Fire Alarm Monitoring System

DS Assignment 2

Group Members

Registration Number	Name
IT18148428	K.G.G.H. Silva
IT18040968	W.A.P.S. Abeyasekare
IT18094800	T. B. Nandisena
IT18000672	D.M.H.E. Dasanayake

Contents

INTRODUCTION

The fire alarm monitoring system is a real time application where users are capable of handling fire sensor details. This system is constructed upon a rich technology stack such as Spring Boot, Java RMI, Java FX and React. The specialty of this system is that it provides real time data to the users by providing a special functionality of sending real time elevated levels of smoke and co2 via SMS and email. For this system to work properly there are several components that needs to work together, all these components are important for this system as a whole,

•A database which stores,

-fire alarm details such as id, floor number, room number, smoke level and CO2 level.

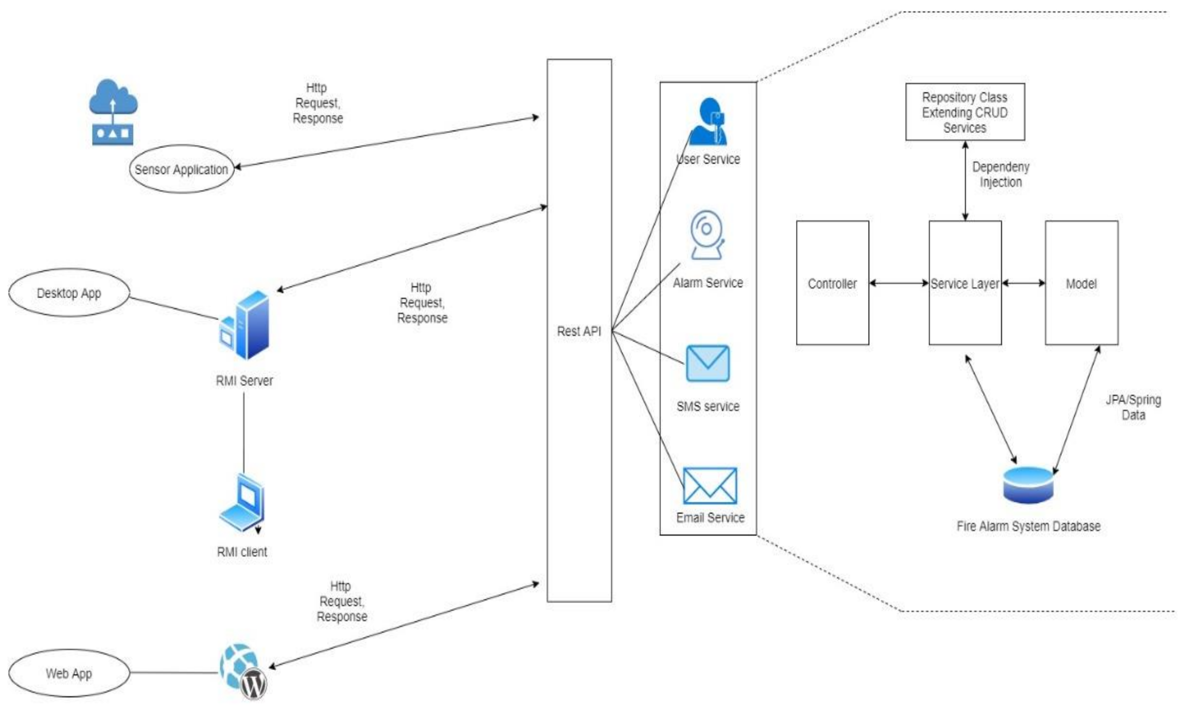
-User details such as email and password.

•A web application where the user can view real time data of each alarm of the system, which automatically fetches updated sensor details every 40 seconds using the REST API.

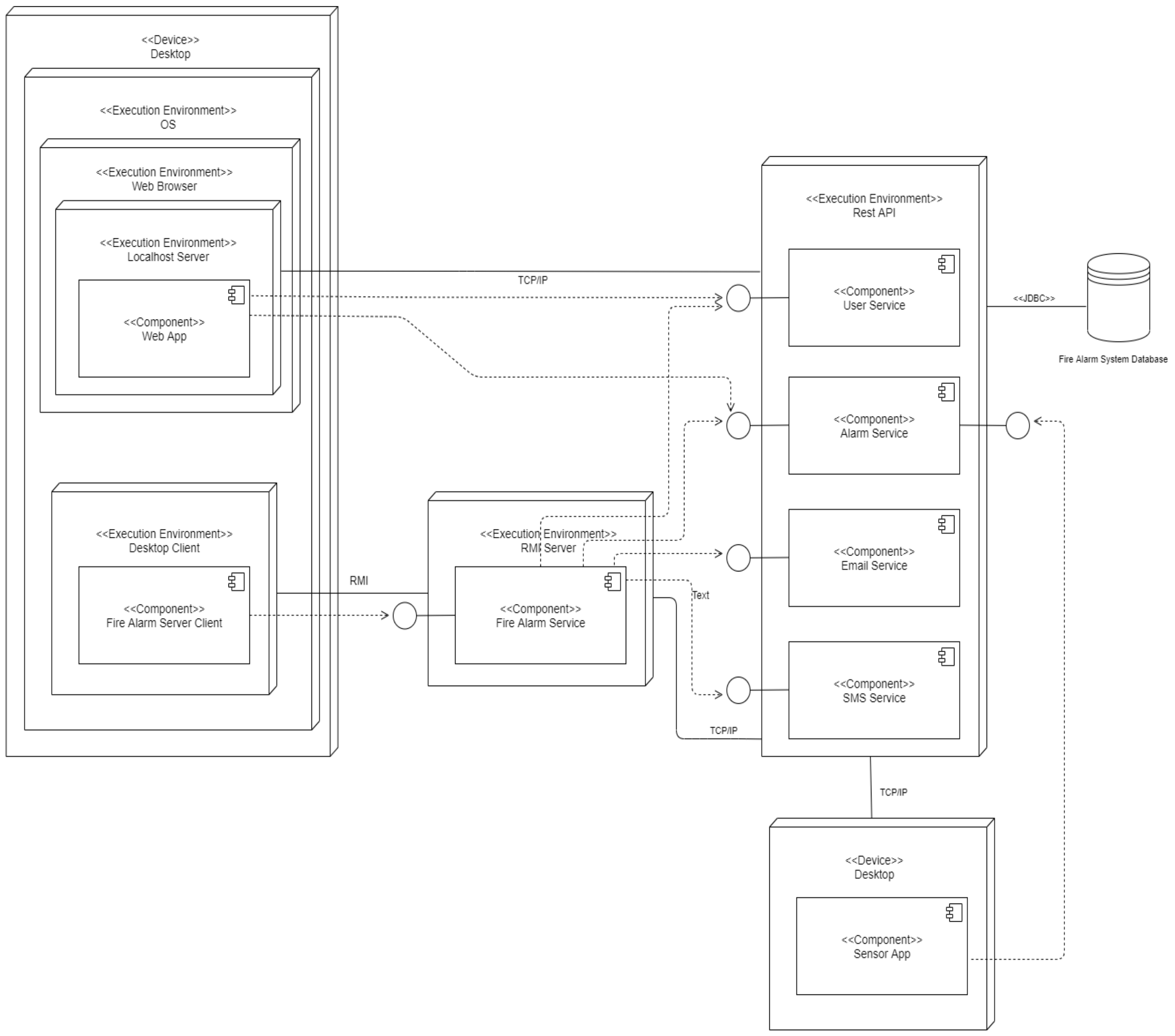
•A desktop client application where the user can view, add, update and delete alarm details, which automatically fetches updated sensor details every 15 seconds using the REST API.

•A sensor application which acts as a dummy sensor which updates the rest API with new sensor details every 10 seconds using the REST API.

HIGH LEVEL ARCHITECTURAL DIAGRAM



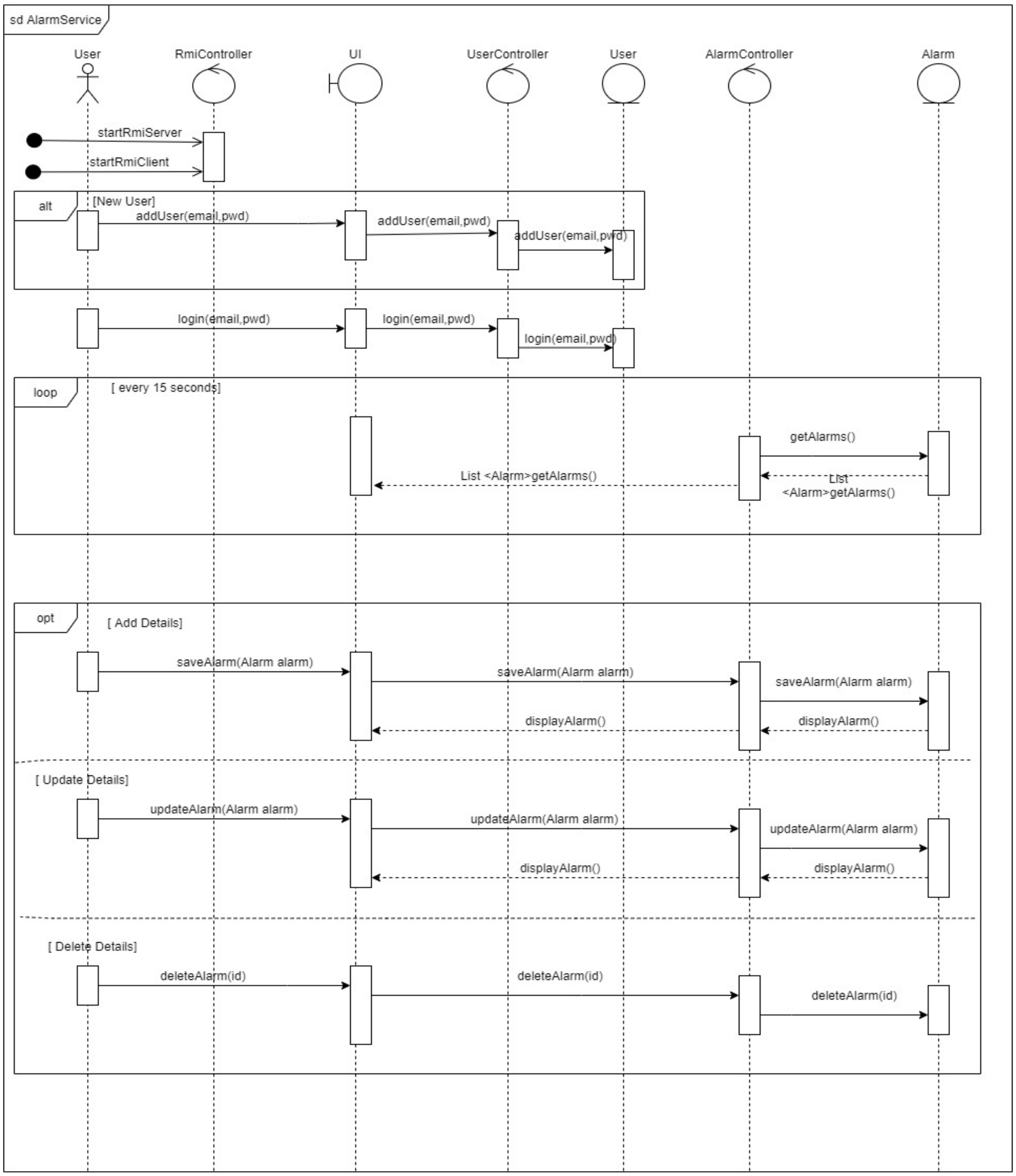
SYSTEM LAYOUT



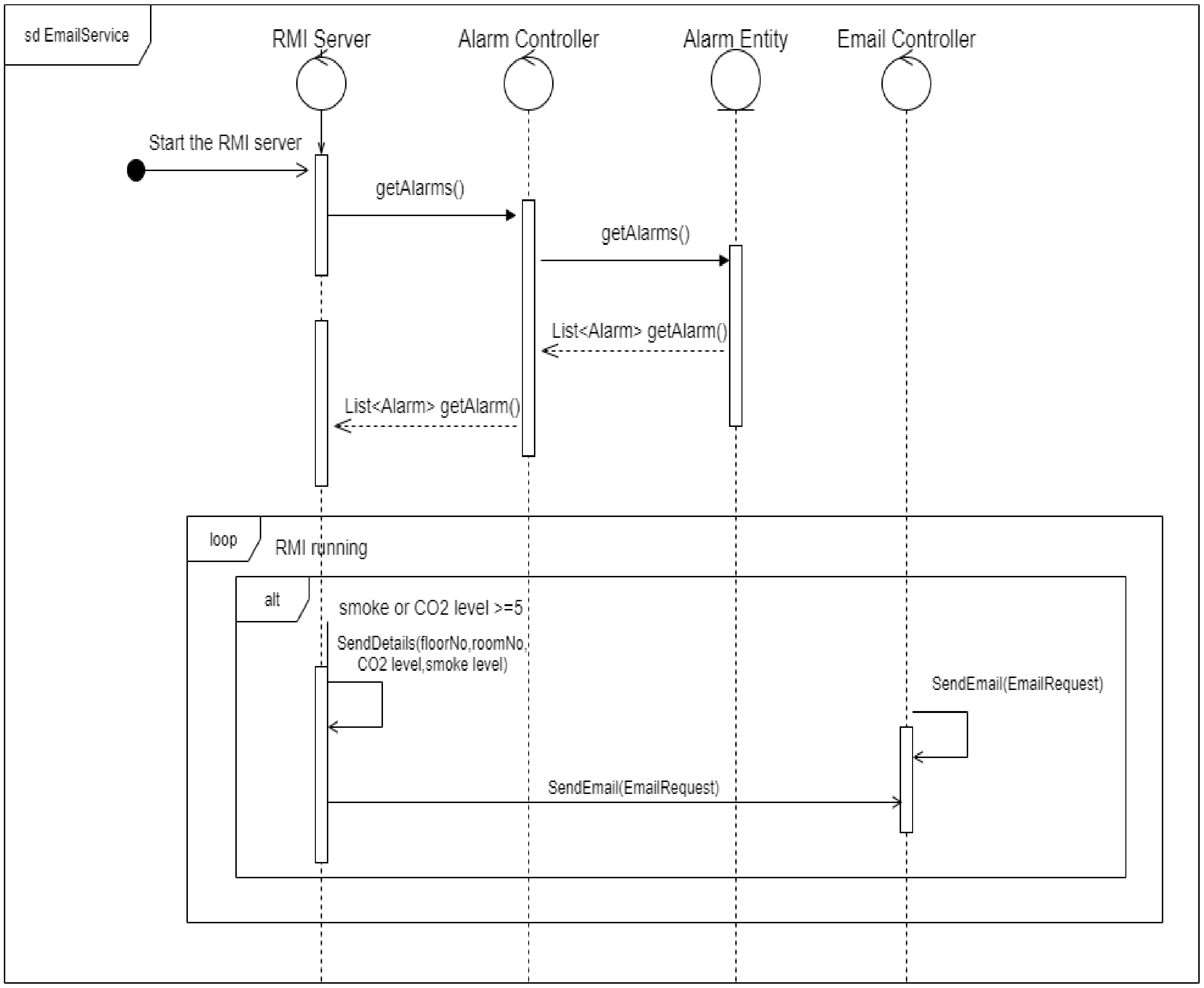
<<Device>> Desktop
<<Execution Environment>> OS
<<Execution Environment>> Web Browser
<<Execution Environment>> Localhost Server
<<Component>> Web App
<<Execution Environment>> Desktop Client
<<Component>> Fire Alarm Server Client
RMI
<<Execution Environment>> RMI Server
<<Component>> Fire Alarm Service
TCP/IP
Text
<<Execution Environment>> Rest API
<<Component>> User Service
<<Component>> Alarm Service
<<Component>> Email Service
<<Component>> SMS Service
<<JDBC>>
Fire Alarm System Database
TCP/IP
TCP/IP
<<Device>> Desktop
<<Component>> Sensor App

SERVICES

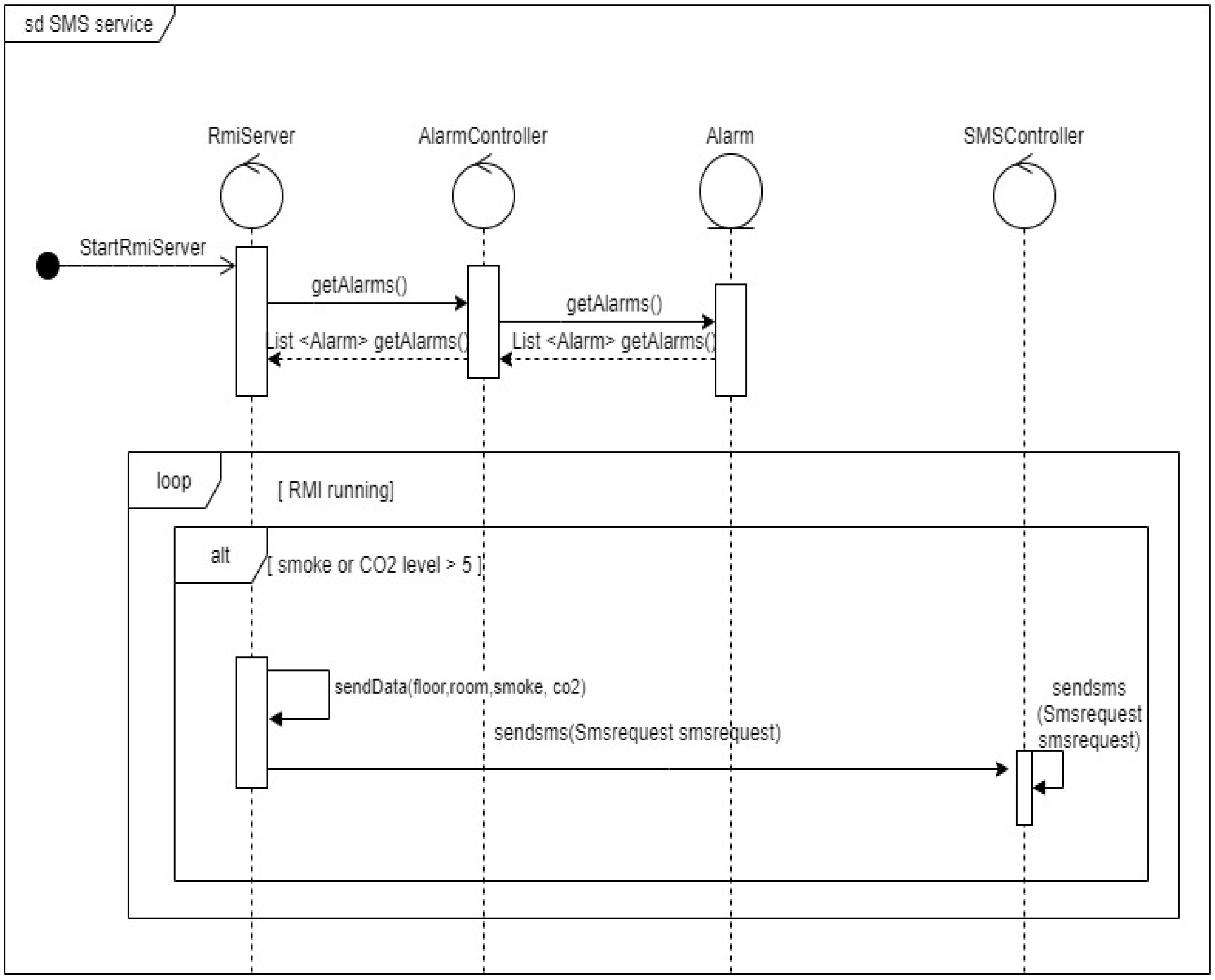
Alarm Service



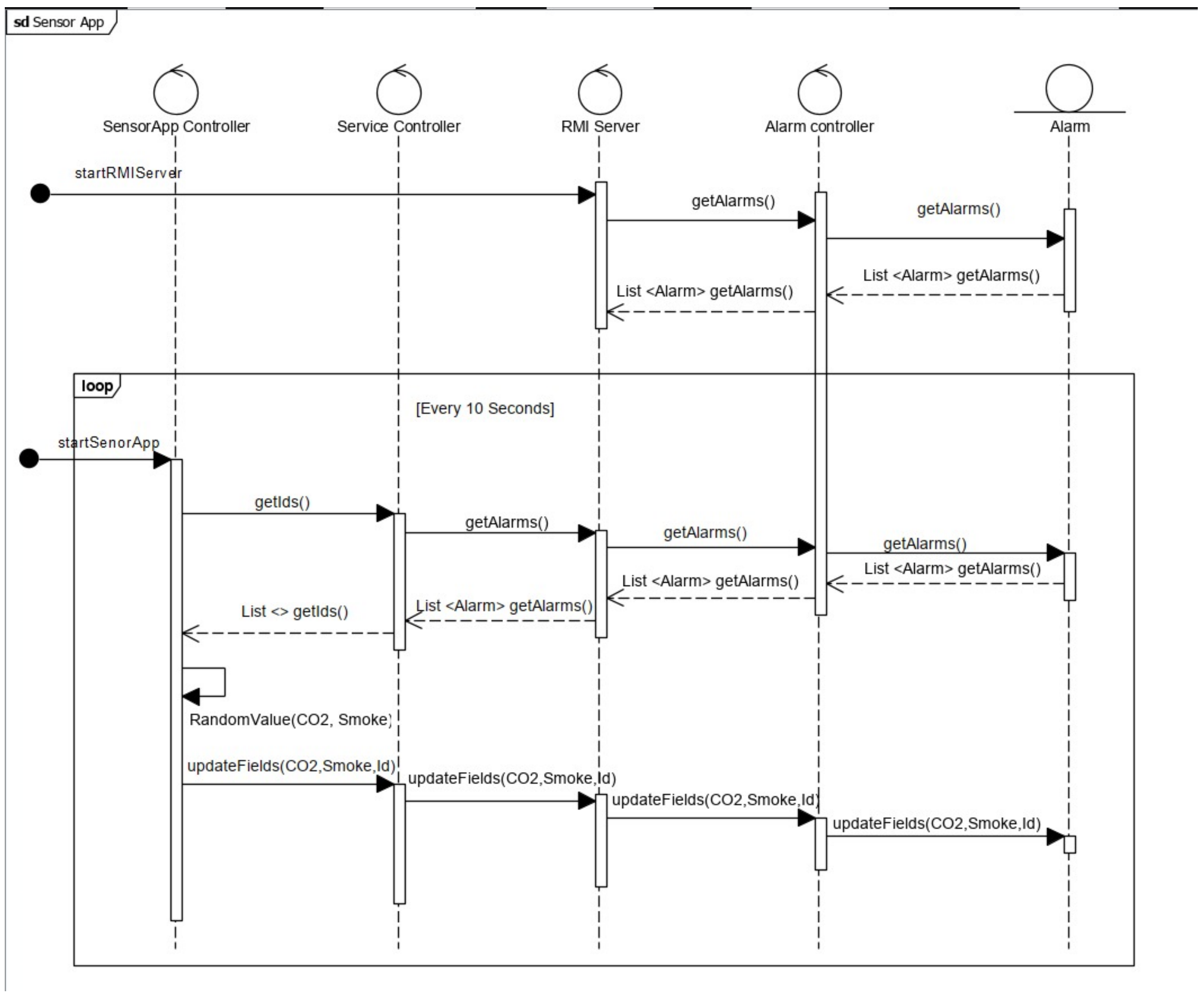
Email Service



SMS Service


sd SMS service
RmiServer
AlarmController
Alarm
SMSController
StartRmiServer
getAlarms()
getAlarms()
List <Alarm> getAlarms()
List <Alarm> getAlarms()
loop
[RMI running]
alt
[smoke or CO2 level > 5]
sendData(floor,room,smoke, co2)
sendsms(Smsrequest smsrequest)
sendsms (Smsrequest smsrequest)

Sensor App


sd Sensor App
SensorApp Controller
Service Controller
RMI Server
Alarm controller
Alarm
startRMIServer
getAlarms()
getAlarms()
List <Alarm> getAlarms()
List <Alarm> getAlarms()
loop
[Every 10 Seconds]
startSenorApp
getIds()
getAlarms()
getAlarms()
getAlarms()
List <Alarm> getAlarms()
List <Alarm> getAlarms()
List <Alarm> getAlarms()
List <> getIds()
RandomValue(CO2, Smoke)
updateFields(CO2,Smoke,Id)
updateFields(CO2,Smoke,Id)
updateFields(CO2,Smoke,Id)
updateFields(CO2,Smoke,Id)

WEB APP

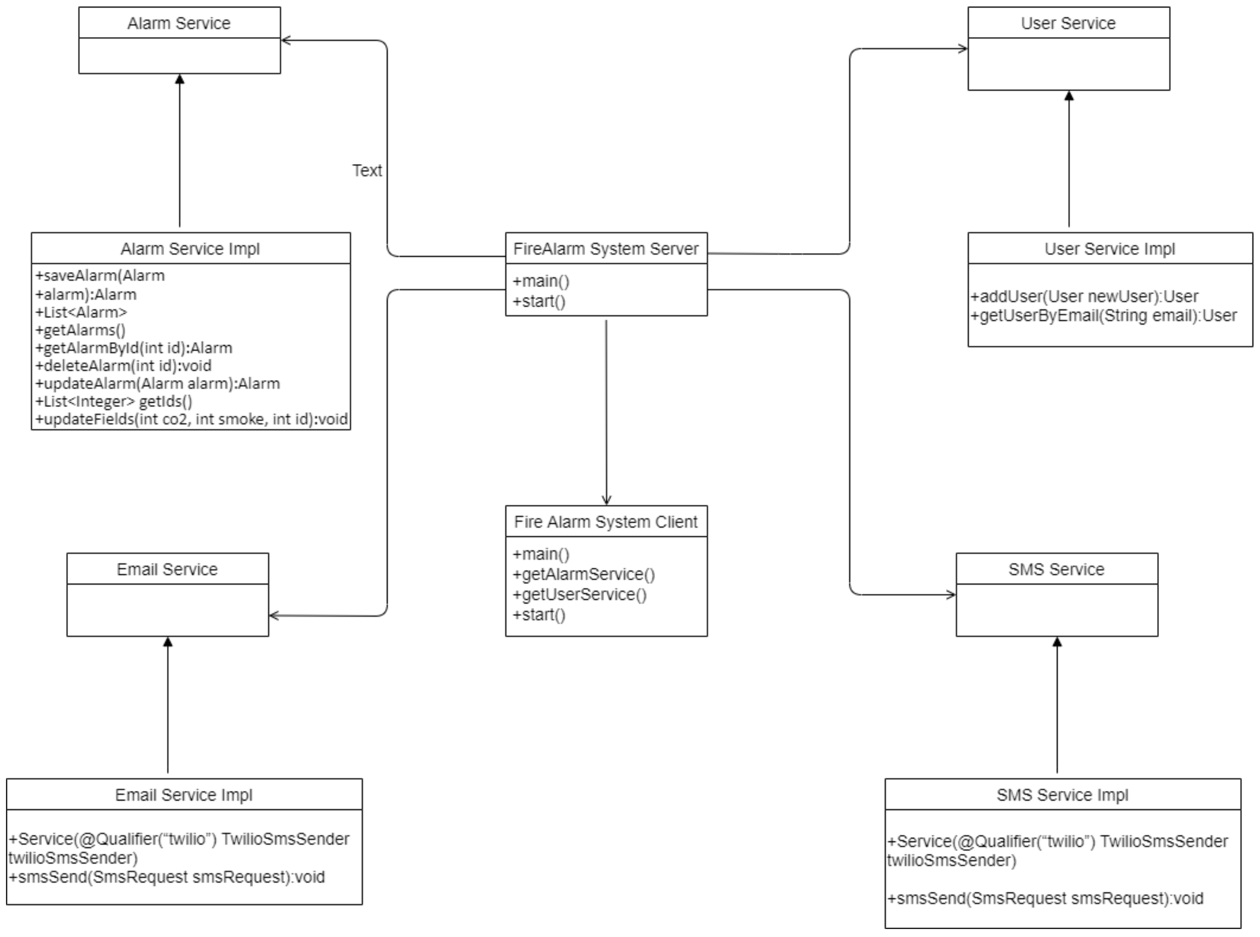
For the web application implementation, we have used React a JavaScript library for building user interfaces. Using axios and fetch method we have created the communication between our REST api and our front-end web application.

REST API

To build our REST API we have used Spring. The Spring framework supports two ways of creating RESTful services. We have used the HTTP message converters instead of the old MVC with ModelAndView. This approach based on HttpMessageConverter and annotations is much more lightweight and easier to implement. The Spring Boot architecture is composed of a controller, service layer and a model. With the inclusion of a database connection a repository class extending JPA is included. In all the services we have implemented using Spring Boot this architecture is followed. Different REST endpoints are implemented as URLs for communication. To handle different request types Spring Boot annotations are used.

RMI SERVER

The RMI Server consists of several interfaces, with different implementations for each interface. All the interfaces are binded by the RMI Server. The RMI Server for every 15 seconds checks for sensor details updates and depending on the provided condition, the server invokes the SMS and Email service. The user and the Alarm service interfaces are invoked by the server through the Fire Alarm System Client.


Alarm Service
Text
User Service
Alarm Service Impl
+saveAlarm(Alarm
+alarm):Alarm
+List<Alarm>
+getAlarms()
+getAlarmById(int id):Alarm
+deleteAlarm(int id):void
+updateAlarm(Alarm alarm):Alarm
+List<Integer> getIds()
+updateFields(int co2, int smoke, int id):void
FireAlarm System Server
+main()
+start()
User Service Impl
+addUser(User newUser):User
+getUserByEmail(String email):User
Fire Alarm System Client
+main()
+getAlarmService()
+getUserService()
+start()
Email Service
SMS Service
Email Service Impl
+Service(@Qualifier("twilio") TwilioSmsSender twilioSmsSender)
+smsSend(SmsRequest smsRequest):void
SMS Service Impl
+Service(@Qualifier("twilio") TwilioSmsSender twilioSmsSender)
+smsSend(SmsRequest smsRequest):void

APPENDIX

Desktop App

1.User

rmiApi/entitiy/User.java

```
package rmiApi.entity;
import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import java.io.Externalizable;
import java.io.IOException;
import java.io.ObjectInput;
import java.io.ObjectOutput;

/*Implement User modal class by defining setters and getters
and constructor and the modal class has implemented the Externalizable interface
by extending the Serializable interface
 */
public class User implements Externalizable {

   private final IntegerProperty id = new SimpleIntegerProperty();
   private  final StringProperty email = new SimpleStringProperty();
   private  final StringProperty password = new SimpleStringProperty();

   public int getId() {
      return id.get();
   }

   public IntegerProperty idProperty() {
      return id;
   }

   public void setId(int id) {
      this.id.set(id);
   }

   public String getEmail() {
      return email.get();
   }

   public StringProperty emailProperty() {
      return email;
   }
```

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

  public String getPassword() {
    return password.get();
  }

  public StringProperty passwordProperty() {
    return password;
  }

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

  @Override
  public void writeExternal(ObjectOutput out) throws IOException {
    out.writeInt(getId());
    out.writeObject(getEmail());
    out.writeObject(getPassword());
  }

  @Override
  public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {
    setId(in.readInt());
    setEmail((String) in.readObject());
    setPassword((String) in.readObject());
  }
}
```

rmiApi/entityservice/userService.java

```
package rmiApi.entityService;
import rmiApi.entity.User;
import java.rmi.Remote;
import java.rmi.RemoteException;

/*The UserService interface is created by extending Remote interface
which will provide the descriptions of methods which can be invoked by
remote clients
 */
public interface UserService extends Remote {

  public User getUser(String email)throws RemoteException;
  public User addUser(User newUser) throws RemoteException;
}
```

rmiServer/serviceImpl/UserServiceImpl.java

```
package rmiServer.serviceImpl;
import com.google.gson.Gson;
import org.apache.http.HttpEntity;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.hildan.fxgson.FxGson;
import rmiApi.entity.User;
import rmiApi.entityService.UserService;
import java.io.IOException;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;


/* Default constructor to throw RemoteException
   from its parent constructor */
public class UserServiceImpl extends UnicastRemoteObject implements UserService {

  public UserServiceImpl() throws RemoteException {
  }

  //method implemenation to getUser details by passing the email as a paramteer
  @Override
  public User getUser(String email) throws RemoteException {
    //Set user to null
    User user = null;
     /*Initialize the apache httpclient used to send http requests and response to the
    rest client
     */
    try (CloseableHttpClient httpclient = HttpClients.createDefault()) {

      //HTTP GET method allows to get user details according to the email using the rest api
      HttpGet httpget = new HttpGet("http://localhost:8080/getUser/"+email);
      System.out.println("Executing request " + httpget.getRequestLine());

      // Create a custom response handler
      ResponseHandler< String > responseHandler = response -> {
        int status = response.getStatusLine().getStatusCode();
        if (status >= 200 && status < 300) {
          HttpEntity entity = response.getEntity();
          return entity != null ? EntityUtils.toString(entity) : null;
        } else {
          throw new ClientProtocolException("Unexpected response status: " + status);
        }
      };
      String responseBody = httpclient.execute(httpget, responseHandler);
      System.out.println("----------------------------------------");
```

```
        System.out.println(responseBody);

        //Convert JSON response to user object
        Gson gson = FxGson.coreBuilder().setPrettyPrinting().disableHtmlEscaping().create();
         user = gson.fromJson(responseBody, User.class);

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

    //Return user
    return user;
}

//method implementation to add a new user by passing a user reference
@Override
public User addUser(User newUser) throws RemoteException {
        //get values from User reference passed
    newUser.getEmail();
    newUser.getPassword();
    /*Initialize the apache httpclient used to send http requests and response to the
    rest client
     */
    try(CloseableHttpClient httpclient = HttpClients.createDefault()){
        //HTTP POST method url add new user using rest api
        HttpPost httpPost = new HttpPost("http://localhost:8080/addUser");
        httpPost.setHeader("Accept", "application/json");
        httpPost.setHeader("Content-type", "application/json");

        //Convert user object to JSON response
        Gson gson = FxGson.coreBuilder().setPrettyPrinting().disableHtmlEscaping().create();
        String json = gson.toJson(newUser);

        //pass the json string request in the entity
        StringEntity stringEntity = new StringEntity(json);
        httpPost.setEntity(stringEntity);
        System.out.println("Executing request " + httpPost.getRequestLine());


        ResponseHandler < String > responseHandler = response -> {
            int status = response.getStatusLine().getStatusCode();
            if (status >= 200 && status < 300) {
                HttpEntity entity = response.getEntity();
                return entity != null ? EntityUtils.toString(entity) : null;
            } else {
                throw new ClientProtocolException("Unexpected response status: " + status);
            }
        };

        String responseBody = httpclient.execute(httpPost, responseHandler);
        System.out.println("----------------------------------------");
        System.out.println(responseBody);

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

    //Return the newly added user
    return newUser;
  }
}
```

views/Register.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>

<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefWidth="600.0" styleClass="login-view" xmlns="http://javafx.com/javafx/8.0.171"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="rmiClient.registerForm.Register">
  <children>
    <HBox styleClass="login-title" />
    <VBox styleClass="login-body">
      <children>
        <GridPane hgap="10.0" vgap="6.0">
          <columnConstraints>
            <ColumnConstraints halignment="RIGHT" hgrow="SOMETIMES" maxWidth="219.0"
minWidth="10.0" prefWidth="142.0" />
            <ColumnConstraints hgrow="SOMETIMES" maxWidth="281.0" minWidth="10.0" prefWidth="280.0"
/>
          </columnConstraints>
          <rowConstraints>
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
              <RowConstraints maxHeight="31.0" minHeight="10.0" prefHeight="31.0" vgrow="SOMETIMES" />
              <RowConstraints maxHeight="30.0" minHeight="10.0" prefHeight="29.0" vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
          </rowConstraints>
           <children>
              <Label fx:id="errorLbl" text="User Registration" GridPane.columnIndex="1" />
              <Label text="Enter Email" GridPane.rowIndex="1" />
              <Label text="Enter Password" GridPane.rowIndex="2" />
              <TextField fx:id="emailID" promptText="Enter User Email" GridPane.columnIndex="1"
GridPane.rowIndex="1" />
              <PasswordField fx:id="password" promptText="Enter User Password" GridPane.columnIndex="1"
GridPane.rowIndex="2" />
              <HBox spacing="10.0" styleClass="al-center-left" GridPane.columnIndex="1"
```

```
GridPane.rowIndex="3">
              <children>
                <Button fx:id="loginBtn" mnemonicParsing="false" onAction="#login" styleClass="wd-120"
text="Login" />
                <Button fx:id="regBtn" mnemonicParsing="false" onAction="#regUSer" text="Register" />
              </children>
            </HBox>
          </children>
        </GridPane>
      </children>
    </VBox>
  </children>
</VBox>
```

views/Login.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.layout.VBox?>

<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefWidth="600.0" styleClass="login-view" xmlns="http://javafx.com/javafx/8.0.171"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="rmiClient.loginForm.Login">
  <children>
    <HBox styleClass="login-title" />
    <VBox styleClass="login-body">
      <children>
        <GridPane hgap="10.0" vgap="6.0">
          <columnConstraints>
            <ColumnConstraints halignment="RIGHT" hgrow="SOMETIMES" maxWidth="219.0"
minWidth="10.0" prefWidth="142.0" />
            <ColumnConstraints hgrow="SOMETIMES" maxWidth="281.0" minWidth="10.0" prefWidth="280.0"
/>
          </columnConstraints>
          <rowConstraints>
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
              <RowConstraints maxHeight="31.0" minHeight="10.0" prefHeight="31.0" vgrow="SOMETIMES" />
              <RowConstraints maxHeight="30.0" minHeight="10.0" prefHeight="29.0" vgrow="SOMETIMES" />
            <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
          </rowConstraints>
           <children>
              <Label fx:id="errorLbl" text="User Login" GridPane.columnIndex="1" />
              <Label text="User Email" GridPane.rowIndex="1" />
              <Label text="User Password" GridPane.rowIndex="2" />
              <TextField fx:id="emailID" promptText="Enter User Email" GridPane.columnIndex="1"
GridPane.rowIndex="1" />
              <PasswordField fx:id="password" promptText="Enter User Password" GridPane.columnIndex="1"
GridPane.rowIndex="2" />
              <HBox spacing="10.0" styleClass="al-center-left" GridPane.columnIndex="1"
GridPane.rowIndex="3">
              <children>
                <Button fx:id="loginBtn" mnemonicParsing="false" onAction="#login" styleClass="wd-120"
text="Login" />
                <Button fx:id="regBtn" mnemonicParsing="false" onAction="#Register" text="Register" />
              </children>
              </HBox>
           </children>
        </GridPane>
      </children>
    </VBox>
  </children>
</VBox>
```

rmiClient/regsiterForm/Register.java

```
package rmiClient.registerForm;
import javafx.event.ActionEvent;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.stage.Modality;
import javafx.stage.Stage;
import rmiApi.entity.User;
import rmiApi.entityService.UserService;
import rmiClient.client.Main;
import rmiClient.loginForm.Login;
import java.io.IOException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.ResourceBundle;

public class Register implements Initializable {
   public TextField emailID;
   public PasswordField password;
   public Button loginBtn;
   public Button regBtn;
```

```java
private Main main;
private UserService userService;

//loginBtn event handler
public void login(ActionEvent actionEvent) throws IOException {
   //On button click show the login view
   Stage stage = (Stage) loginBtn.getScene().getWindow();
   stage.close();

   Stage primaryStage = new Stage();
   FXMLLoader loader = new FXMLLoader(getClass().getResource("/views/Login.fxml"));

   Parent root = loader.load();
   Login login = loader.getController();
   login.setMain(this.main);
   Scene scene = new Scene(root);
   scene.getStylesheets().add(getClass().getResource("/styles/application.css").toExternalForm());
   primaryStage.setScene(scene) ;

   primaryStage.setTitle("Sensor Dashboard Login ");
   primaryStage.show();
}

//regBtn event handler
public void regUSer(ActionEvent actionEvent) throws RemoteException {
   String email = emailID.getText();
   if(checkEmail()){
      if(isValid(email)){
         if(EmailCheck(email)) {
             /* A user reference is initialized and the values from the form fields
            are set to the user reference */
            User u = new User();
            u.setEmail(emailID.getText());
            u.setPassword(password.getText());
            User newU = userService.addUser(u);
            //If there is no user details added to the form, an error alert is shown
            if (newU == null) {
               Alert alert = new Alert(Alert.AlertType.ERROR);
               alert.initModality(Modality.APPLICATION_MODAL);
               alert.setTitle("Unable to Add");
               alert.setHeaderText("Error");
               alert.setContentText("Please Try Again");
               alert.showAndWait();
            } //If correct user details added to the form, an alert is shown indicating the success
            else {
               Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
               alert.initModality(Modality.APPLICATION_MODAL);
               alert.setTitle("Successfully Added ");
               alert.setHeaderText("User Details Registered Successfully");
               alert.setContentText("Please Login");
               alert.showAndWait();
               emailID.setText("");
               password.setText("");
            }
         }
```

```
        }
    }

}


//Checks if the email and password are correct
public boolean checkEmail() {
    String email = emailID.getText();
    String pw = password.getText();

    //Create an error alert
    if(emailID.getText() == null || emailID.getText().isEmpty() || password.getText()== null||
password.getText().isEmpty()){
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.initModality(Modality.APPLICATION_MODAL);
        alert.setTitle("Field Empty");
        alert.setHeaderText("Please correct invalid fields");
        alert.setContentText("Please Type Email or Password");
        alert.showAndWait();
        return false;
    }else{
        return true;
    }
}

//Checks the email syntax
public boolean isValid(String email) {
    String regex = "^[\\w-_\\.+]*[\\w-_\\.]\\@([\\w]+\\.)+[\\w]+[\\w]$";
    if (email.matches(regex)) {
        //Return true if the syntax is correct
        return true;
    }else{
        //If the syntax is incorrect, create an error alert
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.initModality(Modality.APPLICATION_MODAL);
        alert.setTitle("Invalid Email");
        alert.setHeaderText("Please correct invalid fields");
        alert.setContentText("Please Enter a Valid Email");
        alert.showAndWait();
        return false;
    }

}
//checks whether if the email is already registered
public boolean EmailCheck(String email) throws RemoteException {
   User u =  userService.getUser(email);

   if(u == null){
      return true;
   }else{
      //Show error alert if the user already exist
      Alert alert = new Alert(Alert.AlertType.ERROR);
      alert.initModality(Modality.APPLICATION_MODAL);
      alert.setTitle("Email Exist");
```

```
            alert.setHeaderText("A User already Exist");
            alert.setContentText("Please Enter a Different Email");
            alert.showAndWait();
            return false;
        }
    }


    public void setMain(Main main) {
        this.main = main;
        this.userService = main.getUserService();
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {

    }


}
```

rmiClient/regsiterForm/Login.java

```
package rmiClient.loginForm;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.stage.Modality;
import javafx.stage.Stage;
import rmiApi.entity.User;
import rmiApi.entityService.UserService;
import rmiClient.alarmForm.AlarmForm;
import rmiClient.client.Main;
import rmiClient.registerForm.Register;
import java.io.IOException;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.ResourceBundle;

public class Login implements Initializable {


    public Button regBtn;
```

```
@FXML
private TextField emailID;

@FXML
private PasswordField password;

@FXML
private Button loginBtn;

private Main main;
private UserService userService;

//login button event handler
@FXML
public void login() throws IOException {

	if(checkEmail()){
		if(checkPassword()){
			//scene is set
			Stage stage = (Stage) loginBtn.getScene().getWindow();
			stage.close();

			Stage primaryStage = new Stage();
			FXMLLoader loader = new FXMLLoader(getClass().getResource("/views/alarmForm.fxml"));

			Parent root = loader.load();
			AlarmForm form = loader.getController();
			form.setMain(this.main);
			primaryStage.setScene(new Scene(root)) ;

			primaryStage.setTitle("Sensor Dashboard ");
			primaryStage.show();

		}

	}

}

// method to check if the email exists in the DB
public boolean checkEmail() throws RemoteException {
	String email = emailID.getText();

	User u =userService.getUser(email);
	//checks whether the user information is correct
	if(u == null){
		Alert alert = new Alert(Alert.AlertType.ERROR);
		alert.initModality(Modality.APPLICATION_MODAL);
		alert.setTitle("Invalid Fields");
		alert.setHeaderText("Please correct invalid fields");
		alert.setContentText("Email Does not Exist");
		alert.showAndWait();
		return false;
	}else{
		return true;
```

```
        }
    }

    //method to check if the password is correct
    public boolean checkPassword() throws RemoteException {
        //get the email and password from login form
        String email = emailID.getText();
        String pass = password.getText();

        User u = userService.getUser(email);
        //If email and password match, allow the user to login
        if(email.equals(u.getEmail()) && pass.equals(u.getPassword())){
            return true;
        }//If email and password incorrect, show error alert
        else{
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.initModality(Modality.APPLICATION_MODAL);
            alert.setTitle("Invalid Fields");
            alert.setHeaderText("Please correct invalid fields");
            alert.setContentText("Password Does not Exist");
            alert.showAndWait();
            return false;
        }

    }

    //Register view action event
    public void Register(ActionEvent actionEvent) throws IOException {
        //On button click show the register view
        Stage stage = (Stage) regBtn.getScene().getWindow();
        stage.close();

        Stage primaryStage = new Stage();
        FXMLLoader loader = new FXMLLoader(getClass().getResource("/views/Register.fxml"));

        Parent root = loader.load();
        Register reg = loader.getController();
        reg.setMain(this.main);
        Scene scene = new Scene(root);
        scene.getStylesheets().add(getClass().getResource("/styles/application.css").toExternalForm());
        primaryStage.setScene(scene) ;

        primaryStage.setTitle("Sensor Dashboard Register ");
        primaryStage.show();
    }


    public void setMain(Main main) {
        this.main = main;
        this.userService = main.getUserService();
    }

    @Override
    public void initialize(URL location, ResourceBundle resources) {
    }


}
```

2.Alarm

rmiApi/entitiy/Alarm.java

```
package rmiApi.entity;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import java.io.*;

/*Implement Alarm modal class by defining setters and getters
and constructor and the modal class has implemented the Externalizable interface
by extending the Serializable interface
 */

public class Alarm implements Externalizable {


   private final IntegerProperty id = new SimpleIntegerProperty();
   private  final IntegerProperty floorNum = new SimpleIntegerProperty();
   private  final IntegerProperty roomNum = new SimpleIntegerProperty();
   private final IntegerProperty smokeLevel = new SimpleIntegerProperty();
   private  final IntegerProperty co2level = new SimpleIntegerProperty();

   public Alarm() {
   }

   public int getId() {
      return id.get();
   }

   public IntegerProperty idProperty() {
      return id;
   }

   public void setId(int id) {
      this.id.set(id);
   }

   public int getFloorNum() {
      return floorNum.get();
   }
```

```
public IntegerProperty floorNumProperty() {
    return floorNum;
}

public void setFloorNum(int floorNum) {
    this.floorNum.set(floorNum);
}

public int getRoomNum() {
    return roomNum.get();
}

public IntegerProperty roomNumProperty() {
    return roomNum;
}

public void setRoomNum(int roomNum) {
    this.roomNum.set(roomNum);
}

public int getSmokeLevel() {
    return smokeLevel.get();
}

public IntegerProperty smokeLevelProperty() {
    return smokeLevel;
}

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

public int getCo2level() {
    return co2level.get();
}

public IntegerProperty co2levelProperty() {
    return co2level;
}

public void setCo2level(int co2level) {
    this.co2level.set(co2level);
}

@Override
public void writeExternal(ObjectOutput out) throws IOException {
    out.writeInt(getId());
    out.writeInt(getFloorNum());
    out.writeInt(getRoomNum());
    out.writeInt(getSmokeLevel());
    out.writeInt(getCo2level());
}

@Override
public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException {
        setId(in.readInt());
        setFloorNum(in.readInt());
        setRoomNum(in.readInt());
        setSmokeLevel(in.readInt());
        setCo2level(in.readInt());
    }
}
```

rmiApi/entityservice/alarmService.java

```
package rmiApi.entityService;
import rmiApi.entity.Alarm;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;

/*The alarmService interface is created by extending Remote interface
which will provide the descriptions of methods which can be invoked by
remote clients
 */
public interface alarmService extends Remote {

    public Alarm saveAlarm(Alarm alarm) throws RemoteException;
    public List<Alarm> getAlarms() throws RemoteException;
    public void deleteAlarm(int id)throws RemoteException;
    public Alarm updateAlarm(Alarm alarm)throws RemoteException;

}
```

rmiServer/serviceImpl/AlarmServiceImpl.java

```
package rmiServer.serviceImpl;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import org.apache.http.HttpEntity;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.methods.*;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.util.EntityUtils;
import org.hildan.fxgson.FxGson;
import rmiApi.entity.Alarm;
import rmiApi.entityService.alarmService;
import java.io.IOException;
import java.lang.reflect.Type;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.ArrayList;
import java.util.List;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.HttpClients;


//The AlarmServiceImpl extends UnicastRemoteObject
public class AlarmServiceImpl extends UnicastRemoteObject implements alarmService {

  /* Default constructor to throw RemoteException
    from its parent constructor */
  public AlarmServiceImpl() throws RemoteException {
  }

  //method implementation to save an alarm by passing an alarm object and returning the new alarm
  @Override
  public Alarm saveAlarm(Alarm alarm) throws RemoteException {

    //get values from the alarm reference passed
    alarm.getFloorNum();
    alarm.getRoomNum();
    alarm.getSmokeLevel();
    alarm.getCo2level();

    /*Initialize the apache httpclient used to send http requests and response to the
    rest client
     */
    try(CloseableHttpClient httpclient = HttpClients.createDefault()){
      //HTTP POST method url to add alarms using rest api
      HttpPost httpPost = new HttpPost("http://localhost:8080/addAlarm");
      httpPost.setHeader("Accept", "application/json");
      httpPost.setHeader("Content-type", "application/json");

      //Convert alarm object to JSON response
      Gson gson = FxGson.coreBuilder().setPrettyPrinting().disableHtmlEscaping().create();
     String json = gson.toJson(alarm);

      //pass the json string request in the entity
      StringEntity stringEntity = new StringEntity(json);
      httpPost.setEntity(stringEntity);
      System.out.println("Executing request " + httpPost.getRequestLine());

      //Check the response status
      ResponseHandler < String > responseHandler = response -> {
        int status = response.getStatusLine().getStatusCode();
        if (status >= 200 && status < 300) {
          HttpEntity entity = response.getEntity();
          return entity != null ? EntityUtils.toString(entity) : null;
        } else {
          throw new ClientProtocolException("Unexpected response status: " + status);
        }
      };
```

```
        String responseBody = httpclient.execute(httpPost, responseHandler);
        System.out.println("----------------------------------------");
        System.out.println(responseBody);

    } catch (IOException e) {
        e.printStackTrace();
    }
    //Returns the newly created alarm object
    return alarm;
}

//method implementation to get all the alarms as a list
@Override
public List<Alarm> getAlarms() throws RemoteException {
    ArrayList <Alarm> alarms = null;
    /*Initialize the apache httpclient used to send http requests and response to the
    rest client
     */
    try (CloseableHttpClient httpclient = HttpClients.createDefault()) {

        //HTTP GET method url to retrieve alarms using rest api
        HttpGet httpget = new HttpGet("http://localhost:8080/alarms");
        System.out.println("Executing request " + httpget.getRequestLine());

        // Create a custom response handler
        ResponseHandler < String > responseHandler = response -> {
        int status = response.getStatusLine().getStatusCode();
        if (status >= 200 && status < 300) {
            HttpEntity entity = response.getEntity();
            return entity != null ? EntityUtils.toString(entity) : null;
        } else {
            throw new ClientProtocolException("Unexpected response status: " + status);
        }
        };
        String responseBody = httpclient.execute(httpget, responseHandler);
        System.out.println("----------------------------------------");
        System.out.println(responseBody);

        //Allow GSON to identify array list of alarm type
        Type alarmListType = new TypeToken<ArrayList<Alarm>>(){}.getType();
        //Convert JSON response to alarm list
        Gson gson = FxGson.coreBuilder().setPrettyPrinting().disableHtmlEscaping().create();
         alarms = gson.fromJson(responseBody,alarmListType);

    } catch (IOException e) {
        e.printStackTrace();
    }
    //Returns alarm list
    return alarms;
}


//method implementation to delete an alarm by passing the id
@Override
```

```
public void deleteAlarm(int id) throws RemoteException {
  /*Initialize the apache httpclient used to send http requests and response to the
  rest client
   */
  try (CloseableHttpClient httpclient = HttpClients.createDefault()) {
    //HTTP DELETE method url to delete alarms according to their id using rest api
    HttpDelete httpDelete = new HttpDelete("http://localhost:8080/alarmDelete/"+id);
    System.out.println("Executing request " + httpDelete.getRequestLine());

    // Create a custom response handler
    ResponseHandler < String > responseHandler = response -> {
      int status = response.getStatusLine().getStatusCode();
      if (status >= 200 && status < 300) {
        HttpEntity entity = response.getEntity();
        return entity != null ? EntityUtils.toString(entity) : null;
      } else {
        throw new ClientProtocolException("Unexpected response status: " + status);
      }
    };
    String responseBody = httpclient.execute(httpDelete, responseHandler);
    System.out.println("----------------------------------------");
    System.out.println(responseBody);
  } catch (IOException e) {
    e.printStackTrace();
  }

}

//method implementation to update an existing alarm
@Override
public Alarm updateAlarm(Alarm alarm) throws RemoteException {
  //get values from the alarm reference passed
  alarm.getId();
  alarm.getFloorNum();
  alarm.getRoomNum();
  alarm.getSmokeLevel();
  alarm.getCo2level();


  /*Initialize the apache httpclient used to send http requests and response to the
  rest client
   */
    try (CloseableHttpClient httpclient = HttpClients.createDefault()) {
      //HTTP PUT method url to update alarms according to their id using rest api
      HttpPut httpPut = new HttpPut("http://localhost:8080/updateAlarm");
      httpPut.setHeader("Accept", "application/json");
      httpPut.setHeader("Content-type", "application/json");

      //Convert alarm object to JSON response
      Gson gson = FxGson.coreBuilder().setPrettyPrinting().disableHtmlEscaping().create();
      String json = gson.toJson(alarm);

      //pass the json string request in the entity
      StringEntity stringEntity = new StringEntity(json);
      httpPut.setEntity(stringEntity);
```

```
            System.out.println("Executing request " + httpPut.getRequestLine());

            // Create a custom response handler
            ResponseHandler < String > responseHandler = response -> {
            int status = response.getStatusLine().getStatusCode();
            if (status >= 200 && status < 300) {
               HttpEntity entity = response.getEntity();
               return entity != null ? EntityUtils.toString(entity) : null;
            } else {
               throw new ClientProtocolException("Unexpected response status: " + status);
            }
            };
            String responseBody = httpclient.execute(httpPut, responseHandler);
            System.out.println("----------------------------------------");
            System.out.println(responseBody);

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


      //Returns the updated alarm object
      return alarm;
   }



}
```

views/alarmForm.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.text.Font?>

<AnchorPane prefHeight="713.0" prefWidth="814.0" xmlns="http://javafx.com/javafx/8.0.171"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="rmiClient.alarmForm.AlarmForm">
  <children>
```

```
    <GridPane hgap="2.0" layoutX="22.0" layoutY="14.0" vgap="5.0">
      <columnConstraints>
        <ColumnConstraints hgrow="SOMETIMES" maxWidth="1.7976931348623157E308" minWidth="-
Infinity" />
        <ColumnConstraints hgrow="SOMETIMES" maxWidth="200.0" minWidth="200.0" prefWidth="200.0"
/>
      </columnConstraints>
      <rowConstraints>
         <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
        <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
         <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />
      </rowConstraints>
       <children>
         <Label text="Floor Number" GridPane.rowIndex="1" />
         <Label prefHeight="21.0" prefWidth="116.0" text="Room Number" GridPane.rowIndex="2" />
         <Label text="Smoke Level" GridPane.rowIndex="3" />
         <Label text="Co2 Level" GridPane.rowIndex="4" />
         <TextField fx:id="txtFloorNum" GridPane.columnIndex="1" GridPane.rowIndex="1">
           <GridPane.margin>
             <Insets />
           </GridPane.margin>
         </TextField>
         <TextField fx:id="txtRoomNum" GridPane.columnIndex="1" GridPane.rowIndex="2" />
         <TextField fx:id="txtSmokeLevel" GridPane.columnIndex="1" GridPane.rowIndex="3" />
         <TextField fx:id="txtCo2Level" GridPane.columnIndex="1" GridPane.rowIndex="4" />
         <Label text="ID" />
         <TextField fx:id="txtId" disable="true" editable="false" GridPane.columnIndex="1" />
       </children>
    </GridPane>
    <HBox alignment="CENTER" layoutX="407.0" layoutY="161.0" prefHeight="94.0" prefWidth="363.0"
spacing="10.0">
       <children>
         <Button fx:id="btnInsert" mnemonicParsing="false" onAction="#onInsert" text="Insert" />
         <Button fx:id="btnUpdate" mnemonicParsing="false" onAction="#onUpdate" text="Update" />
         <Button fx:id="btnDelete" mnemonicParsing="false" onAction="#onDelete" text="Delete" />
         <Button fx:id="btnRefresh" mnemonicParsing="false" onAction="#onRefresh" text="Refresh" />
       </children>
       <padding>
         <Insets top="5.0" />
       </padding>
    </HBox>
    <TableView fx:id="tableView" layoutX="29.0" layoutY="280.0" prefHeight="353.0" prefWidth="756.0"
style="-fx-alignment: CENTER;">
      <columns>
        <TableColumn fx:id="colId" prefWidth="71.0" text="ID" />
        <TableColumn fx:id="colFloorNum" prefWidth="129.0" text="Floor Number" />
         <TableColumn fx:id="colRoomNum" prefWidth="148.0" text="Room Number" />
         <TableColumn fx:id="colSmokeLevel" prefWidth="141.0" text="Smoke Level" />
         <TableColumn fx:id="colCo2level" prefWidth="148.0" text="Co2 Level" />
      </columns>
       <columnResizePolicy>
         <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
       </columnResizePolicy>
    </TableView>
    <Label alignment="CENTER" layoutX="382.0" layoutY="63.0" prefHeight="31.0" prefWidth="145.0"
text="Smoke Level : 1-10  " textAlignment="CENTER" textFill="#ee1111" />
    <Label alignment="CENTER" layoutX="398.0" layoutY="99.0" prefHeight="21.0" prefWidth="136.0"
text="C02 Level : 1-10  " textAlignment="CENTER" textFill="#c91414" />
    <Label fx:id="lblTimer" layoutX="654.0" layoutY="47.0" prefHeight="94.0" prefWidth="116.0" />
    <Label alignment="CENTER" layoutX="274.0" layoutY="655.0" prefHeight="21.0" prefWidth="290.0"
text="WARNING" textAlignment="CENTER" textFill="#c61212">
      <font>
        <Font name="System Bold" size="15.0" />
      </font>
    </Label>
    <Label layoutX="218.0" layoutY="676.0" prefHeight="21.0" prefWidth="403.0" text=" Every 15 seconds
New Data Are Fetched And The Table Is Refreshed" textFill="#d7290e">
      <font>
        <Font name="System Italic" size="13.0" />
      </font>
    </Label>
  </children>
</AnchorPane>
```

rmiClient/alarmForm/ AlarmForm.java

```
package rmiClient.alarmForm;
import javafx.animation.KeyFrame;
import javafx.animation.KeyValue;
import javafx.animation.Timeline;
import javafx.application.Platform;
import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.concurrent.ScheduledService;
import javafx.concurrent.Task;
import javafx.event.ActionEvent;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.paint.Color;
import javafx.stage.Modality;
import javafx.util.Duration;
import rmiApi.entity.Alarm;
import rmiApi.entityService.alarmService;
import rmiClient.client.Main;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.List;
import java.util.ResourceBundle;
```

```java
public class AlarmForm implements Initializable {


  public TextField txtFloorNum;
  public TextField txtRoomNum;
  public TextField txtSmokeLevel;
  public TextField txtCo2Level;
  public TextField txtId;
  public TableView<Alarm> tableView;
  public TableColumn<Alarm, Integer> colId;
  public TableColumn<Alarm, Integer> colFloorNum;
  public TableColumn<Alarm, Integer> colRoomNum;
  public TableColumn<Alarm, Integer> colSmokeLevel;
  public TableColumn<Alarm, Integer> colCo2level;
  public Label lblTimer;
  private Main main;
  private alarmService as;
  private static final Integer STARTTIME = 15;
  private Timeline timeline;
  private IntegerProperty timeSeconds = new SimpleIntegerProperty(STARTTIME);

  //method to initilize the timer
  public void startTimer(){
    Platform.runLater(() ->{
      if (timeline != null) {
        timeline.stop();
      }
      timeSeconds.set(STARTTIME);
      timeline = new Timeline();
      timeline.getKeyFrames().add(
          new KeyFrame(Duration.seconds(STARTTIME),
              new KeyValue(timeSeconds, 0)));
      timeline.playFromStart();
    });

  }

  //insert button event handler
  public void onInsert(ActionEvent actionEvent) throws RemoteException {
  if(isFieldValid()){
    if(isCorrectRange()){
      if(checkForDuplicates()){
        try {
            /* An Alarm refernece is intialized and the values from the form fields
        are set to the alarm reference */
          Alarm alarm = new Alarm();
          alarm.setRoomNum(Integer.parseInt(txtRoomNum.getText()));
          alarm.setFloorNum(Integer.parseInt(txtFloorNum.getText()));
          alarm.setSmokeLevel(Integer.parseInt(txtSmokeLevel.getText()));
          alarm.setCo2level(Integer.parseInt(txtCo2Level.getText()));

          //The saveAlarm() method of AlarmServiceImpl is executed by passing the alarm reference
          alarm = as.saveAlarm(alarm);
          //The new alarm is displayed in the table using the getItems() method defined in the tableView
          tableView.getItems().add(alarm);
```

```
            tableView.getItems().setAll(as.getAlarms());
            clearField();

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


      }


    }

  }

  //method to check for existing values of floor and room number in the database
  public boolean checkForDuplicates() throws RemoteException {

    List<Alarm> al = as.getAlarms();
    boolean dup = true;

    for (Alarm a : al) {

      if (a.getFloorNum() == Integer.parseInt(txtFloorNum.getText()) && a.getRoomNum() ==
Integer.parseInt(txtRoomNum.getText())) {
          Alert alert = new Alert(Alert.AlertType.ERROR);
          alert.initModality(Modality.APPLICATION_MODAL);
          alert.setTitle("Duplicate Values");
          alert.setHeaderText("Floor number and the Room number already exists");
          alert.setContentText("Please Change Room and Floor Numbers");
          alert.showAndWait();
          dup=false;

      }

    }
    return dup;
  }


  //checks if the form fields are null
  private boolean isFieldValid() {
    String errorMessage="";

    if(txtFloorNum.getText() == null || txtFloorNum.getText().isEmpty()){
      errorMessage += "Please enter a Floor Number !\n";
    }
    if(txtRoomNum.getText() == null || txtRoomNum.getText().isEmpty()){
      errorMessage += "Please enter a Room Number !\n";
    }
    if(txtSmokeLevel.getText() == null || txtSmokeLevel.getText().isEmpty()){
      errorMessage += "Please enter a Smoke level !\n";
    }
    if(txtCo2Level.getText() == null || txtCo2Level.getText().isEmpty()){
      errorMessage += "Please enter a C02 Number !\n";
    }


    if(errorMessage.length() == 0){
      return true;
    }else{
      Alert alert = new Alert(Alert.AlertType.ERROR);
      alert.initModality(Modality.APPLICATION_MODAL);
      alert.setTitle("Invalid Fields");
      alert.setHeaderText("Please correct invalid fields");
      alert.setContentText(errorMessage);
      alert.showAndWait();
      return false;
    }

  }

  //Checks if the smoke level and the CO2 levels are in range of 1 and 10
  private boolean isCorrectRange() {

    String errorMessage="";
    if(Integer.parseInt(txtSmokeLevel.getText()) > 10 || Integer.parseInt(txtSmokeLevel.getText()) < 0){
      errorMessage += "Please enter a smoke level between 1 and 10 !\n";
    }
    if(Integer.parseInt(txtCo2Level.getText()) > 10 || Integer.parseInt(txtCo2Level.getText()) < 0){
      errorMessage += "Please enter a c02 level between 1 and 10 !\n";
    }

    if(errorMessage.length() == 0){
      return true;
    }else{
      Alert alert = new Alert(Alert.AlertType.ERROR);
      alert.initModality(Modality.APPLICATION_MODAL);
      alert.setTitle("Invalid Fields");
      alert.setHeaderText("Please correct invalid fields");
      alert.setContentText(errorMessage);
      alert.showAndWait();
      return false;
    }
  }
  //Set form fields to null after submitting form
  private void clearField() {
    txtId.setText("");
    txtFloorNum.setText("");
    txtRoomNum.setText("");
    txtSmokeLevel.setText("");
    txtCo2Level.setText("");

  }

  // update button event handler
  public void onUpdate(ActionEvent actionEvent) throws RemoteException {
    //Pass the table row index selected to an index varaible
    int index = tableView.getSelectionModel().getSelectedIndex();
    System.out.println(index);
    //Checks if an alarm is selected from the table
    if(index == -1){
      Alert alert = new Alert(Alert.AlertType.ERROR);
      alert.initModality(Modality.APPLICATION_MODAL);
      alert.setTitle("Update");
      alert.setHeaderText("No Alarm selected");
      alert.setContentText("Please select an alarm from the table");
      alert.showAndWait();
      return ;
    }

    if(isFieldValid()){
      if(isCorrectRange()){
      if(checkForDuplicates()){
          /* An Alarm reference is initialized and the values from the form fields
        are set to the alarm reference */
         Alarm a = new Alarm();
         a.setId(Integer.valueOf(txtId.getText()));
         a.setFloorNum(Integer.valueOf(txtFloorNum.getText()));
         a.setRoomNum(Integer.valueOf(txtRoomNum.getText()));
         a.setSmokeLevel(Integer.valueOf(txtSmokeLevel.getText()));
         a.setCo2level(Integer.valueOf(txtCo2Level.getText()));

         try {
           //Execute updateAlarm() method in alarmService and pass the updated alarm data to the
  tableView
           as.updateAlarm(a);
           tableView.getItems().set(index,a);
           tableView.getItems().setAll(as.getAlarms());
           clearField();
         } catch (RemoteException e) {
           e.printStackTrace();
         }
      }

      }
    }

  }


  //udelete button event handler
  public void onDelete(ActionEvent actionEvent) {
    //Retrieve the selected row index value of table to pass as the alarm id
    Alarm alarm = tableView.getSelectionModel().getSelectedItem();
    //Checks if an alard id is selected
    if(alarm == null){
      return;
    }

    try {
       /*The deleteAlarm() method of AlarmServiceImpl is executed by passing the
      alarm id and the respective alarm data is removed from the table
```

```
         */
        as.deleteAlarm(alarm.getId());
        tableView.getItems().remove(alarm);

        clearField();

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


}

//refresh button event handler
public void onRefresh(ActionEvent actionEvent) {
    //Invoke the getAlarms() method on click of the refresh button and reset timer
    try {
        tableView.getItems().setAll(as.getAlarms());
        startTimer();
    } catch (RemoteException e) {
        e.printStackTrace();
    }

    clearField();
}

//Fill the table colomns with the id, floor number, room number, smoke level and co2 level
public void TableRefresh(){
    colId.setCellValueFactory(new PropertyValueFactory<Alarm,Integer>("id"));
    colFloorNum.setCellValueFactory(new PropertyValueFactory<>("floorNum"));
    colRoomNum.setCellValueFactory(new PropertyValueFactory<>("roomNum"));
    colSmokeLevel.setCellValueFactory(new PropertyValueFactory<>("smokeLevel"));
    colCo2level.setCellValueFactory(new PropertyValueFactory<>("co2level"));
}

//javafx method used to refresh gui components in the background
ScheduledService<Integer> service = new ScheduledService<Integer>() {
    @Override
    protected Task<Integer> createTask() {
        return new Task<Integer>() {
            @Override
            protected Integer call() throws Exception {
                tableView.getItems().setAll(as.getAlarms());
                startTimer();
                //System.out.println("i run");
                return 0;
            }
        };
    }
};

//when the AlarmForm is loaded first thing to be executed
@Override
public void initialize(URL location, ResourceBundle resources) {
    //Execute the timer
    lblTimer.textProperty().bind(timeSeconds.asString());
    lblTimer.setTextFill(Color.RED);
    lblTimer.setStyle("-fx-font-size: 4em;");

    TableRefresh();
    startTimer();

    // add listners to every row in the table, when row clicked retrieve the values and set them in text fields
    tableView.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<Alarm>() {
      @Override
      public void changed(ObservableValue<? extends Alarm> observable, Alarm oldValue, Alarm newValue)
{
        if (newValue != null) {
          txtId.setText(Integer.toString(newValue.getId()));
          txtFloorNum.setText(Integer.toString(newValue.getFloorNum()));
          txtRoomNum.setText(Integer.toString(newValue.getRoomNum()));
          txtSmokeLevel.setText(Integer.toString(newValue.getSmokeLevel()));
          txtCo2Level.setText(Integer.toString(newValue.getCo2level()));
        } else {
          clearField();
        }
      }
    });

  }




  //Execute method when loading
  public void setMain(Main main){
    this.main = main;
    this.as = main.getAlarmService();
    //background tasks refreshes every 15 seconds
    service.setPeriod(Duration.seconds(15));
    service.start();

    try {
      tableView.getItems().setAll(as.getAlarms());
    } catch (RemoteException e) {
      e.printStackTrace();
    }
  }


}
```

3.Email

rmiApi/entitiy/MailRequest.java

```
package rmiApi.entity;

//implement emailRequest model class
public class MailRequest {

  private String name;
  private String to;
  private String from;
  private String subject;

  public MailRequest() {
  }


  public MailRequest(String name, String to, String from, String subject) {
    this.name = name;
    this.to = to;
    this.from = from;
    this.subject = subject;
  }

  public String getName() {
    return name;
  }

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

  public String getTo() {
    return to;
  }

  public void setTo(String to) {
    this.to = to;
  }

  public String getFrom() {
    return from;
  }

  public void setFrom(String from) {
    this.from = from;
  }

  public String getSubject() {
    return subject;
  }
```

```java
  public void setSubject(String subject) {
    this.subject = subject;
  }
}
```

rmiApi/entityservice/Emailservice.java

```java
package rmiApi.entityService;
import rmiApi.entity.MailRequest;
import java.rmi.Remote;
import java.rmi.RemoteException;
/*The EmailService interface is created by extending Remote interface
which will provide the descriptions of methods which can be invoked by
remote clients
 */
public interface Emailservice extends Remote {

   public void sendEmail(MailRequest request) throws RemoteException;

   public MailRequest mailRequest(int floorLevel, int RoomNum) throws RemoteException;
}
```

rmiServer/serviceImpl/EmailServiceImpl.java

```java
package rmiServer.serviceImpl;

import com.google.gson.Gson;
import org.apache.http.HttpEntity;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.hildan.fxgson.FxGson;
import rmiApi.entity.MailRequest;
import rmiApi.entityService.Emailservice;

import java.io.IOException;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class EmailServiceImpl extends UnicastRemoteObject implements Emailservice {
   public EmailServiceImpl() throws RemoteException {
   }
```

```
//method implementation to send email
@Override
public void sendEmail(MailRequest request) throws RemoteException  {
   //get the data passed from the request parameter
   request.getFrom();
   request.getTo();
   request.getSubject();
   request.getName();

   /*Initialize the apache httpclient used to send http requests and response to the
   rest client
    */
   try(CloseableHttpClient httpclient = HttpClients.createDefault()){
      //HTTP POST request to send an email using the REST api
      HttpPost httpPost = new HttpPost("http://localhost:8080/sendEmail");
      httpPost.setHeader("Accept", "application/json");
      httpPost.setHeader("Content-type", "application/json");

      //convert the request object to JSON
      Gson gson = FxGson.coreBuilder().setPrettyPrinting().disableHtmlEscaping().create();
      String json = gson.toJson(request);


      StringEntity stringEntity = new StringEntity(json);
      httpPost.setEntity(stringEntity);
      System.out.println("Executing request " + httpPost.getRequestLine());

      //custom header to handle http status
      ResponseHandler< String > responseHandler = response -> {
         int status = response.getStatusLine().getStatusCode();
         if (status >= 200 && status < 300) {
            HttpEntity entity = response.getEntity();
            return entity != null ? EntityUtils.toString(entity) : null;
         } else {
            throw new ClientProtocolException("Unexpected response status: " + status);
         }
      };

      String responseBody = httpclient.execute(httpPost, responseHandler);
      System.out.println("----------------------------------------");
      System.out.println(responseBody);

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


}

//method implementation to set the mail request
@Override
public MailRequest mailRequest(int floorLevel, int RoomNum) throws RemoteException {
   MailRequest req = new MailRequest();
   req.setTo("distributedsystems123@gmail.com");
   req.setFrom("distributedsystems123@gmail.com");
```

```
    req.setSubject("Hazardous levels in Floor Number "+floorLevel+" Room Number "+RoomNum);
    req.setName("Fire Alarm System");

    //returns the req reference with set values
    return req;
  }
}
```

rmiServer/Main.java

```
package rmiServer;
import javafx.application.Application;
import javafx.application.Platform;
import javafx.stage.Stage;
import rmiApi.entity.Alarm;
import rmiApi.entityService.Emailservice;
import rmiApi.entityService.SmsService;
import rmiApi.entityService.alarmService;
import rmiServer.serviceImpl.AlarmServiceImpl;
import rmiServer.serviceImpl.EmailServiceImpl;
import rmiServer.serviceImpl.SmsServiceImpl;
import rmiServer.serviceImpl.UserServiceImpl;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;

//class to retrieve new alarm details from the REST api periodically
class CheckForUpdates extends TimerTask{

  Emailservice emailService;
  SmsService smsService;
  alarmService alarmService;
  @Override
  public void run() {
    try {
      AlarmServiceImpl alarmService = new AlarmServiceImpl();
      //retrieve the  alarm list
      List <Alarm> a =  alarmService.getAlarms();

      /*
      iterate through the alarm list and see if the below conidition is true if so
      invokde sendEmail and sendSms methods
       */
      for(Alarm al : a){
        if(al.getCo2level() > 5 || al.getSmokeLevel() >5){
                  EmailServiceImpl emailService = new EmailServiceImpl();
                  SmsServiceImpl smsService = new SmsServiceImpl();
        }
```

```java
        }


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

  }
}



public class Main extends Application {
  @Override
  public void start(Stage primaryStage) throws Exception {
    //registry created to run the server on port 8081
    Registry registry = LocateRegistry.createRegistry(8081);

    AlarmServiceImpl alarmServiceImpl = new AlarmServiceImpl();
    UserServiceImpl userServiceImpl = new UserServiceImpl();
    EmailServiceImpl emailServiceImpl = new EmailServiceImpl();
    SmsServiceImpl smsServiceImpl = new SmsServiceImpl();

    //binding the remote objects
    registry.rebind("userService",userServiceImpl);
    registry.rebind("alarmService",alarmServiceImpl);
    registry.rebind("emailService",emailServiceImpl);
    registry.rebind("smsService",smsServiceImpl);


    System.out.println("Server is running");
    Platform.exit();
  }


  public static void main(String[] args) {

    launch(args);
    Timer timer = new Timer();
    CheckForUpdates checkForUpdates = new CheckForUpdates();
    //timer to check for alarm status every 15 seconds
    timer.schedule(checkForUpdates, 0, 15000);
  }
}
```

Sensor App

Sensor/service/ServiceImpl.java

```
package service;

import java.io.IOException;
import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.List;

import org.apache.http.HttpEntity;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.ResponseHandler;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

public class ServiceImpl {

  //Method to send HTTP GET request to the REST api and retrieve list of alarm ids.
  public List<Integer> getIds(){

    List <Integer> Ids = null;

     try (CloseableHttpClient httpclient = HttpClients.createDefault()) {

          //HTTP GET method
          HttpGet httpget = new HttpGet("http://localhost:8080/getIDs");
          System.out.println("Executing request " + httpget.getRequestLine());

          // Create a custom response handler
          ResponseHandler < String > responseHandler = response -> {
          int status = response.getStatusLine().getStatusCode();
          if (status >= 200 && status < 300) {
             HttpEntity entity = response.getEntity();
             return entity != null ? EntityUtils.toString(entity) : null;
          } else {
             throw new ClientProtocolException("Unexpected response status: " + status);
          }
          };
          String responseBody = httpclient.execute(httpget, responseHandler);
          System.out.println("----------------------------------------");
          System.out.println(responseBody);

          Gson gson = new Gson();
          Type alarmListIds = new TypeToken<ArrayList<Integer>>(){}.getType();
           Ids = gson.fromJson(responseBody,alarmListIds);
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
        return Ids;
    }


    //Method to send PUT request to the REST api to update sensor details
    public  void updateFields(int co2, int smoke , int id) {

        try (CloseableHttpClient httpclient = HttpClients.createDefault()) {
          //HTTP POST method
            HttpPut httpPut = new HttpPut("http://localhost:8080/updateFields/"+co2+"/"+smoke+"/"+id);
            httpPut.setHeader("Accept", "application/json");
            httpPut.setHeader("Content-type", "application/json");

            // Create a custom response handler
            ResponseHandler < String > responseHandler = response -> {
            int status = response.getStatusLine().getStatusCode();
            if (status >= 200 && status < 300) {
                HttpEntity entity = response.getEntity();
                return entity != null ? EntityUtils.toString(entity) : null;
            } else {
                throw new ClientProtocolException("Unexpected response status: " + status);
            }
            };
            String responseBody = httpclient.execute(httpPut, responseHandler);
            System.out.println("----------------------------------------");
            System.out.println(responseBody);

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

    }



}
```

sensorApp/Main.java

```
package sensorApp;

import java.util.List;
import java.util.Random;
import java.util.Timer;
import java.util.TimerTask;

import service.ServiceImpl;
```

```
class SmokeValueGen extends TimerTask{
  //Reference to the ServicImpl class
  ServiceImpl serviceImpl = new ServiceImpl();

  @Override
  public void run() {
    // TODO Auto-generated method stub
    //Retrieve list of alarm ids using the getIds method.
    List<Integer> ids = serviceImpl.getIds();

    //generate random integer values between 1 and 10 to pass as co2 and smoke level
    int s = (int) (10.0 * Math.random() + 1);
    int c = (int) (10.0 * Math.random() + 1);
    int smoke = smokeLevel(s);
    int co2 = co2Level(c);
    //from the list of alarm ids pick a random alarm id
    int id = getRandomElement(ids);

    //update the co2 and smoke levels of a particular alarm imitating a real life sensor.
    serviceImpl.updateFields(co2, smoke, id);
    System.out.println("Sensor App running c02 "+co2+" smoke "+smoke+ " id "+id);

  }

  public int smokeLevel(int s) {
    return s;
  }

  public int co2Level(int c) {
    return c;
  }

  public int sensorid(int i) {
    return i;
  }

  //method to retrieve a random alarm id
  public int getRandomElement(List<Integer> list)
   {
      Random rand = new Random();
      return list.get(rand.nextInt(list.size()));
   }

}



public class Main {

  public static void main(String[] args) {
    // TODO Auto-generated method stub
    SmokeValueGen smokeValueGen = new SmokeValueGen();
    Timer timer = new Timer();

    //Every 10 seconds execute, to send sensor data to REST api
    timer.schedule(smokeValueGen, 000,10000);
  }

}
```

Springboot

1.User

model/User.java

```
package com.example.dsassignment.fireAlarmSystem.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;
import java.util.Set;

@AllArgsConstructor
@NoArgsConstructor
@Entity
@Data
@Table(name = "user")
public class User {

   @Id
   @GeneratedValue
   private int id;


   private String email;
```

```java
  private String password;

//   @OneToMany
//   private Set<Alarm> alarm;

}
```

services/UserServices.java

```java
package com.example.dsassignment.fireAlarmSystem.service;

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

import com.example.dsassignment.fireAlarmSystem.model.User;
import com.example.dsassignment.fireAlarmSystem.repository.UserRepository;

@Service //initialized as a service class
public class UserService {

  @Autowired //spring injects UserRepository when the USerService is created
  private UserRepository userRepository;

  //method to add user by passing a user type parameter to the DB and returning the saved user
  public User addUser(User newUser) {
    return userRepository.save(newUser);
  }

  //method to list out a unique user by passing an email and retreieving the user from DB
  public User getUserByEmail(String email) {
    return userRepository.getUserByEmail(email);
  }





}
```

repository/UserRepository.java

```java
package com.example.dsassignment.fireAlarmSystem.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.example.dsassignment.fireAlarmSystem.model.User;

public interface UserRepository extends JpaRepository<User, Integer> {

  //custom method to retrieve the user details by passing email as a parameter
  User getUserByEmail(String email);

}
```

controller/UserController.java

```java
package com.example.dsassignment.fireAlarmSystem.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.example.dsassignment.fireAlarmSystem.model.User;
import com.example.dsassignment.fireAlarmSystem.service.UserService;

@CrossOrigin(origins="*") //allow access to all resources from any domain
@RestController //mark UserController class as a request handler
public class UserController {

  @Autowired //spring injects UserService when the UserController is created
  private UserService userService;

  @PostMapping("/addUser") //handles HTTP POST request to add a user
  public User addUser(@RequestBody User user) {
    return userService.addUser(user);
  }


  @GetMapping("/getUser/{email}") //handles HTTP GET request to get a user by passing email as the
param.
  public User getUserByEmail(@PathVariable String email) {
    return userService.getUserByEmail(email);
  }
}
```

2.Alarm

model/Alarm.java

```java
package com.example.dsassignment.fireAlarmSystem.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;


import javax.persistence.*;

@AllArgsConstructor // initialize constructor with all arguments
@NoArgsConstructor // initialize constructor with no arguments
@Entity  // specify the class as an entity found in the DB
@Data // provide auto generated code for a normal class
@Table(name = "alarm") //table name found in the db
public class Alarm {

   @Id
   @GeneratedValue //id should be auto generated an auto incremented.
   private int id;



   private int floorNum;

   private int roomNum;

   private int smokeLevel;

   private int co2level;



}
```

services/AlarmServices.java

```java
package com.example.dsassignment.fireAlarmSystem.service;



import java.util.Collection;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.dsassignment.fireAlarmSystem.model.Alarm;
import com.example.dsassignment.fireAlarmSystem.repository.AlarmRepository;


@Service //initialized as a service class
public class AlarmService {
  @Autowired //spring injects AlarmRepository when the AlarmService is created
  private AlarmRepository alarmRepository;


  //method to save alarm in the DB by passing an alarm and returning the saved alarm
  public Alarm saveAlarm(Alarm alarm) {
    return alarmRepository.save(alarm);
  }

  //method to retrieve all the alarms in the DB as an alarm List
  public List<Alarm> getAlarms(){
    return alarmRepository.findAll();
  }



  //method to delete an alarm from the DB by passing the alarm id
  public void deleteAlarm(int id) {
    alarmRepository.deleteById(id);

  }

  //method to update an alarm in the DB by passing an alarm and return the updated alarm
  public Alarm updateAlarm(Alarm alarm) {

    Alarm existingAlarm = alarmRepository.findById(alarm.getId()).orElse(null);
    existingAlarm.setFloorNum(alarm.getFloorNum());
    existingAlarm.setRoomNum(alarm.getRoomNum());
    existingAlarm.setSmokeLevel(alarm.getSmokeLevel());
    existingAlarm.setCo2level(alarm.getCo2level());

    return alarmRepository.save(existingAlarm);
  }


  //method to retrieve all the alarm ids in the DB
  public List<Integer> getIds(){
    return alarmRepository.findIDs();
  }

  //method to update some of the fields in the alarm table in the DB
  public void updateFields(int co2, int smoke, int id) {
     alarmRepository.updateFields(co2, smoke, id);
  }
}
```

repository/AlarmRepository.java

```
package com.example.dsassignment.fireAlarmSystem.repository;


import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.transaction.annotation.Transactional;

import com.example.dsassignment.fireAlarmSystem.model.Alarm;

public interface AlarmRepository extends JpaRepository<Alarm, Integer> {



  // custom query to retrieve all the ids of alarms in the DB
  @Query(value="SELECT id from alarm",nativeQuery = true)
   List<Integer> findIDs();


  // custom query to modify an existing record of an alarm in the DB
  @Transactional
  @Modifying
  @Query(value="UPDATE alarm a set a.co2level=?1,a.smoke_level=?2 WHERE a.id=?3",nativeQuery = true)
  void updateFields(int co2level,int smokelevel, int id);

}
```

controller/AlarmController.java

```
package com.example.dsassignment.fireAlarmSystem.controller;

import java.util.List;

import org.omg.CORBA.PUBLIC_MEMBER;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```java
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.example.dsassignment.fireAlarmSystem.model.Alarm;
import com.example.dsassignment.fireAlarmSystem.service.AlarmService;

@RestController //mark AlarmController class as a request handler
public class AlarmController {

  @Autowired //spring injects AlarmService when the AlarmController is created
  private AlarmService alarmService;

  @PostMapping("/addAlarm") //handles HTTP POST request to add Alarm
  public Alarm addAlarm(@RequestBody Alarm alarm) {
    return alarmService.saveAlarm(alarm);
  }

  @GetMapping("/alarms")//handles HTTP GET request to get all Alarms
  public List<Alarm> findAllAlarms(){
    return alarmService.getAlarms();
  }


  @PutMapping("/updateAlarm")//handles HTTP PUT method to update an alarm
  public Alarm updateAlarm(@RequestBody Alarm alarm) {
    return alarmService.updateAlarm(alarm);
  }


  @DeleteMapping("/alarmDelete/{id}")//handles HTTP DELETE method to delete an alarm
  public void deleteAlarm(@PathVariable int id) {
     alarmService.deleteAlarm(id);
  }


  @GetMapping("/getIDs")//handles HTTP GET request to get all Alarm ids
  public List<Integer> findAllIds(){
    return alarmService.getIds();
  }

  @PutMapping("/updateFields/{co2}/{smoke}/{id}")//handles HTTP PUT method to UPDATE particular
details of an alarm
    public void updateFields(@PathVariable int co2,@PathVariable int smoke,@PathVariable int id){
     alarmService.updateFields(co2, smoke, id);
  }



}
```

3.Email

model/MailRequest.java

```
package com.example.dsassignment.fireAlarmSystem.model;

import lombok.Data;

@Data // provide auto generated code for a normal class
public class MailRequest {

  private String name;
  private String to;
  private String from;
  private String subject;


}
```

model/MailResponse.java

```
package com.example.dsassignment.fireAlarmSystem.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data // provide auto generated code for a normal class
@AllArgsConstructor  // initialize constructor with all arguments
@NoArgsConstructor // initialize constructor with no arguments
public class MailResponse {

  private String message;
  private boolean status;
}
```

services/EmailService.java

```java
package com.example.dsassignment.fireAlarmSystem.service;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.util.Map;

import javax.mail.MessagingException;
import javax.mail.internet.MimeMessage;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.io.ClassPathResource;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.stereotype.Service;
import org.springframework.ui.freemarker.FreeMarkerTemplateUtils;

import com.example.dsassignment.fireAlarmSystem.model.MailRequest;
import com.example.dsassignment.fireAlarmSystem.model.MailResponse;

import freemarker.template.Template;
import freemarker.template.TemplateException;
import freemarker.template.Configuration;


@Service //initialized as a service class
public class EmailService {

  @Autowired //spring injects JavaMailSender when the EmailService is created
  private JavaMailSender sender;

  @Autowired //spring injects Configuration when the EmailService is created
  private Configuration config;


  //method implementation to send the eamil
  public MailResponse sendEmail(MailRequest request, Map<String, Object> model) {
    MailResponse response = new MailResponse(); //initilaize a reference to the MailResponse class
    MimeMessage message = sender.createMimeMessage(); //creates a new JavaMail mime message.
    try {
      // set mediaType
      MimeMessageHelper helper = new MimeMessageHelper(message,
MimeMessageHelper.MULTIPART_MODE_MIXED_RELATED,
          StandardCharsets.UTF_8.name());


      // reference to the template to be used in the eamil
      Template t = config.getTemplate("email-template.ftl");
      String html = FreeMarkerTemplateUtils.processTemplateIntoString(t, model);

      //set the values
      helper.setTo(request.getTo());
      helper.setText(html, true);
      helper.setSubject(request.getSubject());
      helper.setFrom(request.getFrom());
      sender.send(message);
      response.setMessage("mail send to : " + request.getTo());
      response.setStatus(Boolean.TRUE);

    } catch (MessagingException | IOException | TemplateException e) {
      response.setMessage("Mail Sending failure : "+e.getMessage());
      response.setStatus(Boolean.FALSE);
    }

    return response;
  }
}
```

email/config/ApiConfig.java

```
package com.example.dsassignment.fireAlarmSystem.email.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.ui.freemarker.FreeMarkerConfigurationFactoryBean;

@Configuration //automatically configures beans
public class ApiConfig {

  @Primary // this particular bean must be given preference
  @Bean //instatntate this as a bean therebey get all the methods in a bean
  public FreeMarkerConfigurationFactoryBean factoryBean() {
    FreeMarkerConfigurationFactoryBean bean = new FreeMarkerConfigurationFactoryBean();
    bean.setTemplateLoaderPath("classpath:/templates");
    return bean;
  }

}
```

controller/EmailController.java

```
package com.example.dsassignment.fireAlarmSystem.controller;

import java.util.HashMap;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.example.dsassignment.fireAlarmSystem.model.MailRequest;
import com.example.dsassignment.fireAlarmSystem.model.MailResponse;
import com.example.dsassignment.fireAlarmSystem.service.EmailService;

@RestController //mark EmailController class as a request handler
public class EmailController {

  @Autowired //spring injects EmailService when the EmailController is created
  private EmailService emailService;


  @PostMapping("/sendEmail")  //handles HTTP POST request to send an Email
  public MailResponse sendEmail(@RequestBody MailRequest request) {

    Map<String, Object> model = new HashMap<>();
    model.put("Name", request.getName());

    return emailService.sendEmail(request, model);
  }


}
```

4.SMS

Model/SMSRequest.java

```
package com.example.dsassignment.fireAlarmSystem.model;

import javax.validation.constraints.NotBlank;

import lombok.AllArgsConstructor;
import lombok.Data;

@Data // provide auto generated code for a normal class
@AllArgsConstructor // initialize constructor with all arguments
public class SmsRequest {

  @NotBlank // the value cannot be null
  private final String phoneNumber;

  @NotBlank // the value cannot be null
  private final String message;
}
```

Model/TwilloConfiguration.java

```
package com.example.dsassignment.fireAlarmSystem.model;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;

import lombok.Data;
import lombok.NoArgsConstructor;

@Configuration //automatically configures beans
@ConfigurationProperties("twilio") //automatically configures beans to twillio
@Data // provide auto generated code for a normal class
@NoArgsConstructor // initialize constructor with no arguments
public class TwilloConfiguration {

  private String accountSid;
  private String authToken;
  private String trialNumber;


}
```

service/TwilloService.java

```
package com.example.dsassignment.fireAlarmSystem.service;



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

import com.example.dsassignment.fireAlarmSystem.model.SmsRequest;
import com.example.dsassignment.fireAlarmSystem.sms.SmsSender;
import com.example.dsassignment.fireAlarmSystem.sms.TwilioSmsSender;;

@org.springframework.stereotype.Service
public class TwilioService {

  private final SmsSender smsSender;

  @Autowired
  public TwilioService(@Qualifier("twilio") TwilioSmsSender twilioSmsSender) {
    this.smsSender= twilioSmsSender;
  }

  //method implemenation to send out a sms
  public void smsSend(SmsRequest smsRequest) {
    smsSender.sendSms(smsRequest);
  }

}
```

sms/SmsSender.java

```
package com.example.dsassignment.fireAlarmSystem.sms;

import com.example.dsassignment.fireAlarmSystem.model.SmsRequest;

public interface SmsSender {

  // abstract method to sendSms
  public void sendSms(SmsRequest smsRequest);
}
```

sms/TwilloSmsSender.java

```
package com.example.dsassignment.fireAlarmSystem.sms;


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

import com.example.dsassignment.fireAlarmSystem.model.SmsRequest;
import com.example.dsassignment.fireAlarmSystem.model.TwilloConfiguration;
import com.twilio.rest.api.v2010.account.Message;
import com.twilio.rest.api.v2010.account.MessageCreator;
import com.twilio.type.PhoneNumber;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@Service("twilio") //is a service class
public class TwilioSmsSender implements SmsSender {

  //reference
  private final TwilloConfiguration twillioConfg;
```

```
 private final static Logger LOGGER = LoggerFactory.getLogger(TwillioInitializer.class);

 @Autowired
  public TwilioSmsSender(TwilloConfiguration twilloConfiguration) {
    // TODO Auto-generated constructor stub
    this.twillioConfg = twilloConfiguration;
 }

 //method sendSms implementation
 /*
  * checks if the method is valid and if so  the receiver phone number is requested
  * and then the sender number is requested ,net the message to sent is requested
  * and then the message is created.
  */


 @Override
 public void sendSms(SmsRequest smsRequest) {
    // TODO Auto-generated method stub
    if(isPhoneNumberValid(smsRequest.getPhoneNumber())){
    PhoneNumber to = new PhoneNumber(smsRequest.getPhoneNumber());
    PhoneNumber from = new PhoneNumber(twillioConfg.getTrialNumber());
    String message = smsRequest.getMessage();
    MessageCreator creator = Message.creator(to, from, message);
    creator.create();
    LOGGER.info("Send sms" +smsRequest);

    }else {
      throw new IllegalArgumentException(
           "phone number ["+smsRequest.getPhoneNumber()+" not valid");
    }


 }

 //method to check the validit of the phone number
 private boolean isPhoneNumberValid(String phString) {
    return true;
 }

}
```

sms/TwilloInitializer.java

```
package com.example.dsassignment.fireAlarmSystem.sms;



import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;

import com.example.dsassignment.fireAlarmSystem.model.TwilloConfiguration;
import com.twilio.Twilio;



@Configuration //automatically configures beans
public class TwillioInitializer {

  //add logging support to the REST app
  private final static Logger LOGGER = LoggerFactory.getLogger(TwillioInitializer.class);

  //reference
  private final TwilloConfiguration twilloConfiguration;

  /*
   * Twilio init method is called to get the account sid and the authentication token
   * in the application.properties
   */
  @Autowired
  public TwillioInitializer(TwilloConfiguration twilloConfiguration) {
    this.twilloConfiguration = twilloConfiguration;
    Twilio.init(twilloConfiguration.getAccountSid(),
        twilloConfiguration.getAuthToken());

    LOGGER.info("Twillio initilaized with "+twilloConfiguration.getAccountSid());
  }



}
```

controller/SmsController.java

```
package com.example.dsassignment.fireAlarmSystem.controller;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.convert.ReadingConverter;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.dsassignment.fireAlarmSystem.model.SmsRequest;
import com.example.dsassignment.fireAlarmSystem.service.TwilioService;
```

```
@RestController //mark SmsController class as a request handler
@RequestMapping("api/sms") //provide the url path to the service
public class SmsController {

  private final TwilioService service;

  @Autowired //an instance of service is injected to the constructor when SmsController is created.
  public SmsController(TwilioService service) {
  // TODO Auto-generated constructor stub
    this.service = service;
  }

  @PostMapping //handles HTTP POST request to send a sms
  public void sendSms(@Valid @RequestBody SmsRequest smsRequest) {
    service.smsSend(smsRequest);
  }
}
```

5.Application

/FireAlarmSystemApplication.java

```
package com.example.dsassignment.fireAlarmSystem;


import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.CrossOrigin;
@CrossOrigin(origins="*")
@SpringBootApplication
public class FireAlarmSystemApplication {

  //main method which runs the spring boot application
  public static void main(String[] args) {
    SpringApplication.run(FireAlarmSystemApplication.class, args);
  }

}
```

Web App

/Login.js

```
import React, {Component} from 'react';
import {Link, Redirect,useHistory} from 'react-router-dom';
import Logo from "./images/navicon.png";
import axios from 'axios';

class Login extends Component {
  //initalize state
  state = {
    email: '',
    password: '',
    redirect: false
  }

  //on change set the email state
  handleEmail(text){
    this.setState({email: text.target.value})
  }

  //on change set the password state
  handlePassword(text){
    this.setState({password: text.target.value})
  }


  //method to execute the login functinality
  login = e =>{

    e.preventDefault();
    console.log(this.state);
    //check if the fields are empty or not
    if(this.state.email==='' || this.state.password===''){

      alert('Please fill out email and password')

    }
    else {
      //using axios fetch the user details and validate the user
      axios.get("http://localhost:8080/getUser/"+this.state.email)
        .then(response =>{

          if(response.data.password===this.state.password){
            console.log(response)
            this.props.history.push('/sensor')
          }else{
            alert('Invalid password or email');
          }

        })
        .catch(error =>{
          console.log(error)
        })
    }
  }




    //user interface for the login component
  render() {
    return (
      <div>
        <nav className="navbar navbar-light bg-dark mb-4">
          <div className="navbar-brand text-light">
            <img src={Logo} width="30" height="30" className="d-inline-block align-top mr-2" alt=""/>
            Fire Alarm System
          </div>
        </nav>
        <div className="col-sm-4 mt-3" style={{marginLeft:'30%'}}>
          <div className="card" style={{marginTop:'40%'}}>
            <div className="card-body">
              <h3 className="card-title text-center mb-3">Login</h3>
              <form className="login-form" onSubmit={this.login}>
                <div className="form-group">
                  <input type="email" className="form-control" id="exampleInputEmail1"
                      aria-describedby="emailHelp" placeholder="Email" onChange={(text)
=>{this.handleEmail(text)}}/>
                </div>
                <div className="form-group">
                  <input type="password" className="form-control" id="exampleInputPassword1"
                      placeholder="Password" onChange={(text) =>{this.handlePassword(text)}}/>
                </div>
                {/*<Link to={'/sensor'}><button type="submit" className="btn btn-primary btn-lg btn-
block" onClick={() => this.login()}>Login</button></Link>*/}
                <button type="submit" className="btn btn-primary btn-lg btn-block">Login</button>
              </form>
            </div>
          </div>
        </div>
      </div>
    );
  }
}

export default Login;
```

/Sensor.js

```
import React, {Component} from 'react';
import * as ReactDOM from 'react-dom';
import {
  RadialGauge
} from '@progress/kendo-react-gauges';
import '@progress/kendo-theme-default/dist/all.css';
import Logo from './images/navicon.png';
import {Link} from 'react-router-dom';

class Sensor extends Component {
  //initilaize the state of the variables
  state = {
    isLoading: true,
    alarms: [],
    value: 0
  }
  intervalID;

  async componentDidMount() {
    /*when conponentDidMount executes run getData method and setInterval to 4 seconds
      to automatically retrieve data using the REST api
    */
    await this.getData();

    this.intervalID = setInterval(this.getData.bind(this), 40000);
    setInterval( () =>{
      this.setState({
        value:this.state.value
      })
    },0);
  }

   //countdown implementation
    getTimer(){
      this.myInterval = setInterval(() =>{
        this.setState(prevState =>({
          count:this.state.count -1

        }))
      },1000)


    }

    //safetly clear the interval
  componentWillMount(){
    clearInterval(this.myInterval)
  }

  //method to fetch the alarms through the REST api
  async getData(){
    const response = await fetch('/alarms')
    const body = await response.json();
    this.setState({alarms : body,isLoading: false});
  }

  //user interface for the Sensor component
  render() {
    const {alarms,isLoading} = this.state;
    if (isLoading)
      return (<div><h3 className="text-dark">Loading...</h3></div>)
    const radialOptions = {
      pointer: {
        value: this.state.value
      }
    };

    return (
      <div>
        <nav className="navbar navbar-light bg-dark mb-2">
          <div className="navbar-brand text-light">
            <img src={Logo} width="30" height="30" className="d-inline-block align-top mr-2" alt=""/>
               Fire Alarm System
          </div>
          <ul className="navbar-nav">
            <li className="nav-item active">
              <Link to={'/'} className="nav-link text-light" href="#">Log out <span className="sr-
only">(current)</span></Link>
            </li>
          </ul>
        </nav>
        <div className="text-danger mb-4 text-center">*New data will be fetched every 40 seconds</div>
        <div className="text-danger mb-4 text-center">Timer: {count} seconds left</div>
        <div className="row ml-3 mr-3">
          {
            alarms.map(alarm =>
              <div className="col-sm-4 mb-4">
                <div className="card">
                  <div className="card-body" id={alarm.id}>
                    <h3 className="card-title text-center">Sensor {alarm.id}</h3>
                    <h5><span className="mr-2">Floor No.</span> : <span className="ml-
2">{alarm.floorNum}</span></h5>
                    <h5><span className="mr-2">Room No.</span>: <span className="ml-
2">{alarm.roomNum}</span></h5>
                    <div className="w-50 mb-2" style={{marginLeft:'25%'}}>
                      <RadialGauge {...{value:alarm.co2level*10}}/>
                      {alarm.co2level>5 ? <div className="text-center text-danger">CO<small>2</small>
Level : {alarm.co2level}</div> :
                        <div className="text-center text-success">CO<small>2</small> Level :
{alarm.co2level}</div>}
                    </div>
                    <div className="w-50" style={{marginLeft:'25%'}}>
                      <RadialGauge {...{value:alarm.smokeLevel*10}}/>
                      {alarm.smokeLevel>5 ? <div className="text-center text-danger">Smoke Level :
{alarm.smokeLevel}</div> :
                        <div className="text-center text-success">Smoke Level :
{alarm.smokeLevel}</div>}
                    </div>
                  </div>
                </div>
              </div>
            )}
        </div>
      </div>
    );
  }
}



export default Sensor;
```