Fire Alarm Monitoring System Assignment Report

Batch: 20.1

	Student Registration Number	Student Name
1	IT18018288	Ranawaka M. N.
2	IT18016376	L. K. Sandeepa
3	IT17036658	K. A. D. G. Methmini
4	IT18140958	W. S. I. Fernando

Table of Contents

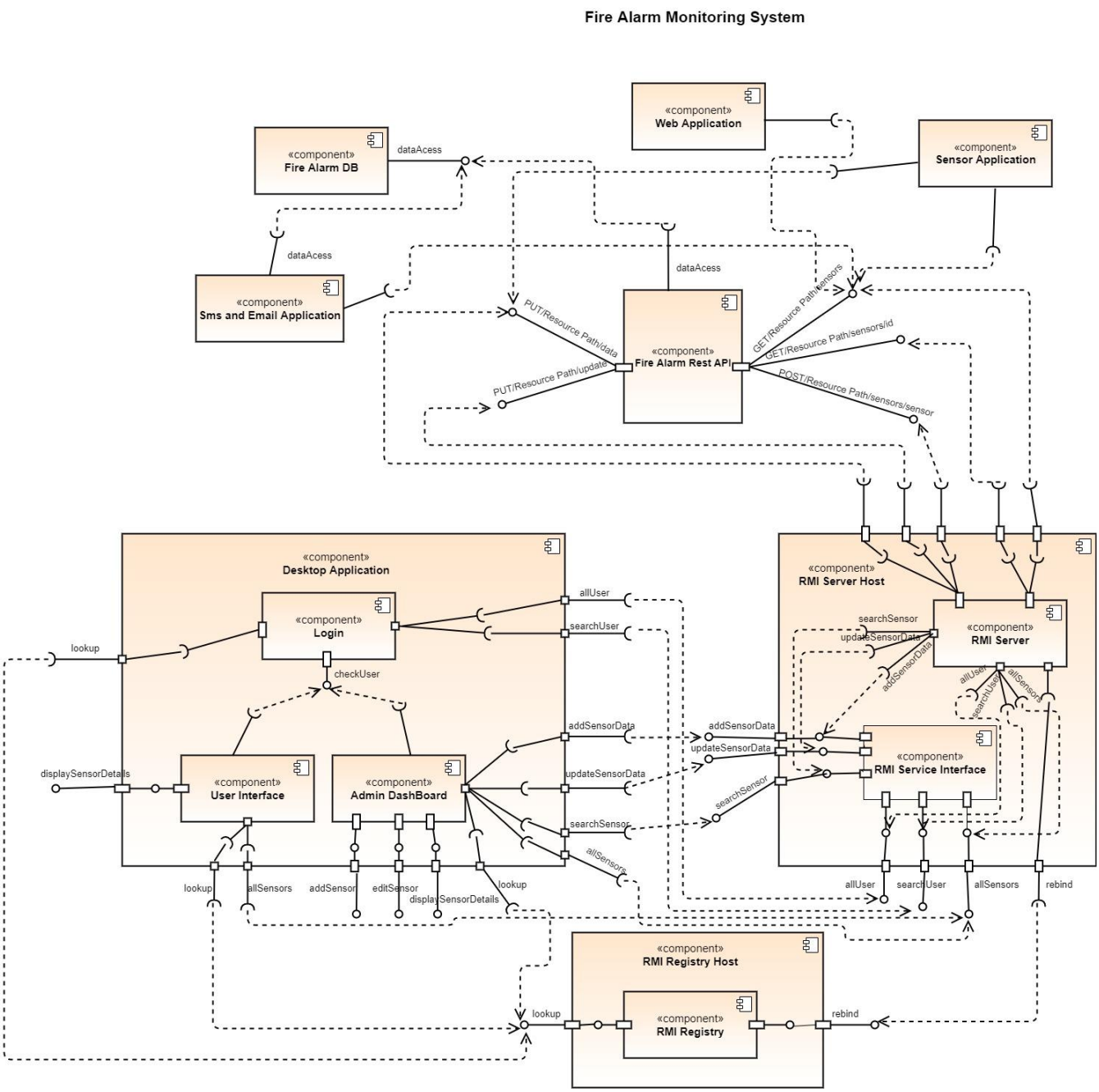
1. Introduction

The fire alarm monitoring system is developed using different technologies such as Maven and Jersey version 01 using JAVA for REST API architecture, Web client using React, and for styling bootstrap and font-awesome. Java for sensor app RMI Client/Server. SMS & email using Gmail API and Twilio API for java and Database using MySQL.

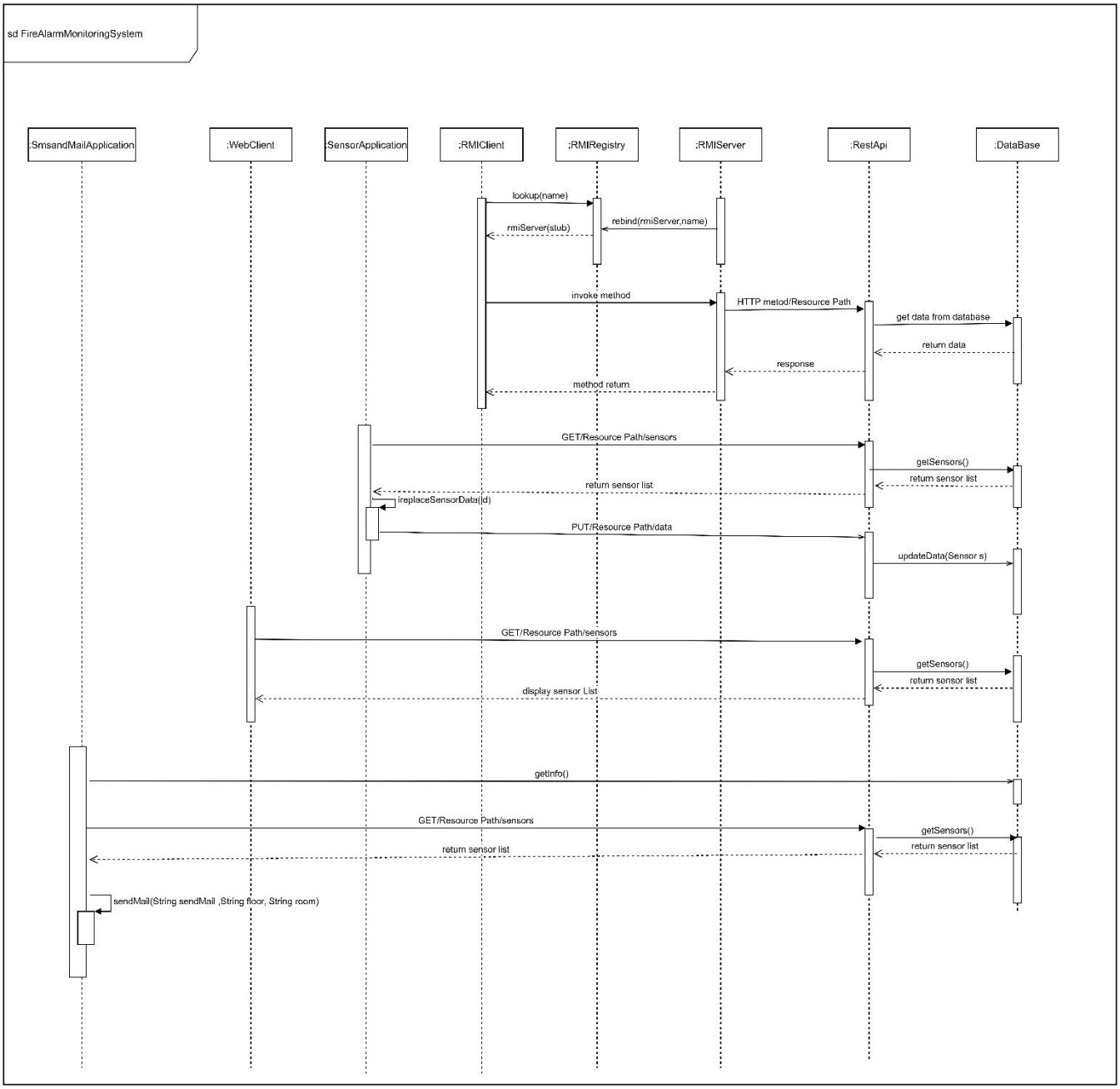
This system is used for users who can monetarize the fire alarm system. Admin of the system can edit and update information on the fire alarm system. Fire Sensor App will update sensor data every 10 seconds. The web client will update every 40 seconds, and the RMI server will update every 15 seconds, and the RMI client will update every 30 seconds.

If Sensor data, smoke level, or co2 level goes more than five web client dashboards will mark them in red color, and also desktop client will automatically update and email notification, and SMS notification will send to registered email and mobile number. We have explained this scenario in this report with some screenshots.

2. High Level Architectural Diagram



3. System Workflow Diagram



4. System Explanation

4.1 REST API

Project Name: firealamrest

We create REST API using Maven and Jersey version 01 for java. We create properties for sensorID, sensorStatus, sensorLocationFloorNo, sensorLocationRoomNo, smokeLevel, co2Level. SensorRepository class will work with the database and add and return the data that other methods are requesting. SensorResource class has the REST API method, such as GET, PUT, and POST methods, to add, update, and retrieve data. CORSFilter method is used to modify the header to enable other applications to access API.

This app will use mysql-connector-java as maven dependencies.

4.2 Sensor App

Project Name: SensorUpdateApp

Sensor update class will act as a real server in this project. Sensor update class has getRandom(), getList(), replaceSensorData() and main() method. getRandom() class will generate random number with some probability. getList() method will get all data in REST API and list them and call replaceSensorData() class for every sensor. replaceSensorData class will get random number form getRandom() class and update that sensor co2 and smoke level. Main class will call getList() class every 10 seconds.

Sensor Update will only update only sensor status TRUE sensors only. If sensor status is false it means sensor is deactivate so it will no longer read data from that sensor. So, that sensor will not update.

This app will use json-simple-1.1.1.jar file

4.3 Web App

Project Name: firealarmweb

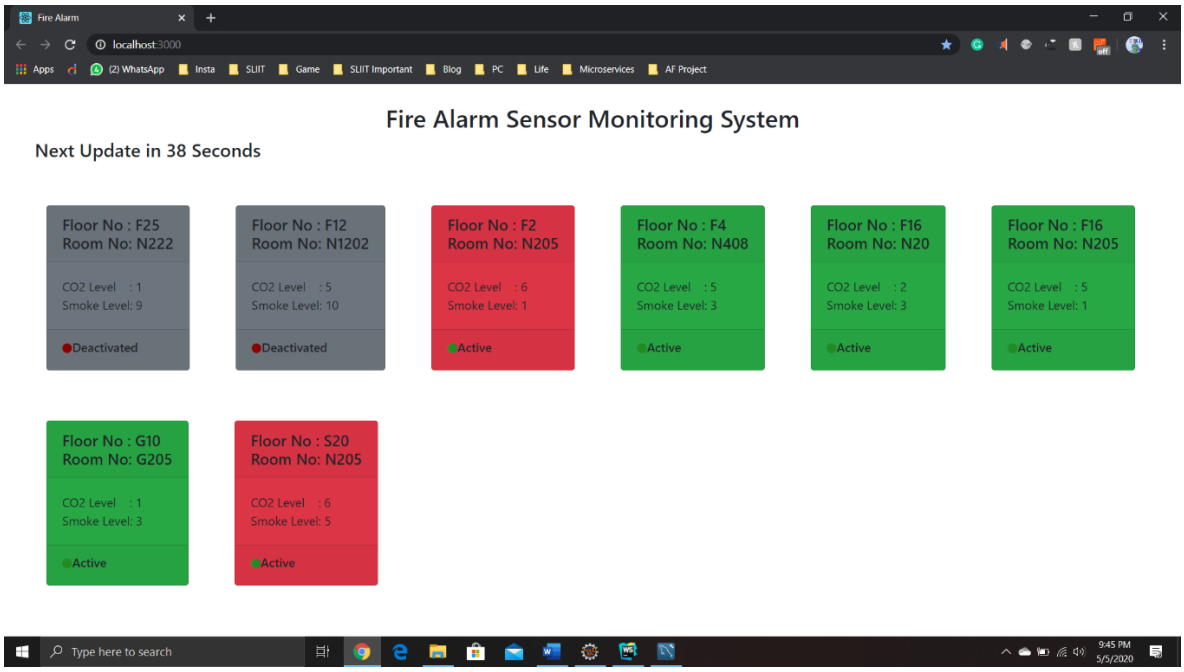
We build fire alarm class using React. First we declare loading status, sensor list and countdown starting from 40 seconds. tick() method and componentWillUnmount() life cycle method will handle the countdown. componentDidMount() life cycle method will handle count down and also Rest API. In componentDidMount() method get API data for first time load and it will enter 40 second timeout catch block. So that code block will get data from API and store them in sensor array every 40 seconds and countdown will reset back to 40.

In body using list map() method it will create a card for every sensor and display that sensor with some color code using bootstrap for styling.

Gray for Deactivated Sensors.

Red for warning sensor. Sensors where Co2 or Smoke level higher than 5

Green for Active sensors.



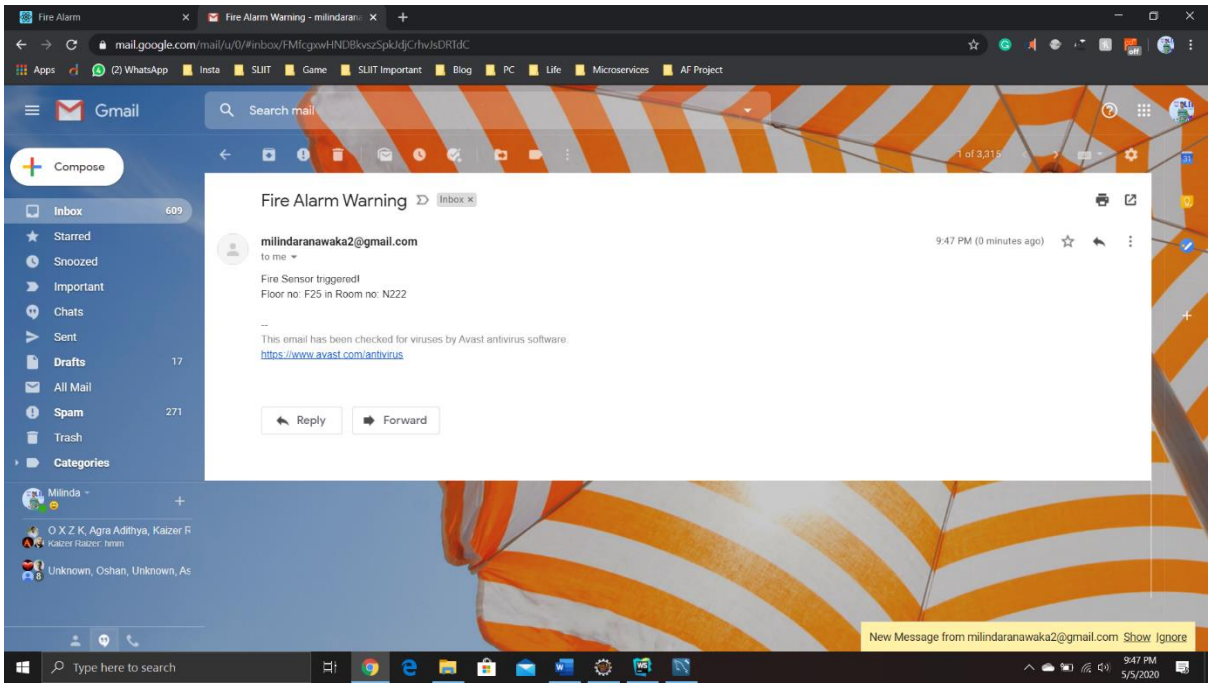
4.4 Email & SMS App

Project Name: SendEmail

We developed this class using Gmail API and Twilio API. This app will check sensor every 5 seconds and send email and SMS to register users in every 5 seconds.

Using getInfo() method admin contact No and Email address will be stored in String variables. getList() method will check if sensor hit Co2 or Smoke level larger number than 5. The commented SMSSend() can be used to send SMS to user mobile phone. Since twilio is paid method we could not get our Auth token. If we have auth token and uncomment SMSSend() method and then add auth token to AUTH_TOKEN variable and un comment commented line in getList() method also sms can be send to user.

In main class getList() method will call every 5 seconds and getList method look in every sensor in API and if a sensor hit larger number that 5 it will automatically send email and print user number and sms send message in console.




```
Email sent
SMS Send to : +94718956912 Number
```

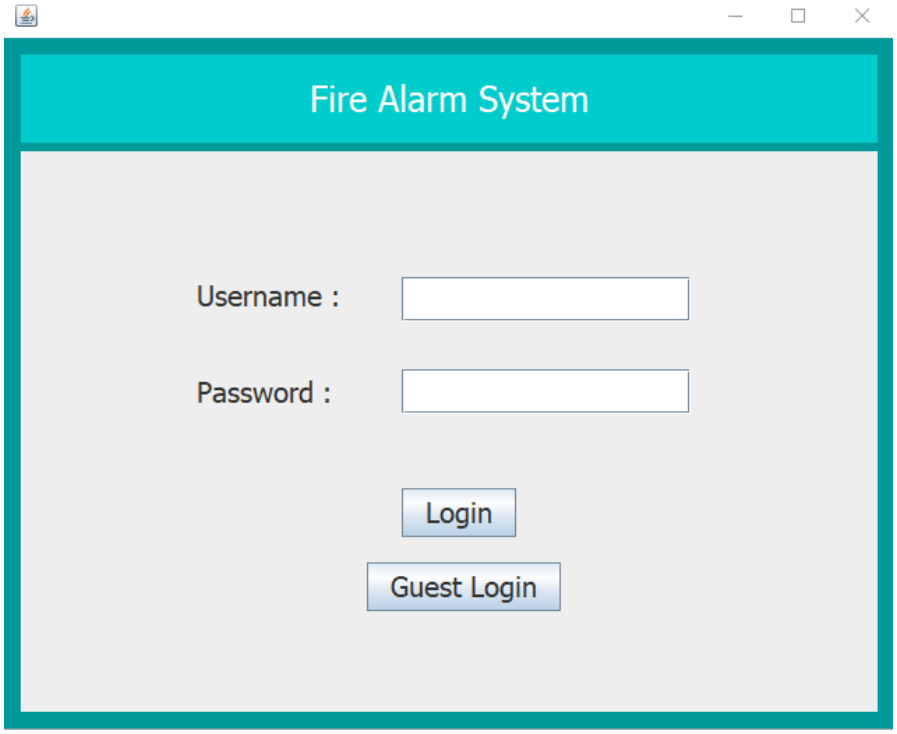
4.5 RMI Server

Project Name: fireAlarmRMI

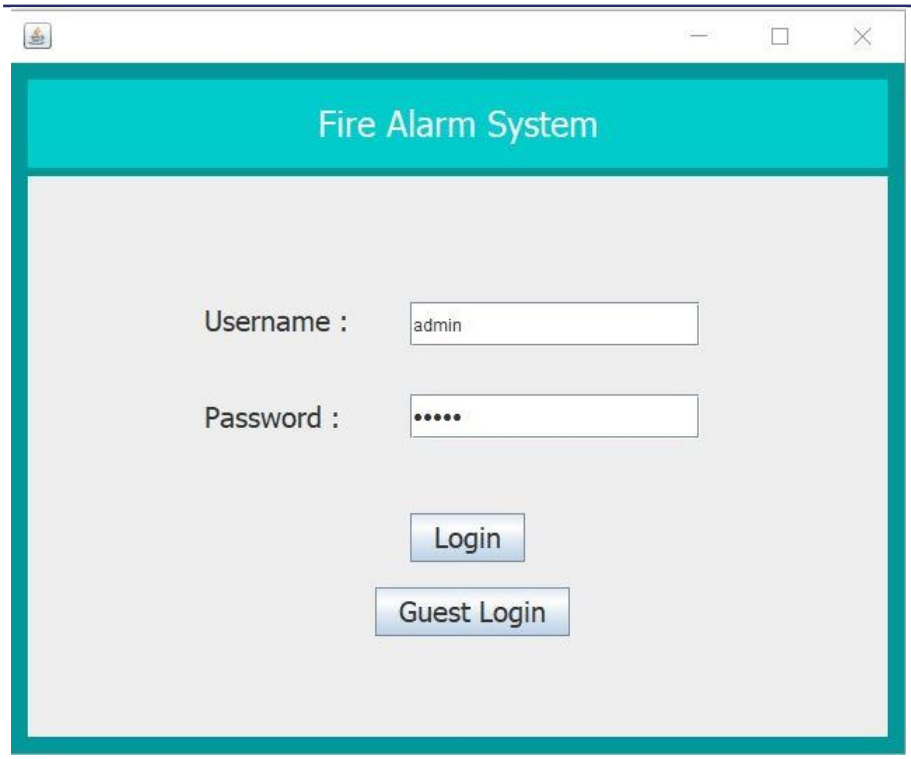
RMI has 6 main methods. addSensorData(Sensor s) and updateSensorData(Sensor s) methods are used to add new sensor details and edit existing sensor details. searchSensor(Sensor s) and searchUser(User u) methods are used to find sensor or user details in the given id. allSensors() and allUser() methods return all sensors and all users in the database.

4.6 RMI Client

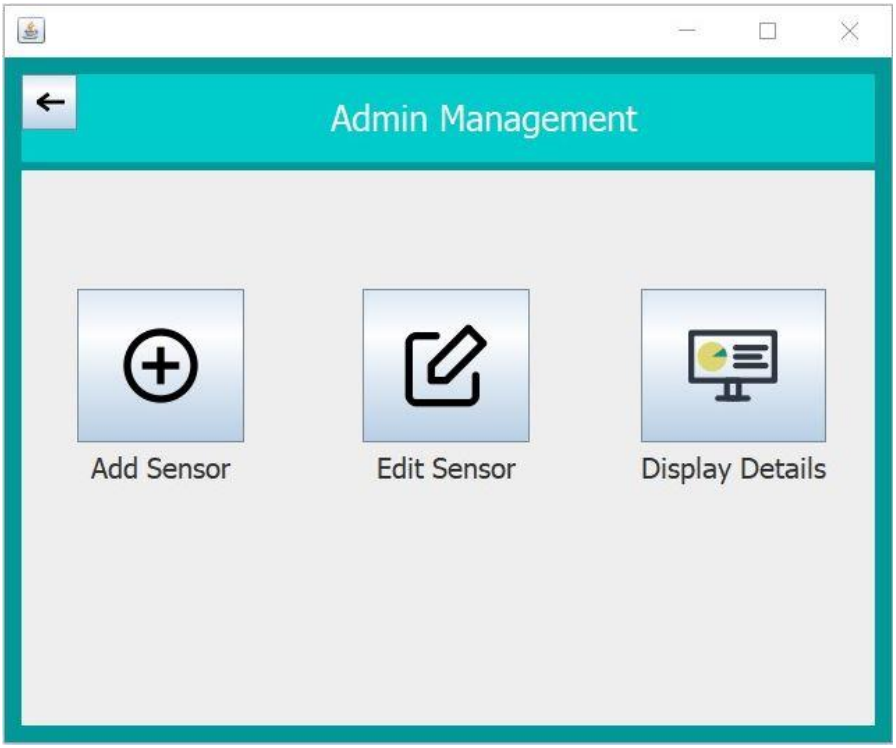
In Fire Alarm System first you have to login by providing the username and password.



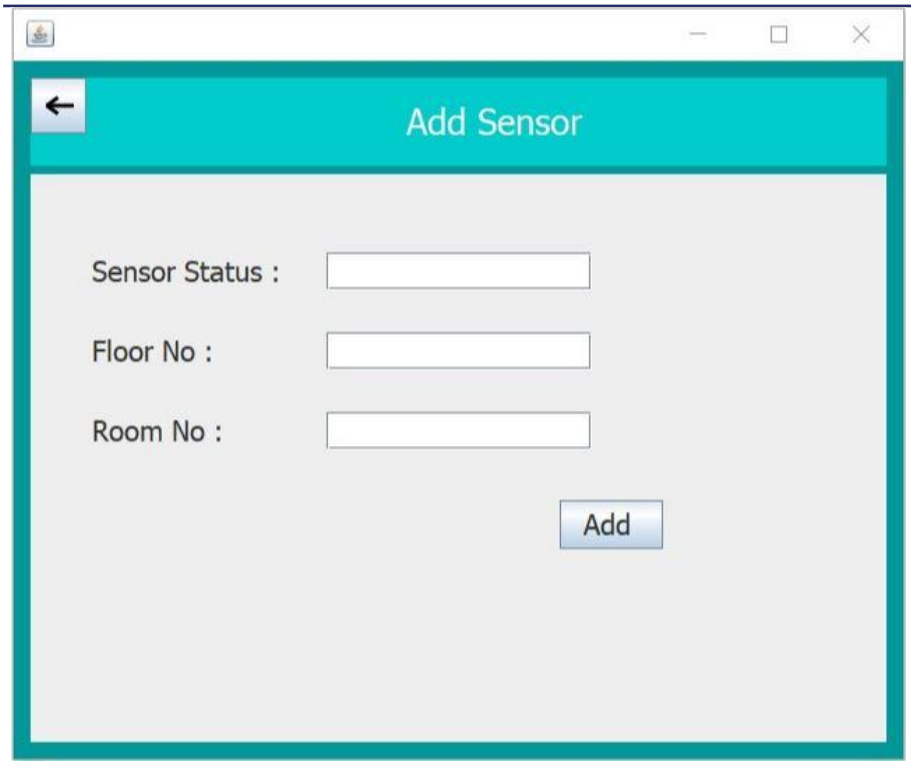
If you are an admin click on “Login” button after giving the username and password.



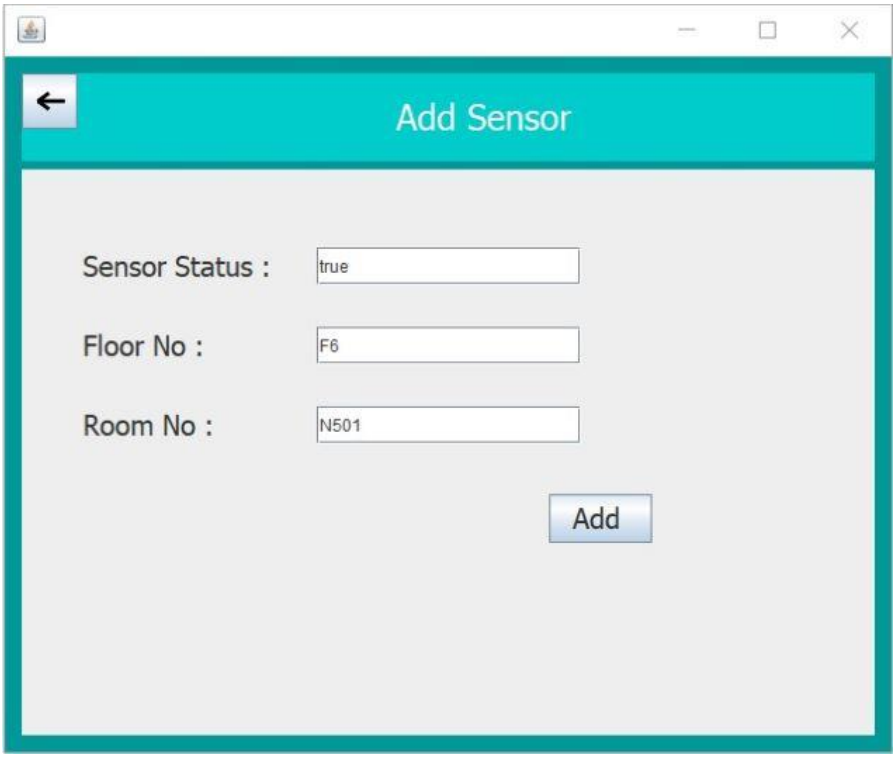
After login as an admin it will move to “Admin Management” page where an admin can add new sensors, edit sensors or display sensor details.



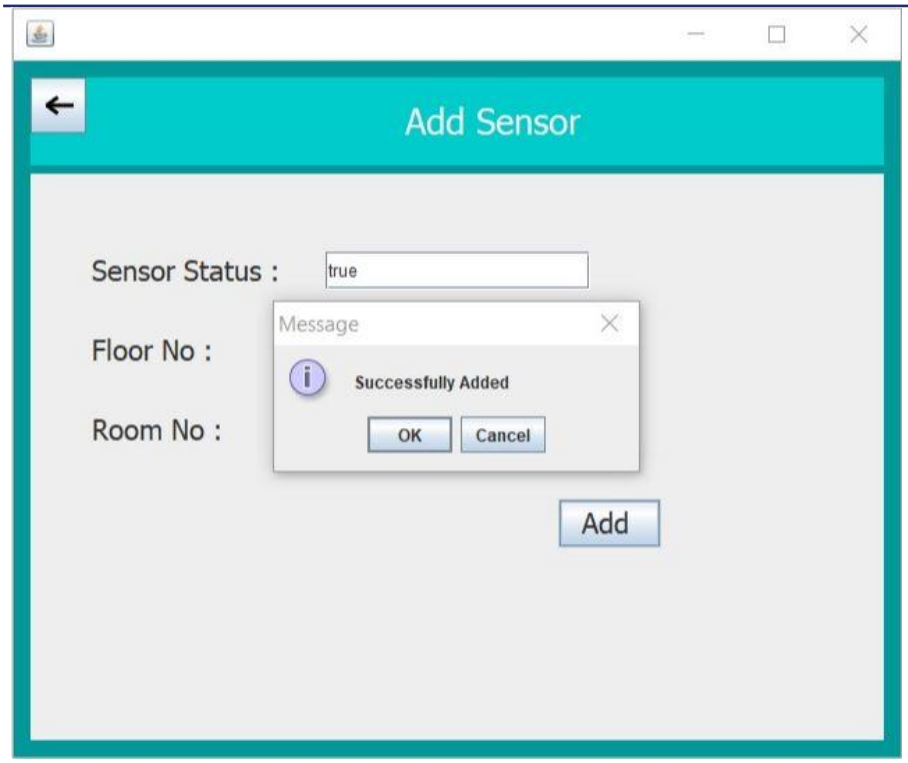
If admin wants to add a new sensor click on “Add Sensor” button in admin management page and it will move to add sensor page.



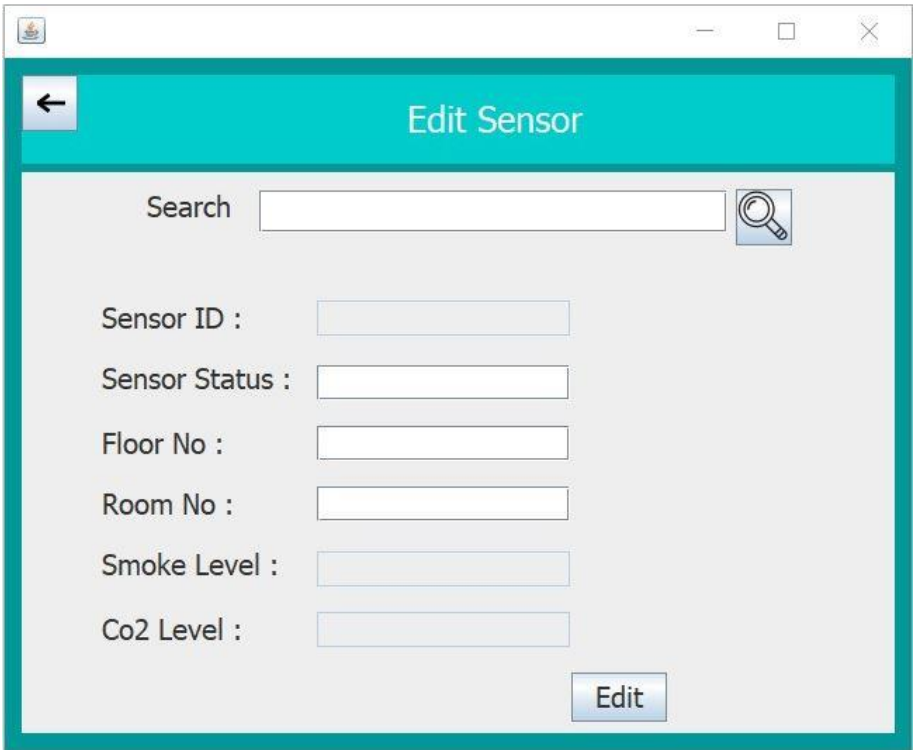
After providing sensor status, floor number and room number of a new sensor click on "Add" button.



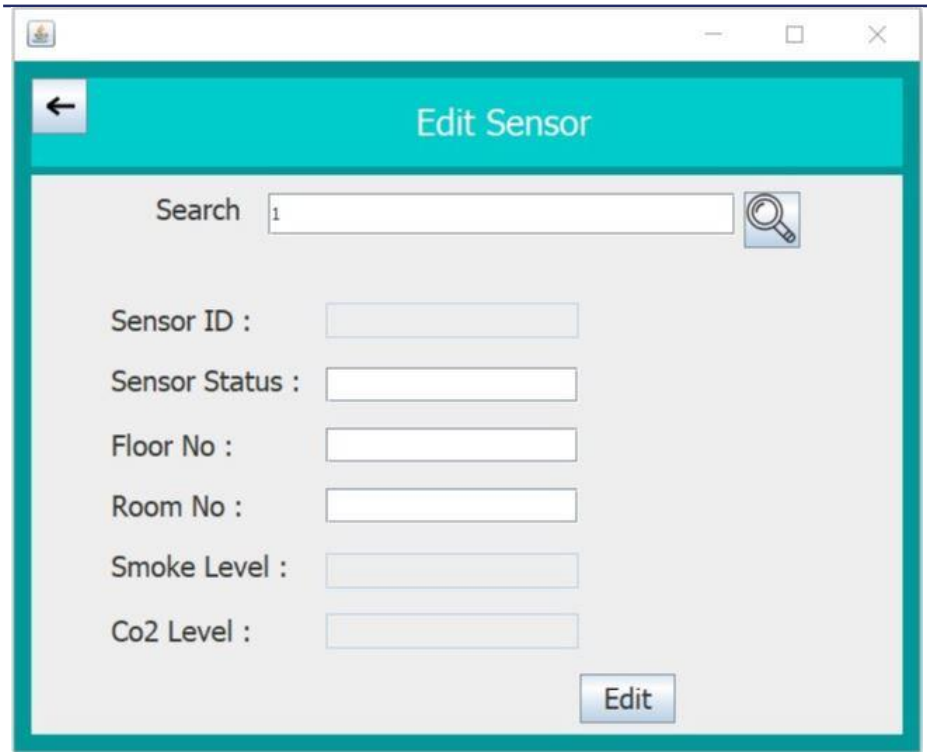
If the new sensor was added to the database a success message will be displayed.



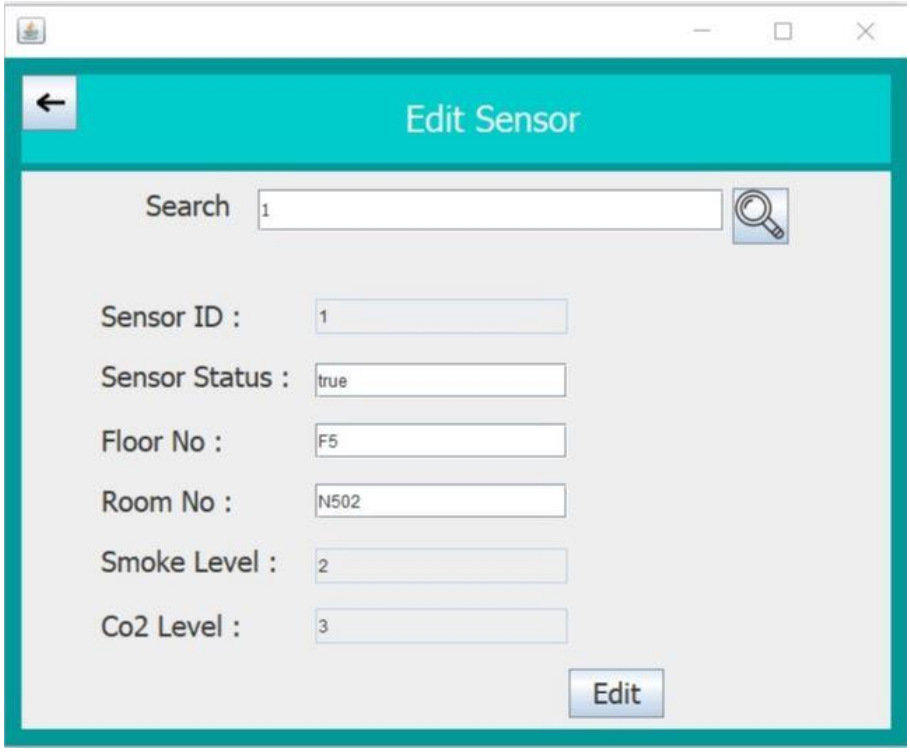
If admin wants to edit sensor details, click on "Edit Sensor" button in admin management page and it will move to edit sensor page.



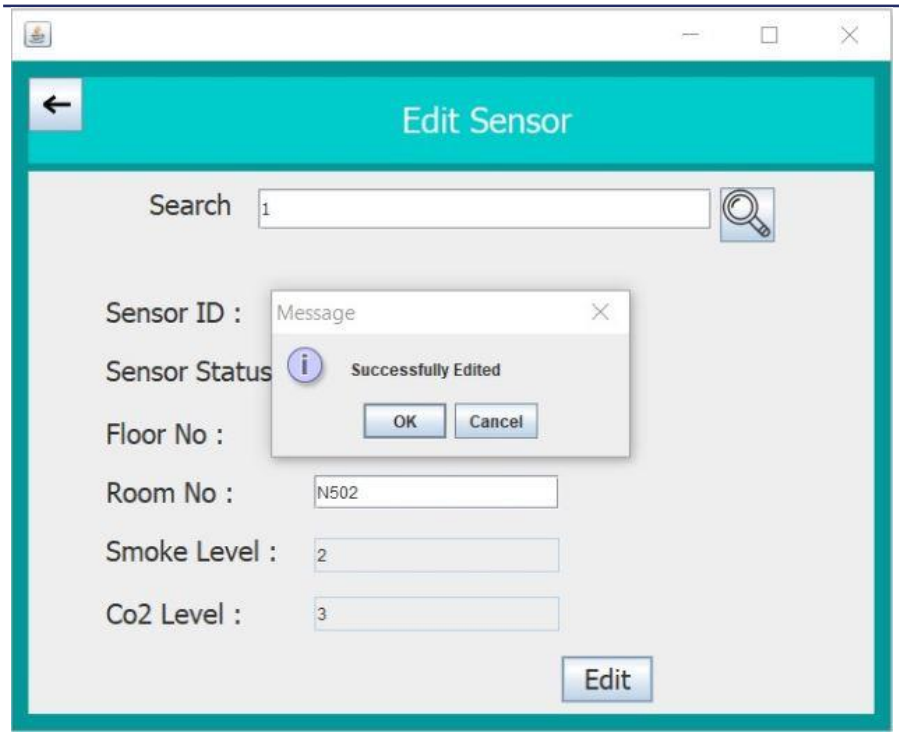
Search a particular sensor by providing the sensor id number in the search bar and click on search icon button.



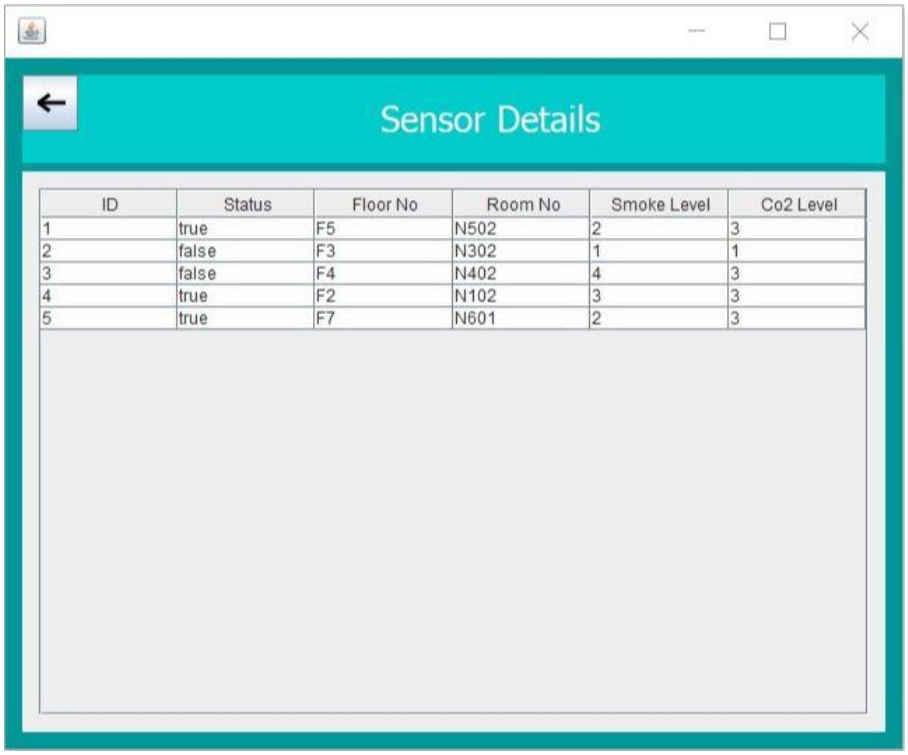
Once the search icon button is clicked after providing an id in the search bar, details relevant to that sensor id will be displayed in the fields.



If admin wants to edit those details provide necessary details and click on “Edit” button and a success message will be displayed. (admin can edit only the sensor status, floor number and room number)

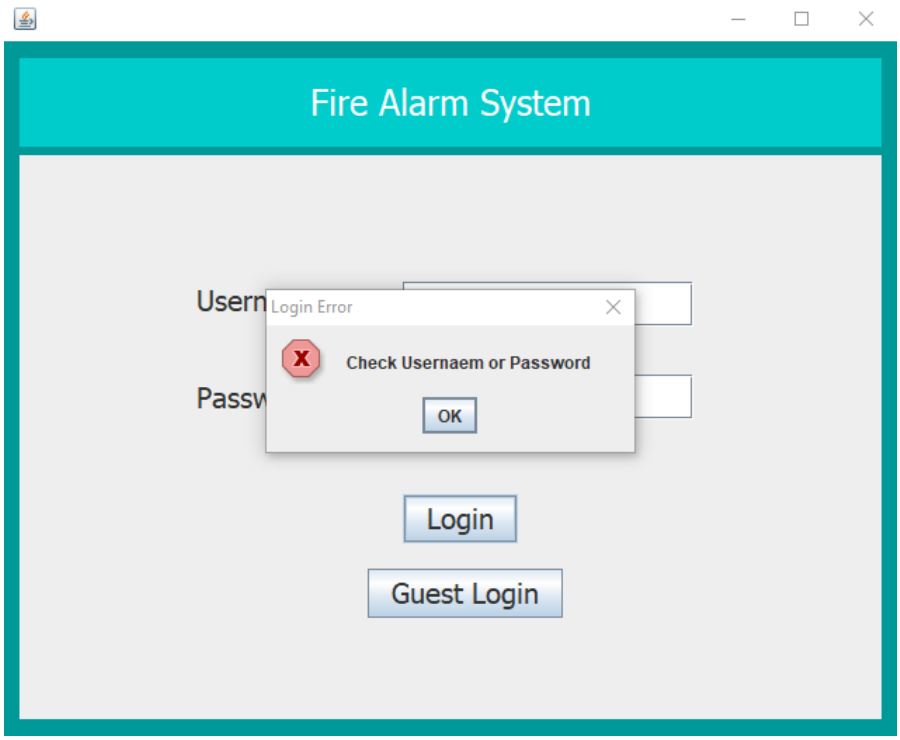


If admin wants to display sensor details, click on "Display Details" button in admin management page and it will move to sensor details page. (Details are auto updating after 30 seconds)

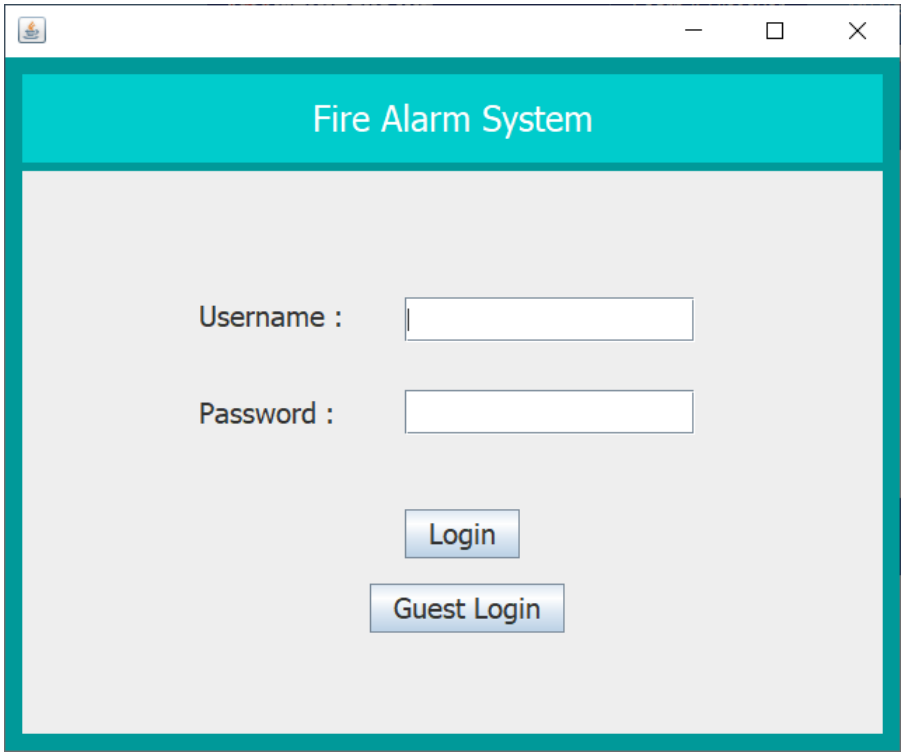


ID	Status	Floor No	Room No	Smoke Level	Co2 Level
1	true	F5	N502	2	3
2	false	F3	N302	1	1
3	false	F4	N402	4	3
4	true	F2	N102	3	3
5	true	F7	N601	2	3

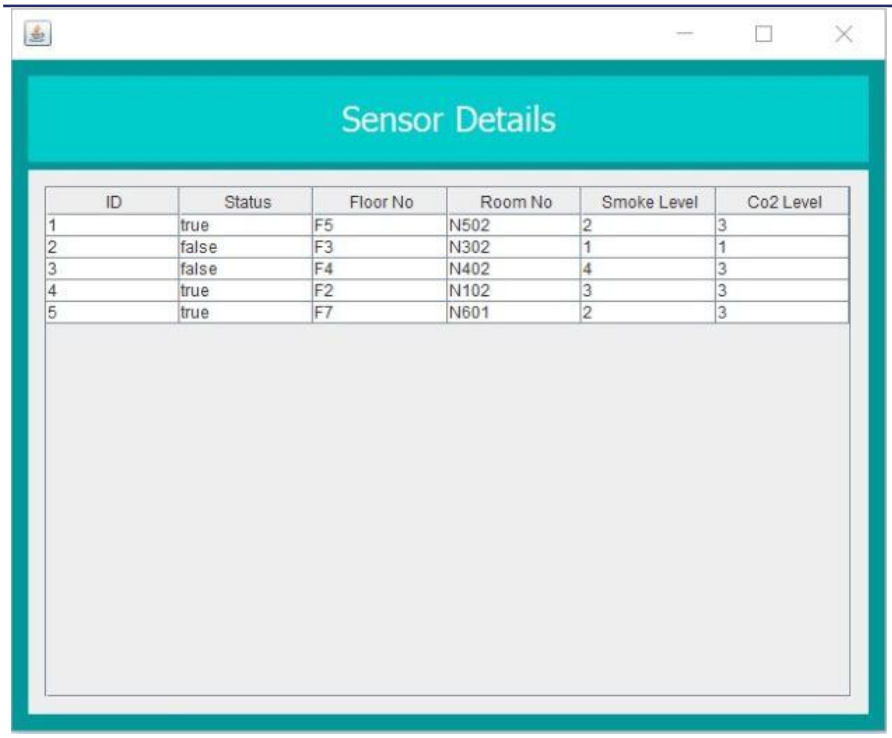
If admin enters an invalid username or password error message will be displayed.



If you are a user click on “Guest Login” button without entering username or password.



Users can only view the sensor details after login by clicking “Guest Login” button it will move to guest sensor details page. (Details are auto updating after 30 seconds)



ID	Status	Floor No	Room No	Smoke Level	Co2 Level
1	true	F5	N502	2	3
2	false	F3	N302	1	1
3	false	F4	N402	4	3
4	true	F2	N102	3	3
5	true	F7	N601	2	3

4.7 Database

We create Database using MySql. We create users class for store admin data in database and sensors class for store sensor data in database.

5. Appendix

5.1 REST API

CORSFilter.java

```
package com.firealarm.firealamrest;

import com.sun.jersey.spi.container.ContainerRequest;
import com.sun.jersey.spi.container.ContainerResponse;
import com.sun.jersey.spi.container.ContainerResponseFilter;
public class CORSFilter implements ContainerResponseFilter {

      //used to modify header of the code to access API
    @Override
    public ContainerResponse filter(ContainerRequest request,
            ContainerResponse response) {

        response.getHttpHeaders().add("Access-Control-Allow-Origin", "*");
        response.getHttpHeaders().add("Access-Control-Allow-Headers",
                "origin, content-type, accept, authorization");
        response.getHttpHeaders().add("Access-Control-Allow-Credentials", "true");
        response.getHttpHeaders().add("Access-Control-Allow-Methods",
                "GET, POST, PUT, DELETE, OPTIONS, HEAD");

        return response;
    }
}
```

Sensor.java

```
package com.firealarm.firealamrest;

import javax.xml.bind.annotation.XmlRootElement;

//Class of a sensor
@XmlRootElement
public class Sensor {

      private int sensorID;
      private boolean sensorStatus;
      private String sensorLocationFloorNo;
      private String sensorLocationRoomNo;
      private int smokeLevel;
      private int co2Level;

      public int getSensorID() {
            return sensorID;
      }
      public void setSensorID(int sensorID) {
```

```
		this.sensorID = sensorID;
	}
	public boolean isSensorStatus() {
		return sensorStatus;
	}
	public void setSensorStatus(boolean sensorStatus) {
		this.sensorStatus = sensorStatus;
	}
	public String getSensorLocationFloorNo() {
		return sensorLocationFloorNo;
	}
	public void setSensorLocationFloorNo(String sensorLocationFloorNo) {
		this.sensorLocationFloorNo = sensorLocationFloorNo;
	}
	public String getSensorLocationRoomNo() {
		return sensorLocationRoomNo;
	}
	public void setSensorLocationRoomNo(String sensorLocationRoomNo) {
		this.sensorLocationRoomNo = sensorLocationRoomNo;
	}
	public int getSmokeLevel() {
		return smokeLevel;
	}
	public void setSmokeLevel(int smokeLevel) {
		this.smokeLevel = smokeLevel;
	}
	public int getCo2Level() {
		return co2Level;
	}
	public void setCo2Level(int co2Level) {
		this.co2Level = co2Level;
	}
	@Override
	public String toString() {
		return "Sensor [sensorID=" + sensorID + ", sensorStatus=" +
sensorStatus + ", sensorLocationFloorNo="
			+ sensorLocationFloorNo + ", sensorLocationRoomNo=" +
sensorLocationRoomNo + ", smokeLevel="
			+ smokeLevel + ", co2Level=" + co2Level + "]";
	}


}
```

SensorRepository.java

```
package com.firealarm.firealamrest;

import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class SensorRepository {

	Connection con = null;

	//DB Connection
	public SensorRepository() {

		//TODO : Make sure line 19, 20, 21 port Username and PWD same with
your computer
		String url = "jdbc:mysql://localhost:3307/fireAlarmAPI";
		String un = "root";
		String pwd = "root";
		try {
			Class.forName("com.mysql.jdbc.Driver");
			con = DriverManager.getConnection(url,un,pwd);
		} catch (SQLException e) {
			e.printStackTrace();
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}
	}

	//To get all sensor information from DB
	public ArrayList<Sensor> getSensors() {

		ArrayList<Sensor> sensors = new ArrayList<Sensor>();
		String sql = "select * from sensors";

		try {
			Statement st = con.createStatement();
			ResultSet rs = st.executeQuery(sql);

			while(rs.next()) {
				Sensor s = new Sensor();

				s.setSensorID(rs.getInt(1));
				s.setSensorStatus(rs.getBoolean(2));
				s.setSensorLocationFloorNo(rs.getString(3));
				s.setSensorLocationRoomNo(rs.getString(4));
				s.setSmokeLevel(rs.getInt(5));
				s.setCo2Level(rs.getInt(6));
```

```
                        sensors.add(s);
                    }
                }
                catch (Exception e) {
                    System.out.println(e);
                }

                return sensors;
            }

            //To get sensor details of specific sensor details from DB
            public Sensor getSensor(int id) {
                String sql = "select * from sensors where sensorID ="+id;
                Sensor s = new Sensor();

                try {
                    Statement st = con.createStatement();
                    ResultSet rs = st.executeQuery(sql);

                    while(rs.next()) {
                        s.setSensorID(rs.getInt(1));
                        s.setSensorStatus(rs.getBoolean(2));
                        s.setSensorLocationFloorNo(rs.getString(3));
                        s.setSensorLocationRoomNo(rs.getString(4));
                        s.setSmokeLevel(rs.getInt(5));
                        s.setCo2Level(rs.getInt(6));
                    }
                }
                catch (Exception e) {
                    System.out.println(e);
                }

                return s;
            }


            //To add new sensor to DB
            public void createSensor(Sensor s) {
                String sql = "insert into
sensors(sensorStatus,sensorLocationFloorNo,sensorLocationRoomNo,smokeLevel,co2Leve
l) values(?,?,?,?,?)";

                try {
                    PreparedStatement st = con.prepareStatement(sql);
                    st.setBoolean(1, s.isSensorStatus());
                    st.setString(2, s.getSensorLocationFloorNo());
                    st.setString(3, s.getSensorLocationRoomNo());
                    st.setInt(4, s.getSmokeLevel());
                    st.setInt(5, s.getCo2Level());
                    st.executeUpdate();
                }
                catch(Exception e) {
                    System.out.println(e);
                }
            }
```

```
	//To Update smoke and co2 level details of a sensor in DB
	public void updateData(Sensor s) {

		String sql = "UPDATE sensors SET smokeLevel=?, co2Level=? WHERE
sensorID=?";

		try {
			PreparedStatement st = con.prepareStatement(sql);
			st.setInt(1, s.getSmokeLevel());
			st.setInt(2, s.getCo2Level());
			st.setInt(3, s.getSensorID());

			st.executeUpdate();
		}
		catch(Exception e) {
			System.out.println(e);
		}
	}

	//To update sensor status sensor floor no and room no in DB
	public void updateSensor(Sensor s) {
		String sql = "UPDATE sensors SET sensorStatus=?,
sensorLocationFloorNo=?, sensorLocationRoomNo=? WHERE sensorID=?";

		try {
			PreparedStatement st = con.prepareStatement(sql);
			st.setBoolean(1, s.isSensorStatus());
			st.setString(2, s.getSensorLocationFloorNo());
			st.setString(3, s.getSensorLocationRoomNo());
			st.setInt(4, s.getSensorID());

			st.executeUpdate();
		}
		catch(Exception e) {
			System.out.println(e);
		}
	}
}
```

SensorResource.java

```
package com.firealarm.firealamrest;

import java.util.ArrayList;

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

//This runs in side /webresources
// url url would be http://localhost:8080/firealamrest/webresources/
@Path("sensors")
public class SensorResource {

	SensorRepository sensorRepo = new SensorRepository();

	//To get all sensors details
	@GET
	@Produces(MediaType.APPLICATION_JSON)
	public ArrayList<Sensor> getSensors() {

		return sensorRepo.getSensors();
	}

	//To get specific sensor details
	@GET
	@Path("{id}")
	@Produces(MediaType.APPLICATION_JSON)
	public Sensor getSensor(@PathParam("id") int id) {
		return sensorRepo.getSensor(id);
	}

	//To add a sensor
	@POST
	@Path("sensor")
	public Sensor addSensor(Sensor s) {
		sensorRepo.createSensor(s);

		return s;
	}

	//To update sensor co2 and smoke level
	@PUT
	@Path("data")
	public Sensor updateSensorData(Sensor s) {
		sensorRepo.updateData(s);
		return s;
	}
```

```
	//To update sensor status and floor number and room no
	@PUT
	@Path("update")
	public Sensor updateSensor(Sensor s) {
		sensorRepo.updateSensor(s);
		return s;
	}

}
```

To pom.xml dependency

```
<dependency>
	<groupId>mysql</groupId>
	<artifactId>mysql-connector-java</artifactId>
	<version>8.0.19</version>
</dependency>
```

To web.xml <init-param>

```
<init-param>
	<param-name>com.sun.jersey.spi.container.ContainerResponseFilters</param-name>
	<param-value>com.firealarm.firealamrest.CORSFilter</param-value>
</init-param>
```

5.2 Sensor App

UpdateSensor.java

```
package mainpck;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Random;

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

//Update Sensor data
public class UpdateSensor {
	private static HttpURLConnection conn;

	//Generate Random Number
	public static int getRandom() {
		Random random = new Random();
		int rand = 0;
		while (true){
		    rand = random.nextInt(101);
		    if(rand !=0) break;
		}

		//Generate Random number with probability
		if (rand <= 1) {
			return 10;
		} else if (rand <= 3) {
			return 9;
		} else if (rand <= 7) {
			return 8;
		} else if (rand <= 13) {
			return 7;
		} else if (rand <= 20) {
			return 6;
		} else if (rand <= 28) {
			return 5;
		} else if (rand <= 38) {
			return 4;
		} else if (rand <= 54) {
			return 3;
		} else if (rand <= 75) {
			return 2;
		}
		else {
			return 1;
		}
```

```
	}

	//To get all Sensor List
	public static void getList() {
		BufferedReader reader;
		String line;
		StringBuffer responseContent = new StringBuffer();
		try{
			URL url = new
URL("http://localhost:8080/firealamrest/webresources/sensors");
			conn = (HttpURLConnection) url.openConnection();

			conn.setRequestMethod("GET");
			conn.setConnectTimeout(5000);
			conn.setReadTimeout(5000);

			int status = conn.getResponseCode();
			System.out.println(status);

			if (status>299) {
				reader = new BufferedReader(new
InputStreamReader(conn.getErrorStream()));
				while((line = reader.readLine()) != null) {
					responseContent.append(line);
				}
				reader.close();
			}
			else {
				reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
				while((line = reader.readLine()) != null) {
					responseContent.append(line);
				}
				reader.close();
			}

	System.out.println(responseContent.toString().substring(10,responseContent.
toString().substring(1).length()));

			JSONArray sensors = new
JSONArray(responseContent.toString().substring(10,responseContent.toString().subst
ring(1).length()));

			for(int i=0; i<sensors.length(); i++) {
				JSONObject sensor = sensors.getJSONObject(i);
				int id = sensor.getInt("sensorID");
				boolean sensorStatus =
sensor.getBoolean("sensorStatus");
				System.out.println(id);
				System.out.println(sensorStatus);
				if(sensorStatus) {
					replaceSensorData(id);
				}
			}
		} catch(Exception e) {
			e.printStackTrace();
```

```
            }
        }

        //This method will update Sensor data (co2 level and smoke level)
        public static void replaceSensorData(int sensorID) {
            BufferedReader reader;
            String line;
            StringBuffer responseContent = new StringBuffer();
            int co2 = getRandom();
            int smoke = getRandom();
            try {
                URL url = new
URL("http://localhost:8080/firealamrest/webresources/sensors/data");
                conn = (HttpURLConnection) url.openConnection();

                conn.setRequestMethod("PUT");
                conn.setRequestProperty("Content-Type", "application/json");
                conn.setDoOutput(true);
                String jsonInputString = "{\"sensorID\":
\""+sensorID+"\",\"co2Level\": \""+co2+"\",\"smokeLevel\": \""+smoke+"\"}";
                try(OutputStream os = conn.getOutputStream()) {
                    byte[] input = jsonInputString.getBytes("utf-8");
                    os.write(input, 0, input.length);
                }
                conn.setConnectTimeout(5000);
                conn.setReadTimeout(5000);

                int status = conn.getResponseCode();
                System.out.println(status);

                if (status>299) {
                    reader = new BufferedReader(new
InputStreamReader(conn.getErrorStream()));
                    while((line = reader.readLine()) != null) {
                        responseContent.append(line);
                    }
                    reader.close();
                }
                else {
                    reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                    while((line = reader.readLine()) != null) {
                        responseContent.append(line);
                    }
                    reader.close();
                }
                System.out.println(responseContent.toString());
            } catch (Exception e) {
                e.printStackTrace();
            } finally {
                conn.disconnect();
            }
        }


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

```
		//Calling getList every 10 second
		while(true) {
			getList();
			try {
				Thread.sleep(10000);
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
	}
}
```

5.3 Web App

App.js

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
import Countdown from "react-countdown";

export default class DataFetch extends React.Component{

  //Declare count down,  Sensor data and loading status
  state = {
    loading: true,
    sensor: null,
    seconds: 40
  };

  //Update countdown
  tick() {
    this.setState(state => ({
      seconds: state.seconds - 1
    }));
  }

  //Get data
  async componentDidMount(){

    //Get data for first loading
    const url = "http://localhost:8080/firealamrest/webresources/sensors";

    const response = await fetch(url);
    const data = await response.json();

    this.setState({
      sensor: data.sensor,
      loading: false,
      seconds: 40
    })

    this.interval = setInterval(() => this.tick(), 1000);

    //Update data every 40 seconds
    try {
      setInterval(async () => {
        const url = "http://localhost:8080/firealamrest/webresources/sensors";

        const response = await fetch(url);
        const data = await response.json();

        this.setState({
          sensor: data.sensor,
          loading: false,
          seconds: 40
        })
        const renderer = ({ seconds }) => {
          return <span>{seconds}</span>;
```

```
    };
   }, 40000);
  } catch(e) {
    console.log(e);
  }
}

//To update countdown
componentWillUnmount() {
  clearInterval(this.interval);
}

//HTML coding and jsx events
//Styling using Bootstrap
render() {
  const renderer = ({ seconds }) => {
    return <span>{seconds}</span>;
  };

  return (
     <div>
      <div className="bubbles">

      </div>

      {this.state.loading ? (
         <div className="spinner-border" role="status">
          <span className="sr-only">Loading...</span>
         </div>
      ) :(
         <div className="container-fluid">
          <h2 className="text-center">Fire Alarm Sensor Monitoring System</h2>
          <h4>Next Update in {this.state.seconds} Seconds</h4>
          <div className="row">
            {this.state.sensor.map(item => {
             if (item.sensorStatus===="true") {
              if (item.co2Level > 5 || item.smokeLevel > 5){
               return(
                 <div>
                  <div className="card bg-danger mb-3 our-card">
                   <h5 className="card-header">
                    Floor No : {item.sensorLocationFloorNo}<br/>
                    Room No: {item.sensorLocationRoomNo}
                   </h5>
                   <div className="card-body">
                    <p className="card-text">
                     CO2 Level    : {item.co2Level}<br/>
                     Smoke Level: {item.smokeLevel}
                    </p>
                   </div>
                   <div className="card-footer">
                    {item.sensorStatus===="true"
                      ? <h6><i className="fas fa-circle circle-active"></i>Active</h6>
                      : <h6><i className="fas fa-circle circle-deactivate"></i>Deactivated</h6>
                    }
                   </div>
                  </div>
                 </div>
               )
             }
             else {
               return(
```

```
                <div>
                 <div className="card bg-success mb-3 our-card">
                  <h5 className="card-header">
                   Floor No : {item.sensorLocationFloorNo}<br/>
                   Room No: {item.sensorLocationRoomNo}
                  </h5>
                  <div className="card-body">
                   <p className="card-text">
                    CO2 Level    : {item.co2Level}<br/>
                    Smoke Level: {item.smokeLevel}
                   </p>
                  </div>
                  <div className="card-footer">
                   {item.sensorStatus=="true"
                     ? <h6><i className="fas fa-circle circle-active"></i>Active</h6>
                     : <h6><i className="fas fa-circle circle-deactivate"></i>Deactivated</h6>
                   }
                  </div>
                 </div>
                </div>
              )
             }
            } else {
             return(
               <div>
                <div className="card bg-secondary mb-3 our-card">
                 <h5 className="card-header">
                  Floor No : {item.sensorLocationFloorNo}<br/>
                  Room No: {item.sensorLocationRoomNo}
                 </h5>
                 <div className="card-body">
                  <p className="card-text">
                   CO2 Level    : {item.co2Level}<br/>
                   Smoke Level: {item.smokeLevel}
                  </p>
                 </div>
                 <div className="card-footer">
                  {item.sensorStatus=="true"
                    ? <h6><i className="fas fa-circle circle-active"></i>Active</h6>
                    : <h6><i className="fas fa-circle circle-deactivate"></i>Deactivated</h6>
                  }
                 </div>
                </div>
               </div>
             )
            }
           })}

          </div>
        </div>

      )}
     </div>
   );
  }
}
```

App.css

```
.fa-circle{
  font-size: 13px;
}

.circle-active{
  color: forestgreen;
}

.circle-deactivate{
  color: darkred;
}

.our-card{
  margin: 50px 30px 10px;
}
```

5.4 Email & SMS App

MainClass.java

```
import javax.mail.*;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

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

import com.twilio.Twilio;
import com.twilio.type.PhoneNumber;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Properties;

public class MainClass {
	private static HttpURLConnection conn;
	private static Connection sqlConn;

	//Get Admin Details
	public static ArrayList<String> getInfo() {

		String url = "jdbc:mysql://localhost:3307/fireAlarmAPI";
		String un = "root";
		String pwd = "root";
		try {
			Class.forName("com.mysql.jdbc.Driver");
			sqlConn = DriverManager.getConnection(url,un,pwd);
		} catch (SQLException e) {
			e.printStackTrace();
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		}

		ArrayList<String> strArr = new ArrayList<String>();
		String sql = "select email,contactno from users";
		String email = "";
		String contactNo = "";
		try {
			Statement st = sqlConn.createStatement();
			ResultSet rs = st.executeQuery(sql);
```

```
                    while(rs.next()) {
                            email = rs.getString(1);
                            contactNo = rs.getString(2);
                            strArr.add(email);
                            strArr.add(contactNo);
                    }
                    sqlConn.close();
            }
            catch (Exception e) {
                    System.out.println(e);
            }

            return strArr;
    }

    //To send email to relevant Admin
    public static void sendMail(String sendMail,String floor, String room) {
            final String username = "milindaranawaka2@gmail.com";
      final String password = "This_is_Temporary_pwd_225588";

      Properties prop = new Properties();
            prop.put("mail.smtp.host", "smtp.gmail.com");
      prop.put("mail.smtp.port", "587");
      prop.put("mail.smtp.auth", "true");
      prop.put("mail.smtp.starttls.enable", "true"); //TLS
      prop.put("mail.smtp.ssl.trust", "smtp.gmail.com");

      Session session = Session.getInstance(prop,
              new javax.mail.Authenticator() {
                  protected PasswordAuthentication getPasswordAuthentication() {
                      return new PasswordAuthentication(username, password);
                  }
              });

      try {

          Message message = new MimeMessage(session);
          message.setFrom(new InternetAddress(username));
          message.setRecipients(
                  Message.RecipientType.TO,
                  InternetAddress.parse(sendMail)
          );
          message.setSubject("Fire Alarm Warning");
          String bodyText="Fire Sensor triggered!\nFloor no: "+floor+" in Room
no: "+room+"";
          message.setText(bodyText);

          Transport.send(message);

          System.out.println("Email sent");

      } catch (MessagingException e) {
          e.printStackTrace();
      }
    }
```

```
	//Get Sensor details and check co2 or smoke level is greater than 5
	public static void getList() {
		BufferedReader reader;
		String line;
		StringBuffer responseContent = new StringBuffer();
		try{
			URL url = new
URL("http://localhost:8080/firealamrest/webresources/sensors");
			conn = (HttpURLConnection) url.openConnection();

			conn.setRequestMethod("GET");
			conn.setConnectTimeout(5000);
			conn.setReadTimeout(5000);

			int status = conn.getResponseCode();

			if (status>299) {
				reader = new BufferedReader(new
InputStreamReader(conn.getErrorStream()));
				while((line = reader.readLine()) != null) {
					responseContent.append(line);
				}
				reader.close();
			}
			else {
				reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
				while((line = reader.readLine()) != null) {
					responseContent.append(line);
				}
				reader.close();
			}

			JSONArray sensors = new
JSONArray(responseContent.toString().substring(10,responseContent.toString().subst
ring(1).length()));

			for(int i=0; i<sensors.length(); i++) {
				JSONObject sensor = sensors.getJSONObject(i);
				int id = sensor.getInt("sensorID");
				int co2 = sensor.getInt("co2Level");
				int smoke = sensor.getInt("smokeLevel");
				String floor =
sensor.getString("sensorLocationFloorNo");
				String room = sensor.getString("sensorLocationRoomNo");

				if((smoke > 5)||(co2 > 5)) {
					sendMail(getInfo().get(0), floor, room);
					System.out.println("SMS Send to :
"+getInfo().get(1)+" Number");

					//Twilio SMS Function Call
					//SMSSend(getInfo().get(1),floor,room);
				}
			}
		} catch(Exception e) {
```

```
                    e.printStackTrace();
                }
        }

        public static void main(String[] args) {
                //calling this every 5 second
                while(true) {
                        getList();
                        try {
                                Thread.sleep(5000);
                        } catch (InterruptedException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }
                }
    }


        //SMS Using twilio message service
        /*
        public static void SMSSend(String to,String floor, String room) {
                String ACCOUNT_SID = "AC5ef872f6da5a21de157d80997a64bd33";
            String AUTH_TOKEN = "TOKEN";

            Twilio.init(ACCOUNT_SID, AUTH_TOKEN);
            com.twilio.rest.api.v2010.account.Message message =
com.twilio.rest.api.v2010.account.Message
                    .creator(new PhoneNumber(to), new PhoneNumber("+94776603675"),
                      "Floor no: F5 in Room no: N502 \nFloor no: "+floor+" in Room
no: "+room+"")
                    .create();
                  System.out.println(message.getSid());
                  }
         */
}
```

To pom.xml dependency

```
<dependency>
        <groupId>com.sun.mail</groupId>
        <artifactId>javax.mail</artifactId>
        <version>1.6.2</version>
</dependency>
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.19</version>
</dependency>
<dependency>
        <groupId>com.twilio.sdk</groupId>
        <artifactId>twilio</artifactId>
        <version>7.50.0</version>
</dependency>
```

5.5 RMI Server

RMIServiceInterface.java

```
package firesensor;

import Models.Sensor;
import Models.User;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

public interface RMIServiceInterface extends Remote{

    void addSensorData(Sensor s)throws RemoteException;

    void updateSensorData(Sensor s)throws RemoteException;

    User searchUser(User u) throws RemoteException;

    Sensor searchSensor(Sensor s)throws RemoteException;

    List<User> allUser() throws RemoteException;

    List<Sensor> allSensors() throws RemoteException;

}
```

RMIServer.java

```
package firesensor;

import Models.Sensor;
import Models.User;
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;

import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
```

```
import java.sql.*;

public class RMIServer extends UnicastRemoteObject implements RMIServiceInterface {

   private List<User> userList;
   private List<Sensor> sensorList;
   private Connection con = null;


private static HttpURLConnection conn;

   public RMIServer() throws RemoteException {
      super();
      System.out.println("Server start");
      initializeData();
   }

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

         Registry reg = LocateRegistry.createRegistry(9999);
         reg.rebind("rmi_server", new RMIServer());
         System.err.println("Server ready");

      } catch (Exception e) {

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

      }
   }



   private void initializeData() {

      userList = new ArrayList<>();
      sensorList = new ArrayList<>();

      serverUpdate();

   }

   public void addSensorData(Sensor s) {

      BufferedReader reader;
      String line;
      StringBuffer responseContent = new StringBuffer();

      try {

         URL url = new URL("http://localhost:8080/firealamrest/webresources/sensors/sensor");
         conn = (HttpURLConnection) url.openConnection();

         conn.setRequestMethod("POST");
         conn.setRequestProperty("Content-Type", "application/json");
         conn.setDoOutput(true);
```

```
        String jsonInputString = "{\"co2Level\": \"0\",\"sensorID\":
\""+s.getSensorID()+"\",\"sensorLocationFloorNo\":
\""+s.getSensorLocationFloorNo()+"\",\"sensorLocationRoomNo\":
\""+s.getSensorLocationRoomNo()+"\",\"sensorStatus\": \""+s.getSensorStatus()+"\",\"smokeLevel\":
\"0\"}";

        try(OutputStream os = conn.getOutputStream()) {
           byte[] input = jsonInputString.getBytes("utf-8");
           os.write(input, 0, input.length);
        }

        conn.setConnectTimeout(5000);
        conn.setReadTimeout(5000);

        int status = conn.getResponseCode();
        System.out.println(status);

        if (status>299) {
          reader = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
          while((line = reader.readLine()) != null) {
            responseContent.append(line);
          }
          reader.close();
        }
        else {
          reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
          while((line = reader.readLine()) != null) {
            responseContent.append(line);
          }
          reader.close();
        }
        System.out.println(responseContent.toString());
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        conn.disconnect();
    }
  }

  public void updateSensorData(Sensor s) {

    BufferedReader reader;
    String line;
    StringBuffer responseContent = new StringBuffer();

    try {
      URL url = new URL("http://localhost:8080/firealamrest/webresources/sensors/update");
      conn = (HttpURLConnection) url.openConnection();

      conn.setRequestMethod("PUT");
      conn.setRequestProperty("Content-Type", "application/json");
      conn.setDoOutput(true);
      String jsonInputString = "{\"sensorID\": \""+s.getSensorID()+"\",\"sensorLocationFloorNo\":
\""+s.getSensorLocationFloorNo()+"\",\"sensorLocationRoomNo\":
\""+s.getSensorLocationRoomNo()+"\",\"sensorStatus\": \""+s.getSensorStatus()+"\"}";
```

```
    try(OutputStream os = conn.getOutputStream()) {
      byte[] input = jsonInputString.getBytes("utf-8");
      os.write(input, 0, input.length);
    }
    conn.setConnectTimeout(5000);
    conn.setReadTimeout(5000);

    int status = conn.getResponseCode();
    System.out.println(status);

    if (status>299) {
      reader = new BufferedReader(new InputStreamReader(conn.getErrorStream()));
      while((line = reader.readLine()) != null) {
        responseContent.append(line);
      }
      reader.close();
    }
    else {
      reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
      while((line = reader.readLine()) != null) {
        responseContent.append(line);
      }
      reader.close();
    }
    System.out.println(responseContent.toString());
  } catch (Exception e) {
    e.printStackTrace();
  } finally {
    conn.disconnect();
  }
}

@Override
public User searchUser(User user) throws RemoteException {

  Predicate<User> predicate = x -> x.getUserID() == user.getUserID();

  return userList.stream().filter(predicate).findFirst().get();

}

@Override
public List<User> allUser() throws RemoteException {

  return userList;

}

@Override
public Sensor searchSensor(Sensor s) throws RemoteException {

  Predicate<Sensor> predicate = x -> x.getSensorID() == s.getSensorID();

  return sensorList.stream().filter(predicate).findFirst().get();

}
```

```
@Override
public List<Sensor> allSensors() throws RemoteException {

    return sensorList;
}

private void serverUpdate() {
    System.out.println("Server Update Function Execute");

    String url = "jdbc:mysql://localhost:3307/fireAlarmAPI";
    String un = "root";
String pwd = "root";

    try {
        Class.forName("com.mysql.jdbc.Driver");
        con = DriverManager.getConnection(url,un,pwd);
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }

    String sql = "select * from users";

    try {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery(sql);

        while(rs.next()) {
            User user = new User();
            user.setUserID(rs.getInt(1));
            user.setUsername(rs.getString(2));
            user.setPassword(rs.getString(3));
            user.setEmail(rs.getString(4));
            user.setContactno(rs.getString(5));

            userList.add(user);
        }
    }
    catch (Exception e) {
        System.out.println(e);
    }

    Thread thread = new Thread(new Runnable() {

        @Override
        public void run() {
            try {
                System.out.println("Server Thread Start");
                apiRequests();
                System.out.println("15sec Delay");
                Thread.sleep(15000);
                serverUpdate();
            } catch (Exception e) {
                System.out.println("Thread Exception : " + e);
```

```
            }

        }
    }
    );

    thread.start();

}

private 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;
}



private void apiRequests() {
    System.out.println("Send API Requests");

    String sensor_response = jsonRequest("http://localhost:8080/firealamrest/webresources/sensors");
    System.out.println("sensor_response"+sensor_response);

    if (sensor_response != null) {
        try {

            JSONParser jsonParser = new JSONParser();

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

            JSONArray array = (JSONArray) responseObj.get("sensor");

            sensorList.clear();

            for (Object obj : array) {
                JSONObject jSONObject = (JSONObject) obj;
                sensorList.add(
                        new Sensor(
                                Integer.parseInt(jSONObject.get("sensorID").toString()),
                                Integer.parseInt(jSONObject.get("co2Level").toString()),
                                (jSONObject.get("sensorLocationFloorNo") == null) ? "" :
```

```
jSONObject.get("sensorLocationFloorNo").toString(),
                        (jSONObject.get("sensorLocationRoomNo") == null) ? "" :
jSONObject.get("sensorLocationRoomNo").toString(),
                        Boolean.parseBoolean(jSONObject.get("sensorStatus").toString()),
                        Integer.parseInt(jSONObject.get("smokeLevel").toString()))
            );
          }

          System.out.println("Response from " + sensorList.size() + " sensors");

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

    }

}
```

RMIClient.java

```
package firesensor;

import interfaces.Login;
import java.net.MalformedURLException;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;


public class RMIClient {

    private static RMIServiceInterface rmiInterface;


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

        System.out.println("client started");

        RMIClient client = new RMIClient();
        client.connectRemote();

        Login login = new Login();
        login.setVisible(true);

    }

    private void connectRemote() throws RemoteException, NotBoundException {

        Registry reg = LocateRegistry.getRegistry("localhost", 9999);
        rmiInterface =(RMIServiceInterface) reg.lookup("rmi_server");

        System.out.println("connectRemote excuted");
```

```
    }
}
```

Models/User.java

```
package Models;


import java.io.Serializable;

public class User implements Serializable {
    private int userID;
    private String username;
    private String password;
    private String email;
    private String contactno;

    public User() {
    }

    public User(int userID, String username, String password, String email, String contactno) {
        this.userID = userID;
        this.username = username;
        this.password = password;
        this.email = email;
        this.contactno = contactno;
    }

    public int getUserID() {
        return userID;
```

```
}

public void setUserID(int userID) {
    this.userID = userID;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getPassword() {
    return password;
}

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

public String getEmail() {
    return email;
}

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

```
    }


    public String getContactno() {

        return contactno;

    }


    public void setContactno(String contactno) {

        this.contactno = contactno;

    }





}
```

Models/Sensor.java

```
package Models;

import java.io.Serializable;

public class Sensor implements Serializable {

    public int sensorID;
    public int co2Level;
    public String sensorLocationFloorNo;
    public String sensorLocationRoomNo;
    public Boolean sensorStatus;
    public int smokeLevel;


    public Sensor() {
    }

    public Sensor(int sensorID, int co2Level, String sensorLocationFloorNo, String sensorLocationRoomNo,
Boolean sensorStatus, int smokeLevel) {
```

```
    this.sensorID = sensorID;
    this.co2Level = co2Level;
    this.sensorLocationFloorNo = sensorLocationFloorNo;
    this.sensorLocationRoomNo = sensorLocationRoomNo;
    this.sensorStatus = sensorStatus;
    this.smokeLevel = smokeLevel;
  }


  public int getSensorID() {
    return sensorID;
  }

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

  public int getCo2Level() {
    return co2Level;
  }

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

  public String getSensorLocationFloorNo() {
    return sensorLocationFloorNo;
  }

  public void setSensorLocationFloorNo(String sensorLocationFloorNo) {
    this.sensorLocationFloorNo = sensorLocationFloorNo;
  }

  public String getSensorLocationRoomNo() {
    return sensorLocationRoomNo;
  }

  public void setSensorLocationRoomNo(String sensorLocationRoomNo) {
    this.sensorLocationRoomNo = sensorLocationRoomNo;
  }

  public Boolean getSensorStatus() {
    return sensorStatus;
  }

  public void setSensorStatus(Boolean sensorStatus) {
    this.sensorStatus = sensorStatus;
  }

  public int getSmokeLevel() {
    return smokeLevel;
  }

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

5.6 RMI Client

Login.java

```
        private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try{
      Registry  reg = LocateRegistry.getRegistry("localhost",9999);
      RMIServiceInterface rmiInterface =(RMIServiceInterface) reg.lookup("rmi_server");

      List<User> userList = rmiInterface.allUser();

      System.out.println(userList.size());
      userList.size();
      User user = userList.get(0);

      String us = usr.getText();
      String pass = new String(pwd.getPassword());
      if((us.equals(user.getUsername())) && (pass.equals(user.getPassword()))){
        new AdminManagement().setVisible(true);
        this.setVisible(false);
      }else{
        this.setVisible(true);
        JOptionPane.showMessageDialog(this, "Check Usernaem or Password", "Login Error",
JOptionPane.ERROR_MESSAGE);
      }
    } catch(Exception e){
      e.printStackTrace();
    }
        }

        private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    new GuestSensorDetails().setVisible(true);
    this.setVisible(false);
        }
```

AddSensor.java

```
        private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminManagement().setVisible(true);
    this.setVisible(false);
  }

        private void addActionPerformed(java.awt.event.ActionEvent evt) {
```

```
    try{
        Registry  reg = LocateRegistry.getRegistry("localhost",9999);
        RMIServiceInterface rmiInterface =(RMIServiceInterface) reg.lookup("rmi_server");

        Sensor  s2 = new Sensor();

        String st = status.getText();
        boolean b = Boolean.parseBoolean(st);
        s2.setSensorStatus(b);
        s2.setSensorLocationFloorNo(floor.getText());
        s2.setSensorLocationRoomNo(room.getText());

        rmiInterface.addSensorData(s2);

        JOptionPane.showConfirmDialog(this,"Successfully Added","Message",
JOptionPane.OK_CANCEL_OPTION, JOptionPane.INFORMATION_MESSAGE);

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

EditSensor.java

```
        private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminManagement().setVisible(true);
    this.setVisible(false);
  }

        private void searchActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        Registry  reg = LocateRegistry.getRegistry("localhost",9999);
        RMIServiceInterface rmiInterface =(RMIServiceInterface) reg.lookup("rmi_server");

        Sensor  s1 = new Sensor();

        s1.setSensorID(Integer.parseInt(searchID.getText()));

        Sensor s2 = rmiInterface.searchSensor(s1);

        int sid = s2.getSensorID();
        String sd = Integer.toString(sid);
        id.setText(sd);
        boolean st = s2.getSensorStatus();
        String ss = Boolean.toString(st);
```

```
        status.setText(ss);
        String fn = s2.getSensorLocationFloorNo();
        floor.setText(fn);
        String rm = s2.getSensorLocationRoomNo();
        room.setText(rm);
        int sl = s2.getSmokeLevel();
        String sm = Integer.toString(sl);
        smoke.setText(sm);
        int c = s2.getCo2Level();
        String c2 = Integer.toString(c);
        co2.setText(c2);

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

          private void editActionPerformed(java.awt.event.ActionEvent evt) {
      try{
        Registry  reg = LocateRegistry.getRegistry("localhost",9999);
        RMIServiceInterface rmiInterface =(RMIServiceInterface) reg.lookup("rmi_server");

        Sensor  se = new Sensor();

        se.setCo2Level(Integer.parseInt(co2.getText()));
        se.setSensorID(Integer.parseInt(id.getText()));
        se.setSensorLocationFloorNo(floor.getText());
        se.setSensorLocationRoomNo(room.getText());
        se.setSensorStatus(Boolean.parseBoolean(status.getText()));
        se.setSmokeLevel(Integer.parseInt(smoke.getText()));

        rmiInterface.updateSensorData(se);

        JOptionPane.showConfirmDialog(this,"Successfully Edited","Message",
JOptionPane.OK_CANCEL_OPTION, JOptionPane.INFORMATION_MESSAGE);

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

SensorDetails.java

```
        public SensorDetails() {
    initComponents();
    showSensorDetails();
    DetailsUpdate();
}

        private void DetailsUpdate() {
    Thread th = new Thread(new Runnable() {

      @Override
      public void run() {
        try {
          showSensorDetails();
          Thread.sleep(30000);
          DetailsUpdate();
        } catch (Exception e) {
          System.out.println("Thread Exception : " + e);
        }

      }
    }
    );

    th.start();

}

        public void showSensorDetails(){
    try{
      Registry  reg = LocateRegistry.getRegistry("localhost",9999);
      RMIServiceInterface rmiInterface =(RMIServiceInterface) reg.lookup("rmi_server");

      List<Sensor> lst = rmiInterface.allSensors();

      DefaultTableModel md = (DefaultTableModel) usertable.getModel();

      md.setRowCount(0);

      Object rowData[] = new Object[6];

      for(int i=0;i<lst.size();i++){
        rowData[0] = lst.get(i).getSensorID();
        rowData[1] = lst.get(i).getSensorStatus();
```

```
            rowData[2] = lst.get(i).getSensorLocationFloorNo();
            rowData[3] = lst.get(i).getSensorLocationRoomNo();
            rowData[4] = lst.get(i).getSmokeLevel();
            rowData[5] = lst.get(i).getCo2Level();

            md.addRow(rowData);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}

        private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    new AdminManagement().setVisible(true);
    this.setVisible(false);
}
```

AdminManagement.java

```
        private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    new SensorDetails().setVisible(true);
    this.setVisible(false);
}

        private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    new AddSensor().setVisible(true);
    this.setVisible(false);
}

        private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    new EditSensor().setVisible(true);
    this.setVisible(false);
}

        private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    new Login().setVisible(true);
    this.setVisible(false);
}
```

RMIClient.java

```
        Login login = new Login();
        login.setVisible(true);
```

GuestSensorDetails.java

```
    public GuestSensorDetails() {
     initComponents();
     showSensorDetails();
     DetailsUpdate();
  }

        private void DetailsUpdate() {
     Thread th = new Thread(new Runnable() {

        @Override
        public void run() {
           try {
              showSensorDetails();
              Thread.sleep(30000);
              DetailsUpdate();
           } catch (Exception e) {
              System.out.println("Thread Exception : " + e);
           }

        }
     }
     );

     th.start();

  }

        public void showSensorDetails(){
     try{
        Registry  reg = LocateRegistry.getRegistry("localhost",9999);
        RMIServiceInterface rmiInterface =(RMIServiceInterface) reg.lookup("rmi_server");

        List<Sensor> lst = rmiInterface.allSensors();

        DefaultTableModel md = (DefaultTableModel) usertable.getModel();

        md.setRowCount(0);

        Object rowData[] = new Object[6];

        for(int i=0;i<lst.size();i++){
           rowData[0] = lst.get(i).getSensorID();
           rowData[1] = lst.get(i).getSensorStatus();
```

```
            rowData[2] = lst.get(i).getSensorLocationFloorNo();
            rowData[3] = lst.get(i).getSensorLocationRoomNo();
            rowData[4] = lst.get(i).getSmokeLevel();
            rowData[5] = lst.get(i).getCo2Level();

            md.addRow(rowData);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
```

5.7 Database

```
#Database Creation
create database fireAlarmAPI;
use fireAlarmAPI;

#Table
create table sensors(
        sensorID int NOT NULL AUTO_INCREMENT,
        sensorStatus boolean,
        sensorLocationFloorNo varchar(20),
        sensorLocationRoomNo varchar(20),
        smokeLevel int,
        co2Level int,
  primary key (sensorID)
);

create table users(
        userID int NOT NULL AUTO_INCREMENT,
        username varchar(20),
        password varchar(20),
  email varchar(500),
        contactno varchar(20),
  primary key (userID)
);

#Data
insert into
sensors(sensorStatus,sensorLocationFloorNo,sensorLocationRoomNo,smokeLevel,co2Level)
values(true,"F5","N502",2,3);
insert into
sensors(sensorStatus,sensorLocationFloorNo,sensorLocationRoomNo,smokeLevel,co2Level)
values(false,"F3","N302",1,1);

insert into users(username,password,email,contactno)
values('admin','admin','milindaranawaka@gmail.com','+94718956912');
#Verify
select * from sensors;
select * from users;
```