BSc (Hons) in Information Technology

Assignment 2

SE3020 – Distributed System

Year 3, Semester I, 2020

Fire Alarm Monitoring System

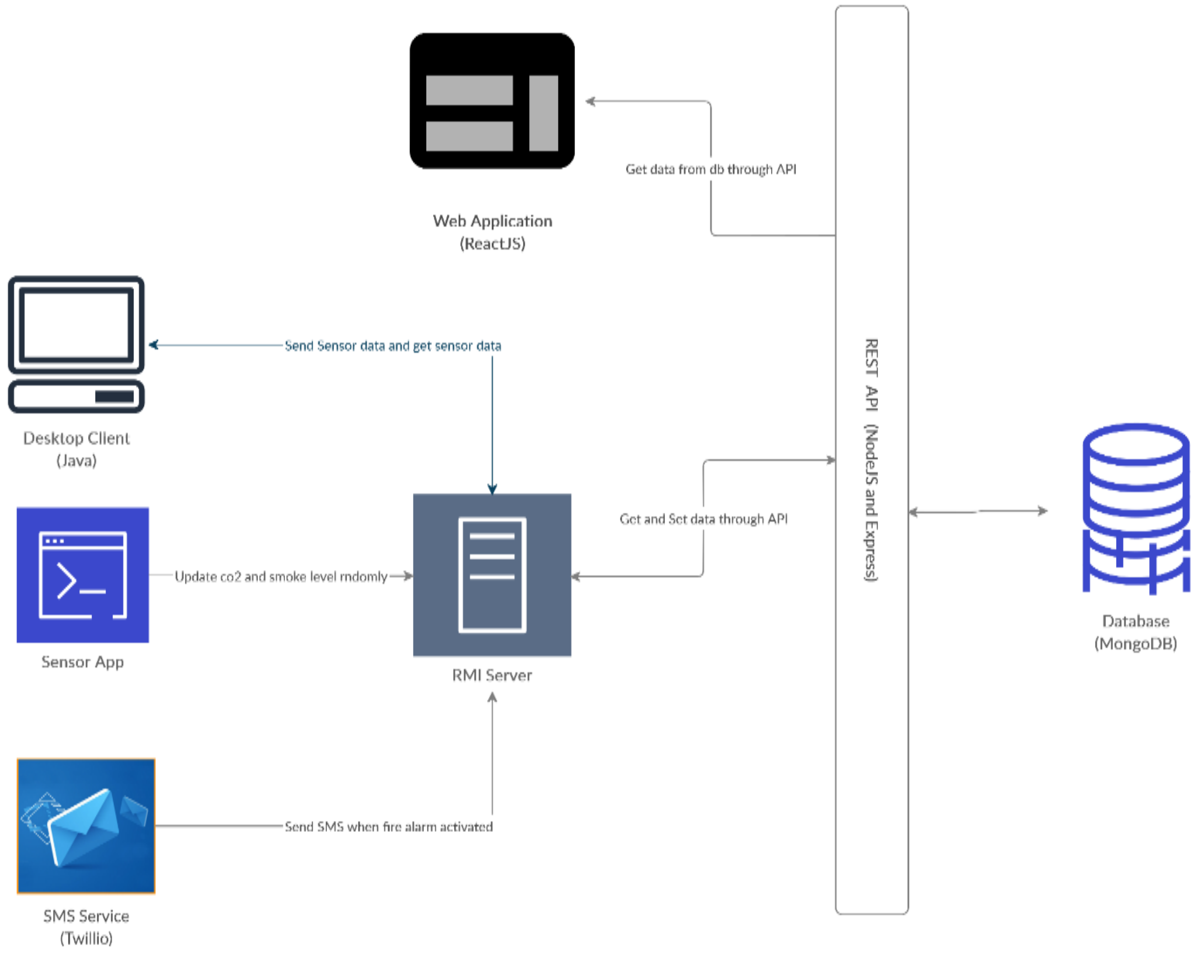
Assignment 2

Distributed System

Registration Number	Name
IT18001808	S.W.P.N.M.Weerasinghe (Leader)
IT18012866	L.G.I.Sathsara
IT17084796	H.S.D.N.Gunasekara
IT18066944	R.M.D.D.Rajapaksha

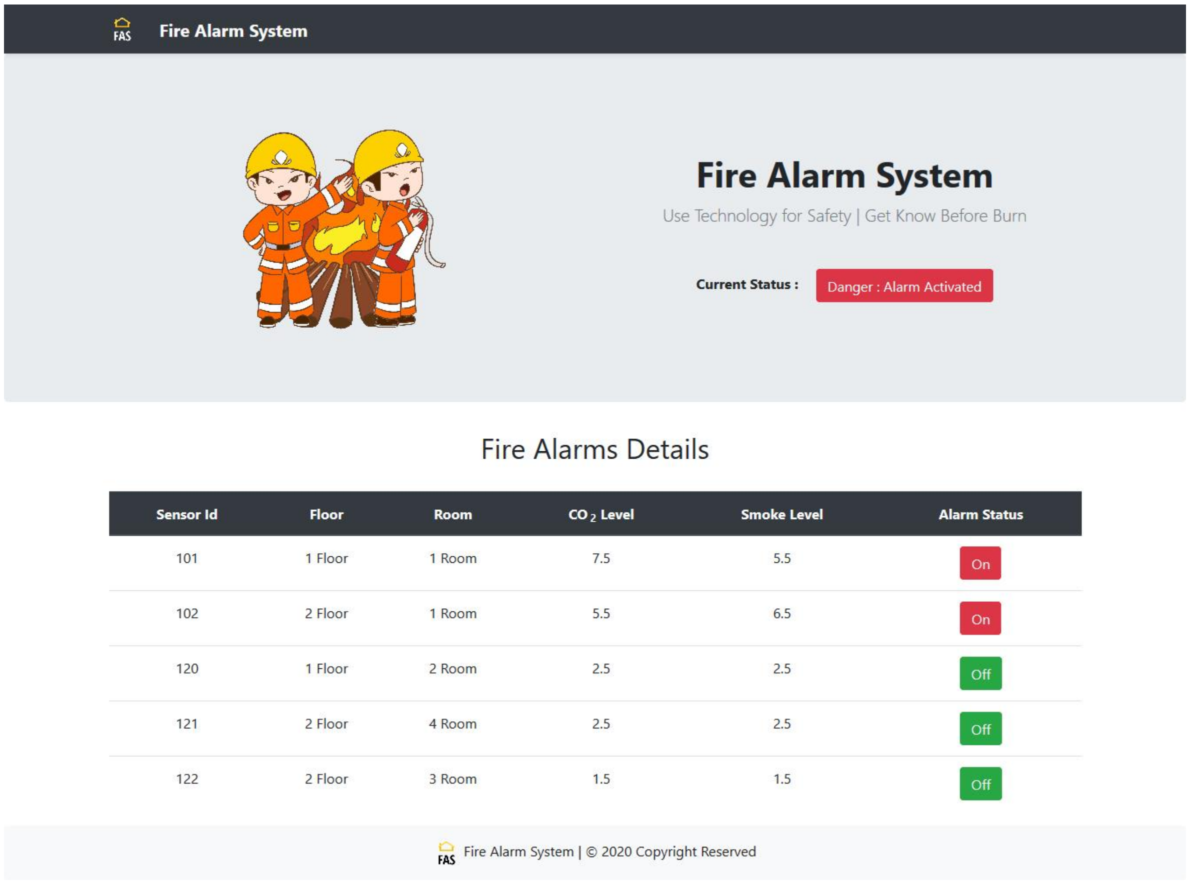
1) High Level Architectural Diagram

Fire Alarm Monitoring System contains REST API, RMI server and client application, Web client and a database. REST API has implemented using Node Js and Express Js and it handles the http requests which came from the web client and also it handles http requests which came from the RMI server. Web client has implemented using React Js and it displays the details of fire alarms. Web client send http requests to REST API and REST API response with the details which obtained from the database. RMI server and client is a desktop application which use to monitor the fire alarm sensors. Desktop client can send http requests through RMI server and those requests are handled by REST API. The RMI sensor checks API from 15 seconds and update the desktop client. If gas levels have increased, the RMI sensor sends a SMS alert. Desktop client can add or edit sensor details. A dummy app called Sensor App has used to simulate the behaviors of the sensors.


Web Application
(ReactJS)
Get data from db through API
Send Sensor data and get sensor data
Desktop Client
(Java)
REST API (NodeJS and Express)
Get and Set data through API
Update co2 and smoke level rndomly
Sensor App
RMI Server
Database
(MongoDB)
Send SMS when fire alarm activated
SMS Service
(Twillio)

2) System Interfaces

I. Web Application



Fire Alarm System

Fire Alarm System
Use Technology for Safety | Get Know Before Burn

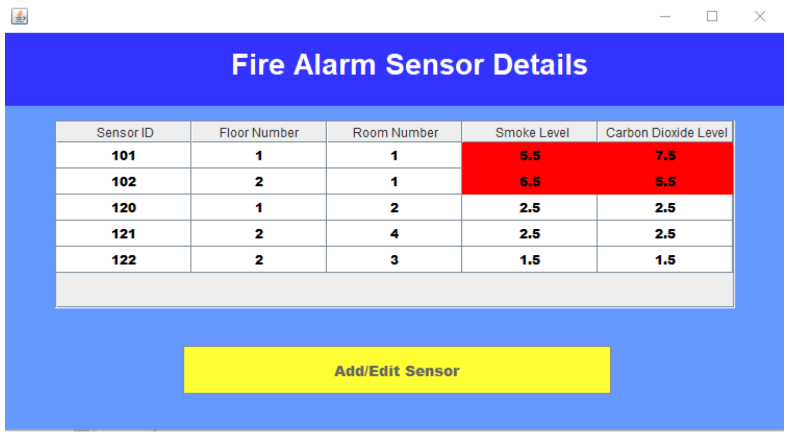
Current Status : Danger : Alarm Activated

Fire Alarms Details

Sensor Id	Floor	Room	CO_2 Level	Smoke Level	Alarm Status
101	1 Floor	1 Room	7.5	5.5	On
102	2 Floor	1 Room	5.5	6.5	On
120	1 Floor	2 Room	2.5	2.5	Off
121	2 Floor	4 Room	2.5	2.5	Off
122	2 Floor	3 Room	1.5	1.5	Off

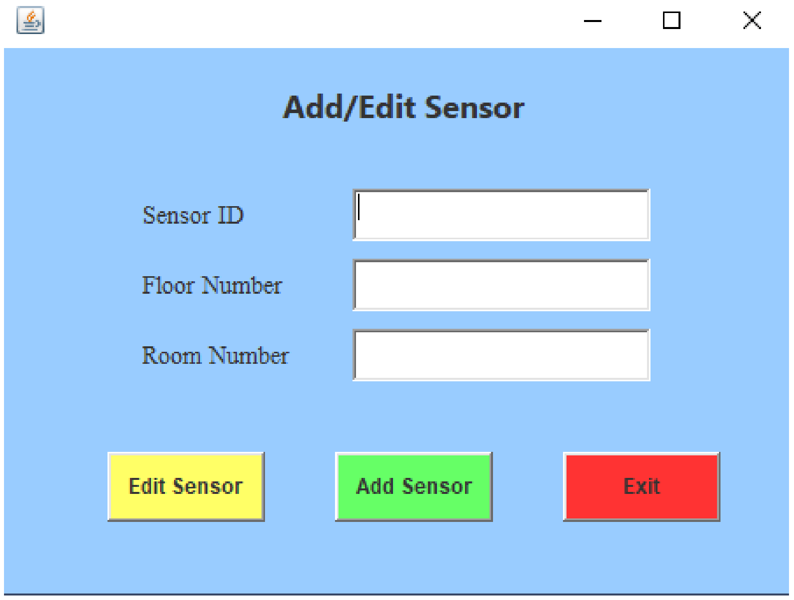
Fire Alarm System | © 2020 Copyright Reserved

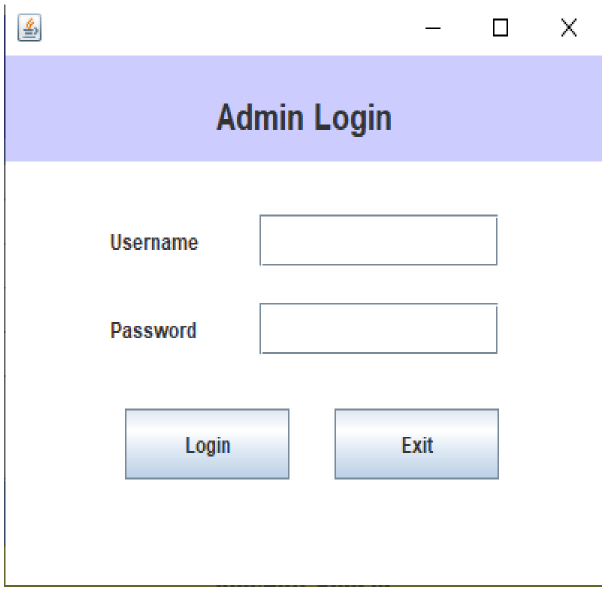
II. Desktop Application



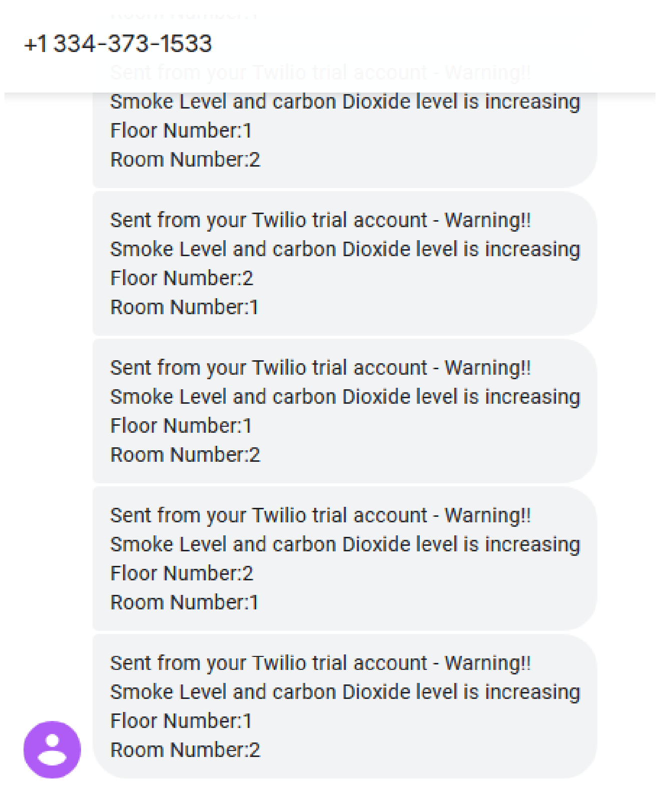
Fire Alarm Sensor Details

Sensor ID	Floor Number	Room Number	Smoke Level	Carbon Dioxide Level
101	1	1	5.5	7.5
102	2	1	6.5	5.5
120	1	2	2.5	2.5
121	2	4	2.5	2.5
122	2	3	1.5	1.5





III) Send SMS to users when smoke and co2 get increase



+1 334-373-1533

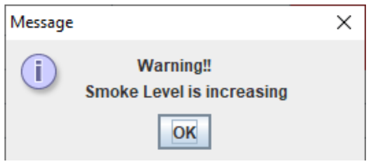
Smoke Level and carbon Dioxide level is increasing
Floor Number:1
Room Number:2

Sent from your Twilio trial account - Warning!!
Smoke Level and carbon Dioxide level is increasing
Floor Number:2
Room Number:1

Sent from your Twilio trial account - Warning!!
Smoke Level and carbon Dioxide level is increasing
Floor Number:1
Room Number:2

Sent from your Twilio trial account - Warning!!
Smoke Level and carbon Dioxide level is increasing
Floor Number:2
Room Number:1

Sent from your Twilio trial account - Warning!!
Smoke Level and carbon Dioxide level is increasing
Floor Number:1
Room Number:2



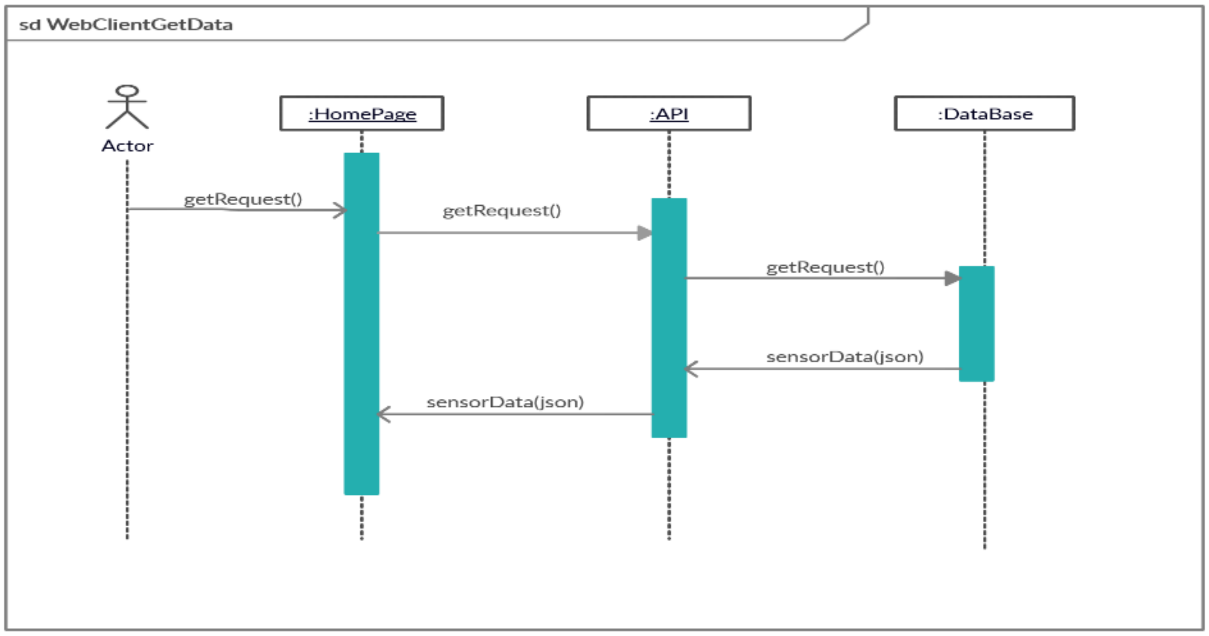
Message

Warning!!
Smoke Level is increasing

3) System Workflow

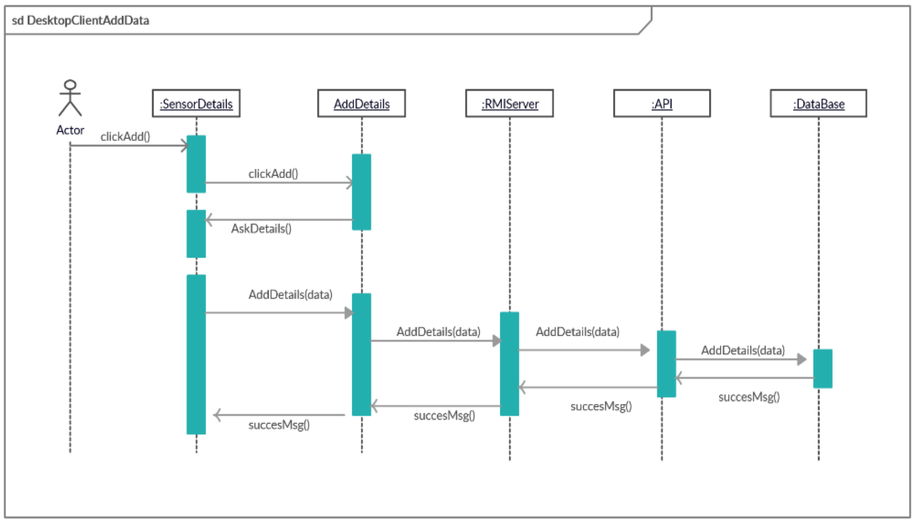
I. Web Application

Web client sends http get() requests to the API and API handles that request by retrieving sensor details from the database and send http response to the client in json format.



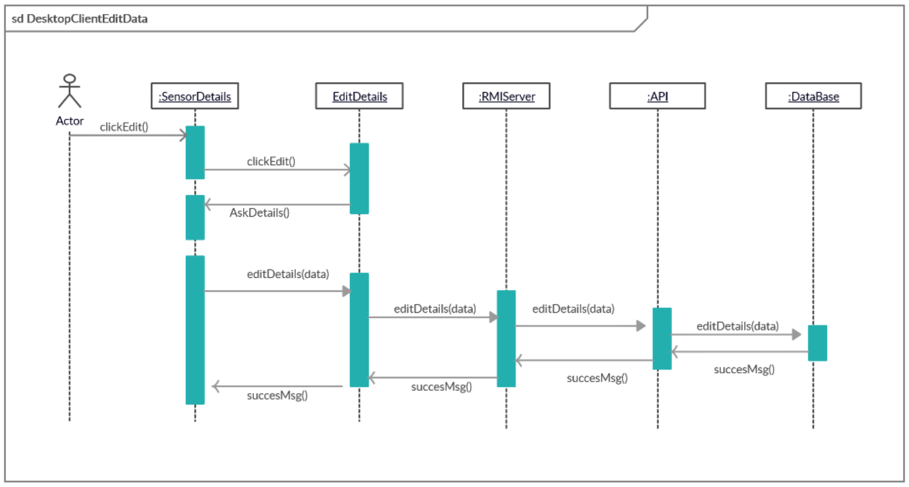
II. Desktop Application – Add Sensor

New sensors can register to the system through desktop client. RMI server send http post() request to the API and it stores the new sensor in the database. After that API sends an success msg to the client through RMI server.



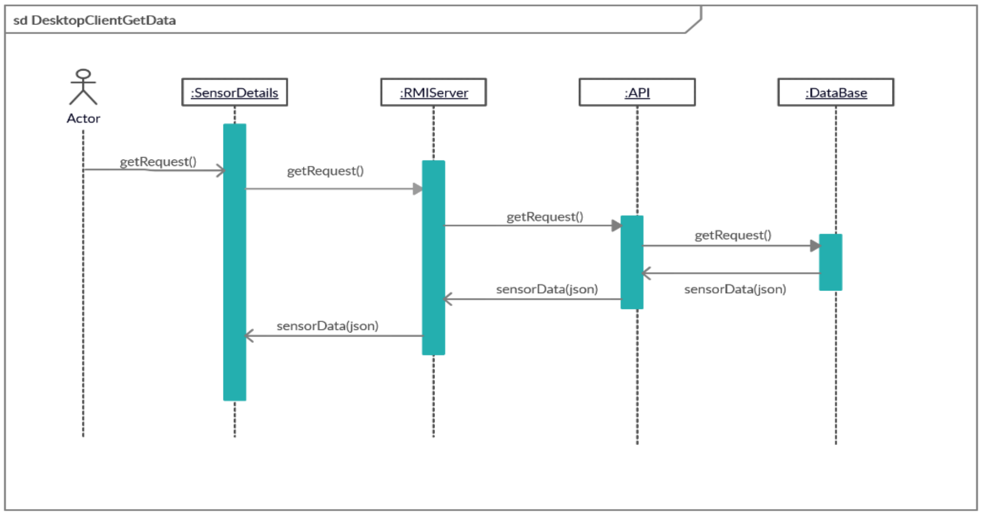
III. Desktop Application – Edit Sensor

Sensor details can edit through RMI server. RMI server send http request to the API and it stores the new details of edited sensor in the database. After that API sends an success msg to the client through RMI server.



IV. Desktop Application – Get Data

RMI server send http get() request to the API and API retrieve the sensor details from the database and sends them to the RMI server and desktop client displays the details in a table which retrieve from the server.



4) Appendix

API – app.js

```
const express = require('express');
const app = express();
const mongoose = require('mongoose');
const bodyparser = require('body-parser');
const cors = require('cors');
require('dotenv/config');


//import routes
const sensorRoute = require('./Routes/Sensors');
const userRoute = require('./Routes/Users');

//middleware
app.use(bodyparser.json());
app.use(cors());
app.use('/sensors',sensorRoute);
app.use('/users',userRoute);

//db
mongoose.connect(
    process.env.DB_CONNECTION,
    {useNewUrlParser:true,useUnifiedTopology: true},
    ()=>console.log('Database connected!')
)

app.listen(4000);
```

API – Models / SensorDetails.js

```
const mongoose = require('mongoose');

const PostSchema = mongoose.Schema({
    sensorid:{type:Number,required:true},
    floor:{type:Number,required:true},
    room:{type:Number,required:true},
    colevel:{type:Number,default:0},
    smokelevel:{type:Number,default:0}
});

module.exports=mongoose.model('SensorDetails',PostSchema);
```

API – Models / UserDetails.js

```
const mongoose = require('mongoose');

const UserSchema = mongoose.Schema({
    userid:{type:Number,required:true},
    username:{type:String,required:true},
    password:{type:String,required:true}

});

module.exports=mongoose.model('UserDetails',UserSchema);
```

API – Routes / Sensors.js

```
const express = require('express');
const router = express.Router();
const SensorDetails = require('../Models/SensorDetails');

//get all sensor
router.get('/',async (req,res)=>{
    try{
        const sensors = await SensorDetails.find();
        res.json(sensors);
    }catch(err){
        res.json({message:err})
    }
})

//submit sensor details
router.post('/',(req,res)=>{
    const sensor = new SensorDetails({
        sensorid:req.body.sensorid,
        floor:req.body.floor,
        room:req.body.room,
        colevel:req.body.colevel,
        smokelevel:req.body.smokelevel
    });

    console.log(req.body);
    sensor.save()
    .then(data=>{
        res.json(data);
    })
    .catch(err=>{
        res.json({message:err})
    })
})

//specific sensor details
router.get('/:sensorid',async (req,res)=>{
    try{
        const specificSensor = await SensorDetails.findById(req.params.sensorid);
        res.json(specificSensor);
    }catch(err){
        res.json({message:err});
    }
})
//delete sensor
router.delete('/:sensorid',async (req,res)=>{
    try{
        const deleteSensor =await SensorDetails.remove({_id: req.params.sensorid});
        res.json(deleteSensor);
    }catch(err){
        res.json({message:err});
    }
})
//update sensor
router.post('/:sensorid',async (req,res)=>{
    try{
        const updateSensor =await SensorDetails.updateOne({_id:req.params.sensorid},
            {$set : {sensorid:req.body.sensorid,
                    floor:req.body.floor,
                    room:req.body.room,
                    colevel:req.body.colevel,
                    smokelevel:req.body.smokelevel}});

            res.json(updateSensor);


    }catch(err){
        res.json({message:err});
    }
})

module.exports=router;
```

API – Routes / Users.js

```
const express = require('express');
const router = express.Router();
const UserDetails = require('../Models/UserDetails');
//get all users
router.get('/',async (req,res)=>{
    try{
        const users = await UserDetails.find();
        res.json(users);
    }catch(err){
        res.json({message:err})
    }
})
//submit user details
router.post('/',(req,res)=>{
    const user = new UserDetails({
        userid:req.body.userid,
        username:req.body.username,
        password:req.body.password
    });

    console.log(req.body);

    user.save()
    .then(data=>{
        res.json(data);
    })
    .catch(err=>{
        res.json({message:err})
    })
})
//view users
router.get('/:userid',async (req,res)=>{
    try{
        const specificUser = await UserDetails.findById(req.params.userid);
        res.json(specificUser);
    }catch(err){
        res.json({message:err});
    }
})

module.exports=router;
```

Web Application – app.js

```
import React,{useEffect,useState} from 'react';
import Navbar from './component/Navbar/Navbar'
import MainContent from './component/MainContent/MainContent'
import SensorDetails from './component/SensorDetails/SensorDetails'
import Footer from './component/Footer/Footer'

const App = ()=> {

  const[sensors,setSensors]= useState([]);

  useEffect(()=>{

    setInterval(function(){
      getSensors();
    },40000);

  },[]);

  const getSensors = async ()=>{
    const response = await fetch('http://localhost:4000/sensors');
    const data = await response.json();
    setSensors(data);
  }

  return (
    <div className="App">
      <Navbar />
      <MainContent sensorDetail={sensors} key={sensors._id}/>
      <SensorDetails sensorDetail={sensors} key={sensors._id}/>
      <Footer />
    </div>
  );
}

export default App;
```

Web Application – SensorDetails.js

```
import React from 'react'

const SensorDetails = ({sensorDetail}) => {
    //"btn btn-danger"
        return (
            <div className="container">
                <center><h2>Fire Alarms Details</h2></center><br />
                <table className="table table-hover border-rounded mb-
3" style={{textAlign:'center'}}>
                    <thead className="thead-dark">
                    <tr>
                        <th scope="col">Sensor Id</th>
                        <th scope="col">Floor</th>
                        <th scope="col">Room</th>
                        <th scope="col">CO<sub> 2</sub> Level</th>
                        <th scope="col">Smoke Level</th>
                        <th scope="col">Alarm Status</th>
                    </tr>
                    </thead>
                    <tbody>
                        {sensorDetail.map(sensor=>(
                            <tr>
                                <td>{sensor.sensorid}</td>
                                <td>{sensor.floor} Floor</td>
                                <td>{sensor.room} Room</td>
                                <td>{sensor.colevel}</td>
                                <td>{sensor.smokelevel}</td>
                                <td><button className={(sensor.colevel>5 || sensor.smokelevel
>5)?"btn btn-danger":"btn btn-success"}>
                                {(sensor.colevel>5 || sensor.smokelevel>5)?"On":"Off"}
                                </button></td>
                            </tr>
                        ))}
                    </tbody>
                 </table>
            </div>
        )

}

export default SensorDetails;
```

Web Application – MainContent.js

```
import React from 'react';
import logo from './fire.png'

const MainContent = ({sensorDetail}) => {

        //check one of fire active
        var status = false;

        {sensorDetail.map(sensor=>{
            if(sensor.colevel>5 || sensor.smokelevel>5){
                status = true;
            }
        })}

        return (
            <div>
                <main role="main" >
                    <section className="jumbotron text-center"  style={{height:'100%'}}>
                        <div className="container">
                        <div className="row">
                            <div className="col-sm">
                                <img src={logo} style={{width:'50%'}} />
                            </div>
                            <div className="col-lg">
                            <br />
                            <br />
                            <h1><b>Fire Alarm System</b></h1>
                            <p className="lead text-
muted">Use Technology for Safety | Get Know Before Burn</p>
                            <br />
                            <p>
                            <b>Current Status : </b>
                            <a href="#" className={(status)?"btn btn-danger ml-3 my-
2":"btn btn-success ml-3 my-2"}>
                            {(status)?"Danger : Alarm Activated":"Normal"}
                            </a>
                            </p>
                            </div>
                            </div>
                        </div>
                    </section>
                </main>
            </div>
        )

}

export default MainContent;
```

Desktop App – SensorApp.java

```
public class sensorApp implements  Runnable{

    static int num;
    static ArrayList<Sensor> sensorList = new ArrayList<>() ;
    @Override
    public void run() {

        //---generate random number-----
        Random r = new Random();


         for(;;) {

            try {

                Thread.sleep(10000);
            }
            catch(InterruptedException ie) {

            }

             this.updateArrayList();
             num = r.nextInt();

             for(Sensor s:sensorList) {

                 double smoke = s.getSmokeLevel();
                 double cd = s.getCdLevel();
                 //-----
if generated number > 0, smoke level and cd level of all the sensors will decrease by 0.2---
                 if(num > 0) {

                     smoke = smoke - 0.5;
                     cd = cd - 0.5;

                     if(smoke > 0 && cd >0) {

                         s.setCdLevel(cd);
                         s.setSmokeLevel(smoke);
                     }
                 }else if(num < 0) {

               //-----
if generated number > 0, smoke level and cd level of all the sensors will increase by 0.2
                    smoke = smoke + 0.5;
                     cd = cd + 0.5;

                     s.setCdLevel(cd);
                     s.setSmokeLevel(smoke);
                 }
             }

            try {

                this.updateAPI(sensorList);
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }
    }
    public static void main(String[] args) throws ParseException, JsonProcessingException, IO
Exception {


        sensorApp sp = new sensorApp();
        Thread thread = new Thread(sp);

        thread.start();
    }

    //----this method updates the API with the changes of the gas levels------
    public void updateAPI(ArrayList<Sensor> sensorList) throws IOException {


        for(Sensor s:sensorList) {

        final String REQ_BODY = "{\n" +
                "\"sensorid\":" +s.id+",\r\n" +
                "    \"floor\":"+s.floorNo+",\r\n" +
                "    \"colevel\":"+ s.cdLevel+",\r\n" +
                "    \"smokelevel\":"+ s.smokeLevel+",\r\n" +
                "    \"room\":"+s.RoomNo+"" +
                "\n}";



            URL obj = new URL("http://localhost:4000/sensors/"+s._id);
            HttpURLConnection postConnection = (HttpURLConnection) obj.openConnection();
            postConnection.setRequestMethod("POST");
            postConnection.setRequestProperty("Content-Type", "application/json");
            postConnection.setDoOutput(true);
            OutputStream os = postConnection.getOutputStream();
            os.write(REQ_BODY.getBytes());
            os.flush();
            os.close();

            int responseCode = postConnection.getResponseCode();
            System.out.println("POST Response Code :  " + responseCode);
            System.out.println("POST Response Message : " + postConnection.getResponseMessage
());
            if (responseCode == HttpURLConnection.HTTP_CREATED) { //success
                BufferedReader in = new BufferedReader(new InputStreamReader(
                    postConnection.getInputStream()));
                String inputLine;
                StringBuffer response = new StringBuffer();
                while ((inputLine = in .readLine()) != null) {
                    response.append(inputLine);
                } in .close();
                // print result
                System.out.println(response.toString());
            } else {
                System.out.println(" ");
            }
        }

    }

    //-----This method updates the arraylist according to the changes of the database------
    public void updateArrayList() {


                //----get response from api---------------

                Client client = ClientBuilder.newClient();
                WebTarget target = client.target("http://localhost:4000/sensors");

                //----parse response to JSON Object----------
                JSONParser parser = new JSONParser();
                Object obj;





                try {
                     obj = parser.parse(target.request(MediaType.TEXT_XML).get(String.class))
;
                     JSONArray array = (JSONArray)obj;

                     //-----Iterate through Json array and update ArryList-----------
                     for(int i = 0 ;i<array.size();++i) {
                         JSONObject obj2 = (JSONObject)array.get(i);
                         Sensor sensor = new Sensor();
                         sensor.set_id(obj2.get("_id").toString());
                         sensor.setId(obj2.get("sensorid").toString());
                         sensor.setFloorNo(new Integer(obj2.get("floor").toString()));
                         sensor.setRoomNo(obj2.get("room").toString());
                         sensor.setSmokeLevel(new Double( obj2.get("smokelevel").toString()))
;
                         sensor.setCdLevel(new Double( obj2.get("colevel").toString()));

                         sensorList.add(sensor);

                     }
                } catch (ParseException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

    }


}
```

Desktop App – Sensor.java

```
import java.io.Serializable;

public class Sensor implements Serializable {

   public String id;
   public String _id;
   public int floorNo;
   public String RoomNo;
   public double smokeLevel;
   public double cdLevel;

   public Sensor() {

   }


   public Sensor(String id,int floorNo, String roomNo, double smokeLevel, double cdLevel) {
    super();
    this.id = id;
    this.floorNo = floorNo;
    this.RoomNo = roomNo;
    this.smokeLevel = smokeLevel;
    this.cdLevel = cdLevel;
}


   public String get_id() {
        return _id;
    }
    public void set_id(String _id) {
        this._id = _id;
    }

   public String getId() {
        return id;
    }


    public void setId(String id) {
        this.id = id;
    }

public int getFloorNo() {
    return floorNo;
}


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


public String getRoomNo() {
    return RoomNo;
}


public void setRoomNo(String roomNo) {
    RoomNo = roomNo;
}


public double getSmokeLevel() {
    return smokeLevel;
}


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

```
public double getCdLevel() {
    return cdLevel;
}


public void setCdLevel(double cdLevel) {
    this.cdLevel = cdLevel;
}




}
```

FireAlarmSensorServer.java

```
public class FireAlarmSensorServer extends UnicastRemoteObject implements
FireAlarmSensor, Runnable,Serializable  {






    //---Account SID for twilio-------
     public static final String ACCOUNT_SID = "AC2c49b397d2035b2e9a478ca8172bbc3c";

    //---Account Authentication Token for twilio-------
     public static final String AUTH_TOKEN = "fc88a0cac0f5b1d06b1d671b0b70e362";


    private static final long serialVersionUID = 1L;

    public   List<Sensor> sensorList = new CopyOnWriteArrayList<Sensor>();
    public   List<Admin> adminList = new CopyOnWriteArrayList<>();
    public   Sensor newSensor;
    private  List<FireAlarmClient> clientList = new CopyOnWriteArrayList<>();





    public FireAlarmSensorServer() throws java.rmi.RemoteException{
        //static data to test
        /*sensorList.add(new Sensor("11A",1,"1AAA",3,4));
        sensorList.add(new Sensor("22A",2,"2A",5,2));
        sensorList.add(new Sensor("22B",2,"2B",1,0));
        sensorList.add(new Sensor("33A",3,"3A",0,6));
        adminList.add(new Admin("admin","admin"));
        adminList.add(new Admin("dulini","dulini"));*/

    }

    public void saveSensorsToList() {


          //----get response from api---------------

        Client client = ClientBuilder.newClient();
        WebTarget target = client.target("http://localhost:4000/sensors");

        //----parse response to JSON Object----------
        JSONParser parser = new JSONParser();
        Object obj;





        try {
             obj = parser.parse(target.request(MediaType.TEXT_XML).get(String.class));
             JSONArray array = (JSONArray)obj;

             List<Sensor> sList = new CopyOnWriteArrayList<Sensor>();

             //-----Iterate through Json array and update ArryList-----------
             for(int i = 0 ;i<array.size();++i) {
                 JSONObject obj2 = (JSONObject)array.get(i);
                 Sensor sensor = new Sensor();

                 sensor.set_id(obj2.get("_id").toString());
                 sensor.setId(obj2.get("sensorid").toString());
                 sensor.setFloorNo(new Integer(obj2.get("floor").toString()));
                 sensor.setRoomNo(obj2.get("room").toString());
                 sensor.setSmokeLevel(new Double( obj2.get("smokelevel").toString()));
                 sensor.setCdLevel(new Double( obj2.get("colevel").toString()));

                 sList.add(sensor);




             }

             sensorList = sList;


        } catch (ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }



    public void addClient(FireAlarmClient client) throws java.rmi.RemoteException {

        System.out.println("adding client -" + client);
        clientList.add(client);
    }

    public void removeClient(FireAlarmClient client) throws java.rmi.RemoteException {
        System.out.println("Remove client -" + client);
        clientList.remove(client);

    }


    //---------Get Admins from API--------

    public void getAdminsFromApi() {

         //----get response from api---------------

        Client client = ClientBuilder.newClient();
        WebTarget target = client.target("http://localhost:4000/users");

        //----parse response to JSON Object----------
        JSONParser parser = new JSONParser();
        Object obj;





        try {
             obj = parser.parse(target.request(MediaType.TEXT_XML).get(String.class));
             JSONArray array = (JSONArray)obj;



             //-----Iterate through Json array and update ArryList-----------
             for(int i = 0 ;i<array.size();++i) {
                  JSONObject obj2 = (JSONObject)array.get(i);
                 Admin admin = new Admin();

                 admin.setUserName(obj2.get("username").toString());
                 admin.setPassword(obj2.get("password").toString());
                 adminList.add(admin);


             }




        } catch (ParseException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
    //-----returns admin list-------------
    @Override
    public List<Admin> getAdmins() throws RemoteException {

        return adminList;
    }

    //-----returns sensor list-------------
    @Override
    public List<Sensor> getSensors() throws RemoteException {


        return sensorList ;
    }

     //----------Edit the sensor details-----------------

    public void EditAPISensordetails(Sensor sensor) throws IOException {

        //-------check the relevent sensor in sensor list

        Sensor updated = new Sensor();
        for(Sensor s:sensorList) {

            if(s.getId().equals(sensor.getId())) {

                  updated.set_id(s.get_id());
                  updated.setId(sensor.getId());
                  updated.setFloorNo(sensor.getFloorNo());
                  updated.setRoomNo(sensor.getRoomNo());
                  updated.setCdLevel(s.cdLevel);
                  updated.setSmokeLevel(s.getSmokeLevel());
            }
        }

        final String EDIT_SENSOR = "{\n" +
                "\"sensorid\":"+ updated.getId()+",\r\n" +
                "    \"floor\":"+updated.getFloorNo()+",\r\n" +
                "    \"colevel\":"+updated.getCdLevel()+",\r\n" +
                "    \"smokelevel\":"+updated.getSmokeLevel()+",\r\n" +
                "    \"room\":"+ updated.getRoomNo()+""+
                "\n}";


            System.out.println(updated.getRoomNo());
            URL obj = new URL("http://localhost:4000/sensors/"+updated._id);
            HttpURLConnection postConnection = (HttpURLConnection) obj.openConnection();
            postConnection.setRequestMethod("POST");
            postConnection.setRequestProperty("Content-Type", "application/json");
            postConnection.setDoOutput(true);
            OutputStream os = postConnection.getOutputStream();
            os.write(EDIT_SENSOR.getBytes());
            os.flush();
            os.close();
            int responseCode = postConnection.getResponseCode();
            System.out.println("POST Response Code :  " + responseCode);
            System.out.println("POST Response Message : " + postConnection.getResponseMessage
());
            if (responseCode == HttpURLConnection.HTTP_CREATED) { //success
                BufferedReader in = new BufferedReader(new InputStreamReader(
                    postConnection.getInputStream()));
                String inputLine;
                StringBuffer response = new StringBuffer();
                while ((inputLine = in .readLine()) != null) {
                    response.append(inputLine);
                } in .close();
                // print result
                System.out.println(response.toString());
            } else {
                System.out.println("POST NOT WORKED");
            }


    }
    @Override
    public void EditSensor(Sensor sensor) throws RemoteException {

        try {
            EditAPISensordetails(sensor);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }






        }






    //----------Add new sensors-----------------
```

```java
@Override
public void addSensor(Sensor s) {


    try {
        AddNeWSensorToApi(s);
    } catch (MalformedURLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

}

public void AddNeWSensorToApi(Sensor s) throws IOException {


    final String NEW_SENSOR = "{\n" +
            "\"sensorid\":"+ s.getId()+",\r\n" +
            "    \"floor\":"+s.getFloorNo()+",\r\n" +
            "    \"colevel\":"+0+",\r\n" +
            "    \"smokelevel\":"+0+",\r\n" +
            "    \"room\":"+ s.getRoomNo()+""+
            "\n}";


        System.out.println(NEW_SENSOR);
        URL obj = new URL("http://localhost:4000/sensors");
        HttpURLConnection postConnection = (HttpURLConnection) obj.openConnection();
        postConnection.setRequestMethod("POST");
        postConnection.setRequestProperty("userId", "a1bcdefgh");
        postConnection.setRequestProperty("Content-Type", "application/json");
        postConnection.setDoOutput(true);
        OutputStream os = postConnection.getOutputStream();
        os.write(NEW_SENSOR.getBytes());
        os.flush();
        os.close();
        int responseCode = postConnection.getResponseCode();
```

```
            System.out.println("POST Response Code :  " + responseCode);
            System.out.println("POST Response Message : " + postConnection.getResponseMessage
());
            if (responseCode == HttpURLConnection.HTTP_CREATED) { //success
                BufferedReader in = new BufferedReader(new InputStreamReader(
                    postConnection.getInputStream()));
                String inputLine;
                StringBuffer response = new StringBuffer();
                while ((inputLine = in .readLine()) != null) {
                    response.append(inputLine);
                } in .close();
                // print result
                System.out.println(response.toString());
            } else {
                System.out.println("POST NOT WORKED");
            }

    }

    //----------Send sms when carbondioxide or smoke level goes up----------------
    public void sendSMS(int floor, String room, double smokeLevel, double cdLevel) throws Rem
oteException {

            Twilio.init(ACCOUNT_SID, AUTH_TOKEN);
            Message.creator(
                    //----My phone number----
                    new com.twilio.type.PhoneNumber("+94715443619"),
                    //---Purchased number from twilio-----
                    new com.twilio.type.PhoneNumber("+13343731533"),
                    "Warning!!\n"+"Smoke Level and carbon Dioxide level is increasing\n"+"Flo
or Number:"+floor+"\n"+"Room Number:"+room)
                .create();

    }



    @Override
    public void run() {

        //Random r = new Random();

        for(;;) {
            try {

                Thread.sleep(15000);
            }
            catch(InterruptedException ie) {

            }



            try {





                saveSensorsToList();
                notifyClients();

                   for(Sensor s:sensorList) {

                    if(s.getSmokeLevel() >5 || s.getCdLevel() > 5) {
                        sendSMS(s.getFloorNo(),s.getRoomNo(),s.getSmokeLevel(),s.getCdLevel()
);


                    }
                }

            } catch (RemoteException e) {

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

                e.printStackTrace();
            }

        }

    }



    private void notifyClients() throws RemoteException,AlreadyBoundException {
```

```java
        for(FireAlarmClient c :clientList){
            c.getSensorDetails(sensorList);

        }
    }



    public static void main(String[] args) {

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


            System.out.println("Loading RMI server");




            // Registering the server to the RMI registry
            try {
                FireAlarmSensorServer fsensor = new FireAlarmSensorServer();
                String registry = "localhost";

                String registration = "rmi://" + registry + "/FireAlarmSensor";


                Naming.rebind(registration, fsensor);

                Thread thread = new Thread(fsensor);
                fsensor.saveSensorsToList();
                fsensor.getAdminsFromApi();
                thread.start();
            } catch (RemoteException re) {
                System.err.println("Remote Error - " + re);
            } catch (Exception e) {
                System.err.println("Error - " + e);
            }
        }

}
```

Client.java

```java
public class Client extends UnicastRemoteObject implements Serializable,
FireAlarmClient, Runnable {

    public static List<Sensor> sensorList;
    public static sensorui sList = new sensorui();
    public boolean isLoggedIn = false;


    private static final long serialVersionUID = 1L;

    public Client() throws RemoteException{

    }


    public static void main(String[] args) throws Exception {

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

                //--------
Searching the appropriate server in RMI registry using lookup() method-----
            try {

                String registration = "//localhost/FireAlarmSensor";
                Remote remoteService = Naming.lookup(registration);
                FireAlarmSensor sensor = (FireAlarmSensor) remoteService;


                sensorList = sensor.getSensors();

                Client c = new Client();

                sensor.addClient(c);

                sList.sensorList(sensorList);
                sList.show_sensor();
                sList.getServer(sensor);
                sList.getClient(c);
                sList.setVisible(true);

                c.run();

            } catch (MalformedURLException mue) {
                System.out.println(mue);
            } catch (RemoteException re) {
                System.out.println(re);
            } catch (NotBoundException nbe) {
                System.out.println(nbe);
            }
        }


    @Override
    public void getSensorDetails(List<Sensor> sensorList) {



        this.sensorList = sensorList;
        sList.sensorList(sensorList);
        sList.selectSID();
        sList.refreshTable();
        try {
            sList.show_sensor();
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }


    public void setLoginStatus(boolean result) {

        this.isLoggedIn = result;
    }

     public boolean getLoginStatus() {

        return this.isLoggedIn;
    }
    @Override
    public void run() {


        for (;;) {
            //count++;

        // note that this might only work on windows console
            //System.out.print("\r" + count);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException ie) {
            }

        }

    }

}
```

END