Sri Lanka Institute of Information Technology

BSc (Hons) In Information Technology –Software Engineering

Fire Alarm Monitoring System

Project Report

Distributed Systems (SE3020)

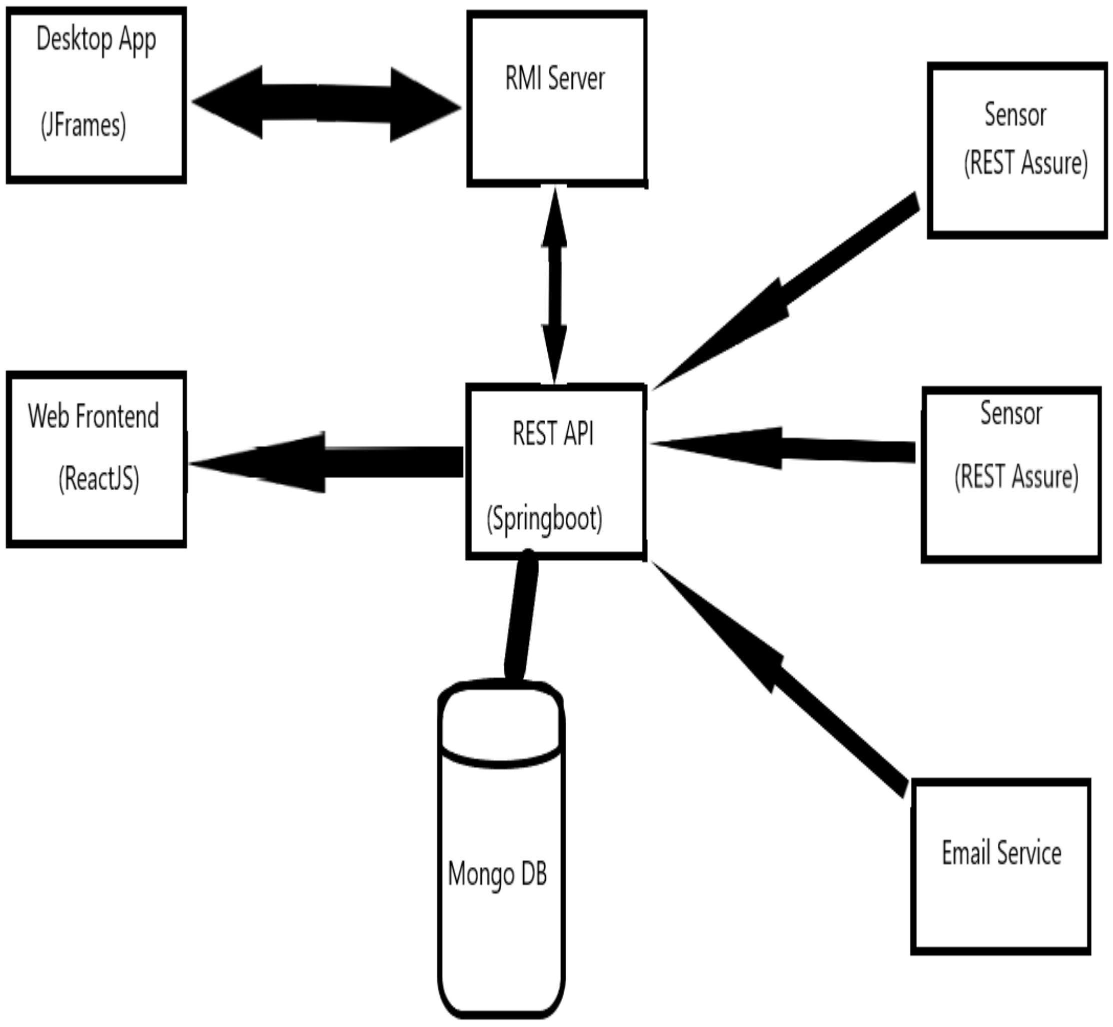
Year 3, Semester 1 – 2020

Contents

Group Details

Registration Number	**Student Name**
IT18125412	Thalakumbura T. M. D. D.
IT18112160	Bandara M. L. S.
IT18026580	De Seram E. M. C. J. D.
IT18143928	Perera B. A. E. K.

High level architectural diagram



Service Interfaces

Desktop application service interface

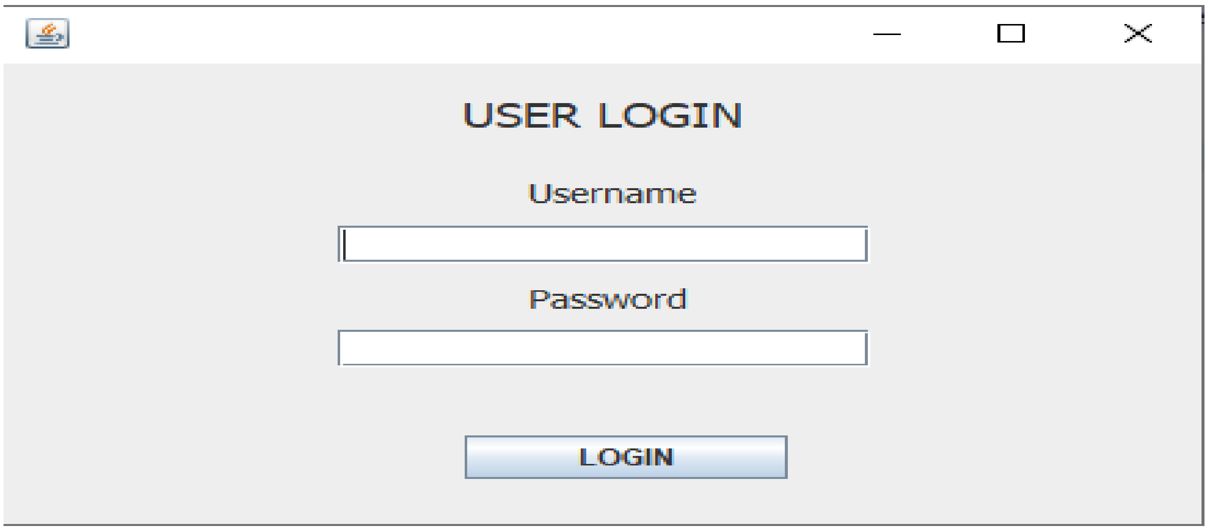
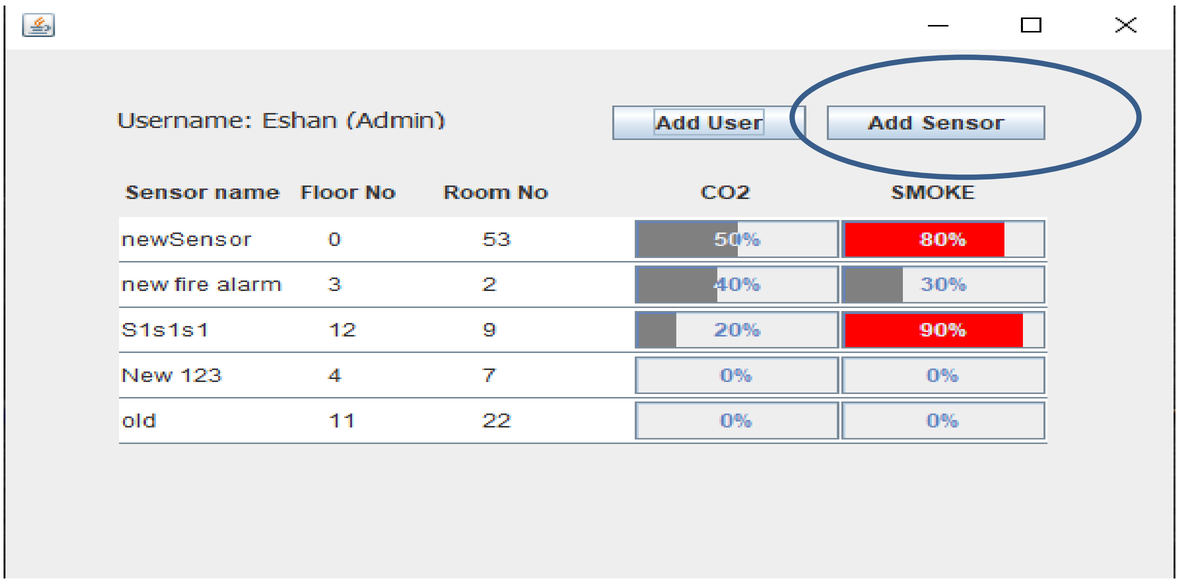


Figure 1: Login interface

The above figure 1 show the login UI of the system, here the system will verify if the current user is an admin user or not. If the user has admin access, there will be special services provided to the user. Else if the user does not have admin privileges, the user can view the sensor status.

If user has admin privileges, the system provides the below services. Upon login admin user will come to the Admin Interface (Figure 2).



Username: Eshan (Admin)

Sensor name	Floor No	Room No	CO2	SMOKE
newSensor	0	53	50%	80%
new fire alarm	3	2	40%	30%
S1s1s1	12	9	20%	90%
New 123	4	7	0%	0%
old	11	22	0%	0%

Figure 2: Admin interface

The above interface (Figure 2) shows the current status of the sensor. And if the admin user wants to add a new sensor user can click the 'Add sensor' button and will be shown the below interface (Figure 3). This will enable user to enter related details; name, floor number and room number to add a new sensor.

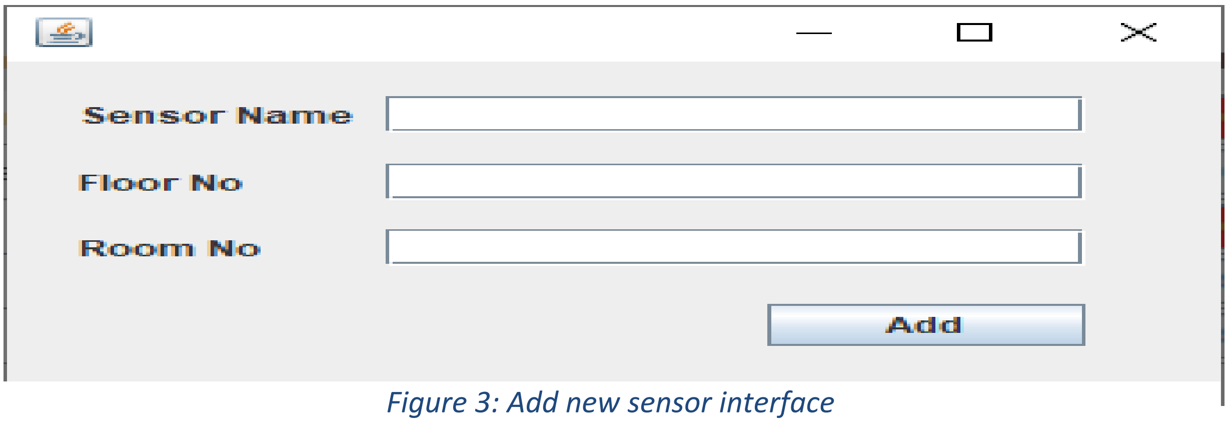
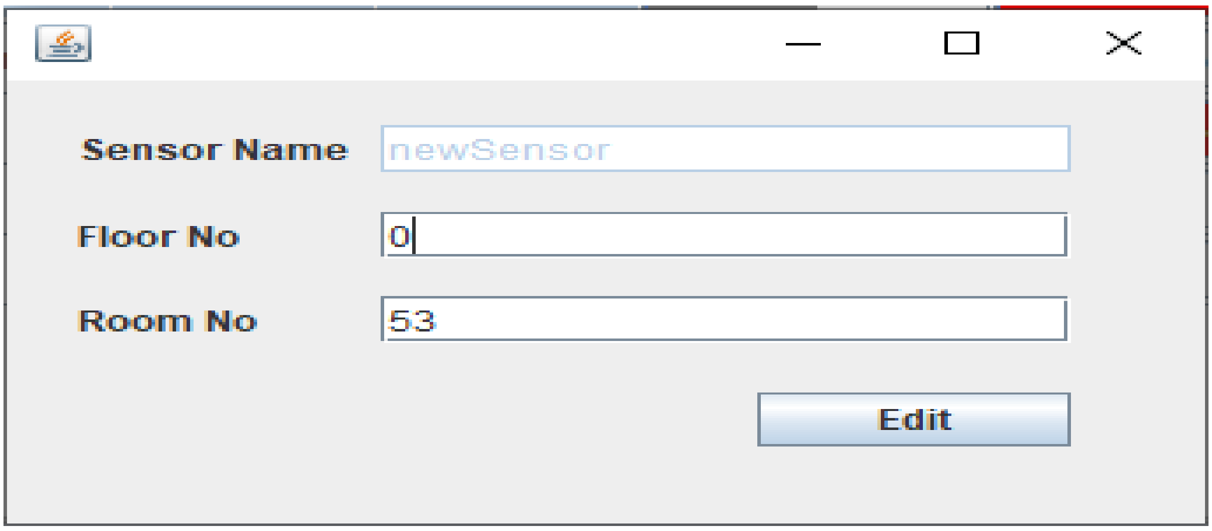


Figure 3: Add new sensor interface

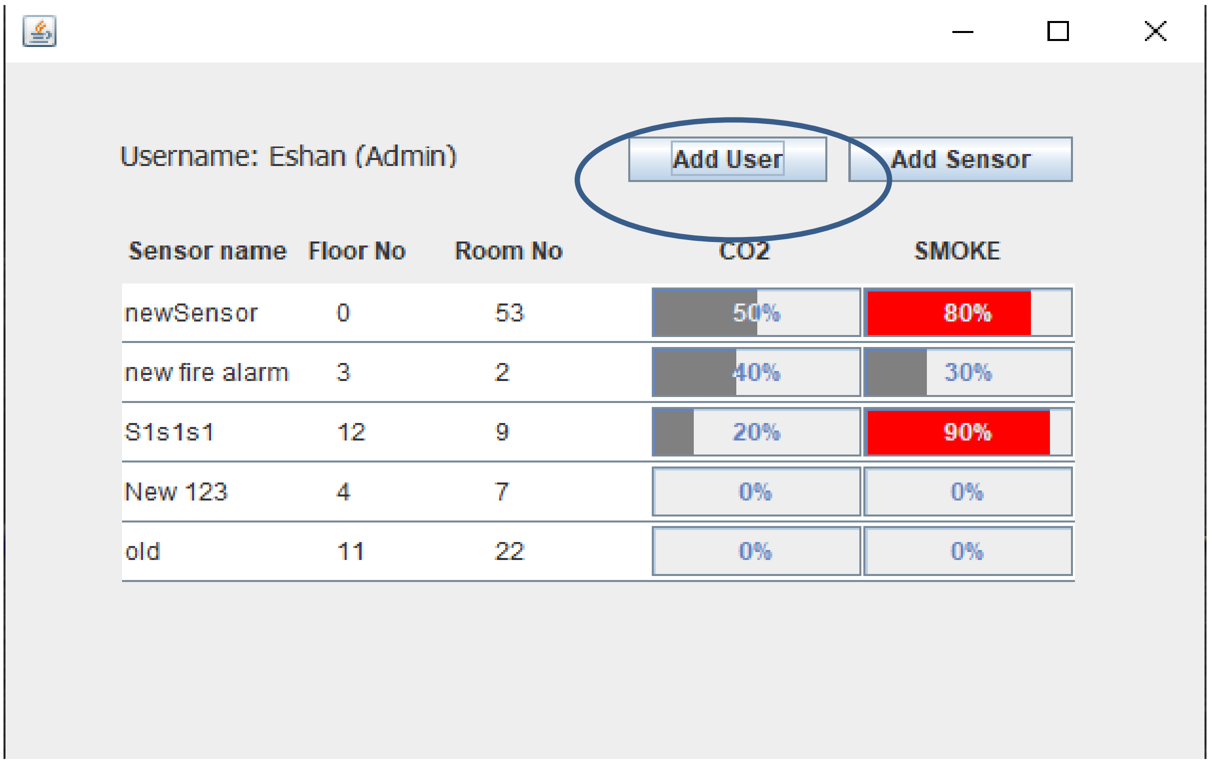
If admin wishes to update a sensor, admin can click on the sensor name and will be redirected to the below interface (Figure 4).



Sensor Name: newSensor
Floor No: 0
Room No: 53

Figure 4: Update sensor details

If admin wishes to add a new user to the system, you can do so by clicking "Add User" button as shown below(Figure 5).



Username: Eshan (Admin)

Sensor name	Floor No	Room No	CO2	SMOKE
newSensor	0	53	50%	80%
new fire alarm	3	2	40%	30%
S1s1s1	12	9	20%	90%
New 123	4	7	0%	0%
old	11	22	0%	0%

Figure 5: Add user in admin interface

An admin also has the privilege of adding a new user as shown in the below interface

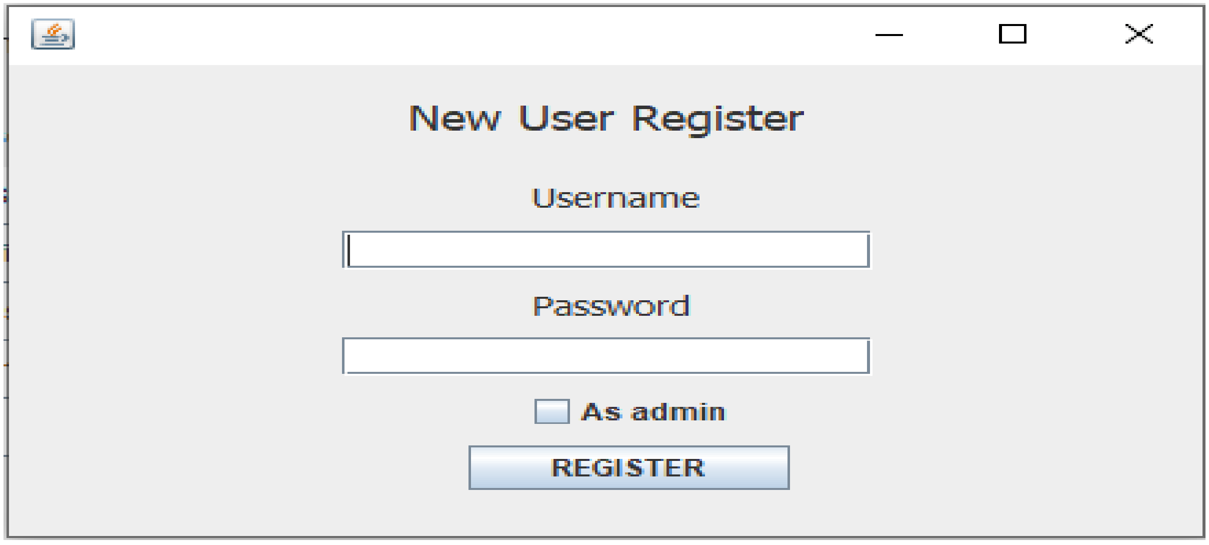
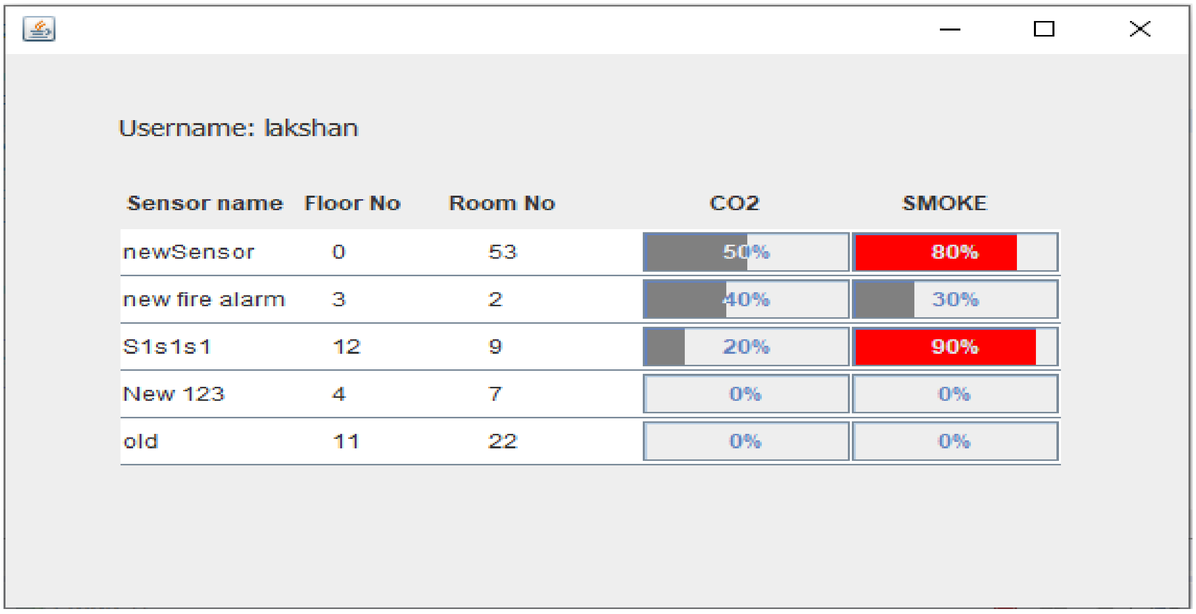


Figure 6: Register user interface

If admin wishes to add new user as an admin, then time admin can press the "As admin" checkbox.

If it a normal user login, the system desktop application will display the below interface which shows the status on fire alarming system(Figure 7).


Username: lakshan

Sensor name	Floor No	Room No	CO2	SMOKE
newSensor	0	53	50%	80%
new fire alarm	3	2	40%	30%
S1s1s1	12	9	20%	90%
New 123	4	7	0%	0%
old	11	22	0%	0%

Figure 7: Normal user interface

-

Web application service interface

If the user logs into the system using the web application, they can see sensor status on fire alarming system(Figure 9) from below interface upon login(Figure 8).

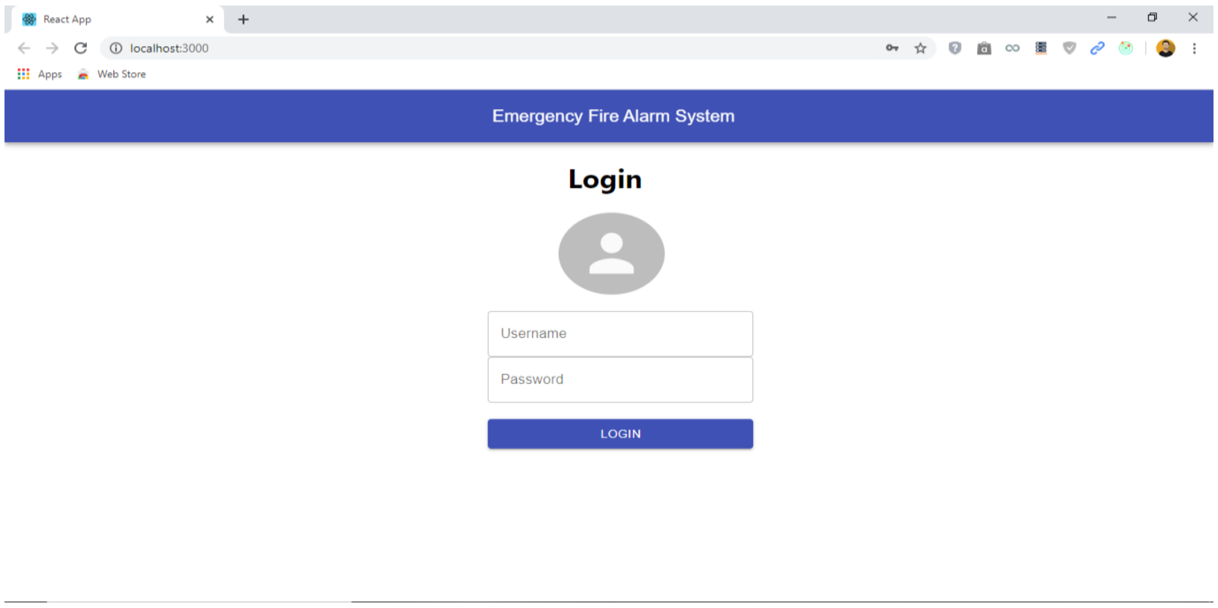
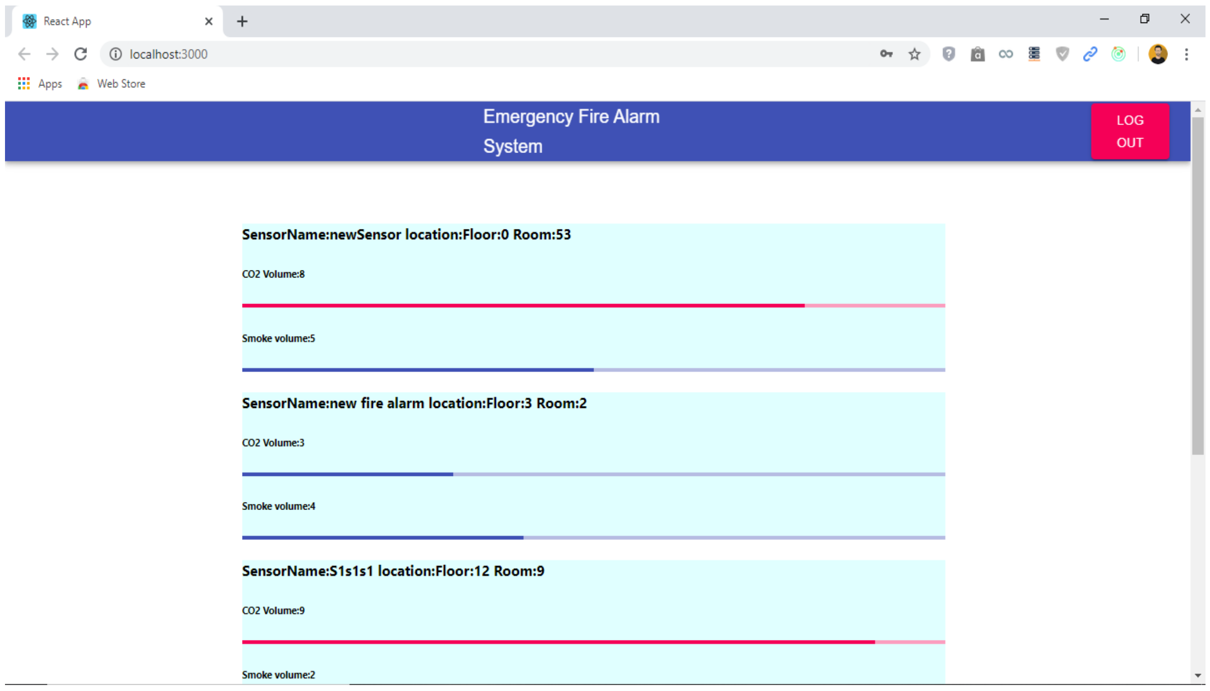


Figure 8: User login interface



SensorName:newSensor location:Floor:0 Room:53

CO2 Volume:8

Smoke volume:5

SensorName:new fire alarm location:Floor:3 Room:2

CO2 Volume:3

Smoke volume:4

SensorName:S1s1s1 location:Floor:12 Room:9

CO2 Volume:9

Smoke volume:2

Figure 9: User interface

workflow using sequence diagram

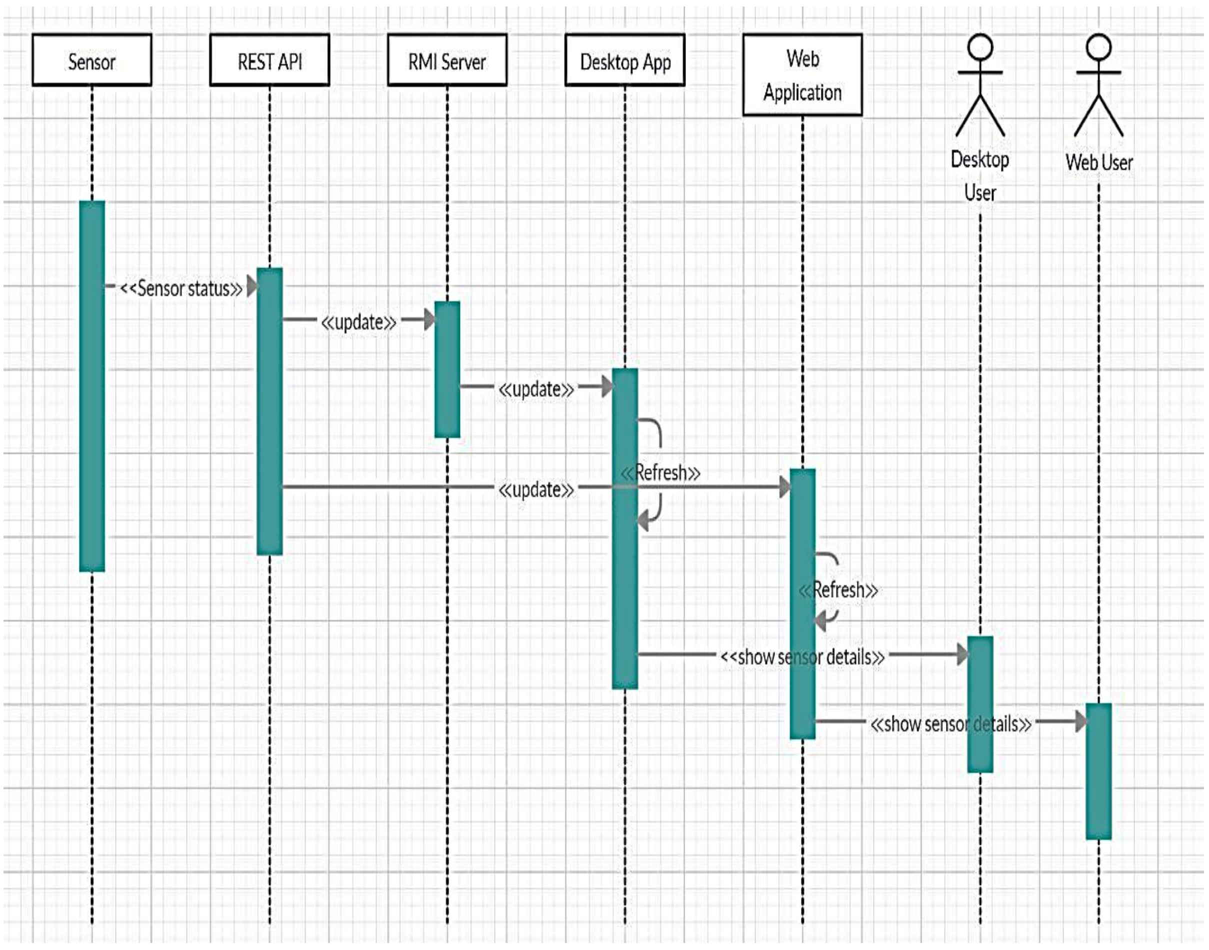


Figure 10: Sequence diagram

This represents how to show sensor status to our users.

First, we update RMI server about the sensor status in every 15 seconds. Then we give those sensor status data to our desktop application and web application. In that time user can see sensor status through desktop app or web application. And the Desktop application refreshes sensor status in every 20 seconds and web application refreshes sensor status in every 45 seconds.

Appendix

Desktop Application

AddEditSensor.java

```
package
clientApp;
            import java.awt.BorderLayout;
            import java.awt.EventQueue;
            import java.awt.HeadlessException;
            import javax.swing.JFrame;
            import javax.swing.JPanel;
            import javax.swing.border.EmptyBorder;
            import rmi_server.AlarmServer;
            import rmi_server.SensorService;
            import javax.swing.JLabel;
            import javax.swing.JOptionPane;
            import javax.swing.SwingConstants;
            import javax.swing.JTextField;
            import javax.swing.JButton;
            import java.awt.event.ActionListener;
            import java.net.MalformedURLException;
            import java.rmi.Naming;
            import java.rmi.NotBoundException;
            import java.rmi.RemoteException;
            import java.awt.event.ActionEvent;
            public class AddEditSensor extends JFrame {
                   private JPanel contentPane;
                   private JTextField sensorName;
                   private JTextField floorNo;
                   private JTextField roomNo;
                   private JButton addBtn;
```

```
public AddEditSensor() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 356, 228);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JPanel panel = new JPanel();
        panel.setBounds(10, 11, 320, 167);
        contentPane.add(panel);
        panel.setLayout(null);

        JLabel lblNewLabel_1 = new JLabel("Floor No");
        lblNewLabel_1.setBounds(10, 48, 76, 14);
        lblNewLabel_1.setHorizontalAlignment(SwingConstants.LEFT);
        panel.add(lblNewLabel_1);

        JLabel lblNewLabel = new JLabel("Sensor Name");
        lblNewLabel.setBounds(10, 11, 89, 14);
        lblNewLabel.setHorizontalAlignment(SwingConstants.LEFT);
        panel.add(lblNewLabel);

        JLabel lblNewLabel_2 = new JLabel("Room No");
        lblNewLabel_2.setBounds(10, 84, 76, 14);
        panel.add(lblNewLabel_2);

        sensorName = new JTextField();
        sensorName.setBounds(96, 8, 196, 20);
        panel.add(sensorName);
        sensorName.setColumns(10);

        floorNo = new JTextField();
        floorNo.setBounds(96, 45, 196, 20);
        panel.add(floorNo);
        floorNo.setColumns(10);

        roomNo = new JTextField();
        roomNo.setBounds(96, 81, 196, 20);
        panel.add(roomNo);
        roomNo.setColumns(10);

        addBtn = new JButton("Add");
        addBtn.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
```

```
                              if (addBtn.getText().equals("Add")) {

        System.setProperty("java.security.policy", "file:allowall.policy");
                               SensorService service = null;
                               if(validateFeilds(sensorName.getText(),
floorNo.getText(), roomNo.getText())) {
                               try {
                                   service = (SensorService)
Naming.lookup("//localhost:"+AlarmServer.RMI_PORT+"/AlarmService");
                                   if (service.addSensor(sensorName.getText(),
floorNo.getText(), roomNo.getText()).equals("success")) {
                                      JOptionPane.showMessageDialog(null, "New
Sensor added successfully....");
                                      ClientApp.refreshTable();
                                      setVisible(false);
                                              }
                               } catch (Exception ex) {
                                   System.err.println(ex.getMessage());
                               }
                               }
                              } else {

        System.setProperty("java.security.policy", "file:allowall.policy");
                               SensorService service = null;
                               if(validateFeilds(sensorName.getText(),
floorNo.getText(), roomNo.getText())) {
                               try {
                                   service = (SensorService)
Naming.lookup("//localhost:"+AlarmServer.RMI_PORT+"/AlarmService");
                                   if
(service.editSensors(sensorName.getText(), floorNo.getText(),
roomNo.getText()).equals("success")) {
                                      JOptionPane.showMessageDialog(null,
"Sensor details edited successfully....");
                                      ClientApp.refreshTable();
                                      setVisible(false);
                                              }
                               } catch (Exception ex) {
                                   System.err.println(ex.getMessage());
                               }
                               }
                              }



                          }
                      });
```

```
		addBtn.setBounds(203, 122, 89, 23);
		panel.add(addBtn);
	}




	public void editSensor(String sn, String fn, String rn){
		sensorName.setText(sn);
		floorNo.setText(fn);
		roomNo.setText(rn);
		addBtn.setText("Edit");
		sensorName.disable();
	}
	private boolean validateFeilds(String sn, String fn, String rn) {
		if(sn.isEmpty() || fn.isEmpty() || rn.isEmpty()) {
			JOptionPane.showMessageDialog(null, "please fill all
feilds...","Error",JOptionPane.ERROR_MESSAGE);
			return false;
		}else {

			return true;
		}

		//return true;
	}
}
```

ClientApp.java

```
Package
clientApp;
            import java.net.MalformedURLException;
            import java.rmi.Naming;
            import java.rmi.NotBoundException;
            import java.rmi.RemoteException;
            import java.util.Timer;
            import java.util.TimerTask;
            import rmi_server.AlarmServer;
            import rmi_server.SensorService;
            import java.awt.event.ActionListener;
            import java.awt.event.ActionEvent;
            import java.awt.event.MouseAdapter;
            import java.awt.event.MouseEvent;
            import java.awt.*;
            import javax.swing.*;
            import javax.swing.event.*;
            import javax.swing.table.*;
            import com.sun.tools.xjc.reader.RawTypeSet.Mode;
            public class ClientApp extends Thread {
                   private JFrame frame;
                   private static JTable table;
                   private JLabel userdata;
                   private String un;
                   private JButton btnNewButton;
                   private JButton addUserBtn;
                   private static JPanel panel;
                   private String userType;
                   private static JButton reloadBtn;
```

```
public ClientApp() throws InterruptedException {
        initialize();
}
private void initialize() {
        frame = new JFrame("Smart fire alarm system");
        table = new JTable();
        frame.setBounds(100, 100, 582, 390);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(null);

        panel = new JPanel();
        panel.setBounds(10, 106, 540, 174);
        frame.getContentPane().add(panel);

        JPanel panel_2 = new JPanel();
        panel_2.setBounds(10, 82, 540, 21);
        frame.getContentPane().add(panel_2);
        panel_2.setLayout(null);

        JLabel lblNewLabel = new JLabel("Sensor name");
        lblNewLabel.setBounds(10, 5, 81, 14);
        panel_2.add(lblNewLabel);

        JLabel lblNewLabel_1 = new JLabel("Floor No");
        lblNewLabel_1.setBounds(197, 5, 59, 14);
        panel_2.add(lblNewLabel_1);

        JLabel lblNewLabel_3 = new JLabel("Room No");
        lblNewLabel_3.setBounds(266, 5, 67, 14);
        panel_2.add(lblNewLabel_3);

        JLabel lblNewLabel_4 = new JLabel("CO2");
        lblNewLabel_4.setBounds(374, 5, 30, 14);
        panel_2.add(lblNewLabel_4);

        JLabel lblNewLabel_2 = new JLabel("SMOKE");
        lblNewLabel_2.setBounds(463, 5, 67, 14);
        panel_2.add(lblNewLabel_2);

        JLabel lblNewLabel_5 = new JLabel("Status");
        lblNewLabel_5.setBounds(136, 5, 51, 14);
        panel_2.add(lblNewLabel_5);
```

```
                JPanel panel_3 = new JPanel();
                panel_3.setBounds(10, 28, 540, 43);
                frame.getContentPane().add(panel_3);
                panel_3.setLayout(null);

                userdata = new JLabel("New label");
                userdata.setFont(new Font("Tahoma", Font.PLAIN, 14));
                userdata.setBounds(10, 11, 230, 14);
                panel_3.add(userdata);

                btnNewButton = new JButton("Add Sensor");
                btnNewButton.setBounds(325, 9, 106, 23);
                panel_3.add(btnNewButton);

                addUserBtn = new JButton("Add User");
                addUserBtn.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                LoginRegisterClient login = new
LoginRegisterClient();
                                login.openFrameAsRegister();

        login.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                        }
                });
                addUserBtn.setBounds(221, 9, 94, 23);
                panel_3.add(addUserBtn);

                JButton logoutbtn = new JButton("LOGOUT");
                logoutbtn.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                frame.dispose();
                                LoginRegisterClient client = new
LoginRegisterClient();
                                client.setVisible(true);
                        }
                });
                logoutbtn.setBounds(441, 9, 89, 23);
                panel_3.add(logoutbtn);



                reloadBtn = new JButton("");
                reloadBtn.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                System.out.println("refresh.........");
                                DefaultTableModel defaultTableModel =
(DefaultTableModel) table.getModel();
                                defaultTableModel.setRowCount(0);
                                loadDatatoTable();

                        }
                });
                reloadBtn.setBounds(538, 334, 12, 6);
                frame.getContentPane().add(reloadBtn);


                btnNewButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                                AddEditSensor addSensor = new AddEditSensor();
                                addSensor.setVisible(true);

        addSensor.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

                        }
                });

                loadDatatoTable();

                table.setShowVerticalLines(false);
                table.setRowHeight(30);
                table.setRowMargin(5);
                table.addMouseListener(new MouseAdapter() {
                        @Override
                        public void mouseClicked(MouseEvent arg0) {
                                if(JOptionPane.showConfirmDialog(null, "Update
sensor details") == 0) {
                                        int row = table.getSelectedRow();
                                        System.out.println(table.getValueAt(row,
0));
                                        AddEditSensor addSensor = new
AddEditSensor();

        addSensor.editSensor(table.getValueAt(row,0).toString(),
table.getValueAt(row,2).toString(), table.getValueAt(row,3).toString());
                                        addSensor.setVisible(true);

        addSensor.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                                }


                        }
                });
                table.setBackground(Color.WHITE);

        }

        public static void loadDatatoTable() {

                SwingUtilities.updateComponentTreeUI(table);
                System.setProperty("java.security.policy",
"file:allowall.policy");
         SensorService service = null;
         try {
             service = (SensorService)
Naming.lookup("//localhost:"+AlarmServer.RMI_PORT+"/AlarmService");

             table.setModel(service.getAllSensors());

         } catch (NotBoundException ex) {
             System.err.println(ex.getMessage());
         } catch (MalformedURLException ex) {
             System.err.println(ex.getMessage());
         } catch (RemoteException ex) {
             System.err.println(ex.getMessage());
         }
         ProgressRenderer pcr = new ProgressRenderer(0,10);
                TableColumnModel tcm = table.getColumnModel ();
                tcm.getColumn(5).setCellRenderer (pcr);
                tcm.getColumn(4).setCellRenderer (pcr);
                table.getColumnModel().getColumn(0).setPreferredWidth(130);
                table.getColumnModel().getColumn(2).setPreferredWidth(65);
                table.getColumnModel().getColumn(3).setPreferredWidth(60);
                table.getColumnModel().getColumn(4).setPreferredWidth(100);
                table.getColumnModel().getColumn(5).setPreferredWidth(100);
                panel.add(table);
        }
        public void openClientApp(String username, String userType) throws
InterruptedException {
                userdata.setText("Username: "+username+ " (Admin)");
                frame.setVisible(true);
                System.out.println(username +" "+ userType);

                if (userType.equals("user")) {
                        userdata.setText("Username: "+username);
                        btnNewButton.setVisible(false);
                        addUserBtn.setVisible(false);
                        table.setEnabled(false);
                }

        }

        public static void refreshTable() {
                DefaultTableModel defaultTableModel = (DefaultTableModel)
table.getModel();
                defaultTableModel.setRowCount(0);
                loadDatatoTable();
                System.out.println("refresh method...");
                SwingUtilities.updateComponentTreeUI(table);
        }

        public static void sentEmail() {
                System.setProperty("java.security.policy",
"file:allowall.policy");
         SensorService service = null;
         try {
             service = (SensorService)
Naming.lookup("//localhost:"+AlarmServer.RMI_PORT+"/AlarmService");
             if (service.sentEmail("s", "s", "s").equals("success")) {
                System.out.println("Warning Email has been sent..........");

                        }
         } catch (Exception ex) {
             System.err.println(ex.getMessage());
         }

        }
}
```

```
class ProgressRenderer extends JProgressBar implements TableCellRenderer {

        public ProgressRenderer(int min, int max) {
          super(min, max);
          this.setStringPainted(true);
        }
        public Component getTableCellRendererComponent(JTable table, Object
value,
            boolean isSelected, boolean hasFocus, int row, int column) {
              this.setValue((Integer) value);

          if (((Integer) value).intValue() > 5) {
              this.setForeground(Color.RED);
              //JOptionPane.showMessageDialog(null, "CO2 or smoke level is
high", "System Alert ", JOptionPane.WARNING_MESSAGE);
              ClientApp.sentEmail();

              }else {
                     this.setForeground(Color.GRAY);
              }
          return this;
        }
}
```

LoginRegisterClient.java

```
package
clientApp;
            import java.awt.BorderLayout;
            import java.awt.EventQueue;
            import javax.swing.JFrame;
            import javax.swing.JPanel;
            import javax.swing.border.EmptyBorder;
            import org.json.JSONObject;
            import io.restassured.RestAssured;
            import io.restassured.response.Response;
            import io.restassured.specification.RequestSpecification;
            import rmi_server.AlarmServer;
            import rmi_server.SensorService;
            import javax.swing.JLabel;
            import javax.swing.JOptionPane;
            import java.awt.Font;
            import javax.swing.JTextField;
            import javax.swing.SwingConstants;
            import javax.swing.JButton;
            import javax.swing.JPasswordField;
            import java.awt.event.ActionListener;
            import java.rmi.Naming;
            import java.util.Timer;
            import java.util.TimerTask;
            import java.util.concurrent.Executors;
            import java.util.concurrent.ScheduledExecutorService;
            import java.util.concurrent.TimeUnit;
            import java.awt.event.ActionEvent;
            import javax.swing.JCheckBox;
```

```
public class LoginRegisterClient extends JFrame {
        private JPanel contentPane;
        private JTextField username;
        private JPasswordField password;
        private JLabel headderLable;
        private JCheckBox adminSelect;
        private JButton loginBtn;
        private String asAdmin;
        /**
         * Launch the application.
         * @throws InterruptedException
         */
        public static void main(String[] args) throws InterruptedException {
                Timer timer = new Timer();
                EventQueue.invokeLater(new Runnable() {
                        public void run() {
                                try {
                                        LoginRegisterClient frame = new
LoginRegisterClient();
                                        frame.setVisible(true);
                                } catch (Exception e) {
                                        e.printStackTrace();
                                }
                        }
                });
         TimerTask task = new TimerTask() {
             @Override
             public void run() {
                System.currentTimeMillis();
                 ClientApp.refreshTable();
             }
         };
         timer.schedule(task, 40000,40000);

        }
```

```
	public LoginRegisterClient() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 450, 283);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);

		JPanel panel = new JPanel();
		panel.setBounds(10, 11, 414, 32);
		contentPane.add(panel);

		headderLable = new JLabel("USER LOGIN");
		headderLable.setFont(new Font("Tahoma", Font.PLAIN, 18));
		panel.add(headderLable);

		JPanel panel_1 = new JPanel();
		panel_1.setBounds(10, 45, 414, 186);
		contentPane.add(panel_1);
		panel_1.setLayout(null);

		username = new JTextField();
		username.setBounds(111, 41, 193, 20);
		panel_1.add(username);
		username.setColumns(10);

		JLabel lblNewLabel_1 = new JLabel("Username");
		lblNewLabel_1.setFont(new Font("Tahoma", Font.PLAIN, 14));
		lblNewLabel_1.setBounds(180, 16, 94, 14);
		panel_1.add(lblNewLabel_1);

		JLabel lblNewLabel_2 = new JLabel("Password");
		lblNewLabel_2.setFont(new Font("Tahoma", Font.PLAIN, 14));
		lblNewLabel_2.setBounds(180, 72, 101, 14);
		panel_1.add(lblNewLabel_2);

		loginBtn = new JButton("LOGIN");
		loginBtn.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {

				System.setProperty("java.security.policy",
"file:allowall.policy");
				SensorService service = null;
				if(validateFeilds(username.getText().toString(),
password.getText().toString())) {
```

```
if (loginBtn.getText().toString().equals("REGISTER")) {
                                    try {
                                        service = (SensorService)
Naming.lookup("//localhost:"+AlarmServer.RMI_PORT+"/AlarmService");
                                        if (adminSelect.isSelected()) {
                                                asAdmin = "admin";
                                            }else {
                                                asAdmin = "user";
                                            }
                                        if
(service.addUser(username.getText().toString(), password.getText().toString(),
asAdmin).equals("success")) {

       JOptionPane.showMessageDialog(null, "New User added
successfully....");
                                          setVisible(false);
                                            }

                                     } catch (Exception ex) {
                                        System.err.println(ex.getMessage());
                                     }

                                    }else {
                                          try {
                                        service = (SensorService)
Naming.lookup("//localhost:"+AlarmServer.RMI_PORT+"/AlarmService");
                                        String value =
service.loginUser(username.getText().toString(),
password.getText().toString());
                                        if (value.equals("no_user") ||
value==null) {

       JOptionPane.showMessageDialog(null, "Invalid Login details...");
                                            }else {

       System.out.println(value);
                                                String[] valueArray
= value.split("/");
                                                ClientApp app = new
ClientApp();

       app.openClientApp(valueArray[0], valueArray[1]);
                                                setVisible(false);

                                            }
                                     } catch (Exception ex) {
                                        System.err.println(ex.getMessage());
                                    }
                                   }


                        }
                       }
                });
                loginBtn.setBounds(157, 152, 117, 23);
                panel_1.add(loginBtn);

                password = new JPasswordField();
                password.setBounds(111, 96, 193, 20);
                panel_1.add(password);

                adminSelect = new JCheckBox("As admin");
                adminSelect.setBounds(177, 123, 97, 23);
                adminSelect.setVisible(false);
                panel_1.add(adminSelect);

        }

        public void openFrameAsRegister() {
                headderLable.setText("New User Register");
                adminSelect.setVisible(true);
                loginBtn.setText("REGISTER");
                setVisible(true);
        }

        private boolean validateFeilds(String un, String pass) {
                if(un.isEmpty() || pass.isEmpty()) {
                        JOptionPane.showMessageDialog(null, "Enter username and
password","Error",JOptionPane.ERROR_MESSAGE);
                        return false;
                }else {
                        return true;
                }
        }
}
```

RMI Server

AlarmServer.java

```
package
rmi_server;
              import java.rmi.RemoteException;
              import java.rmi.registry.LocateRegistry;
              import java.rmi.registry.Registry;
              import java.rmi.server.UnicastRemoteObject;
              import javax.swing.table.DefaultTableModel;
              import org.json.JSONArray;
              import org.json.JSONObject;
              import io.restassured.RestAssured;
              import io.restassured.response.Response;
              import io.restassured.specification.RequestSpecification;
              public class AlarmServer extends UnicastRemoteObject implements SensorService
              {

                     private int clientCount;
                     public static final int RMI_PORT = 5097;

                     protected AlarmServer() throws RemoteException {
                            super();
                            clientCount = 0;
                     }
                     public synchronized int countClients() throws RemoteException {
                            // TODO Auto-generated method stub
                            return ++clientCount;
                     }
                     public synchronized String addSensor(String sensorName, String
              floorNo, String roomNo) throws RemoteException {
                            RestAssured.baseURI="http://localhost:8080";
                            RequestSpecification request= RestAssured.given();
                            JSONObject reqparam= new JSONObject();

                            reqparam.put("SensorName",sensorName);
                            reqparam.put("floorNo", floorNo);
                            reqparam.put("roomNo", roomNo);

                            request.header("Content-Type", "application/json");
                            request.body(reqparam.toString());
                            Response response= request.post("/registersensor");


                            if (response.statusCode()==200) {
```

```
                        return "success";
                }else {
                        return null;
                }

        }
        public synchronized String editSensors(String sensorName, String
floorNo, String roomNo) throws RemoteException {
                RestAssured.baseURI="http://localhost:8080";
                RequestSpecification request= RestAssured.given();
                JSONObject reqparam= new JSONObject();

                reqparam.put("SensorName",sensorName);
                reqparam.put("floorNo", floorNo);
                reqparam.put("roomNo", roomNo);

                request.header("Content-Type", "application/json");
                request.body(reqparam.toString());
                Response response= request.post("/updatesensor");

                if (response.statusCode()==200) {
                        return "success";
                }else {
                        return null;
                }
        }
        public synchronized DefaultTableModel getAllSensors() throws
RemoteException {

                Object data[][] = {};
        String col[] = {"SensorName","Status",
"floorNo","roomNo","smoke","co2"};
        DefaultTableModel model = new DefaultTableModel(data,col);

        RestAssured.baseURI="http://localhost:8080";
                RequestSpecification request= RestAssured.given();
                request.header("Content-Type", "application/json");
                Response response= request.post("/getallsensors");
                System.out.println(response.statusCode());
                JSONArray array = new JSONArray(response.print());
                System.out.println("Number of sensors: "+array.length());

                for(int i=0; i<array.length(); i++) {
                        String status;
                        JSONObject jsonObject = array.getJSONObject(i);
                        String sn = jsonObject.getString("sensorName");
```

```
                    String fn = jsonObject.getString("floorNo");
                    String rn = jsonObject.getString("roomNo");
                    int smoke  = jsonObject.getInt("smoke");
                    int co2 = jsonObject.getInt("co2");

                    if (smoke == 0 && co2 == 0) {
                           status = "InActive";
                    }else {
                           status = "Active";
                    }

                    model.insertRow(i,new
Object[]{sn,status,fn,rn,co2,smoke});
             }
             return model;
       }

       public static void main(String[] args) {
             System.setProperty("java.security.policy",
"file:allowall.policy");
        try{
             AlarmServer svr = new AlarmServer();
           Registry registry = LocateRegistry.createRegistry(RMI_PORT);
           registry.rebind("AlarmService", svr);
           System.out.println ("Alarm Service started....");
        }
        catch(RemoteException re){
           System.err.println(re.getMessage());
        }
       }
       public synchronized String addUser(String username, String password,
String type) {
             RestAssured.baseURI="http://localhost:8080";
             RequestSpecification request= RestAssured.given();
             JSONObject reqparam= new JSONObject();

             reqparam.put("username", username);
             reqparam.put("password", password);
             reqparam.put("type", type);

             request.header("Content-Type", "application/json");
             request.body(reqparam.toString());
             Response response= request.post("/regiser");

             if (response.statusCode()==200) {
                    return "success";
```

```
                }else {
                        return null;
                }
        }
        public synchronized String loginUser(String username, String
password) {
                RestAssured.baseURI="http://localhost:8080";
                RequestSpecification request= RestAssured.given();
                JSONObject reqparam= new JSONObject();

                reqparam.put("username",username);
                reqparam.put("password", password);

                request.header("Content-Type", "application/json");
                request.body(reqparam.toString());
                Response response= request.post("/login");

                if (response.statusCode()==200) {
                        if(response.print().isEmpty()) {
                                return "no_user";
                        }else {
                                System.out.println("success");
                                JSONObject jsonObject = new
JSONObject(response.print());
                                String userValues =
jsonObject.getString("username")+"/"+jsonObject.getString("type");
                                return userValues;
                        }
                }else {
                        return null;
                }
        }

        public synchronized String sentEmail(String sensorName, String
floorNo, String roomNo) throws RemoteException {
                RestAssured.baseURI="http://localhost:8080";
                RequestSpecification request= RestAssured.given();
                JSONObject reqparam= new JSONObject();

                reqparam.put("sensorName",sensorName);
                reqparam.put("floorNo", floorNo);
                reqparam.put("roomNo", roomNo);

                request.header("Content-Type", "application/json");
                request.body(reqparam.toString());
                Response response= request.post("/sentemail");
```

```
                    if (response.statusCode()==200) {
                            return "success";
                    }else {
                            return null;
                    }
            }

    }
```

SensorService.java

```
package
rmi_server;
        import java.rmi.Remote;
        import java.rmi.RemoteException;
        import javax.swing.JTable;
        import javax.swing.table.DefaultTableModel;
        public interface SensorService extends Remote {
                public String addSensor(String sensorName, String floorNo, String
        roomNo) throws RemoteException;
                public String editSensors(String sensorName, String floorNo, String
        roomNo) throws RemoteException;
                public DefaultTableModel getAllSensors() throws RemoteException;

                public int countClients() throws RemoteException;
                public String addUser(String username, String password, String
        type)throws RemoteException;
                public String loginUser(String username, String password)throws
        RemoteException;

                public String sentEmail(String sensorName, String floorNo, String
        roomNo) throws RemoteException;

        }
```

REST Backend

Service

SensorService.java

```
package
com.example.demo.Service;
                            import java.util.List;
                            import org.springframework.beans.factory.annotation.Autowired;
                            import org.springframework.stereotype.Service;
                            import com.example.demo.model.Sensor;
                            import com.example.demo.repo.SensorRepo;
                            @Service
                            public class SensorService {

                                   @Autowired
                                   SensorRepo repo;

                                   public String  RegisterSensor(String SensorName,String
                            floorNo,String roomNo) {

                                          try {

                                                 repo.save(new Sensor(SensorName, floorNo,
                            roomNo));
                                                 return "success";
                                          }catch(Exception e) {

                                                 return "error";

                                          }

                                   }

                                   public List<Sensor> getAllSensors(){

                                          return repo.findAll();
                                   }
```

```
        public void updateSensor(String SensorName,String
floorNo,String roomNo) {
                try {
                        List<Sensor> list= repo.findAll();
                        for(Sensor sensor:list) {

        if(sensor.getSensorName().equals(SensorName)) {

        sensor.setFloorNo(floorNo);
                                        sensor.setRoomNo(roomNo);
                                        repo.save(sensor);
                                        System.out.println("
Sensor updated.....");
                                }
                }
                }catch(Exception e){}
        }

        public void updateSensorCO2(String SensorName,int CO2)
{

                try {

                        List<Sensor> list= repo.findAll();

                        for(Sensor sensor:list) {


        if(sensor.getSensorName().equals(SensorName)) {

                                        sensor.setCO2(CO2);

                                        repo.save(sensor);
                                        System.out.println(" CO2
updated.....");

                                }
                }
                }catch(Exception e)
                {


                }
        }
```

```
          public void updateSmoke(String SensorName,int smoke) {

                 try {


                        List<Sensor> list= repo.findAll();

                        for(Sensor sensor:list) {


          if(sensor.getSensorName().equals(SensorName)) {

                                      sensor.setSmoke(smoke);

                                      repo.save(sensor);

                               }
                        }



                 }catch(Exception e)
                 {


                 }
          }
   }
```

UserService.java

```
package
com.example.demo.Service;
                                        import org.springframework.beans.factory.annotation.Autowired;
                                        import org.springframework.stereotype.Service;
                                        import com.example.demo.repo.UserRepo;
                                        import com.example.demo.model.User;
                                        @Service
                                        public class UserService {

                                                @Autowired
                                                UserRepo repo;


                                                public String RegisterUser(String username,String
                                        password, String type) {

                                                        try {
                                                                repo.save(new User(username, password,
                                        type));
                                                                return "success";
                                                        }catch(Exception e) {

                                                                return "error";
                                                        }
                                                }
                                                public User Login(String username,String password) {

                                                        try {
                                                                User user =
                                        repo.findByUsername(username);

                                                                if(user!=null) {

                                                if(user.getPassword().equals(password)) {
                                                                                return user;
                                                                        }else { return null;}
                                                                }else {
                                                                        return null;
                                                                }
                                                        }catch(Exception e) {

                                                                return null;
                                                        }
```

```
        }

    }
```

Controller

EmailController.java

```
package
com.example.demo.controller;
import org.springframework.web.bind.annotation.RequestBody;
import
org.springframework.web.bind.annotation.RequestMapping;
import
org.springframework.web.bind.annotation.RequestMethod;
import
org.springframework.web.bind.annotation.RestController;
import com.example.demo.data.EmailSentRequest;
import com.example.demo.model.Sensor;
@RestController
public class EmailController {
       @RequestMapping(path = "/sentemail",method =
RequestMethod.POST,consumes = "application/json")
       public String sendNotification(@RequestBody
EmailSentRequest request) {

              System.out.println("Email
Sent............");
              System.out.println("Sensorname:
"+request.sensorName+" FloorNo: "+request.floorNo+"
RoomNo"+request.roomNo);


              return "success";
       }


}
```

SensorController.java

```
package
com.example.demo.controll
er;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.Service.SensorService;
import com.example.demo.data.RegisterSensorRequest;
import com.example.demo.data.UpdateRequest;
import com.example.demo.data.UpdateSensorRequest;
import com.example.demo.data.updateCO2Request;
import com.example.demo.model.Sensor;
@RestController
@CrossOrigin(origins = "http://localhost:3000")
public class SensorController {

        @Autowired
        SensorService service;

        @RequestMapping(path = "/registersensor",method =
RequestMethod.POST,consumes = "application/json")
        public String AddSensor(@RequestBody
RegisterSensorRequest request) {

                return
service.RegisterSensor(request.SensorName, request.floorNo,
request.roomNo);

        }

        @RequestMapping("/getallsensors")
        public List<Sensor> getAllSensors(){

                return service.getAllSensors();
        }

        @RequestMapping(method = RequestMethod.POST,
path="/updatesensor")
        public void UpdateSensor(@RequestBody
UpdateSensorRequest request) {

            service.updateSensor(request.SensorName,
request.floorNo, request.roomNo);
       }

       @RequestMapping(method = RequestMethod.POST,
path="/updateco2")
       public void UpdateCO2(@RequestBody updateCO2Request
request) {


service.updateSensorCO2(request.SensorName,request.co2);
       }

       @RequestMapping(path="/updatesmoke",method=RequestMetho
d.POST)
       public void UpdateSmoke(@RequestBody UpdateRequest
request) {


service.updateSmoke(request.SensorName,request.smoke);
       }



}
```

UserController.java

```
package
com.example.demo.controller;
                                                import javax.swing.text.html.FormSubmitEvent.MethodType;
                                                import
                                                org.springframework.beans.factory.annotation.Autowired;
                                                import org.springframework.http.MediaType;
                                                import org.springframework.web.bind.annotation.CrossOrigin;
                                                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.RequestMethod;
                                                import org.springframework.web.bind.annotation.RequestParam;
                                                import
                                                org.springframework.web.bind.annotation.RestController;
                                                import com.example.demo.Service.UserService;
                                                import com.example.demo.data.UserRequest;
                                                import com.example.demo.data.RgisterUserRequest;
                                                import com.example.demo.model.User;
                                                @RestController
                                                @CrossOrigin(origins = "http://localhost:3000")
                                                public class UserController {

                                                        @Autowired
                                                        UserService service;

                                                        @RequestMapping(path = "/regiser",method =
                                                RequestMethod.POST)
                                                        public String registerUser(@RequestBody
                                                RgisterUserRequest request) {

                                                                return service.RegisterUser(request.username,
                                                request.password, request.type);
                                                        }

                                                        @RequestMapping(method = RequestMethod.POST, path =
                                                "/login")
                                                        public User Login(@RequestBody UserRequest request)
                                                {

                                                                System.out.println(request.username+"
                                                "+request.password);
```

```
                              return
                    service.Login(request.username,request.password);


                        }

                    }
```

Data

EmailSentRequest.java

```
package
com.example.demo.data;
                    public class EmailSentRequest {

                        public String sensorName;
                        public String floorNo;
                        public String roomNo;
                    }
```

RegisterSensorRequest.java

```
package
com.example.demo.data;
                    public class RegisterSensorRequest {

                        public String SensorName;
                        public String floorNo;
                        public String roomNo;
                    }
```

RgisterUserRequest.java

```
    package com.example.demo.data;
    public class RgisterUserRequest {

        public String username;
        public String password;
        public String type;
    }
```

UpdateRequest.java

```
        package com.example.demo.data;
        public class UpdateRequest {

                public String  SensorName;
                public int smoke;

        }
```

UpdateSensorRequest.java

```
package
com.example.demo.data;
                         public class UpdateSensorRequest {

                                 public String SensorName;
                                 public String floorNo;
                                 public String roomNo;

                         }
```

UserRequest.java

```
package
com.example.demo.data;
                         public class UserRequest {

                                 public String username;
                                 public String password;


                         }
```

updateCO2Request.java

```
        package com.example.demo.data;
        public class updateCO2Request {


                public String SensorName;
                public int co2;
        }
```

Model

Sensor.java

```
package
com.example.demo.model;
                          import org.springframework.data.annotation.Id;
                          import org.springframework.data.mongodb.core.mapping.Document;
                          @Document
                          public class Sensor {

                                 @Id
                                 private String id;
                                 private String SensorName;
                                 private String floorNo;
                                 private String roomNo;
                                 private int CO2=0;
                                 private int smoke=0;

                          public Sensor(String SensorName,String floorNo,String roomNo ) {

                                 this.floorNo=floorNo;
                                 this.roomNo=roomNo;
                                 this.SensorName=SensorName;

                          }
                                 public String getId() {
                                        return id;
                                 }
                                 public void setId(String id) {
                                        this.id = id;
                                 }
                                 public int getCO2() {
                                        return CO2;
                                 }
                                 public void setCO2(int cO2) {
                                        CO2 = cO2;
                                 }
                                 public int getSmoke() {
                                        return smoke;
                                 }
                                 public void setSmoke(int smoke) {
                                        this.smoke = smoke;
                                 }
                                 public String getFloorNo() {
                                        return floorNo;
```

```
        }
        public void setFloorNo(String floorNo) {
                this.floorNo = floorNo;
        }
        public String getRoomNo() {
                return roomNo;
        }
        public void setRoomNo(String roomNo) {
                this.roomNo = roomNo;
        }
        public String getSensorName() {
                return SensorName;
        }
        public void setSensorName(String sensorName) {
                SensorName = sensorName;
        }



}
```

User.java

```
package
com.example.demo.model;
import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;
@Document
public class User {

        @Id
        private String id;
        private String username;
        private String password;
        private String type;
        public User(String username,String password,String type)
{
                this.username=username;
                this.password=password;
                this.type=type;
        }
        public String getType() {
                return type;
        }
```

```
	public void setType(String type) {
		this.type = type;
	}
	public String getId() {
		return id;
	}
	public void setId(String id) {
		this.id = id;
	}
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}


}
```

Repo

SensorRepo.java

```
package com.example.demo.repo;
import org.springframework.data.mongodb.repository.MongoRepository;
import com.example.demo.model.Sensor;
public interface SensorRepo extends MongoRepository<Sensor, String> {

        public Sensor findByid(String id);
        public Sensor findBySensorName(String SensorName);

}
```

UserRepo.java

```
package
com.example.demo.repo;
                         import
                         org.springframework.data.mongodb.repository.MongoRepository;
                         import org.springframework.stereotype.Repository;
                         import com.example.demo.model.User;
                         @Repository
                         public interface UserRepo extends MongoRepository<User, String> {

                                 public User findByUsername(String username);


                         }
```

RestfullAppApplication.java

```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class RestfullAppApplication {
	public static void main(String[] args) {
		SpringApplication.run(RestfullAppApplication.class, args);
	}
}
```

resources/application.properties

```
spring.data.mongodb.uri=mongodb://localhost:27017/ds
```

Sensor App

Sensor.java

```
import
java.util.Random;
                    import java.util.Scanner;
                    import org.json.JSONArray;
                    import org.json.JSONObject;
                    import io.restassured.RestAssured;
                    import io.restassured.response.Response;
                    import io.restassured.specification.RequestSpecification;
                    public class Sensor {


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

                                  String[] sensors = {"newSen","newfirealarm","new"};
                                  Random random = new Random();

                                  int i=0;
                                  while(true) {




                                         if(i>9) {
                                                i=0;
                                         }



                                   RestAssured.baseURI="http://localhost:8080";
                                   RequestSpecification request= RestAssured.given();
                                   JSONObject reqparam= new JSONObject();

                                   reqparam.put("SensorName",sensors[random.nextInt(3)]);
                                   reqparam.put("smoke", i);

                                   request.header("Content-Type", "application/json");

                                   request.body(reqparam.toString());

                                   Response response= request.post("/updatesmoke");

                                   System.out.println(response.statusCode());

                                   RestAssured.baseURI="http://localhost:8080";
```

```
                            RequestSpecification request2= RestAssured.given();
                            JSONObject reqparam2= new JSONObject();

                            reqparam2.put("SensorName",sensors[random.nextInt(3)]);

                            if(i<5) {

                            reqparam2.put("co2", i+4);
                            }else {
                                   reqparam2.put("co2", i);
                            }

                            request2.header("Content-Type", "application/json");

                            request2.body(reqparam2.toString());

                            Response response2= request2.post("/updateco2");


                            System.out.println(response2.statusCode());


                                  Thread.sleep(25000);

                                  i=i+1;

                           }



                   }
           }
```

Web Frontend

Actions

signin.js

```
const
signin=()=>{
                   return {
                       type:'SIGN_IN'
                   }
               }
               export default  signin;
```

signout.js

```
const
signout=()=>{
                    return {
                        type:'SIGN_OUT'
                    }
                }
                export default signout;
```

Components

Login.js

```
import
React,
{Component
} from
'react';
             import {AppBar,Typography,Toolbar,Avatar,TextField,Button} from "@material-
             ui/core";
             import './Login.css'
             import {useSelector} from "react-redux";
             import axios from  'axios'
             class Login extends Component {
                 constructor(props) {
                     super(props);
                     this.state={
                         username:'',
                         password:''
```

```
        }
    }
    handlechange=(e)=>{
        e.preventDefault();
        this.setState({
            [e.target.name]: e.target.value
        })
    }
    login=(e)=> {
            e.preventDefault();
            axios.post('http://localhost:8080/login',{
                username:this.state.username,
                password:this.state.password
            }).then(response=>{
               if(response.data.hasOwnProperty("username")){
                   this.props.login();
               }
            })
    }
    render() {
        return (
            <div>
                <AppBar position="static">
                    <Toolbar>
                        <Typography variant="h6" style={{marginLeft:'40%'}}>
                           Emergency Fire Alarm System
                        </Typography>
                    </Toolbar>
                </AppBar>
                <div className='login-div'>
                    <h1 style={{marginLeft:'90px'}}>Login</h1>
                    <Avatar
style={{width:'120px',height:'100px',marginLeft:'80px',marginBottom:'20px'}}><
/Avatar>
                    <div className='input-div'>
                    <TextField name='username' id="outlined-basic"
label="Username" variant="outlined" className="textfield"
onChange={this.handlechange} />
                    <br />
                    <TextField name='password' id="outlined-basic"
label="Password" variant="outlined" className="textfield" type='password'
onChange={this.handlechange}/>
                    </div>
                    <Button style={{marginTop:'20px',width:'300px'}}
variant="contained" color="primary" onClick={this.login}>
                        Login
```

```
                                </Button>
                            </div>

                        </div>
                    );
                }
            }
            export default Login;
```

Login.css

```
.login-
div{
              margin-left: 40%;
          }
          .textfield{
              width: 300px;
              margin-top: 10px;
          }
          input-div{
              margin-top: 1000px;
          }
```

Panel.js

```
import
React,
{Component}
from
'react';
              import {AppBar, Toolbar, Typography,LinearProgress,Button} from "@material-
              ui/core";
              import './Panel.css'
              class Panel extends Component {
                  constructor(props) {
                      super(props);
                      this.state={
                          sensors: []
                      }
                  }
              componentWillMount() {
              setInterval(()=> {
                  fetch('http://localhost:8080/getallsensors', {mode: "cors"}).then(res =>
              {
                      return res.json()
                  }).then(data => {
```

```
        console.log(data)
        this.setState({sensors: data})
    })
},5000);
}
    render() {
        return (
            <div>
                <AppBar position="static">
                    <Toolbar>
                        <Typography variant="h6" style={{marginLeft:'40%'}}>
                            Emergency Fire Alarm System
                        </Typography>
                        <Button variant={"contained"} color={"secondary"}
style={{marginLeft:'35%'}} onClick={()=>this.props.logout()}>log Out</Button>
                    </Toolbar>
                </AppBar>
                <div className='container' >
                    {
                        this.state.sensors.map(sensor=>{
                        return (<div className="sensor" key={sensor.id}>
                                <h4>SensorName:{" "+sensor.sensorName}
location:{"Floor: "+sensor.floorNo +"  "+ "Room: "+sensor.roomNo}</h4>
                                <h6>CO2 Volume:{" "+sensor.co2}</h6>
                                <LinearProgress variant="buffer"
value={sensor.co2*10} valueBuffer={100} color={sensor.co2>5? 'secondary':
'primary'}/>
                                <h6>Smoke volume:{" "+sensor.smoke}</h6>
                                <LinearProgress variant="buffer"
value={sensor.smoke*10} valueBuffer={100} color={sensor.smoke>5? 'secondary':
'primary'}/>
                            </div>)
                    })
                    }
                </div>
            </div>
        );
    }
}
export default Panel;
```

Panel.css

```
.container{
                    width: 800px;
                    margin-top: 5%;
                    margin-left: 20%;
                    height: 180px;
                }
                .sensor{
                    margin-bottom: 10px;
                    background: lightcyan;
                }
```

Reducers

index.js

```
import
{combineReducers}
from "redux";
                    import loogedReducer from'./loggedreducer'
                    const allreducers= combineReducers({
                        logged: loogedReducer
                    })
                    export default  allreducers;
```

loggedreducer.js

```
const
loogedReducer
=(state=false,
action)=>{
                     switch(action.type){
                         case 'SIGN_IN':
                             return (!state);
                         default:
                             return state;
                     }
                 }
                 export default loogedReducer;
```

App.js

```
Import
React
from
'react';
           import logo from './logo.svg';
           import './App.css';
           import {useDispatch, useSelector} from "react-redux";
           import signin from "./Actions/signin";
           import Login from "./Components/Login";
           import Panel from "./Components/Panel";
           import signout from "./Actions/signout";
           function App() {
             const log= useSelector(state=>state.logged)
             const dispatch= useDispatch();
             const login=()=>{
               dispatch(signin())
             }
           if(log===false){
             return <div>
               <Login login={login} />
             </div>
           }
           if(log===true){
             return <div>
               <Panel logout={login}/>
             </div>
           }
           }
           export default App;
```

index.js

```
import
React
from
'react';
           import ReactDOM from 'react-dom';
           import './index.css';
           import App from './App';
           import * as serviceWorker from './serviceWorker';
           import {createStore} from "redux";
           import allreducers from "./reducers";
           import {Provider} from "react-redux";
           const store = createStore(allreducers,window.__REDUX_DEVTOOLS_EXTENSION__ &&
           window.__REDUX_DEVTOOLS_EXTENSION__());
           ReactDOM.render(
             <React.StrictMode>
           <Provider store={store}>
               <App />
           </Provider>
             </React.StrictMode>,
             document.getElementById('root')
           );
           // If you want your app to work offline and load faster, you can change
           // unregister() to register() below. Note this comes with some pitfalls.
           // Learn more about service workers: https://bit.ly/CRA-PWA
           serviceWorker.unregister();
```