Distributed Systems

Assignment 2

Registration Number	Name
IT16073760	**V.S. Thenuwara**
IT17172806	**De Silva D.K.G.**
IT18172560	**R.M.G.H. Amarasinghe**
IT18111484	**Harshanga K.P.C.**

Content

1. Introduction

This is a report of the Assignment 2 for the Distributed Systems. This scenario was implementing the Fire Alarm monitoring system. This system mainly has the web and desktop versions. In web application users can view the sensors in the particular floor and room whether they are active or not. Desktop application admin can add sensor to the system. This alarm monitoring system capture the smoke level and CO_2 level of the sensor. If those level above 5 warning message will be displayed. In addition desktop application notifies the user via SMS or email. Sensor app captures the smoke level and CO_2 level and updates the database in every 30 seconds. This system has REST API as backend accomplish by MongoDB database and frontend will be the interfaces of the desktop and web applications.

- Web application – This is implemented by using Angular + CSS + JavaScript pages with Bootstrap to make interfaces more attractive.
- REST API – This was implementing according to the SOA principles with using Java, Springboot framework runs it's embedded tomcat server port 8080 get all the services over the HTTP request.

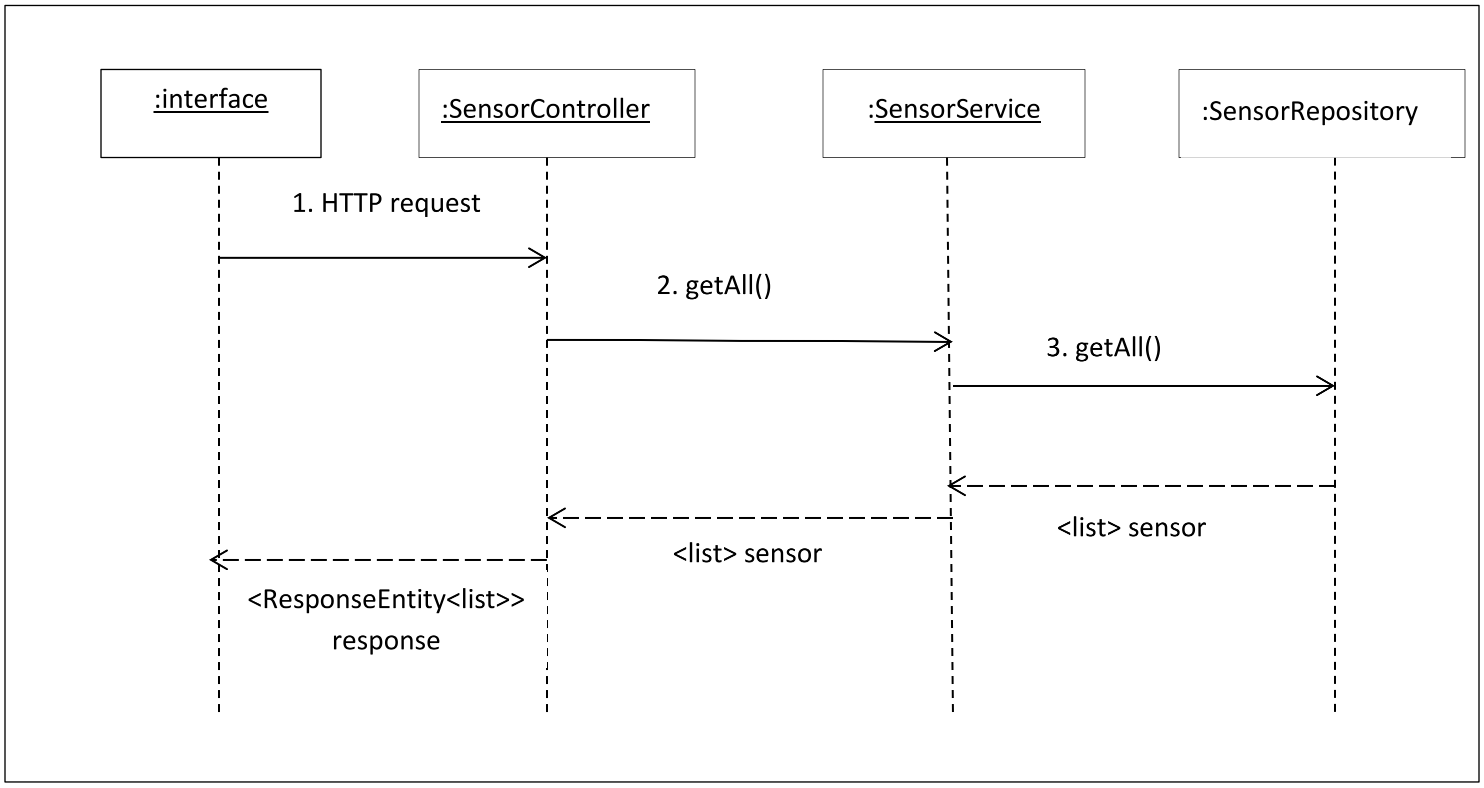
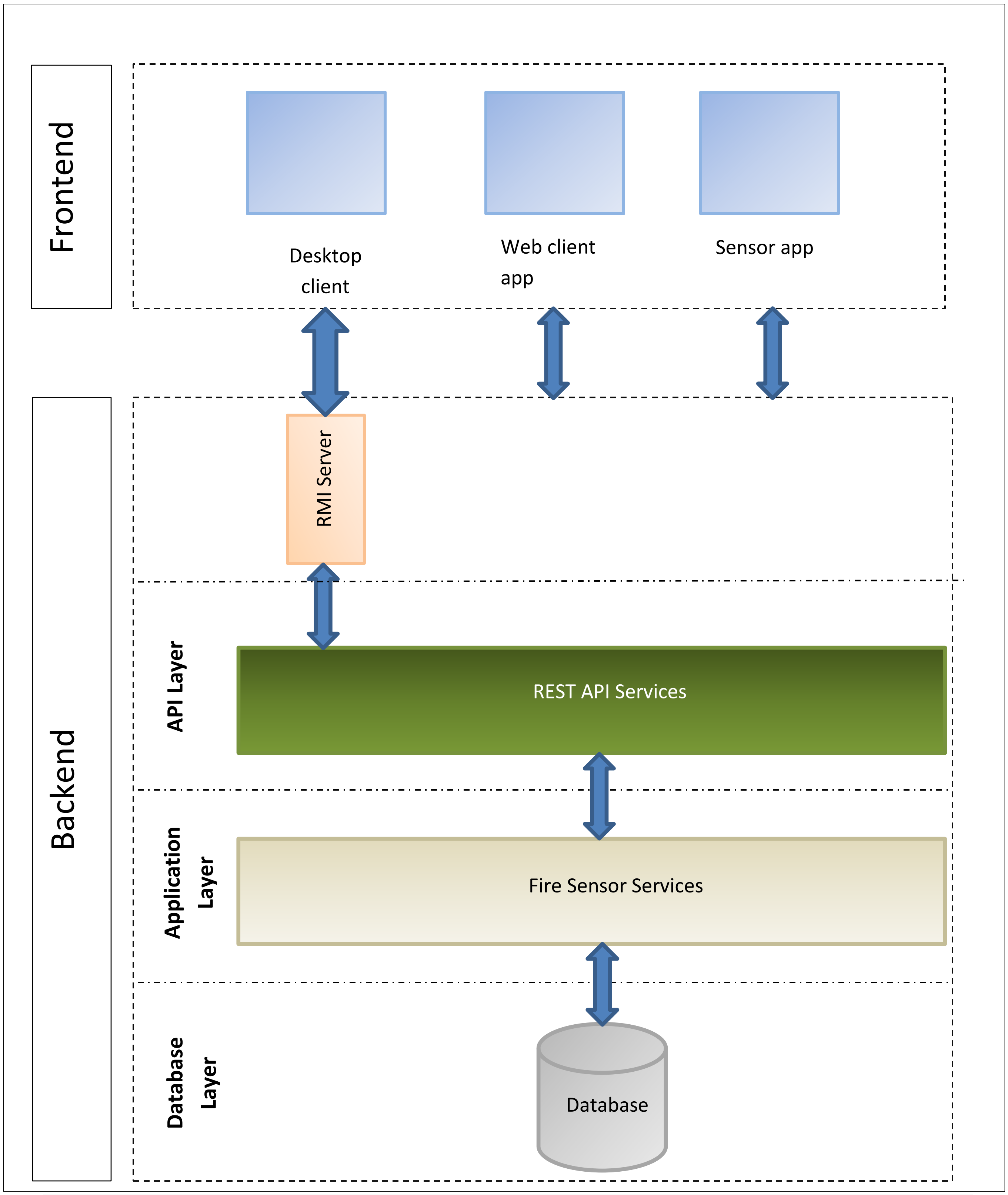


Figure 1: fetch sensor details

2. System Architecture



3. Architecture

The backend implemented SOA concept as mention earlier. Other functionalities and services based on the http request/ response. The main advantage is allowing us to change/ modify the internal methods with minimum changes of the frontend. There are 2 main categories.

- Sensor
- User

They are run independently; this will increase the functionality cohesiveness by decreasing coupling nature. Although the system using same NoSQL database.

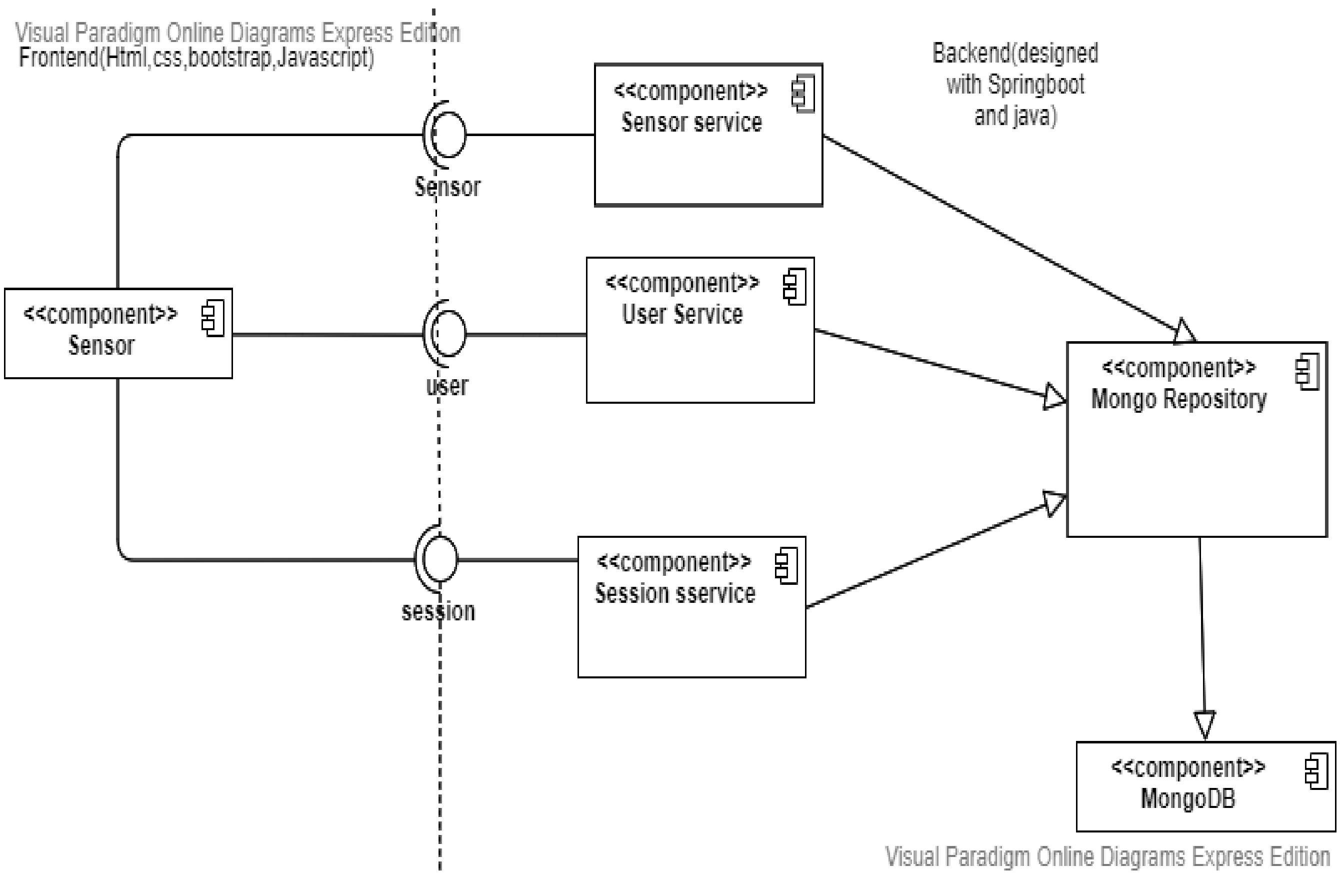

Visual Paradigm Online Diagrams Express Edition
Frontend(Html,css,bootstrap,Javascript)
Backend(designed with Springboot and java)
<<component>> Sensor service
Sensor
<<component>> Sensor
<<component>> User Service
user
<<component>> Mongo Repository
<<component>> Session sservice
session
<<component>> MongoDB
Visual Paradigm Online Diagrams Express Edition

Figure 2: high level diagram

For considering backend, SOA was implemented as a web services in major functions, these functions are defined as API for the users to consume the services. JSON used for the communication between backend and frontend since MongoDB is being well combination with this. According to the diagram the system consist major 3 services. Those services provide set of sub functionalities. Client consumes these sub functionalities according to the authentication that provided.

There are 4 various perspectives implementing the services.

- Controller
- Model
- Service
- Service implementation

Controller mapped service requests to its URL and controls services indepentendently respectively, these services has its own controllers. Models represent the entity such as sensor, users. This models everything resolves which around. Service interfaces used to interact with the sub functionalities due to each services provide large number of sub functionalities. Those sub functionalities commits data manipulation to catch the various requests that services supports.

System divided into frontend and backend, frontend which user can interact with the system and user represented in the frontend. JSON choose as communication medium due to the lightweight, easy to manipulate with javaScript since JavaScript was the only language that browser can understand. Considering the frontend web application interacts with the REST API to view data and desktop application interact with the REST API through the RMI server. Skelton will be the desktop application and sub will be the server.

4. Backend workflow

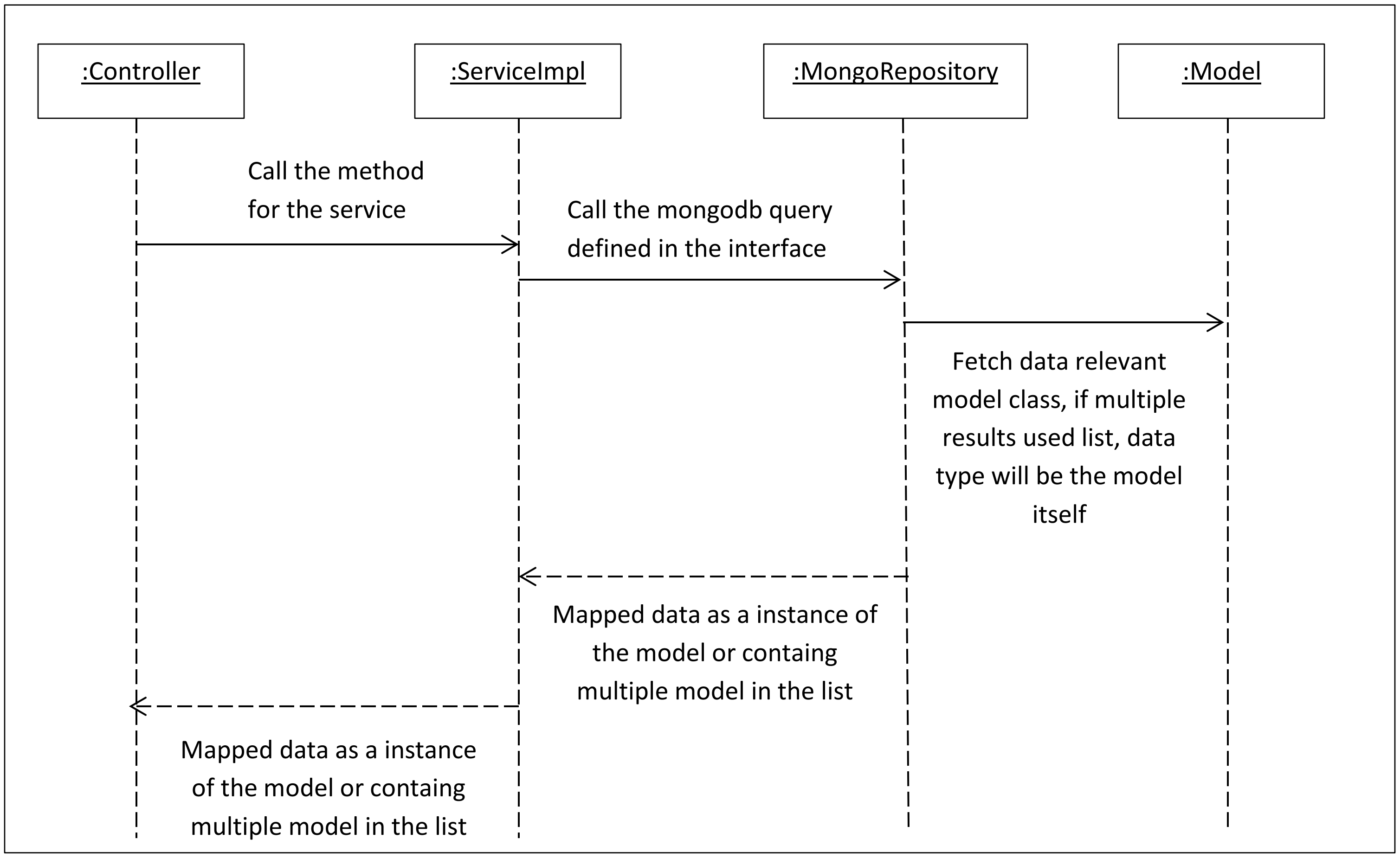

:Controller
:ServiceImpl
:MongoRepository
:Model
Call the method for the service
Call the mongodb query defined in the interface
Fetch data relevant model class, if multiple results used list, data type will be the model itself
Mapped data as a instance of the model or containg multiple model in the list
Mapped data as a instance of the model or containg multiple model in the list

Figure 3: flow of the serving request by the backend

The above diagram is common to the all the backend services. Controller classes responsible for the tasks of the backend for each services. For an example view sensor services all the requests handle by sensorControl class mapped /sensor. According to the URL controller call the respective method. GET request of the URL sensor (/sensor) the controller call the method that display the all the sensors. sensorController call a method in the sensorServiceImp(service implementation class). Inside this method implementation sensorRepository will contact with the MongoDB instance. This sensorRepository we do not implement but we can mention methods as we wish due to the interface extends the MongoRepository interface which we implemented earlier invoke the runtime. sensorRepository defined methods earlier are Java mehods form in the MongoDB queries. In the runtime use sensor model class to map the details of the each sensor an instance of the sensor class once fetch the data from the database. The sensor model class will be the bridge between MongoDB and Java class.

5. Frontend Workflow

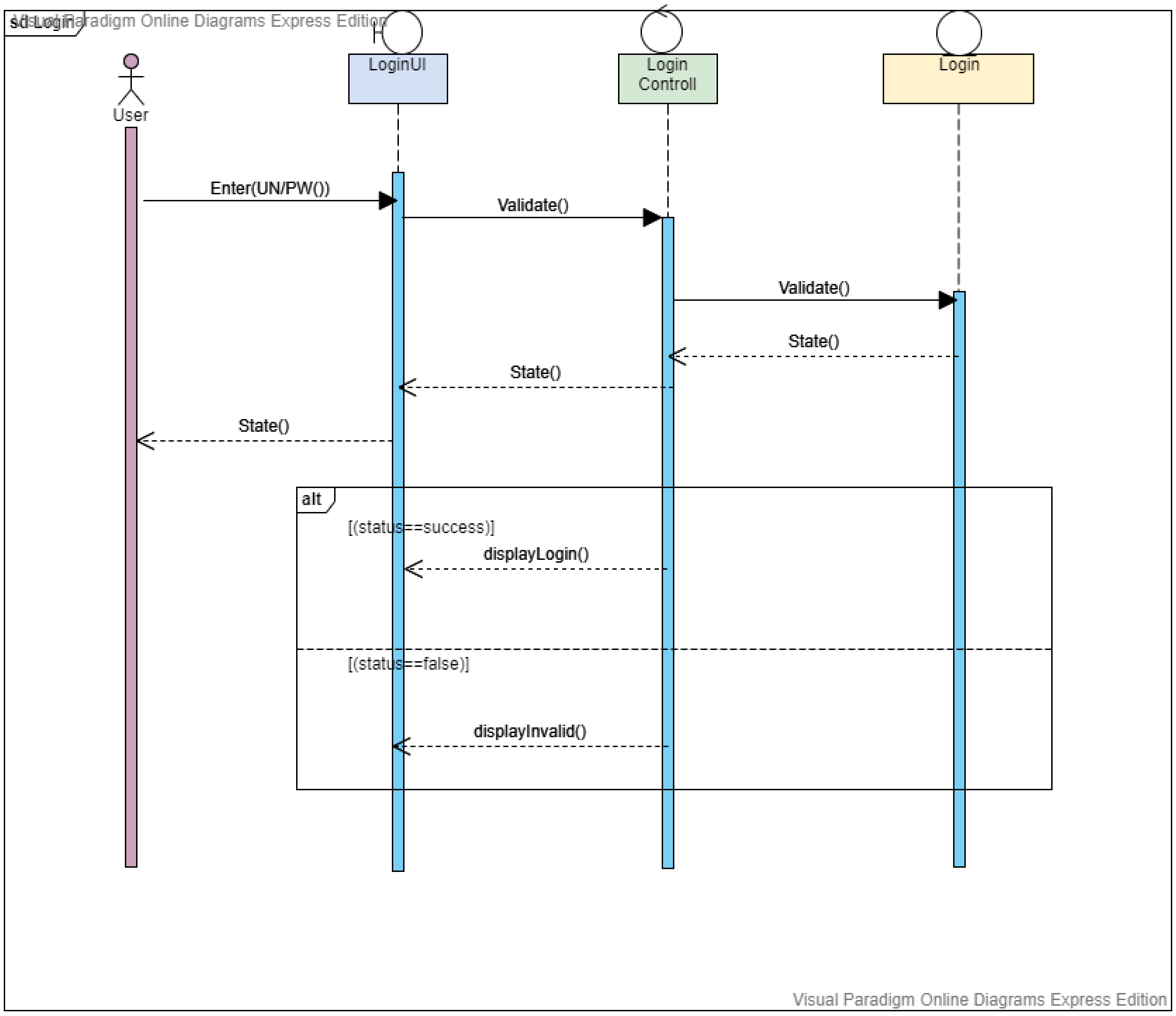
This system basically has two frontends one for web application and other for the desktop application. Web application contains set of web pages implemented by using HTML, CSS and Bootstrap framework. JavaScript used change the content dynamically.

Desktop application is using java, Netbeans IDE to create the pages.

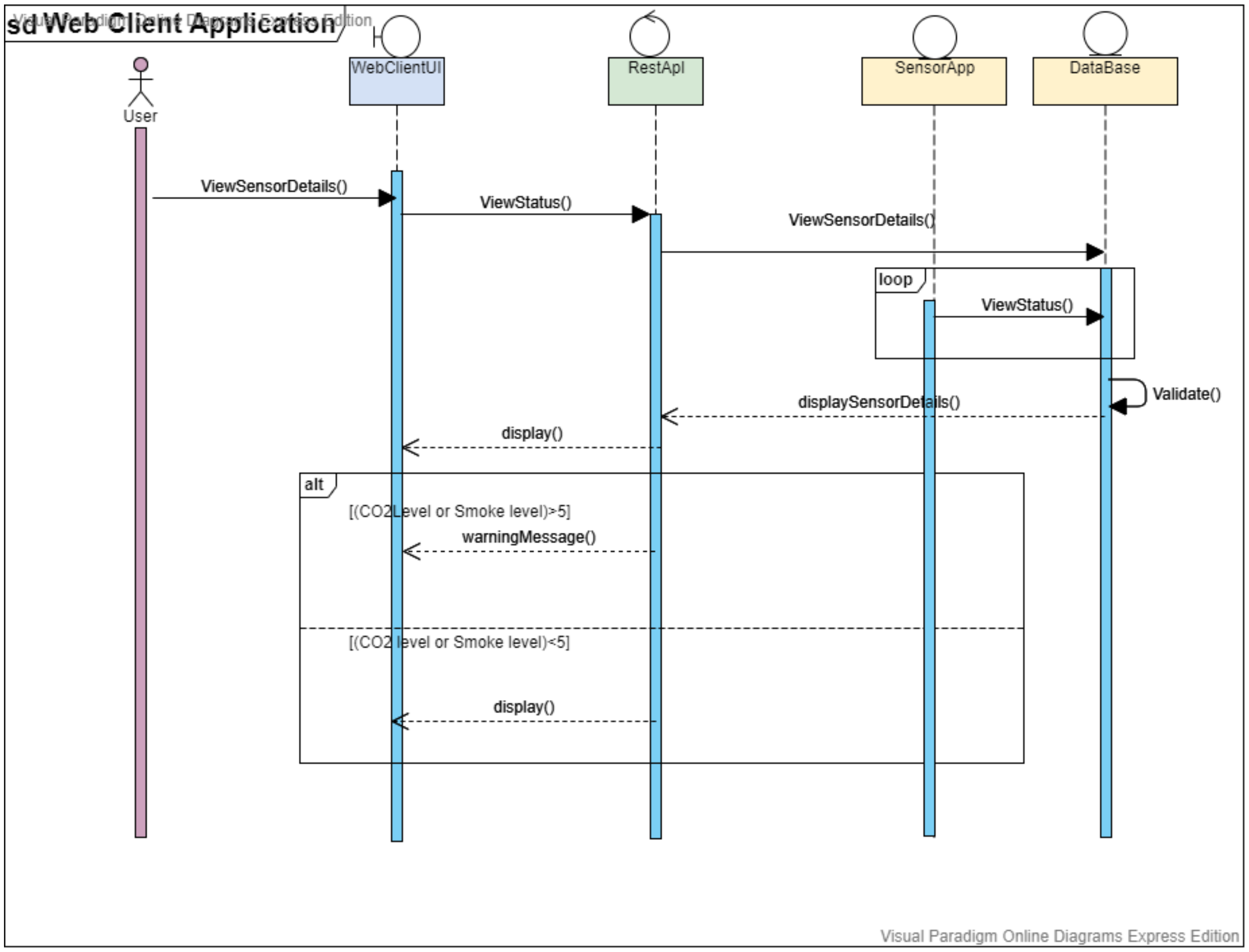
The overall frontend was using MVC architecture. Web pages using as views, controllers will be JavaScript functions but there is no clear model in this system.

6. Sequence diagram for the system

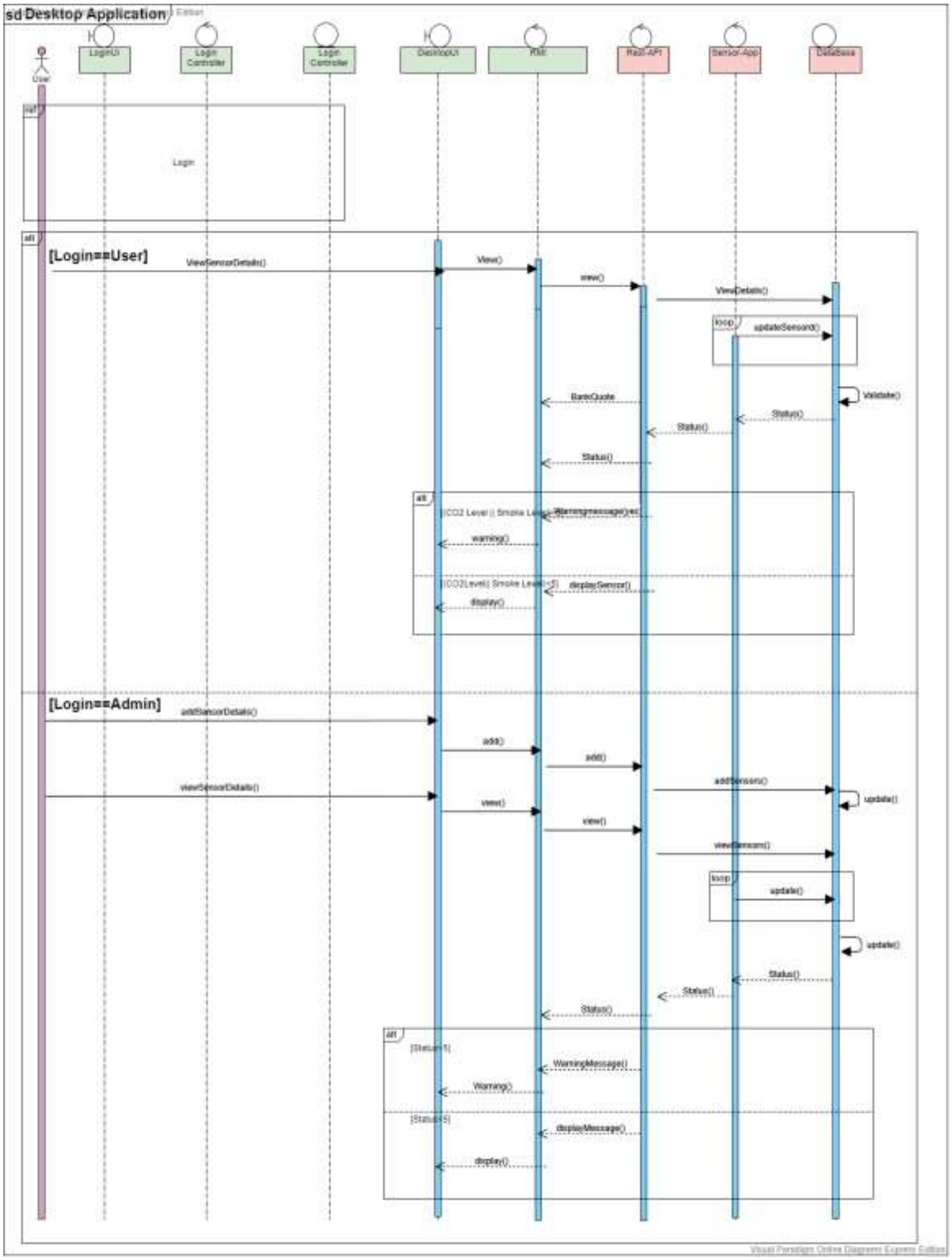
6.1. Login



6.2. Web application


sd Web Client Application
Visual Paradigm Online Diagrams Express Edition
User
WebClientUI
RestApI
SensorApp
DataBase
ViewSensorDetails()
ViewStatus()
ViewSensorDetails()
loop
ViewStatus()
Validate()
displaySensorDetails()
display()
alt
[(CO2Level or Smoke level)>5]
warningMessage()
[(CO2 level or Smoke level)<5]
display()
Visual Paradigm Online Diagrams Express Edition

6.3. Desktop application



7. Appendix

7.1. Web application

Model Controller

```
app.controller('AlarmController', ['$scope',
    function ($scope) {
        $scope.modalShown = false
        $scope.openRisk = function () {
            alert('Your Room is in a high danger .Please contact our service for help and get immediate
actions');
        }
        $scope.openModel = function (i) {
            $scope.modalShown = !$scope.modalShown;
            $scope.selected = i.co2;
            $scope.selected2 = i.smoke;
            $scope.Roomno = i.roomNo;
            if ($scope.selected >= 5 && $scope.selected2 >= 5) {
                $scope.Alert = "Risky";
            } else {
                $scope.Alert = "Good";
           }
        };
        $scope.isSelected = function (i) {
            return $scope.selected === i.co2;
            return $scope.selected2 === i.smoke;

        }

        $scope.getAlert = function (i) {
            if (i.co2 >= 5 && i.smoke >= 5) {
                return " Highly Risky !(click here)";
            }
        }
    }
]);
```

App.js

```
'use strict';

// Declare app level module which depends on views, and core components
angular.module('myApp', [
  'ngRoute',
  'myApp.view1',
  'myApp.view2',
  'myApp.version'
]).
config(['$locationProvider', '$routeProvider', function($locationProvider, $routeProvider) {
  $locationProvider.hashPrefix('!');

  $routeProvider.otherwise({redirectTo: '/view1'});
}]);
let app = angular.module('SensorApp',[]);
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Fire Sensor Alarm</title>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="app.css">
    <link rel="stylesheet" href="lib/html5-boilerplate/dist/css/normalize.css">
    <link rel="stylesheet" href="lib/html5-boilerplate/dist/css/main.css">

    <!-- jQuery library -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <!-- Popper JS -->
    <script
            src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <!-- Latest compiled JavaScript -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-
J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>
</head>
<body ng-app="SensorApp" ng-controller="MainController">

    <div class="box" >
        <img src="fireman.png" style="width: 15%;height:12%;margin-bottom: -13%;margin-left: 20%"/>
            <h1 style="text-align: center;color: white;">Fire-Sensor-Alarm!</h1>
            <br><p>Your Safety is our First Priority<p style="text-align: right;">
        <img class="png-icon" src="call1.jpg" style="width: 30px; height: 30px" >Contact Us-011-
2972929</p></p>
    </div>

    <div class="row">
        <input  type="text" class=" SensorID"  ng-model="search" placeholder="Enter your RoomNo" />
        <mg-button><img class="png-icon" src="search.png" style="width: 30px; height: 30px" ></mg-button>
    </div>
```

```
<div class="Alarms" style="background-color: black; background-image: url('fire3.jpg');background-repeat: no-
repeat;background-size: contain" ng-controller="AlarmController">
        <div class="BoxArrangements">
                <h2 >Sensor Alarms</h2>
            <ul ng-repeat="i in items | searchFor:search" ng-class="{active : isSelected(i)}">
                <div class="SensorBox" >
                    <span style="border-style: inset;border-color: white;border-width: 2px">
                          <p1 ><b>Sensor ID:{{-1}}</b><button ng-click="openModel(i)" class="btn btn-default"
style="background-color: darkred;color: white;float: right">View Status</button></p1><br>
                    </span>
                    <div style="text-align: center">
                           <br><b><p2>Floor No:{{i.floorNo}}</p2>
                           <br><p3>Room No:{{i.roomNo}}</p3></b>
                           <p style="color: orangered">Alarm :{{getStatusFromResponseEnum(i.status)}}</p>
                    </div>
                       <div style="color: red" ng-click="openRisk()" >{{getAlert(i)}}</div>
                </div>
            </ul>
        </div>
    <div style="font-family: Calibri,serif"><Strong>
        <modal  show="modalShown" width='400px' height='60%' >
            <h4 style="background-color: black;border: black;padding-bottom: 5%">Alarm Status<br>Readings of Room
No :{{Roomno}}</h4><br>

                <p style="color: black;text-align: left" >CO2 Level  :{{selected | json}}  </p><br>
                <p style="color: black;text-align: left">Smoke Level : {{selected2 |json}}  </p><br>
            <h5 style="text-align: left">Room Condition :{{Alert}}</h5><br><br>
        </modal></Strong>
    </div>
</div>
<script src="lib/angular/angular.js"></script>
<script src="lib/angular-route/angular-route.js"></script>
<!--Following codes is used to import the app and controller-->
<script src="app.js"></script>

<script src="MainController.js"></script>
<script src="core/version/version.js"></script>
<script src="core/version/version-directive.js"></script>
<script src="core/version/interpolate-filter.js"></script>
<script src="controllers/ModalController.js"></script>
<script src="Directives/modal.directive.js"></script>
</body>
</html>
```

7.2. Sensor application

getRequset.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.TimerTask;
import org.json.JSONObject;
public class GetRequest extends TimerTask{
        @Override
        public void run() {
                // TODO Auto-generated method stub
                try {
                        GetRequest.call_me();
                }catch(Exception e) {
                        e.printStackTrace();
                }
        }
        public static void call_me() throws Exception{

                URL urlForGetRequest = new URL("http://localhost:8080/getAll");
          String readLine = null;
          HttpURLConnection conection = (HttpURLConnection)     urlForGetRequest.openConnection();
          conection.setRequestMethod("GET");
          conection.setRequestProperty("userId", "a1bcdef"); // set userId its a sample here
          int responseCode = conection.getResponseCode();
          if (responseCode == HttpURLConnection.HTTP_OK) {
              BufferedReader in = new BufferedReader(
                  new InputStreamReader(conection.getInputStream()));
              StringBuffer response = new StringBuffer();
              String output = null;
              while ((readLine = in.readLine()) != null) {

              JSONObject myResponse = new JSONObject(readLine.toString());

                    int floorNo = myResponse.getInt("floorNo");
                    int roomNo = myResponse.getInt("roomNo");
                   System.out.println("");

                    PostRequest postRequest = new PostRequest();

                    postRequest.call_me(floorNo, roomNo);

                  response.append(readLine);
              } in .close();

           //print in String
           System.out.println(response.toString());
           //Read JSON response and print


}
```

Main.java

```
import java.util.Timer;
public class Main {
	public static void main(String[] args) {
		// TODO Auto-generated method stub


		Timer timer1 = new Timer();
		timer1.schedule(new GetRequest(), 0, 30000);
	}
}
```

postRequest.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.Reader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.util.LinkedHashMap;
import java.util.Map;
import org.json.JSONObject;
import java.util.Random;
import java.util.TimerTask;
import org.json.JSONObject;
public class PostRequest {
            public static void call_me(int floorNo,int roomNo) throws Exception {

                        Random rand = new Random();
                        int smoke1 = rand.nextInt(10);
      int co21 = rand.nextInt(10);



               URL url = new URL("http://localhost:8080/update");
               Map<String,Object> params = new LinkedHashMap<>();
               params.put("floorNo", floorNo);
               params.put("roomNo", roomNo);
               params.put("smoke", smoke1);
               params.put("CO2", co21);


               StringBuilder postData = new StringBuilder();
               for (Map.Entry<String,Object> params1 : params.entrySet()) {
                   if (postData.length() != 0) postData.append('&');
                   postData.append(URLEncoder.encode(params1.getKey(), "UTF-8"));
                   postData.append('=');
                   postData.append(URLEncoder.encode(String.valueOf(params1.getValue()), "UTF-8"));
               }
               byte[] postDataBytes = postData.toString().getBytes("UTF-8");
               HttpURLConnection conn = (HttpURLConnection)url.openConnection();
               conn.setRequestMethod("POST");
               conn.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
               conn.setRequestProperty("Content-Length", String.valueOf(postDataBytes.length));
               conn.setDoOutput(true);
               conn.getOutputStream().write(postDataBytes);
               Reader in = new BufferedReader(new InputStreamReader(conn.getInputStream(), "UTF-8"));
               StringBuilder sb = new StringBuilder();
               for (int c; (c = in.read()) >= 0;)
                   sb.append((char)c);
               String response = sb.toString();
               System.out.println(response);
```

7.3. REST API

Sensor control

```
package com.example.demo.controller;

import java.util.List;

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

import com.example.demo.model.Sensor;
import com.example.demo.service.SensorService;

@RestController
public class SensorController {

	@Autowired
	private SensorService sensorService;

	@RequestMapping("/create")
	public String create(@RequestParam int floorNo, @RequestParam int roomNo,@RequestParam int smoke, @RequestParam int
CO2,@RequestParam int Status) {
		Sensor s = sensorService.create(floorNo, roomNo, smoke, CO2,Status);
		return s.toString();
	}

	@RequestMapping("/get")
	public Sensor getSensor(int floorNo,int roomNo) {
		return sensorService.getByRoomNo(floorNo,roomNo);
	}

	@RequestMapping("/getFloor")
	public List<Sensor> getRooms(int floorNo) {
		return sensorService.getByFloorNo(floorNo);
	}

	@RequestMapping("/getAll")
	public List<Sensor> getAll(){
		return sensorService.getAll();
	}

	@RequestMapping("/update")
	public String update(@RequestParam int floorNo, @RequestParam int roomNo,@RequestParam int smoke, @RequestParam int
CO2,@RequestParam int Status) {
		Sensor s = sensorService.update(floorNo, roomNo, smoke, CO2, Status);
		return s.toString();
	}
	@RequestMapping("/delete")
	public String delete(int floorNo,int roomNo) {
		sensorService.delete(floorNo,roomNo);
		return "Deleted floor "+floorNo+" room "+roomNo+" sensor. ";
	}
	@RequestMapping("/deleteAll")
	public String deleteAll() {
		sensorService.deleteAll();
		return "Deleted all records.";
```

Sensor model

```
package com.example.demo.model;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

//sensor class
@Document
public class Sensor {
        @Id
        String id;
        int floorNo;
        int roomNo;
        int smoke;
        int co2;
        int Status;
        public int getStatus() {
                return Status;
        }
        public void setStatus(int status) {
                Status = status;
        }
        public Sensor(int floorNo, int roomNo, int smoke, int co2,int Status) {
                super();
                this.floorNo = floorNo;
                this.roomNo = roomNo;
                this.smoke = smoke;
                this.co2 = co2;
                this.Status=Status;
        }
        public int getFloorNo() {
                return floorNo;
        }
        public void setFloorNo(int floorNo) {
                this.floorNo = floorNo;
        }
        public int getRoomNo() {
                return roomNo;
        }
        public void setRoomNo(int roomNo) {
                this.roomNo = roomNo;
        }
        public int getSmoke() {
                return smoke;
        }
        public void setSmoke(int smoke) {
                this.smoke = smoke;
        }
        public int getCo2() {
                return co2;
        }
        public void setCo2(int co2) {
                this.co2 = co2;
        }
        public String toString() {
                return "Floor : "+floorNo+"Room : "+roomNo+"Smoke level: "+smoke+"CO2 level :"+co2+"Status :"+Status;
        }
```

sensorRepository

```
package com.example.demo.repository;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import com.example.demo.model.Sensor;

@Repository
public interface SensorRepository  extends MongoRepository<Sensor, String> {



			public List<Sensor> findByFloorNo(int floorNo);
			public Sensor findByRoomNo(int floorNo, int roomNo);



}
```

remoteRepository

```
package com.example.demo.repository;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import com.example.demo.model.Sensor;

@Repository
public interface SensorRepository  extends MongoRepository<Sensor, String> {



			public List<Sensor> findByFloorNo(int floorNo);
			public Sensor findByRoomNo(int floorNo, int roomNo);

}
```

sensorService

```
package com.example.demo.service;
import java.util.List;
import java.util.Optional;
import java.rmi.Remote;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Example;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Service;
import com.example.demo.model.Sensor;
import com.example.demo.repository.RemoteRepository;
import com.example.demo.repository.SensorRepository;

@Service
public class SensorService{
        @Autowired
        private SensorRepository sensorRepository = new SensorRepository() {
        @Override
                public <S extends Sensor> Optional<S> findOne(Example<S> example) {
                // TODO Auto-generated method stub
                return null;
}
        @Override
        public <S extends Sensor> Page<S> findAll(Example<S> example, Pageable pageable) {
        // TODO Auto-generated method stub
        return null;
}
        @Override
        public <S extends Sensor> boolean exists(Example<S> example) {
        // TODO Auto-generated method stub
        return false;
}
        @Override
        public <S extends Sensor> long count(Example<S> example) {
        // TODO Auto-generated method stub
        return 0;
        }
        @Override
        public <S extends Sensor> S save(S entity) {
                return null;
                }
        @Override
        public Optional<Sensor> findById(String id) {
                // TODO Auto-generated method stub
                return null;
}
        @Override
        public Iterable<Sensor> findAllById(Iterable<String> ids) {
        // TODO Auto-generated method stub
        return null;
        }
```

```
            @Override
            public boolean existsById(String id) {
            // TODO Auto-generated method stub
                        return false;
}
            @Override
            public void deleteById(String id) {
            // TODO Auto-generated method stub
                        }
            @Override
            public void deleteAll(Iterable<? extends Sensor> entities) {
            // TODO Auto-generated method stub
                        }
            @Override
            public void deleteAll() {
            // TODO Auto-generated method stub
                        }
            @Override
            public void delete(Sensor entity) {
            // TODO Auto-generated method stub
                        }
            @Override
            public long count() {
            // TODO Auto-generated method stub
                        return 0;
            }
            @Override
            public Page<Sensor> findAll(Pageable pageable) {
            // TODO Auto-generated method stub
                        return null;
}
            @Override
            public <S extends Sensor> List<S> saveAll(Iterable<S> entities) {
            // TODO Auto-generated method stub
                        return null;
}
            @Override
            public <S extends Sensor> List<S> insert(Iterable<S> entities) {
            // TODO Auto-generated method stub
                        return null;
            }
            @Override
            public <S extends Sensor> S insert(S entity) {
            // TODO Auto-generated method stub
                        return null;
}
            @Override
            public <S extends Sensor> List<S> findAll(Example<S> example, Sort sort) {
            // TODO Auto-generated method stub
                        return null;

            }
            @Override
            public <S extends Sensor> List<S> findAll(Example<S> example) {
            // TODO Auto-generated method stub
                        return null;
}
            @Override
            public List<Sensor> findAll(Sort sort) {
            // TODO Auto-generated method stub
                        return null;
}
            @Override
            public List<Sensor> findAll() {
            // TODO Auto-generated method stub
                        return null;
            }
            @Override
            public Sensor findByRoomNo(int floorNo, int roomNo) {
            // TODO Auto-generated method stub
                        return null;
            }
            @Override
            public List<Sensor> findByFloorNo(int floorNo) {
            // TODO Auto-generated method stub
                        return null;
}
};
            //create operations

            public Sensor create (int floorNo,int roomNo,int smoke,int co2,int Status) {
            Sensor s =new Sensor(floorNo, roomNo, smoke, co2, Status);
            System.out.println(s);
            return sensorRepository.save(s);
            }
            //retrieve operations

            public List<Sensor> getAll(){
            return sensorRepository.findAll();
            }
            public Sensor getByRoomNo(int floorNo,int roomNo) {
            return sensorRepository.findByRoomNo(floorNo,roomNo);


}
```

```
        //update operations

        public Sensor update (int floorNo,int roomNo,int smoke,int co2,int status) {
        Sensor s=sensorRepository.findByRoomNo(floorNo,roomNo);
        System.out.println(s);
        s.setSmoke(smoke);
        s.setCo2(co2);
        s.setStatus(status);
        return sensorRepository.save(s);
        }

        //delete operations

        public void deleteAll () {
        sensorRepository.deleteAll();
}
        public void delete(int floorNo,int roomNo) {
        Sensor s = sensorRepository.findByRoomNo(floorNo,roomNo);
        sensorRepository.delete(s);
}
        public List<Sensor> getByFloorNo(int floorNo) {
        // TODO Auto-generated method stub
                return  sensorRepository.findByFloorNo(floorNo);

}
}
```

REST application

```
package com.example.demo;

import com.example.demo.controller.SensorController;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SuppressWarnings("unused")
@SpringBootApplication
@ComponentScan(basePackages= {"com.example.demo.controller"})
public class RestApiApplication {

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

}
```

7.4. Desktop application

createRestService

```
package com.client;

import org.springframework.boot.CommandLineRunner;
import org.springframework.stereotype.Component;
import org.springframework.web.client.RestTemplate;

import com.model.Sensor;

@Component
public class createRestService{

	private static void createRestService()  {
		RestTemplate restTemplate = new RestTemplate();
		Sensor sensor1 = restTemplate.getForObject("http://localhost:8080/getFloor?floorNo=3", 
Sensor.class);
		System.out.print("Here is the smoke level"+sensor1.getSmoke());
		int smoke =sensor1.getSmoke();
		int co2 =sensor1.getCo2();
		System.out.println("Checkig errors");
	}




}
```

Details controller

```
package com.controller;

import java.util.List;
import java.util.Arrays;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.client.RestTemplate;

public class DetailsController {

        @Autowired
        private RestTemplate restTemplate;

        @GetMapping("/sensors")
        public List<Object> getSensor (){
                String url = "http://localhost:8080/getFloor?floorNo=3";
                Object[] objects = restTemplate.getForObject(url, Object[].class) ;
                System.out.println(objects);
                return Arrays.asList(objects);
        }

}
```

RestApiDesktopApplication

```
package com.example.demo;

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.core.ParameterizedTypeReference;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.web.client.RestTemplate;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.model.Sensor;


@SpringBootApplication
public class RestApiDsDesktopApplication {

	public RestTemplate getRestTemplate() {
		System.out.println("startded restTemp...");
		return new RestTemplate();
	}

	public static void main(String[] args) {
		SpringApplication.run(RestApiDsDesktopApplication.class, args);
		System.out.println("startded...");

		/*
		ObjectMapper mapper = new ObjectMapper();
		try {
			Sensor sensor = mapper.readValue(new URL("http://localhost:8080/getFloor?floorNo=3"),
Sensor.class);
			System.out.println("Json values CO2 value :"+sensor.getCo2());
		} catch (IOException e1) {
			// TODO Auto-generated catch block
			e1.printStackTrace();
		}

		*/

		Sensor sensor;
		RestTemplate restTemplate = new RestTemplate();
		ResponseEntity<List<Sensor>> rateResponse
=restTemplate.exchange("http://localhost:8080/getFloor?floorNo=3", HttpMethod.GET, null, new
ParameterizedTypeReference<List<Sensor>>() {});
		List<Sensor> rates = rateResponse.getBody();
		System.out.println(rates);

		System.out.println("finished...");
```

Sensor model class

```
package com.model;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement
public class Sensor {

        String id;
        int floorNo;
        int roomNo;
        int smoke;
        int co2;
        int Status;
        public int getStatus() {
                return Status;
        }
        public void setStatus(int status) {
                Status = status;
        }
        public Sensor(int floorNo, int roomNo, int smoke, int co2,int Status) {
                super();
                this.floorNo = floorNo;
                this.roomNo = roomNo;
                this.smoke = smoke;
                this.co2 = co2;
                this.Status=Status;
        }
        public int getFloorNo() {
                return floorNo;
        }
        public void setFloorNo(int floorNo) {
                this.floorNo = floorNo;
        }
        public int getRoomNo() {
                return roomNo;
        }
        public void setRoomNo(int roomNo) {
                this.roomNo = roomNo;
        }
        public int getSmoke() {
                return smoke;
        }
        public void setSmoke(int smoke) {
                this.smoke = smoke;

        }
        public int getCo2() {
                return co2;
        }

        public void setCo2(int co2) {

                this.co2 = co2;

        }
        public String toString() {
                return "Floor : "+floorNo+"Room : "+roomNo+"Smoke level: "+smoke+"CO2 level :"+co2+"Status :"+Status;
        }
```

7.5. RMI

Client

```
package com.example.demo.rmi;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.*;

import com.example.demo.model.Sensor;
import com.example.demo.repository.SensorRepository;

public class Client {
        private Client() {}
          public static void main(String[] args)throws Exception {
            try {
              // Getting the registry
              Registry registry = LocateRegistry.getRegistry(null);

              // Looking up the registry for the remote object
              SensorRepository stub = (SensorRepository) registry.lookup("Hello");

              // Calling the remote method using the obtained object
              Scanner sc =new Scanner (System.in);
              int FNO = sc.nextInt();
              List<Sensor> list = (List)stub.findByFloorNo(FNO);

              for (Sensor s:list) {

                System.out.println("Floor: " + s.getFloorNo());
                System.out.println("Room: " + s.getRoomNo());
                System.out.println("Smoke: " + s.getSmoke());
                System.out.println("CO2: " + s.getCo2());
                System.out.println("Status: " + s.getStatus());
              }
            // System.out.println(list);
            } catch (Exception e) {
              System.err.println("Client exception: " + e.toString());
              e.printStackTrace();
            }
          }
```

RegisterClient

```
package com.example.demo.rmi;

import java.net.Inet4Address;
import java.net.UnknownHostException;
import org.springframework.beans.factory.UnsatisfiedDependencyException;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.remoting.rmi.RmiProxyFactoryBean;

import com.example.demo.service.SensorService;

@Configuration
public class RegisterClient {

        @Bean
        public SensorService sensorSenService() {
                return new SensorService();

        }

        //@Bean
          public RmiProxyFactoryBean exporter() throws UnknownHostException {
              RmiProxyFactoryBean rpfb = new RmiProxyFactoryBean();
              rpfb.setServiceInterface(SensorService.class);
              String hostAddress = Inet4Address.getLocalHost()
                                .getHostAddress();
              rpfb.setServiceUrl(String.format("rmi://%s:2099/OrderService", hostAddress));
              return rpfb;
          }

          public static void main(String[] args) {
              AnnotationConfigApplicationContext context =
                      new AnnotationConfigApplicationContext(SensorClient.class);
              SensorBean bean = context.getBean(SensorBean.class);
              bean.clientRun();

          }
}
```

Server

```
package com.example.demo.rmi;

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.UnicastRemoteObject;

import org.springframework.beans.factory.annotation.Autowired;

import java.net.ConnectException;
import java.rmi.Remote;


import com.example.demo.repository.RemoteRepository;
import com.example.demo.repository.SensorRepository;
import com.example.demo.service.SensorService;
import java.rmi.RemoteException;
public class Server extends SensorService{

        public Server() {}

        @Autowired
        public static void main(String args[]) throws ConnectException {
            try {
              // Instantiating the implementation class
              SensorService obj = new SensorService();

              // Exporting the object of implementation class (here we are exporting the remote object to the stub)
//            RemoteRepository stub = (RemoteRepository) UnicastRemoteObject.exportObject(obj, 0);

              // Binding the remote object (stub) in the registry
              Registry registry = LocateRegistry.getRegistry();
             // System.setProperty("java.rmi.server.hostname","192.168.43.163");
//            registry.bind("Hello",(Remote) stub);

              System.err.println("Server ready");
            } catch (Exception e) {
              System.err.println("Server exception: " + e.toString());
              e.printStackTrace();
            }
          }

}
```

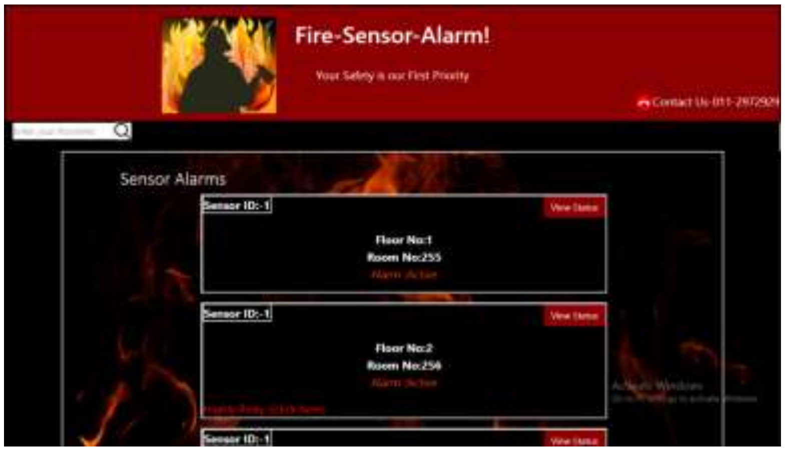
SensorBean

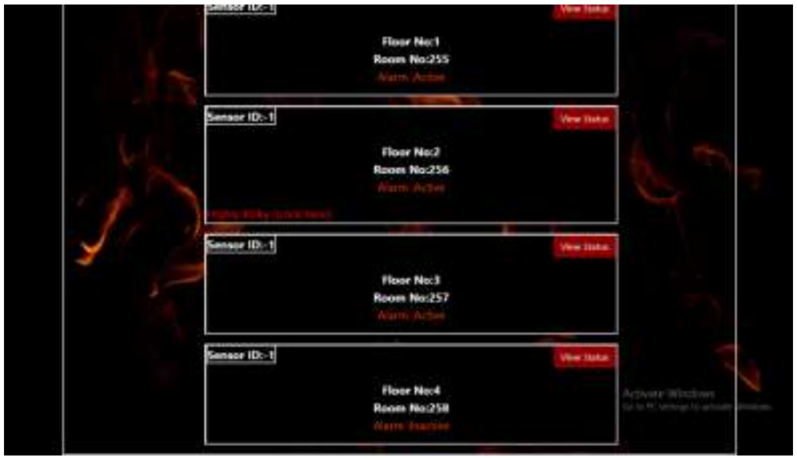
```
package com.example.demo.rmi;
import org.springframework.beans.factory.annotation.Autowired;
import com.example.demo.service.SensorService;

public class SensorBean {

        @Autowired
          private SensorService sensorService;

        public void clientRun() {
                System.out.print("Client Running");
        }

}
```

SensorFacade

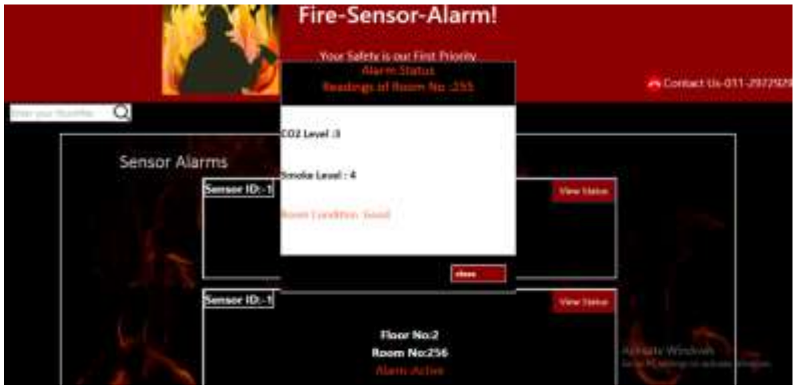
```
package com.example.demo.rmi;

import java.rmi.Remote;
import java.util.List;
import org.springframework.stereotype.Repository;
import com.example.demo.model.Sensor;

public interface SensorFacade extends Remote {

        public List<Sensor> findByFloorNo(int floorNo);
        public Sensor findByRoomNo(int floorNo, int roomNo);


}
```

Web application interfaces

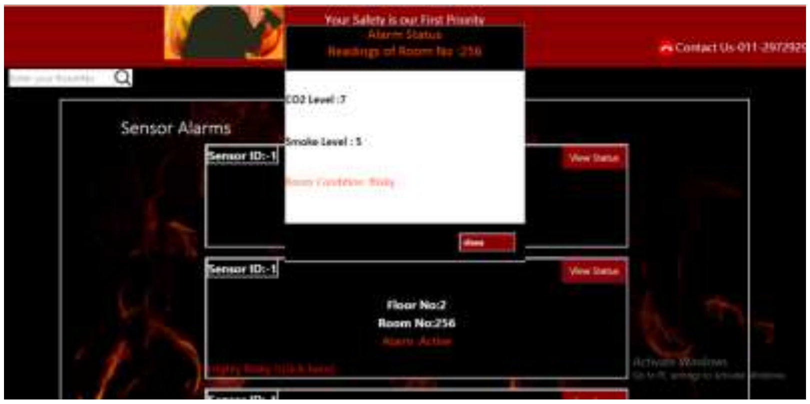




When status normal



When status become greater than 5


Alarm Status

Readings of Room No :256

CO2 Level :7

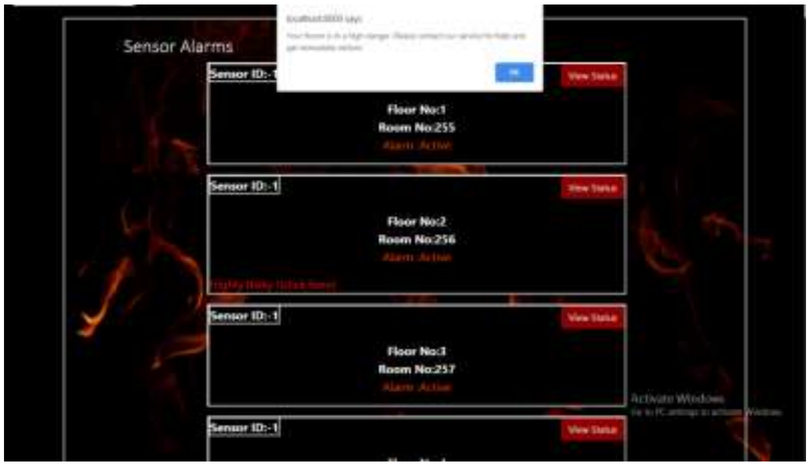
Smoke Level : 5

Room Condition: Risky

Floor No:2
Room No:256
Alarm Active

Highly Risky !!click here!

Notify user status about the risk status


localhost:3000 says

Your Room is in a high danger. Please contact our service for help and get immediate actions

Floor No:1
Room No:255
Alarm Active

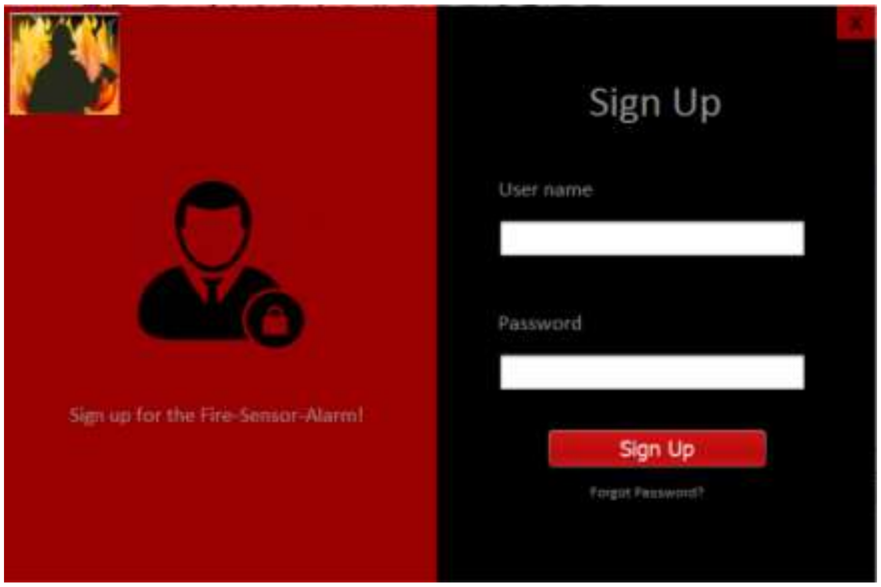
Floor No:2
Room No:256
Alarm Active

Highly Risky !!click here!

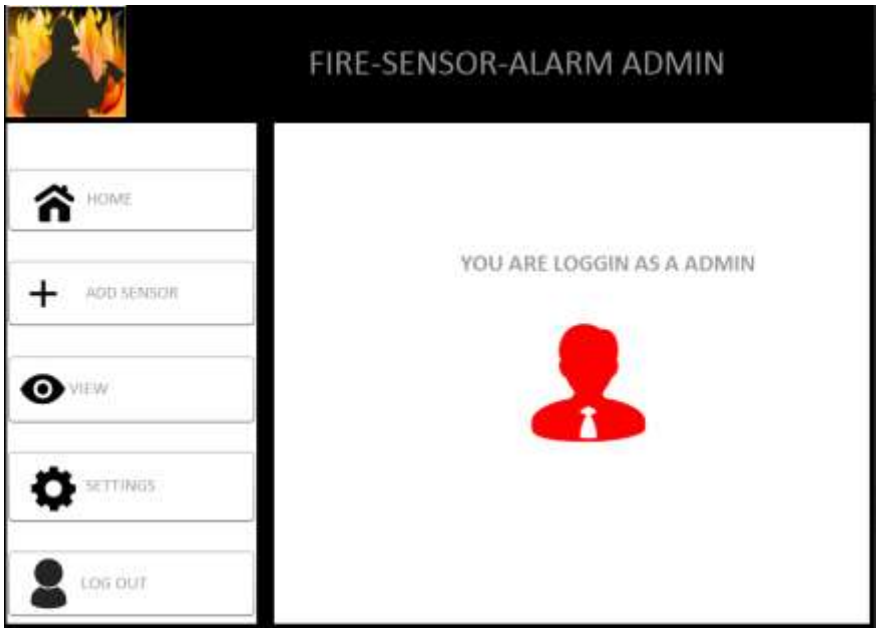
Floor No:3
Room No:257
Alarm Active

Desktop application interfaces

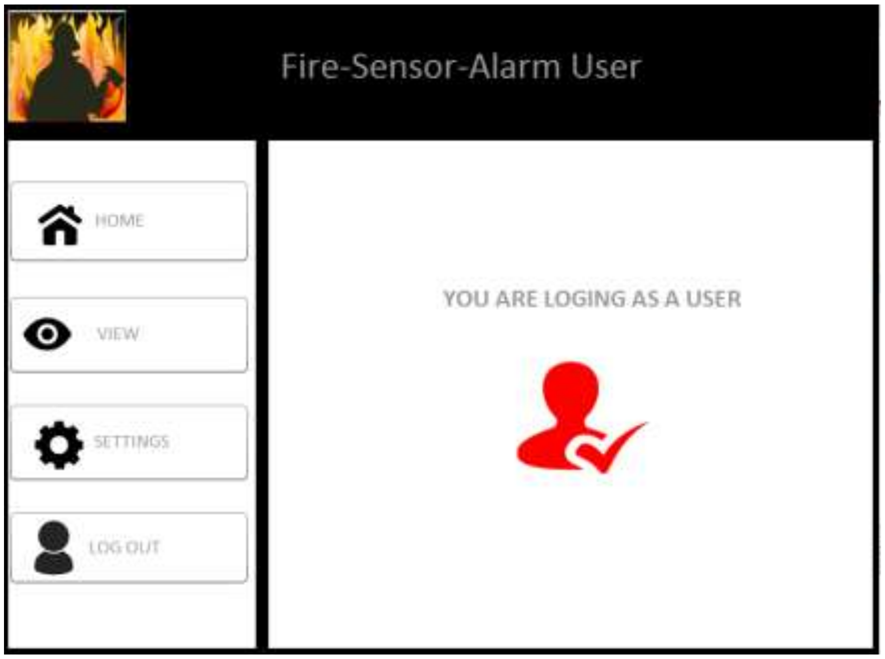
Login



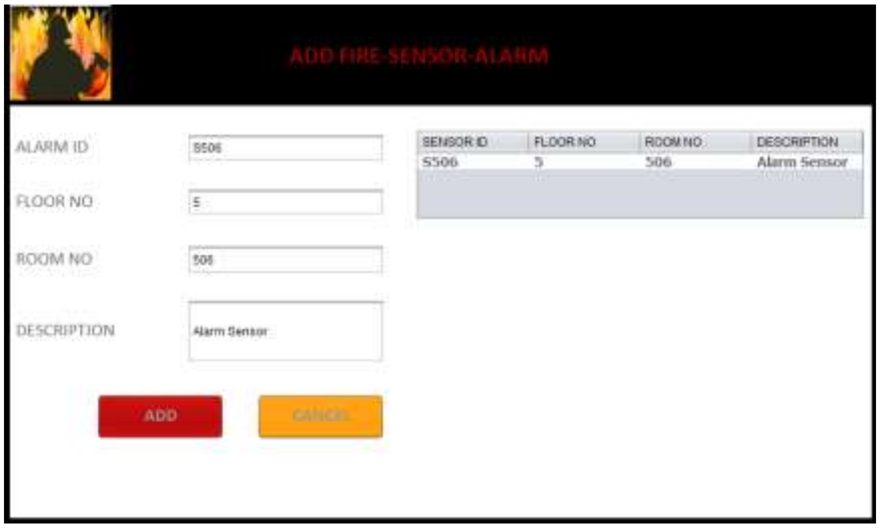
Admin home page



User home page



Admin add sensor details



ALARM ID	S506
FLOOR NO	5
ROOM NO	506
DESCRIPTION	Alarm Sensor

SENSOR ID	FLOOR NO	ROOM NO	DESCRIPTION
S506	5	506	Alarm Sensor