Sri Lanka Institute of Information Technology

Fire Alarm Monitoring System

Distributed System Assignment 2

2020

Submitted by:

IT18197624– Bamunuge H.K.T
IT18126648– Perera W.A.D.H.M
IT18126334– K.S.Shehari
IT17169004– P.A.P.Savindri

1.Introduction

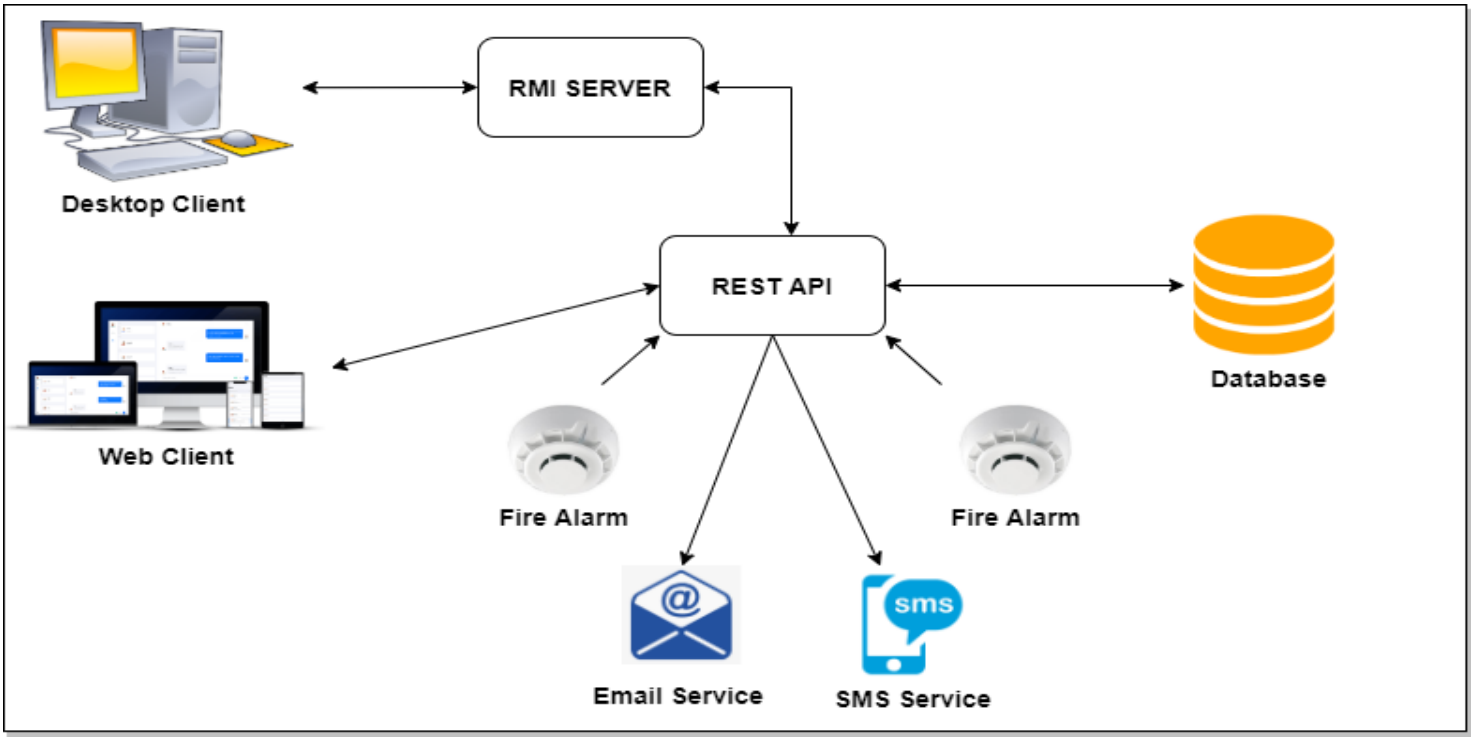
A fire alarm system has a number of devices working together to detect and warn people through visual and audio appliances when smoke, fire, carbon monoxide or other emergencies are present. This report contains the high-level system architectures of a fire alarm monitoring system which we have developed for the 3rd year Distributed Systems module. And it lists out the service interfaces exposed by each service and briefly explains each of the workflows used in the system.

· The system has a web client application where users can view the status of all fire alarm sensors. For each sensor, the web application displays whether the fire alarm sensor is active, the location, smoke level and the CO2 level. The data is sent from a dummy application which we have assumed as a building every 10 seconds. The data in the web client application is refreshed every 15 seconds. Anyone can check carbon dioxide or smoke level by checking the web application. If the smoke level or CO2 level is above 5, then they mark in red for convenience.

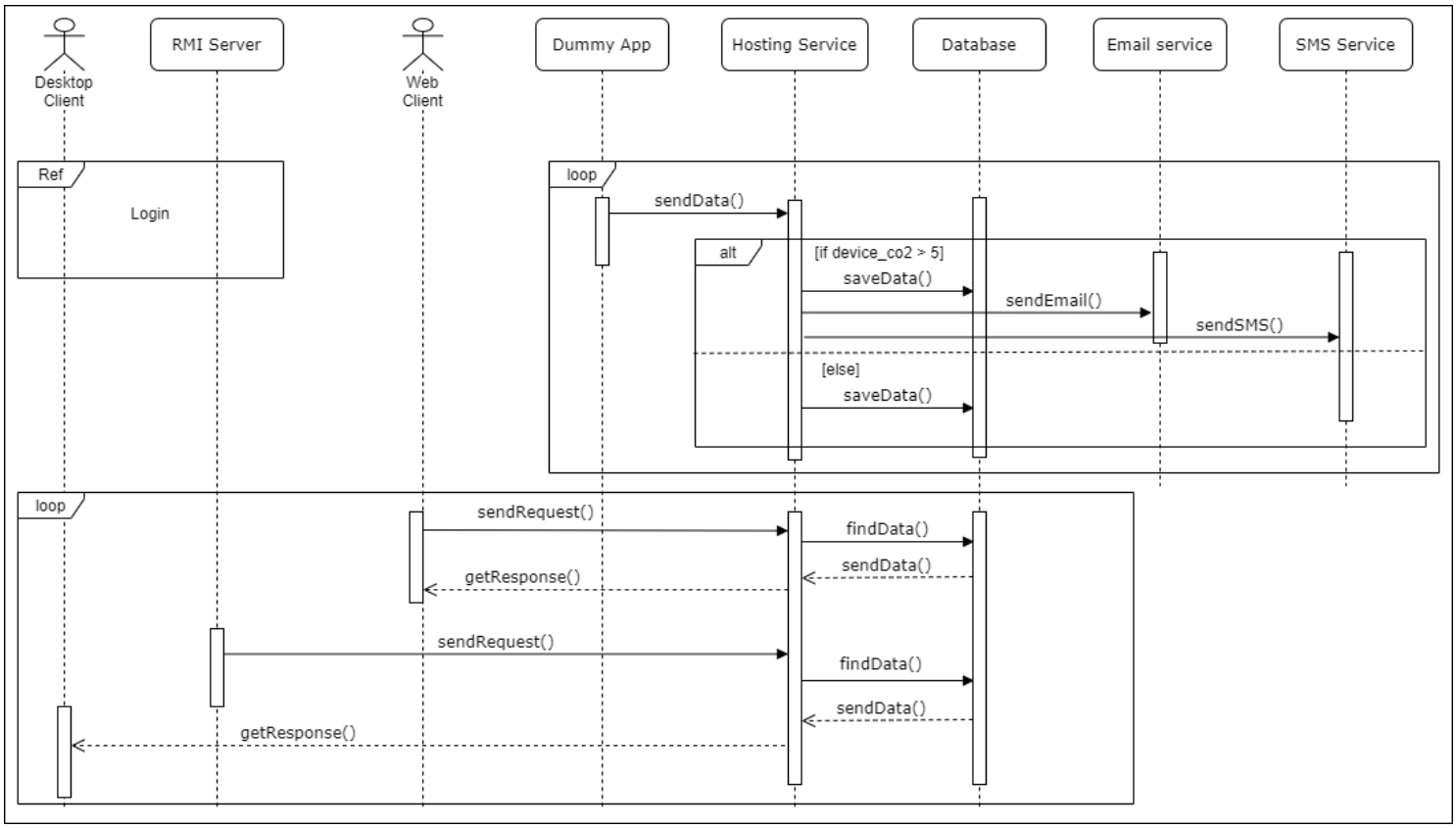
· The system contains a desktop client application where users can view the same information from a desktop client. The information is refreshed every 15 seconds. It has an administrator login, where an administrator can add/register new fire alarm sensors. To register, the floor number and the room no should be given. The administrator can edit sensor details as well as it has an administrator login, where an administrator can add/register new fire alarm sensors. To register, the floor number and the room no should be given. The administrator can edit sensor details as well.

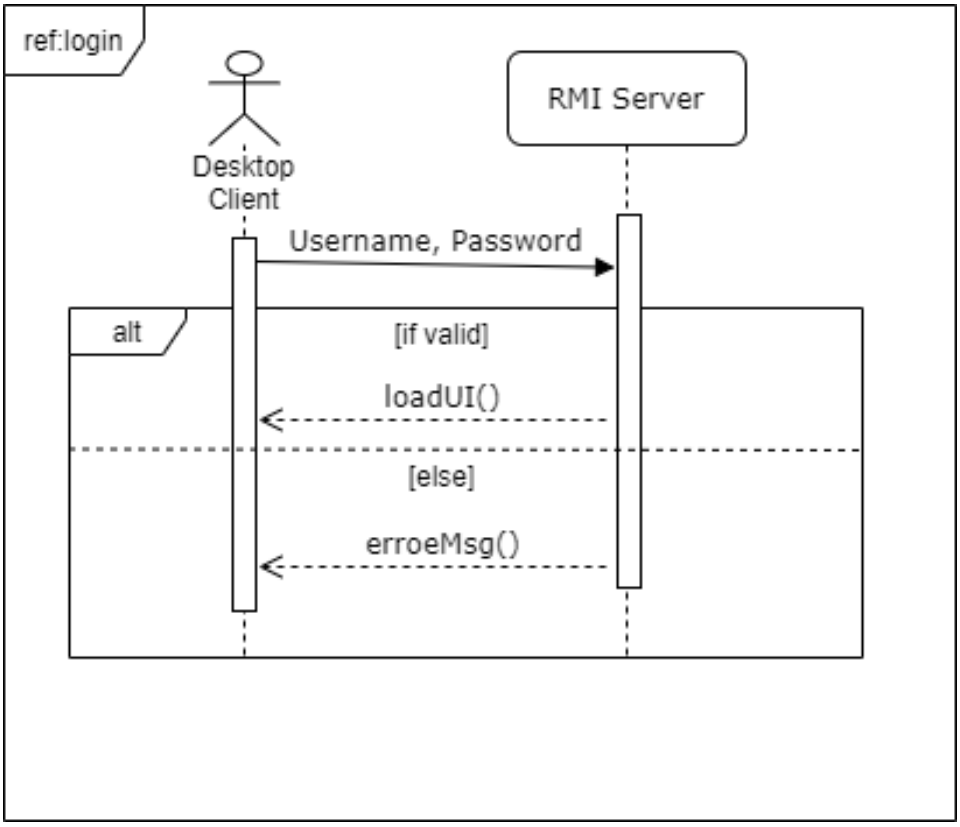
The system identifies a situation as a special fire situation if the carbon dioxide level is above 5. The main purpose of the monitoring system is to send alerts when a fire emergency has occurred. The system sends SMS and emails to the relevant parties when it identifies a fire situation.

2.System Architecture Diagram



3.Sequence Diagram





4.RESTful Web API

The RESTful web API as well as the MySQL database are hosted in a private host service: Baiscope Hosting Service. We have created the API by implementing the GET method, POST method and email sending methods separately. The code will supply below. According to the client's request, data is extracted in json format by GET method from searching the database. This data which is sent by the fire alarm is saved in a relevant table by POST method. URLs of GET (***appendix A***) and POST (***appendix B***) method can be found below.

- GET method - https://www.firealermmonitoring.baishost.com/status.php
- POST method - https://www.firealermmonitoring.baishost.com/sentStatues.php

5.Fire Alarms

We have created a dummy application to act as a fire alarm using Node-Red. We can add one or more fire alarms from that. Data of newly added alarms can be sent to the API. The carbon dioxide level and smoke level are checked from that data.(***appendix C***) If the carbon dioxide level and smoke level is 5 or more than 5, SMS alert and email alert is sent to the relevant parties by the system.(***appendix D***) If the smoke level or CO2 level is above 5 in a particular fire alarm data set, that data set is marked in red in both client applications for convenience. We can run one or more fire alarms at once. The data is sent from every fire alarm to the API in every 10 seconds. That data is sent to the database and saved there.

6.Web Client

The system has a web client application where users can view the status of all fire alarm sensors.(***appendix E***) For each sensor, the web application displays whether the fire alarm sensor is active, the location, smoke level and the CO2 level. The data in the web client application is refreshed every 15 seconds. (***appendix F***) Anyone can check carbon dioxide and smoke levels by checking the relevant page in the web application. If the smoke level or CO2 level is 5 or above 5, then they mark in red for convenience. Anyone can search details of a particular alarm by searching it.

6.Web Client

The system has a desktop client application where users can view the same information from a desktop client. (***appendix G***) The information is refreshed every 15 seconds. If the smoke level or CO2 level is 5 or above 5, then they mark in red for convenience. The client application has an administrator login, where an administrator can add/register new fire alarm sensors. (***appendix H***) To register, the floor number and the room no should be given. The administrator can edit sensor details as well as it has an administrator login, where an administrator can add/register new fire alarm sensors.(***appendix I***) To register, the floor number and the room no should be given. The administrator can edit sensor details as well.

appendix A

```
<?php
header("Access-Control-Allow-Origin: *");

   $servername =
"firealermmonitoring.baishost.com";
   $username = "firealer_firealer";
   $password = "icui4cuK@";
   $dbname = "firealer_monitoringdb";

   $conn = new mysqli($servername, $username,
$password, $dbname);
   // Check connection
   if ($conn->connect_error) {
      die("Connection failed: " . $conn-
>connect_error);
   }
   $sql = "SELECT  * FROM devices";

   $result = $conn->query($sql);

   $dataArray=array();

   if ($result->num_rows > 0) {

      while ($row = $result->fetch_assoc()) {

         array_push($dataArray,$row);

      }
   } else {
      echo "0 results";
   }

 echo json_encode($dataArray);
   $conn->close();
?>
```

appendix B

```
<?php
header("Access-Control-Allow-Origin: *");

$device_id =$_POST['device_id'];
$device_status =$_POST['device_status'];
$device_floor= $_POST['device_floor'];
$device_room= $_POST['device_room'];
$device_smoke= $_POST['device_smoke'];
$device_co2 = $_POST['device_co2'];

   $servername =
"firealermmonitoring.baishost.com";
   $username = "firealer_firealer";
   $password = "icui4cuK@";
   $dbname = "firealer_monitoringdb";

   // Create connection
   $conn = new mysqli($servername, $username,
$password, $dbname);
   // Check connection
   if ($conn->connect_error) {
      die("Connection failed: " . $conn-
>connect_error);
   }

   $sql =  "INSERT INTO devices (device_id,
device_status,
device_floor,device_room,device_smoke,device_c
o2)
VALUES ('$device_id', '$device_status',
'$device_floor','$device_room','$device_smoke','$
device_co2')";

if ($conn->query($sql) === TRUE) {
   echo "Fire Alarm Details Added Successfully";
} else {
   echo "Error: " . $sql . "<br>" . $conn->error;
}

   $conn->close();
?>
```

appendix C

```
        public void sendMYAlert()
        {

                ArrayList<AllData> list =
TempAdd_Grid();
                String alertMessage = "";

                for(int i = 0; i < list.size(); i++) {

                        int id = list.get(i).getId();


        if((list.get(i).getDevice_co2() >= 5) ||
(list.get(i).getDevice_smoke() >= 5)) {



        if(!initialList.stream().filter(o -> o.getId() ==
id).findFirst().isPresent()) {

        alertMessage = alertMessage + "FloorNo: " +
list.get(i).getDevice_floor() + "\nRoomNo: " +
list.get(i).getDevice_room() +

                "\nCO2Level: " +
list.get(i).getDevice_co2() + "\nSmokeLevel: " +
list.get(i).getDevice_smoke() + "\n\n";


        initialList.add(list.get(i));
                                        }
                        }
                        else {


        if(initialList.stream().filter(o -> o.getId() ==
id).findFirst().isPresent()) {

                                                int j=0;


        while(initialList.get(j).getId() != id) {

        j++;
                                                }

                initialList.remove(j);
                                        }
                        }

                }
        }
```

appendix D

```
<?php
// the message
$msg = "Hi, Please Note That the Co2 or Smoke level has
Increased. This May Cause an Emergency
Situation!".print_r($_POST);

// use wordwrap() if lines are longer than 70 characters
$msg = wordwrap($msg,70);

// send email
mail("kasunthilina1000@gmail.com","Fire Alert!",$msg);
echo "send".print_r($_POST);
?>
```

appendix E

```
import React from "react";
import { MDBDataTable } from "mdbreact";

class DatatablePage extends React.Component {
  state = {
    columns: [
      {
        field: 'id',
        label: '#'
      },
      {
        field: 'device_id',
        label: 'ID'
      },
      {
        field: 'device_status',
        label: 'Status'
      },
      {
        field: 'device_floor',
        label: 'Floor'
      },
      {
        field: 'device_room',
        label: 'Room'
      },
      {
        field: 'device_smoke',
        label: 'Smoke Level'
      },
      {
        field: 'device_co2',
        label: 'Co2 Level'
      },
    ],
    rows: []
  };

  componentDidMount() {
    this.getDetails();
    setInterval(this.getDetails, 5000);
  }

  getDetails = () => {


fetch("https://www.firealermmonitoring.baishost.
com/status.php", {
        method: "GET",
    })
      .then(res => res.json())
      .then(json => {
        let rows = [];
        json.forEach(item => rows.push({
          id: item.id,
          device_id: item.device_id,
          device_status: item.device_status,
          device_floor: item.device_floor,
          device_room: item.device_room,
          device_smoke: item.device_smoke,
          device_co2: item.device_co2,

        }));

        console.log(rows);

        this.setState({ rows });
      })
      .catch(err => console.error(err));
  };

  render() {
    return (
      <>
        <MDBDataTable
          style={{width:'98%', marginLeft:'1%'}}
          striped
          bordered
          hover
          data={{ columns: this.state.columns,
rows: this.state.rows }}
        />
      </>
    );
  }
}

export default DatatablePage;
```

appendix F

```
componentDidMount() {
    this.getDetails();
    setInterval(this.getDetails, 15000);
  }
```

appendix G

```
	public static ArrayList<AllData> TempAdd_Grid(){

		ArrayList<AllData> floorDetails = new ArrayList<AllData>();

		try {

			Server_remort =(Server_Remort_INF) Naming.lookup("rmi://localhost:1099/MyServer");
//access read rest api method in server
			String output = Server_remort.readTemp();

			JSONArray  jarr = new JSONArray(output);

			for (int i = 0; i < jarr.length(); i++) {

				AllData All_Flor_Details = new AllData();

				JSONObject obj = jarr.getJSONObject(i);


	All_Flor_Details.setId(obj.getInt("id"));

	All_Flor_Details.setDevice_id(obj.getInt("device_id"));

	All_Flor_Details.setDevice_status(obj.getString("device_status"));

	All_Flor_Details.setDevice_floor(obj.getInt("device_floor"));

	All_Flor_Details.setDevice_room(obj.getInt("device_room"));

	All_Flor_Details.setDevice_smoke(obj.getInt("device_smoke"));

	All_Flor_Details.setDevice_co2(obj.getInt("device_co2"));


	floorDetails.add(All_Flor_Details);
				}


		} catch (Exception e) {
			// TODO: handle exception
			e.printStackTrace();
		}

		return floorDetails;
	}
```

appendix H

```
	public void AddMore(String DeviceId,String roomNo, String floorNumber) {

		try {
			Server_remort =(Server_Remort_INF) Naming.lookup("rmi://localhost:1099/MyServer");
//access read rest api method in server



	Server_remort.addMYSensor(Integer.parseInt(DeviceId),Integer.parseInt(floorNumber), roomNo);


		} catch (MalformedURLException | RemoteException | NotBoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

	}

	public void sendInitialAlert(ArrayList<AllData> alertList) {

	System.getProperty("java.security.policy");
		String alertMessage = "";
		initialList = alertList;

		try {
			Server_remort =(Server_Remort_INF) Naming.lookup("rmi://localhost:1099/MyServer");

			if(initialList.size() > 0) {

				for(int i = 0; i < initialList.size(); i++) {

	alertMessage = alertMessage + "FloorNo: " + initialList.get(i).getDevice_floor()+ "\nRoomNo: " + initialList.get(i).getDevice_room() +

			"\nCO2Level: " + initialList.get(i).getDevice_co2() + "\nSmokeLevel: " + initialList.get(i).getDevice_smoke() + "\n\n";
				}


	Server_remort.sendMYAlert(alertMessage);
			}
		} catch (MalformedURLException | RemoteException | NotBoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
```

appendix I

```
public class Admin_Login extends JFrame {

	private JPanel contentPane;
	private JTextField textUN;
	private JPasswordField textPW;
	static Admin_Login frame = new Admin_Login();

	public static void main(String[] args) {

		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {


	frame.setVisible(true);


				} catch (Exception e) {

	e.printStackTrace();
				}
			}
		});
	}

	public Admin_Login() {

	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 618, 550);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);

		JLabel lblNewLabel = new JLabel("LOG IN TO FIRE ALARM DASH BOARD");
		lblNewLabel.setFont(new Font("Tahoma", Font.BOLD, 24));

	lblNewLabel.setForeground(Color.DARK_GRAY);
		lblNewLabel.setBounds(58, 33, 504, 53);
		contentPane.add(lblNewLabel);

		JLabel lblNewLabel_1 = new JLabel("USER NAME");
		lblNewLabel_1.setFont(new Font("Tahoma", Font.BOLD, 18));
		lblNewLabel_1.setBounds(123, 175, 114, 20);
		contentPane.add(lblNewLabel_1);

		JLabel lblNewLabel_2 = new JLabel("PASSWORD");
		lblNewLabel_2.setFont(new Font("Tahoma", Font.BOLD, 18));
		lblNewLabel_2.setBounds(123, 275, 114, 20);
		contentPane.add(lblNewLabel_2);

		textUN = new JTextField();
		textUN.setBounds(284, 175, 170, 26);
		contentPane.add(textUN);
		textUN.setColumns(10);

		textPW = new JPasswordField();
		textPW.setBounds(284, 275, 170, 26);
		contentPane.add(textPW);


		JButton loginBTN = new JButton("LOG IN");
		loginBTN.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {

				String UserName = textUN.getText();
				String password = textPW.getText();


	if(UserName.contains("Admin")  && password.contains("admin@123"))
				{

	Client_TempDashBord dashbord = new Client_TempDashBord();

	dashbord.New_UI_Main();

	frame.setVisible(false);


				}
				else
				{
					JFrame f = new JFrame();

	JOptionPane.showMessageDialog(f,"SORRY INVALID");

				}
			}
		});

	loginBTN.setForeground(SystemColor.menu);

	loginBTN.setBackground(SystemColor.textHighlight);
		loginBTN.setFont(new Font("Tahoma", Font.BOLD, 18));
```

```
                    loginBTN.setBounds(123, 374, 344,
40);
                    contentPane.add(loginBTN);

                    JButton resetBTN = new
JButton("RESET");

          resetBTN.setForeground(SystemColor.menu);
                    resetBTN.addActionListener(new
ActionListener() {
                              public void
actionPerformed(ActionEvent arg0) {

          textUN.setText(null);

          textPW.setText(null);
                              }
                    });
                    resetBTN.setFont(new
Font("Tahoma", Font.BOLD, 18));

          resetBTN.setBackground(SystemColor.textHig
hlight);
                    resetBTN.setBounds(123, 443, 344,
40);
                    contentPane.add(resetBTN);
          }
}
```

Thank You !