Sri Lanka Institute of Information Technology

Fire Alarm System

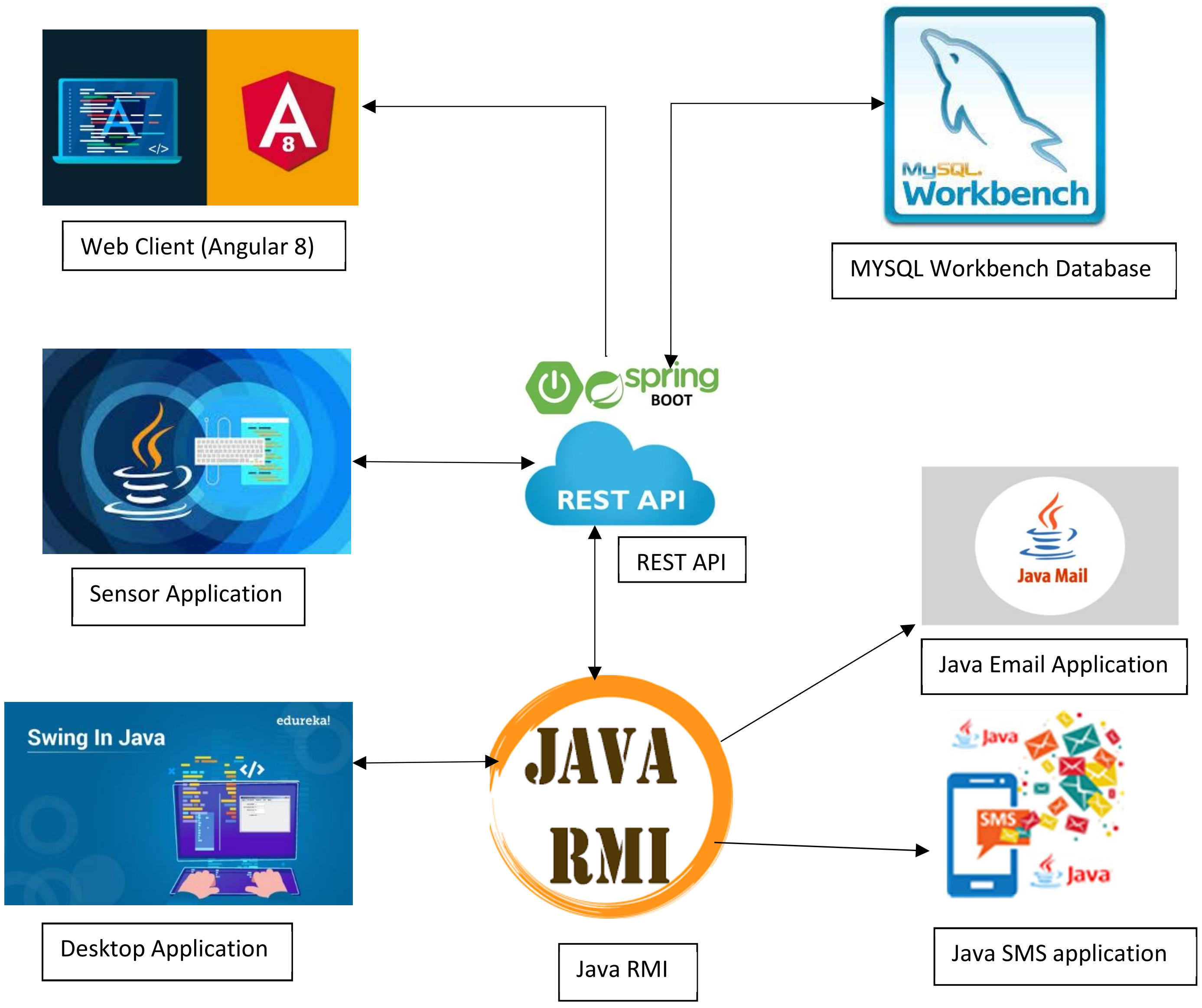
Assignment 2 Report

Distributed Systems Assignment 2
Y3S1.20(WD)

Submitted by:

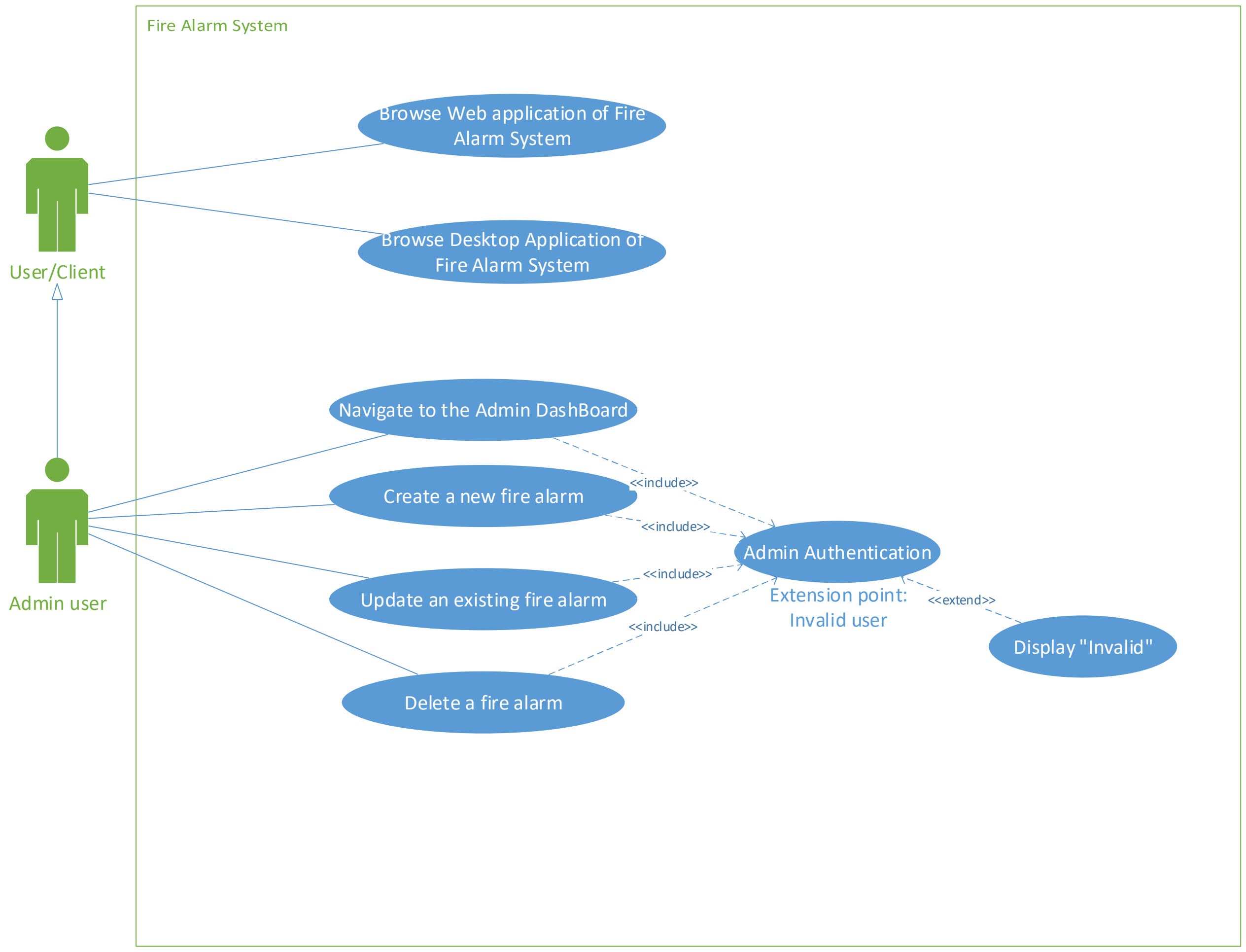
1. IT18110630– (Hewagamage P.L)
2. IT18110944– (Fernando P.D.R)
3. IT18107388– (Perera D.F.Y.D)

System Architecture Diagram



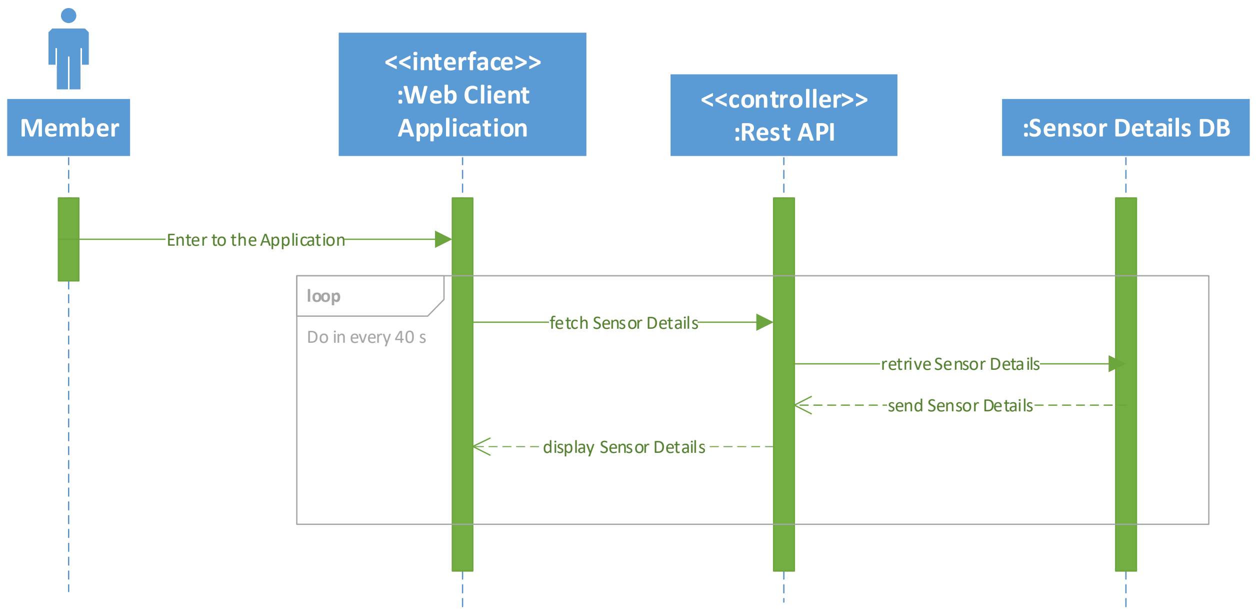
We implement the Desktop application using the "Java Swing". On the other hand web client application was developed using "Angular". We prepared the REST API using "Java Spring Boot Framework". "MYSQL Workbench" was the DBMS we used in our system. RMI server, Sensor application was built based on normal java fundamentals. We used java mail API to send email notification and com.twilio.sdk to sms alert notification in order to simulate those third party services.

Use Case Diagram for Fire Alarm system


Fire Alarm System
User/Client
Admin user
Browse Web application of Fire Alarm System
Browse Desktop Application of Fire Alarm System
Navigate to the Admin DashBoard
Create a new fire alarm
Update an existing fire alarm
Delete a fire alarm
<<include>>
<<include>>
<<include>>
<<include>>
Admin Authentication
Extension point: Invalid user
<<extend>>
Display "Invalid"

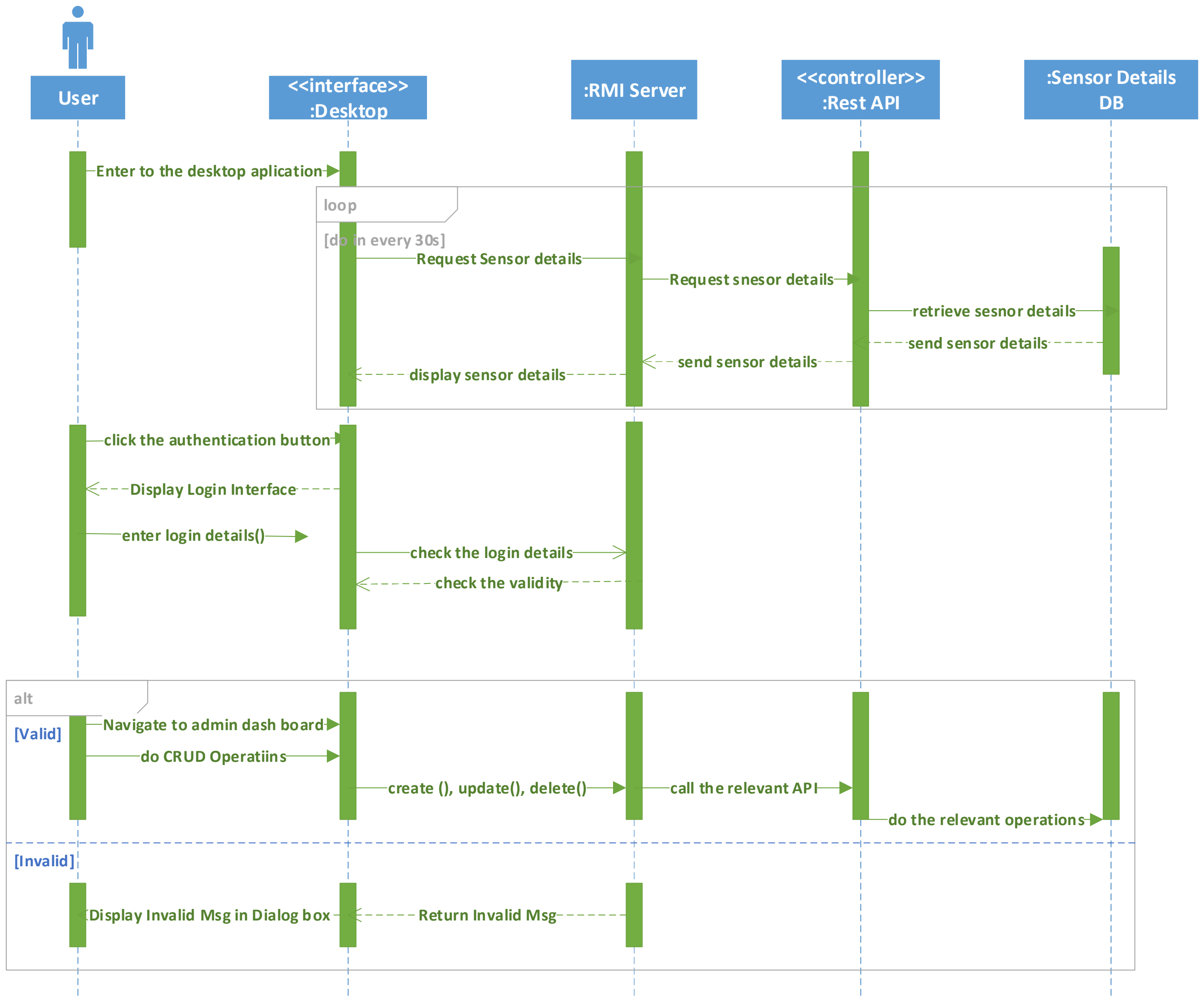
Services of the Fire Alarm system

1. Web Application of the Fire Alarm Sensor



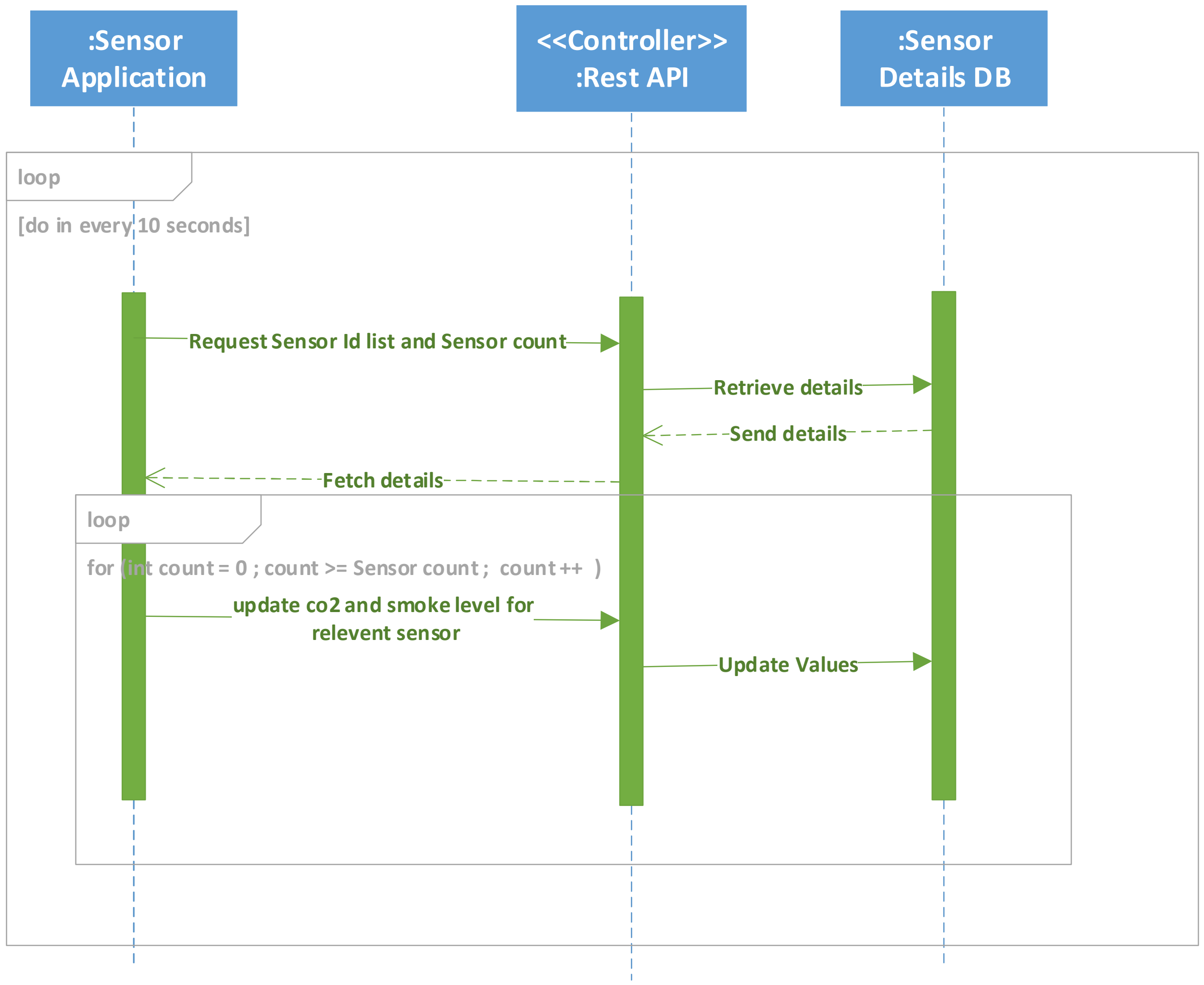
Any client can access the web application without entering any type of user credentials. In the designed web page, it displays Sensor ID, Floor number, Room Number, current co2 level, current smoke level and the status of the web alarm in a table format. If the Co2 level or Smoke level is greater than 5 then the status will be active. Active status will be indicted by a red bulb, while inactive status will be shown as green bulb. This web application is refreshing in every 40 seconds.

2. Desktop Application of the Fire Alarm Sensor



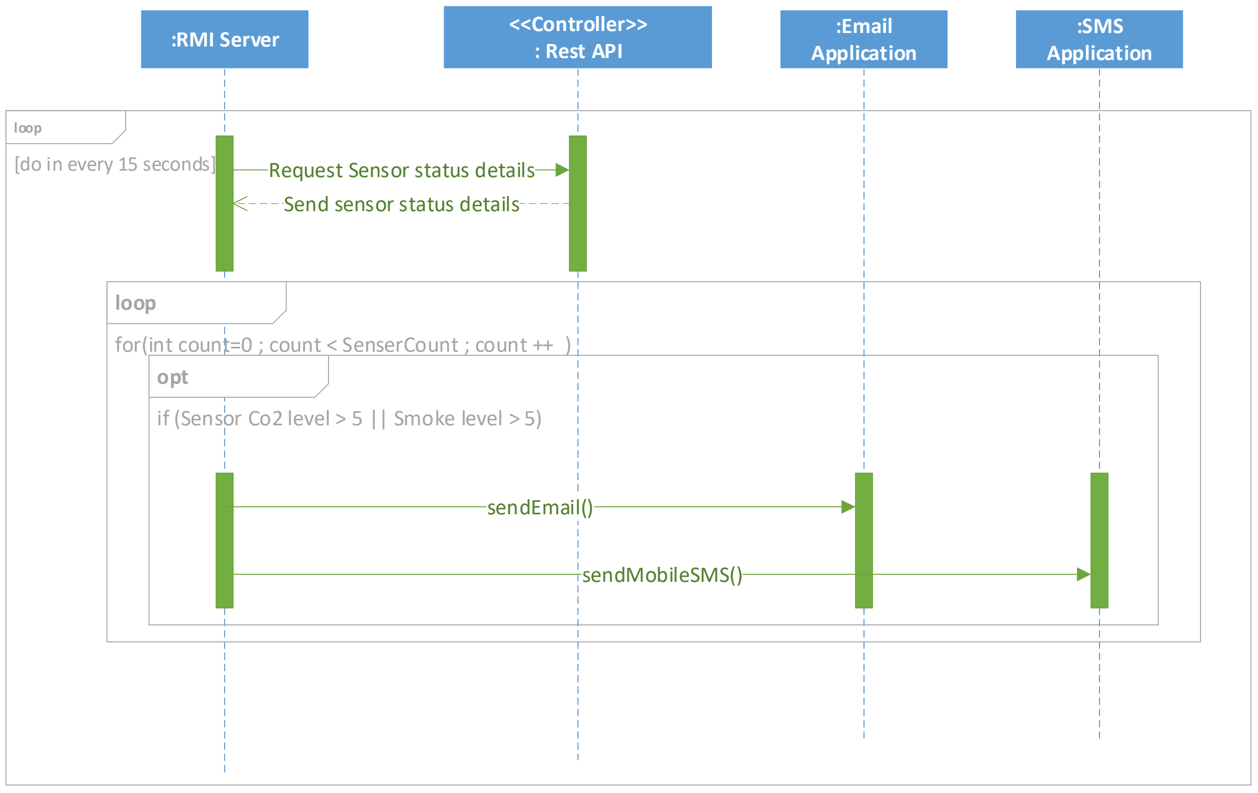
Any client can access the desktop application as well. Desktop application is also displays the same information which are displayed in the web client application in a tabular format. In the desktop application there is an admin login button right upper corner. If we clicked that button, desktop application displays login interface form admin. If the user enter the valid login credentials, user can navigate in to the admin dashboard. But if the user credentials are invalid, then system displays error message in a dialogue box. Well, in the admin dashboard can create a new alarm, update an existing alarm and delete an alarm. This desktop application is refreshing in every 30 seconds.

3. Sensor Application



In the sensor application update the co2 level and smoke level of each and every alarm in every 10 seconds. In this application there is a bit complex process. This application get all the sensor id list and sensor count in the database table through the RESTAPI. Then Iteration process will be occurred in order to change the co2 level and the smoke level of a particular sensor. For example, if the database table consist of the 5 alarms there will be 5 different iterations. One iteration simulates a one sensor. Again this application fetch the list of ids and sensor count from the database table and running an iteration. This process will be continues with interval of 10 seconds period. We just need to run this application only once in order to start this process. Do not need to re-run this application again and again.

4. Email and Mobile SMS application Services


:RMI Server
<<Controller>> : Rest API
:Email Application
:SMS Application
loop
[do in every 15 seconds]
Request Sensor status details
Send sensor status details
loop
for(int count=0 ; count < SenserCount ; count ++)
opt
if (Sensor Co2 level > 5 || Smoke level > 5)
sendEmail()
sendMobileSMS()

RMI retrieve alarm status in every 15 seconds from the REST API using. So this list run in a loop, while checking the status. If the status active (Co2 level > 5 || Smoke level > 5) then the RMI server will send the notification as an email and a sms alert by calling the relevant methods.

Security Measures

We used a simple authentication system in desktop client application in order to build the Admin login. We stored username and the password in a tree map as a pair (Key, Value). In this project,

- User name :- admin
- Password :- admin

Appendix

• RMI and Desktop Application

Server.java

```
public class Server extends UnicastRemoteObject implements LoginFacade, AlarmFacade{

        // properties
        private TreeMap clients = new TreeMap<String, String>();
        Client client = new Client().create();
        SMS sms = new SMS();



        public Server() throws RemoteException {

        }

        @Override
        public String login(String username, String password) throws RemoteException {
                init();
                String response = search(username, password);

                return response;
        }

        // find client
        private String search(String username, String password) {
                String response = "";

                Set set = clients.entrySet();
                Iterator itr = set.iterator();
                boolean flag = false;

                while(itr.hasNext()) {
                        response = "";
                        Map.Entry entry = (Map.Entry) itr.next();
                        String user = entry.getKey().toString();
                        String pass = entry.getValue().toString();

                        if (username.equals(user)) {
                                flag = true;
```

```
                    if (password.equals(pass)) {
                        response = "LOGIN_SUCCESFUL";
                    } else {
                        response = "PASSWORD_INVALID";
                    }
                    break;
                }
            }

            if (!flag) {
                response = "USERNAME_INVALID";
            }

            return response;
        }

        private void init() {
            clients.put("admin", "admin");
        }

        @Override
        public void setSensor(int floor, int room) throws RemoteException {
            try {
                WebResource webResource =
client.resource("http://localhost:8080/location/save");
                JSONObject input = new JSONObject();
                input.put("floor_no", floor);
                input.put("room_no", room);
                ClientResponse response =
webResource.type("application/json").post(ClientResponse.class, input.toString());

                if (response.getStatus() != 200) {
                    throw new RuntimeException("Failed : HTTP error code : " +
response.getStatus());
                }

                System.out.println(response.getStatus());
            } catch (Exception e) {
                e.printStackTrace();
            }

        }

        @Override
        public void updateSensor(int id,int floor, int room) throws RemoteException {
            try {
                WebResource webResource =
client.resource("http://localhost:8080/location/update");
                JSONObject input = new JSONObject();
```

```
				input.put("id", id);
				input.put("floor_no", floor);
				input.put("room_no", room);
				input.put("co2", 1);
				input.put("smokeLvl", 1);

				ClientResponse response =
webResource.type("application/json").put(ClientResponse.class, input.toString());

				if (response.getStatus() != 200) {
					throw new RuntimeException("Failed : HTTP error code : " +
response.getStatus());
				}

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

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

		}


		@Override
		public String getLocation() throws RemoteException {
			try {
				WebResource webResource = client.resource("http://localhost:8080/location");
				ClientResponse response =
webResource.accept("application/json").get(ClientResponse.class);

				if (response.getStatus() != 200) {
					throw new RuntimeException("Failed: HTTP error code: " +
response.getStatus());
				}

				return response.getEntity(String.class);

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

			return "";
		}

		public static void main(String[] args) {
			Registry reg;
			try {
				reg = LocateRegistry.createRegistry(Registry.REGISTRY_PORT);
				Server obj = new Server();
				reg.rebind("rmi://localhost/service", obj);
				System.out.println("Server is running...");



				Timer timeChange = new Timer();

				timeChange.scheduleAtFixedRate(
				  new TimerTask()
				  {
				    public void run()
				    {
				      try {
								obj.sendNotification();
							} catch (RemoteException e) {
								// TODO Auto-generated catch block
								e.printStackTrace();
							}
				    }
				  },
				  0,
				  15000);
			} catch (RemoteException e) {
				System.out.println(e.getMessage());
			}

		}

		@Override
		public void deleteSensor(int id) throws RemoteException {
			// TODO Auto-generated method stub

			try {

			WebResource webResource =
client.resource("http://localhost:8080/location/delete/"+id);
			JSONObject input = new JSONObject();


			ClientResponse response =
webResource.type("application/json").delete(ClientResponse.class, input.toString());

			if (response.getStatus() != 200) {
				throw new RuntimeException("Failed : HTTP error code : " +
response.getStatus());
			}
			System.out.println(response.getStatus());

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

		}

		@Override
		public void sendNotification() throws RemoteException {
			try {
				WebResource webResource = client.resource("http://localhost:8080/location");
				ClientResponse response =
webResource.accept("application/json").get(ClientResponse.class);

				if (response.getStatus() != 200) {
					throw new RuntimeException("Failed: HTTP error code: " +
response.getStatus());
				}


				Gson gson = new Gson();


				Location[] location = gson.fromJson(response.getEntity(String.class),
Location[].class);

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

					if("active".equals(location[i].getStatus())) {

						System.out.println("Alert Notification");
						sms.sendSMS();
						Email.sendMail();

					}else {

				   }



				}

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

```
	}


}
```

LoginFacade.java (Interface which consist methods relavent to the Login)

```
public interface LoginFacade extends Remote {
	public String login(String username, String password) throws RemoteException;
}
```

AlarmFaced.java (Interface which consist methods relavent to the Alarm)

```
public interface AlarmFacade extends Remote {
	public void setSensor(int floor, int room) throws RemoteException;
	public void updateSensor(int id,int floor, int room) throws RemoteException;
	public void deleteSensor(int id)throws RemoteException;
	public String getLocation() throws RemoteException;
	public void sendNotification() throws RemoteException;
}
```

Index.java (Desktop client appliction)

```
public class Index {

	private JFrame frame;
	private JTable table;
	private JPanel contentPane;
	private SMS sms;



	/**
	 * Launch the application.
	 */
	public static void main(String[] args) {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Index window1 = new Index();
					window1.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the application.
	 */
	public Index() {

		initialize();

		Timer timeChange = new Timer();

		timeChange.scheduleAtFixedRate(
		   new TimerTask()
		   {
		      public void run()
		      {
		         TableChange();
		      }
		   },
		   0,
		   30000);
	}

	/**
	 * Initialize the contents of the frame.
	 */
	private void initialize() {

			frame = new JFrame();
			frame.setBounds(100, 100, 585, 476);
			frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
			frame.getContentPane().setLayout(null);

			JButton btnSignIn = new JButton("");
			btnSignIn.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent arg0) {
					Login lg = new Login();
					lg.newLogin();
				}
				});
				btnSignIn.setBackground(SystemColor.menu);
				btnSignIn.setIcon(new
ImageIcon(Index.class.getResource("/com/alarm/rmi/icons8-login-64.png")));
				btnSignIn.setBounds(468, 13, 75, 73);
				frame.getContentPane().add(btnSignIn);

				JTextPane txtpnid = new JTextPane();
				txtpnid.setText("#ID");
				txtpnid.setBounds(49, 166, 75, 20);
				frame.getContentPane().add(txtpnid);

				JTextPane txtpnFloorNo = new JTextPane();
				txtpnFloorNo.setText("Floor No");
				txtpnFloorNo.setBounds(125, 166, 75, 20);
				frame.getContentPane().add(txtpnFloorNo);

				JTextPane txtpnRoomNo = new JTextPane();
				txtpnRoomNo.setText("Room No");
				txtpnRoomNo.setBounds(202, 166, 75, 20);
				frame.getContentPane().add(txtpnRoomNo);

				JTextPane txtpnCoLevel = new JTextPane();
				txtpnCoLevel.setText("Co2 level");
				txtpnCoLevel.setBounds(280, 166, 75, 20);
				frame.getContentPane().add(txtpnCoLevel);

				JTextPane txtpnSmokeLevel = new JTextPane();
				txtpnSmokeLevel.setText("Smoke level");
				txtpnSmokeLevel.setBounds(358, 166, 75, 20);
				frame.getContentPane().add(txtpnSmokeLevel);

				JTextPane txtpnStatus = new JTextPane();
				txtpnStatus.setText("Status");
				txtpnStatus.setBounds(435, 166, 75, 20);
				frame.getContentPane().add(txtpnStatus);



			}

		public void TableChange() {
			table = new JTable();
			table.setBounds(49, 190, 459, 191);
			frame.getContentPane().add(table);



			String[] columnNames = {"#", "Floor No", "Room No", "CO2", "SmokeLvl","status"};
		DefaultTableModel model = new DefaultTableModel(columnNames, 0);
		try {
			Registry reg = LocateRegistry.getRegistry("localhost", 1099);
			AlarmFacade server = (AlarmFacade) reg.lookup("rmi://localhost/service");
			Gson gson = new Gson();

			Location[] location = gson.fromJson(server.getLocation(), Location[].class);



			for(int i=0; i < location.length; i++) {
				Vector<String> row = new Vector<String>();
				row.add(Integer.toString(location[i].getId()));
				row.add(Integer.toString(location[i].getFloor_no()));
				row.add(Integer.toString(location[i].getRoom_no()));
				row.add(Integer.toString(location[i].getCo2()));
				row.add(Integer.toString(location[i].getSmokeLvl()));
				row.add(location[i].getStatus());

				model.addRow(row);


			}


			table.setModel(model);

		} catch (RemoteException | NotBoundException  e) {
			e.printStackTrace();
		}
	}
}
```

Login.java (Desktop Admin Login GUI)

```
public class Login extends JFrame {

	private JPanel contentPane;
	private JTextField textUsername;
	private JPasswordField passwordField;
	private final JPanel panel_1 = new JPanel();

	/**
	 * Launch the application.
	 */
	public static void newLogin() {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Login frame = new Login();
					frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the frame.
	 */
	public Login() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 626, 406);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);

		JLabel lblUsername = new JLabel("Username: ");
		lblUsername.setForeground(Color.WHITE);
		lblUsername.setFont(new Font("Tahoma", Font.PLAIN, 16));
		lblUsername.setBounds(37, 110, 84, 16);
		contentPane.add(lblUsername);

		textUsername = new JTextField();
		textUsername.setBounds(158, 108, 189, 22);
		contentPane.add(textUsername);
		textUsername.setColumns(10);

		JLabel lblPassword = new JLabel("Password: ");
		lblPassword.setForeground(Color.WHITE);
		lblPassword.setFont(new Font("Tahoma", Font.PLAIN, 16));
		lblPassword.setBounds(37, 185, 84, 16);
		contentPane.add(lblPassword);

		JButton btnLogin = new JButton("LOGIN");
		btnLogin.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {
				try {
					Registry reg = LocateRegistry.getRegistry("localhost", 1099);
					LoginFacade server = (LoginFacade)
reg.lookup("rmi://localhost/service");

					// get user input
```

```
                                        String username = textUsername.getText();
                                        String password = passwordField.getText();

                                        String response = server.login(username, password);

                                        if(response.equals("USERNAME_INVALID")) {

                                             JOptionPane.showMessageDialog(null, response);
                                        }
                                        else {
                                             if (response.equals("LOGIN_SUCCESFUL")) {
                                                  Dashboard ds = new Dashboard();
                                                  ds.newWindow();
                                             }
                                             else {
                                                  JOptionPane.showMessageDialog(null,
response);

                                             }
                                        }

                                   } catch (RemoteException | NotBoundException e) {
                                        System.out.println(e.getMessage());
                                   }
                              }
                         });
                         btnLogin.setFont(new Font("Yu Gothic UI Semibold", Font.PLAIN, 13));
                         btnLogin.setForeground(new Color(0, 0, 0));
                         btnLogin.setBackground(new Color(50, 205, 50));
                         btnLogin.setBounds(213, 263, 97, 25);
                         contentPane.add(btnLogin);

                         JPanel panel = new JPanel();
                         panel.setBackground(new Color(0, 204, 255));
                         panel.setBounds(-3, -3, 613, 57);
                         contentPane.add(panel);

                         passwordField = new JPasswordField();
                         passwordField.setBounds(158, 185, 189, 20);
                         contentPane.add(passwordField);
                         panel_1.setBackground(new Color(0, 0, 128));
                         panel_1.setBounds(0, 53, 610, 314);
                         contentPane.add(panel_1);
                    }
          }
```

DashBoard.java (Desktop admin DashBoard)

```
public class Dashboard {

	private JFrame frame;
	private JTextField textFloorNo;
	private JTextField textRoomNo;
	private JButton btnNewLocation;
	private JTable table;
	private JButton btnNewButton_1;
	private int id = 0;

	/**
	 * Launch the application.
	 */
	public static void newWindow() {
		EventQueue.invokeLater(new Runnable() {
			public void run() {
				try {
					Dashboard window = new Dashboard();
					window.frame.setVisible(true);
				} catch (Exception e) {
					e.printStackTrace();
				}
			}
		});
	}

	/**
	 * Create the application.
	 */
	public Dashboard() {
		initialize();
		Timer t = new Timer();

		t.scheduleAtFixedRate(
		    new TimerTask()
		    {
		        public void run()
		        {
		            tableChange();
		        }
		    },
		    0,
                    30000);

        }

        /**
         * Initialize the contents of the frame.
         */
        private void initialize() {
                frame = new JFrame();
                frame.getContentPane().setForeground(new Color(255, 255, 255));
                frame.setBounds(100, 100, 598, 429);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.getContentPane().setLayout(null);

                JLabel lblFloorNumber = new JLabel("Floor number: ");
                lblFloorNumber.setFont(new Font("Tw Cen MT Condensed Extra Bold", Font.PLAIN, 15));
                lblFloorNumber.setBounds(32, 28, 86, 23);
                frame.getContentPane().add(lblFloorNumber);

                textFloorNo = new JTextField();
                textFloorNo.setBounds(158, 28, 189, 22);
                frame.getContentPane().add(textFloorNo);
                textFloorNo.setColumns(10);

                JLabel lblNewLabel = new JLabel("Room number: ");
                lblNewLabel.setFont(new Font("Tw Cen MT Condensed Extra Bold", Font.PLAIN, 15));
                lblNewLabel.setBounds(32, 89, 86, 16);
                frame.getContentPane().add(lblNewLabel);

                textRoomNo = new JTextField();
                textRoomNo.setBounds(158, 86, 189, 22);
                frame.getContentPane().add(textRoomNo);
                textRoomNo.setColumns(10);

                btnNewLocation = new JButton("+");
                btnNewLocation.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                try {
                                        Registry reg = LocateRegistry.getRegistry("localhost", 1099);
                                        AlarmFacade server = (AlarmFacade)
reg.lookup("rmi://localhost/service");

                                        // get use input
                                        int floor_no = Integer.parseInt(textFloorNo.getText());
                                        int room_no = Integer.parseInt(textRoomNo.getText());

                                        server.setSensor(floor_no, room_no);
                                        JOptionPane.showMessageDialog(null, "Successfully Inserted
\n(Following list will be changed in every 30 seconds)");
```

```
					} catch (RemoteException | NotBoundException e) {
						e.printStackTrace();
					}
				}
			});
			btnNewLocation.setBackground(new Color(0, 128, 0));
			btnNewLocation.setBounds(357, 86, 41, 25);
			frame.getContentPane().add(btnNewLocation);



			JButton btnNewButton = new JButton("UPDATE");
			btnNewButton.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent arg0) {
					Registry reg;
					try {
						reg = LocateRegistry.getRegistry("localhost", 1099);
						AlarmFacade server = (AlarmFacade)
reg.lookup("rmi://localhost/service");


						// get use input
						int floor_no = Integer.parseInt(textFloorNo.getText());
						int room_no = Integer.parseInt(textRoomNo.getText());

						System.out.println(id);
						System.out.println(floor_no);

						System.out.println(room_no);

						server.updateSensor(id, floor_no, room_no);

						JOptionPane.showMessageDialog(null, "Successfully Updated
\n(Following list will be changed in every 30 seconds)");



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



				}
```

```
		});
		btnNewButton.setBackground(Color.ORANGE);
		btnNewButton.setBounds(408, 87, 80, 23);
		frame.getContentPane().add(btnNewButton);

		btnNewButton_1 = new JButton("DELETE");
		btnNewButton_1.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {

				Registry reg;
				try {
					reg = LocateRegistry.getRegistry("localhost", 1099);
					AlarmFacade server = (AlarmFacade)
reg.lookup("rmi://localhost/service");


					server.deleteSensor(id);
					JOptionPane.showMessageDialog(null, "Successfully Deleted
\n(Following list will be changed in every 30 seconds)");

				} catch (RemoteException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				} catch (NotBoundException e1) {
					// TODO Auto-generated catch block
					e1.printStackTrace();
				}



			}
		});
		btnNewButton_1.setBackground(Color.RED);
		btnNewButton_1.setBounds(493, 87, 79, 23);
		frame.getContentPane().add(btnNewButton_1);

		JTextPane txtpnid = new JTextPane();
		txtpnid.setText("#ID");
		txtpnid.setBounds(57, 155, 95, 20);
		frame.getContentPane().add(txtpnid);

		JTextPane txtpnFloorNo = new JTextPane();
		txtpnFloorNo.setText("Floor No");
		txtpnFloorNo.setBounds(152, 155, 95, 20);
		frame.getContentPane().add(txtpnFloorNo);

		JTextPane txtpnRoomNo = new JTextPane();
		txtpnRoomNo.setText("Room No");
		txtpnRoomNo.setBounds(246, 155, 95, 20);
		frame.getContentPane().add(txtpnRoomNo);

		JTextPane txtpnCoLevel = new JTextPane();
		txtpnCoLevel.setText("Co2 level");
		txtpnCoLevel.setBounds(339, 155, 95, 20);
		frame.getContentPane().add(txtpnCoLevel);

		JTextPane txtpnSmokeLevel = new JTextPane();
		txtpnSmokeLevel.setText("Smoke Level");
		txtpnSmokeLevel.setBounds(434, 155, 80, 20);
		frame.getContentPane().add(txtpnSmokeLevel);





	}

	public void tableChange() {
		String[] columnNames = {"#", "Floor No", "Room No", "CO2", "SmokeLvl"};
		DefaultTableModel model = new DefaultTableModel(columnNames,0);
		table = new JTable(model);
		JTableHeader header = table.getTableHeader();
		header.setBackground(Color.cyan);
		table.setBounds(58, 177, 457, 156);
		model.setColumnIdentifiers(columnNames);
		table.setModel(model);
//		JScrollPane pane = new JScrollPane(table);


		frame.getContentPane().add(table);

		try {
			Registry reg = LocateRegistry.getRegistry("localhost", 1099);
			AlarmFacade server = (AlarmFacade) reg.lookup("rmi://localhost/service");
			Gson gson = new Gson();



			Location[] location = gson.fromJson(server.getLocation(), Location[].class);

			for(int i=0; i < location.length; i++) {
				Vector<Integer> row = new Vector<Integer>();
				row.add(location[i].getId());
				row.add(location[i].getFloor_no());
				row.add(location[i].getRoom_no());
				row.add(location[i].getCo2());
				row.add(location[i].getSmokeLvl());
				model.addRow(row);
				}

				table.setModel(model);

				JButton btnNewButton = new JButton("UPDATE");
				btnNewButton.addActionListener(new ActionListener() {
					public void actionPerformed(ActionEvent arg0) {
						Registry reg;
						try {
							reg = LocateRegistry.getRegistry("localhost", 1099);
							AlarmFacade server = (AlarmFacade)
reg.lookup("rmi://localhost/service");



							// get use input
							int floor_no =
Integer.parseInt(textFloorNo.getText());
							int room_no =
Integer.parseInt(textRoomNo.getText());

							System.out.println(id);
							System.out.println(floor_no);

							System.out.println(room_no);

							server.updateSensor(id, floor_no, room_no);

							JOptionPane.showMessageDialog(null, "Successfully
Updated \n(Following list will be changed in every 30 seconds)");



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



					}
				});
				btnNewButton.setBackground(Color.ORANGE);
				btnNewButton.setBounds(408, 87, 80, 23);
				frame.getContentPane().add(btnNewButton);

				btnNewButton_1 = new JButton("DELETE");
				btnNewButton_1.addActionListener(new ActionListener() {
					public void actionPerformed(ActionEvent e) {

						Registry reg;
						try {
							reg = LocateRegistry.getRegistry("localhost", 1099);
							AlarmFacade server = (AlarmFacade) 
reg.lookup("rmi://localhost/service");


							server.deleteSensor(id);
							JOptionPane.showMessageDialog(null, "Successfully 
Deleted \n(Following list will be changed in every 30 seconds)");

						} catch (RemoteException e1) {
							// TODO Auto-generated catch block
							e1.printStackTrace();
						} catch (NotBoundException e1) {
							// TODO Auto-generated catch block
							e1.printStackTrace();
						}



					}
				});
				btnNewButton_1.setBackground(Color.RED);
				btnNewButton_1.setBounds(493, 87, 79, 23);
				frame.getContentPane().add(btnNewButton_1);

				JPanel panel = new JPanel();
				panel.setBackground(Color.GRAY);
				panel.setBounds(0, 0, 582, 130);
				frame.getContentPane().add(panel);

			} catch (RemoteException | NotBoundException  e) {
				e.printStackTrace();
			}
			ListSelectionModel listSelectionModel = table.getSelectionModel();
			listSelectionModel.addListSelectionListener(new ListSelectionListener() {

				@Override
				public void valueChanged(ListSelectionEvent e) {
					// TODO Auto-generated method stub
					if (! listSelectionModel.isSelectionEmpty()) {
						int selectedRow = listSelectionModel.getMinSelectionIndex();
						textFloorNo.setText(model.getValueAt(selectedRow, 
1).toString());
```

```
                                        textRoomNo.setText(model.getValueAt(selectedRow,
2).toString());
                                        id = (int) model.getValueAt(selectedRow, 0);
                                }
                        }
                });
        }
}
```

• Web Client Application

body.component.html

```
<div class="container">
    <table class="table table-borderless">
        <thead>
          <tr>
            <th scope="col">#</th>
            <th scope="col">Floor Number</th>
            <th scope="col">Room Number</th>
            <th scope="col">CO2 Level</th>
            <th scope="col">Smoke Level</th>
            <th scope="col">Status</th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let sensor of sensors">
            <td>{{sensor.id}}</td>
            <td>{{sensor.floor_no}}</td>
            <td>{{sensor.room_no}}</td>
            <td>{{sensor.co2}}</td>
            <td>{{sensor.smokeLvl}}</td>
            <td *ngIf="sensor.status == 'active'"><i class="fa fa-circle text-
danger"></i></td>
            <td *ngIf="sensor.status == 'inactive'"><i class="fa fa-circle text-
success"></i></td>
        </tr>
        </tbody>
      </table>
</div>
```

body.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Sensors } from '../model/sensor';
import { SensorService } from 'src/app/sensor.service';
```

```
@Component({
  selector: 'app-body',
  templateUrl: './body.component.html',
  styleUrls: ['./body.component.css']
})
export class BodyComponent implements OnInit {
  sensors: Sensors[];
  constructor(private SensorService: SensorService) {
    setInterval(() => this.getSensorDetails(), 40000);
  }

  ngOnInit() {
    this.getSensorDetails();
  }

  getSensorDetails() {
    this.SensorService.getSensorDetails()
    .subscribe((data:any) => {
      console.log(data)
      this.sensors = data;
    })
  }

  onEverySecond() {
    console.log('second');
  }

}
```

header.component.html

```
<div class="layout">
    <div class="jumbotron jumbotron-fluid">
        <div class="container">
          <h1 class="display-4 text-center font-weight-bold text-uppercase">Fire
Alarm System</h1>
        </div>
      </div>
</div>
```

header.component.ts

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

sensor.ts

```
export class Sensors {
    id: number;
    floor_no: number;
    room_no: number;
    co2: number;
    smokeLvl: number;
    status: string;
}
```

sensor.service.ts

```
import { HttpClient } from '@angular/common/http'
import { Injectable } from '@angular/core';
import { catchError } from 'rxjs/Operators'
import { from, Observable, throwError} from 'rxjs'


import { Sensors } from './components/model/sensor';
```

```
@Injectable({
  providedIn: 'root'
})
export class SensorService {

  constructor(private http: HttpClient) { }

  // fetch data
  getSensorDetails(): Observable<Sensors[]> {
    return this.http.get<Sensors[]>(`http://localhost:8080/location`).pipe(
      catchError(error => {
        return throwError('Something went wrong!');
      })
    )
  }
}
```

- Sensor Application

SensorApp.java

```
public class sensorApp {
        public static void main(String[] args) {
                Client client = new Client().create();
                Gson gson = new Gson();


                while(true) {
                        try {
                                WebResource web = client.resource("http://localhost:8080/location");
                                ClientResponse res =
web.accept("application/json").get(ClientResponse.class);

                                if (res.getStatus() != 200) {
                                        throw new RuntimeException("Failed: HTTP error code: " +
res.getStatus());
                                }

                                Location[] location = new Location[100];
                                location = gson.fromJson(res.getEntity(String.class), Location[].class);
```

```
					Random co2 = new Random();
					Random smokeLvl = new Random();

					int levelCo2,levelSmoke;


					for(int count = 0 ; count < location.length ; count++) {

						WebResource webResource =
client.resource("http://localhost:8080/location/update");
						JSONObject status = new JSONObject();



						levelCo2 = co2.nextInt(10)+1;
						levelSmoke = smokeLvl.nextInt(10)+1;

						System.out.println("Co2 Level " + levelCo2);
						System.out.println("Smoke Level " + levelSmoke);

						status.put("id" , location[count].getId());
						status.put("floor_no", location[count].getFloor_no());
						status.put("room_no", location[count].getRoom_no());
						status.put("co2" ,levelCo2);
						status.put("smokeLvl" ,levelSmoke);

						if(levelCo2 > 5 || levelSmoke > 5) {
							status.put("status" ,"active");
						}

						else {
							status.put("status" ,"inactive");
						}


						ClientResponse response =
webResource.type("application/json").put(ClientResponse.class, status.toString());

						if (response.getStatus() != 200) {
							throw new RuntimeException("Failed : HTTP error
code : " + response.getStatus());
						}

						System.out.println(response.getStatus());
```

```
                        }

                        sensorApp.timer();





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



        }
    }





    public static void timer() {

        try {
            Thread.sleep(10000);

    }catch (Exception e) {

    }

}


}
```

- REST API

Location.java

```
@Entity
@Table(name = "LOCATION")
public class Location {
	@Id
	@GeneratedValue(strategy = GenerationType.AUTO)
	@Column(name = "LOCATION_ID")
	private int id;
	@Column(name = "FLOOR_NO")
	private int floor_no;
	@Column(name = "ROOM_NO")
	private int  room_no;
	@Column(name = "CO2")
	private int co2 = 1;
	@Column(name = "SMOKELVL")
	private int smokeLvl = 1;
	@Column(name = "STATUS")
	private String status;

	public Location() {

	}

	public int getId() {
		return id;
	}

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

	public int getFloor_no() {
		return floor_no;
	}

	public void setFloor_no(int floor_no) {
		this.floor_no = floor_no;
	}

	public int getRoom_no() {
		return room_no;
	}

	public void setRoom_no(int room_no) {
		this.room_no = room_no;
	}

	public int getCo2() {
		return co2;
	}

	public void setCo2(int co2) {
		this.co2 = co2;
	}

	public int getSmokeLvl() {
		return smokeLvl;
	}

	public void setSmokeLvl(int smokeLvl) {
		this.smokeLvl = smokeLvl;
	}

	public String getStatus() {
		return status;
	}

	public void setStatus(String status) {
		this.status = status;
	}


}
```

LocationController.java

```
@RestController
public class LocationController {
	@Autowired
	private LocationService locationService;

	@RequestMapping(value = "/location", method = RequestMethod.GET)
	@CrossOrigin(origins = "*")
	public List<Location> getLocationDetails() {
		return locationService.getAllLocation();
	}

	@RequestMapping(value = "/location/{id}")
	public Location getLocationById(@PathVariable int id) {
		return locationService.getLocation(id);
	}

	@RequestMapping(value = "/location/save", method = RequestMethod.POST)
	public void saveLocation(@RequestBody Location location) {
		locationService.newLocation(location);
	}

	@RequestMapping(value = "/location/update", method = RequestMethod.PUT)
	public void updateLocation(@RequestBody Location location) {
		locationService.updateLocation(location);
	}

	@RequestMapping(value = "/location/delete/{id}", method = RequestMethod.DELETE)
	public void deleteLocation(@PathVariable int id) {
		locationService.deleteLocation(id);
	}
}
```

LocationRepository.java

```
public interface LocationRepository extends CrudRepository<Location, Integer> {

}
```

LocationService.java

```
@Service
public class LocationService {
	@Autowired
	private LocationRepository locationRepository;

	// return location details
	public List<Location> getAllLocation() {
		List<Location> list = new ArrayList<>();
		locationRepository.findAll().forEach(list::add); // find all the data related to Location
		return list;
	}

	// get location details by id
	public Location getLocation(int id) {
		// a container object which may or may not contain a non-null value
		Optional<Location> optionalLocation = locationRepository.findById(id);
		if (optionalLocation.isPresent()) { // check optionalLocation has value  then return it
			return optionalLocation.get();
		}

		return null;
	}

	// save
	public void newLocation(Location location) {
		// save new location
		locationRepository.save(location);
	}

	// update
	public void updateLocation(Location location) {
		// see the id and update particular location
		locationRepository.save(location);
	}

	// delete
	public void deleteLocation(int id) {
		// delete location by id
		locationRepository.deleteById(id);
	}
}
```

- Alert Services

Email.java

```
public class Email
{
   // email ID of Recipient.
   public static final String recipient = "it18107388@my.sliit.lk";
   // email ID of  Sender.
   public static final String sender = "urbanrunes@gmail.com";
   public static final String host = "smtp.gmail.com";

	public static void sendMail() {
		// Getting system properties
```

```
                Properties properties = System.getProperties();

                // Setting up mail server
                properties.put("mail.smtp.host", host);
                properties.put("mail.smtp.port", "465");
                properties.put("mail.smtp.ssl.enable", "true");
                properties.put("mail.smtp.auth", "true");

                // creating session object to get properties
                Session session = Session.getDefaultInstance(properties, new Authenticator() {
                        protected PasswordAuthentication getPasswordAuthentication() {
                                                return new PasswordAuthentication("urbanrunes@gmail.com",
"urbanrunes504");
                                }
                });

                session.setDebug(true);

                        try
                {
                  // MimeMessage object.
                  MimeMessage message = new MimeMessage(session);

                  // Set From Field: adding senders email to from field.
                  message.setFrom(new InternetAddress(sender));

                  // Set To Field: adding recipient's email to from field.
                  message.addRecipient(Message.RecipientType.TO, new InternetAddress(recipient));

                  // Set Subject: subject of the email
                  message.setSubject("Alert");

                  // set body of the email.
                  message.setText("Fire Alarm alert !");

                  // Send email.
                  Transport.send(message);
                  System.out.println("Mail successfully sent");
                }
                catch (MessagingException mex)
                {
                  mex.printStackTrace();
                }
            }

}
```

SMS.java

```
public class SMS {
	public static final String ACCOUNT_SID = "AC6444898123c47140151d8a6f9d6ccabb";
	public static final String AUTH_TOKEN = "cd84eead336cb9e4a9a5038ea5d5b6dd";
	public void sendSMS() {
		Twilio.init(ACCOUNT_SID, AUTH_TOKEN);

		Message message = Message.creator(new PhoneNumber("+940714302153"), new
PhoneNumber("+16623301290"), "Fire Alarm alert !").create();

		System.out.println(message.getSid());
	}
}
```