequest("http://127.0.0.1:8000/api/allfloors");
    }

```

```

if (floor_response != null) {
    try {
        JSONParser jsonParser = new JSONParser();
        JSONObject responseObj = (JSONObject) jsonParser.parse(floor_response);
        JSONArray array = (JSONArray) responseObj.get("floors");
        floorList.clear();
        for (Object obj : array) {
            JSONObject jsonObject = (JSONObject) obj;
            floorList.add(
                new Floor(
                    Integer.parseInt(jsonObject.get("id").toString()),
                    jsonObject.get("no").toString(),
                    Integer.parseInt(jsonObject.get("user_id").toString())));
        }
        System.out.println("Response from all floors: complete-----");
    } catch (Exception e) {
        System.out.println("Server Exception: " + e);
    }
}

//-----Room JSON
//-----

String room_response = new httpJson().jsonRequest("http://127.0.0.1:8000/api/allrooms");
if (room_response != null) {
    try {
        JSONParser jsonParser = new JSONParser();
        JSONObject responseObj = (JSONObject) jsonParser.parse(room_response);
        JSONArray array = (JSONArray) responseObj.get("rooms");
        roomList.clear();
        for (Object obj : array) {
            JSONObject jsonObject = (JSONObject) obj;
            roomList.add(
                new Room(
                    Integer.parseInt(jsonObject.get("id").toString()),
                    jsonObject.get("no").toString(),

```



```

        Integer.parseInt(jsonObject.get("user_id").toString()),
        Integer.parseInt(jsonObject.get("floor_id").toString())));
    }

    System.out.println("Response from all rooms: complete-----");
} catch (Exception e) {
    System.out.println("Server Exception: " + e);
}
}

//-----Room JSON
//-----

String sensor_response = new httpJson().jsonRequest("http://127.0.0.1:8000/api/allsensors");
if (sensor_response != null) {
    try {
        JSONParser jsonParser = new JSONParser();
        JSONObject responseObj = (JSONObject) jsonParser.parse(sensor_response);
        JSONArray array = (JSONArray) responseObj.get("floors");
        sensorList.clear();
        for (Object obj : array) {
            JSONObject jsonObject = (JSONObject) obj;
            sensorList.add(
                new Sensor(
                    Integer.parseInt(jsonObject.get("id").toString()),
                    (jsonObject.get("name") == null) ? "" : jsonObject.get("name").toString(),
                    (jsonObject.get("detail") == null) ? "" : jsonObject.get("detail").toString(),
                    Integer.parseInt(jsonObject.get("user_id").toString()),
                    Integer.parseInt(jsonObject.get("smoke").toString()),
                    Integer.parseInt(jsonObject.get("co2").toString())
                );
            }
        System.out.println("Response from all sensors: complete-----");
    } catch (Exception e) {
        System.out.println("Server Exception: " + e);
    }
}

```

```

    }
//-----FetchAllData JSON
//-----
String fetchAllData_response = new httpJson().jsonRequest("http://127.0.0.1:8000/api/fetchSensors");
//System.out.println(fetchAllData_response);
if (fetchAllData_response != null) {
    try {
        JSONParser jsonParser = new JSONParser();
        JSONArray responseObj = (JSONArray) jsonParser.parse(fetchAllData_response);
        fetchAllDataList.clear();
        for (Object obj : responseObj) {
            JSONObject jsonObject = (JSONObject) obj;
            fetchAllDataList.add(
                new FetchedAllData(
                    Integer.parseInt(jsonObject.get("index").toString()),
                    jsonObject.get("floor").toString(),
                    jsonObject.get("sensor").toString(),
                    jsonObject.get("smoke").toString(),
                    jsonObject.get("co2").toString(),
                    jsonObject.get("room").toString(),
                    jsonObject.get("statusval").toString(),
                    jsonObject.get("status").toString()
                )
            );
        }
        System.out.println("Response from all FetchedData: complete-----");
    } catch (Exception e) {
        System.out.println("Server Exception: " + e);
    }
}
}
}

```