Group Project – Integrative Programming

Instructions:

- This is a group assignment. Each group should not exceed Five(5) members.
- **All the group members should participate in the demonstration and viva session.**
- The deliverables should be submitted to CAL before the deadline.
- Divide the work among group members before starting the work.
- **Copied assignments will be given zero marks.**

Deliverables:

- The git repository link which contains the source code and code commits.
- A pdf report which contains the contribution of each team member and the challenges/issues faced by each group member (there must be a separate section for each group member).

Marking Criteria

- Marks will be given based on the overall progress of the group and your **individual contribution**.
- Marks will be allocated based on the following points
    - Completion of work
    - Usage of software design patterns
    - Code quality
    - Maintainability and extensibility of the source code.

Task:

- Implement the given user requirement using **a REST API** (For backend)**, a frontend app** for the user interface.
- Use any database server to store the data as you wish.
- Use any framework to implement the REST API (Ex: Spring).
- Use any Javascript framework (Such as Angular, React, Vuejs) to implement the front-end.

Assumptions:

- Assume that the system only monitors temperature sensors in this version.
- Assume that the sensor data providers (gateways) can send **POST** requests to an endpoint defined in your backend in order to send the sensor data periodically. The format of the Rest API call is given below.

```
POST ip:port/data
Body:
 {
   "sensor_id": "xxxx",
   "date": "2018-12-01 10:20:12",
   "data_value" : "35C"
 }
```

How to demonstrate (In the ViVa)

- You can mock this API call using **Postman** to demonstrate the application functionality.
- In the demonstration, you should be able to send a mock request to the backend endpoint and show how the data is getting stored in the database. Then, when you send a data reading which is above a certain threshold, it should trigger an alert, and the system should send email, SMS, and phone calls to the user. You should send actual emails but it is not required to send actual SMS or phone calls. You can use print statements for that.
- Show the data reading graphs and event views in the front-end

User Requirement:

“Monitor” is a cloud-based sensor monitoring and alert management platform which can be used to centrally monitor any device regardless of its location. The main functionality of the system is to alert on certain events based on the readings coming from the sensors.

The system should be able to collect sensor data readings from different sensors such as Temperature sensors, Humidity sensors, Pressure sensors etc.

Monitor system performs monitoring part only (The alert generation and notifying the user when there is a device failure). We should use the third party sensor data providers such as Aretas, Imonnit and Helium which can send the sensor readings to our system.

The alert generation logic is simple for temperature sensors. If the current sensor reading goes above a certain threshold the system should trigger an alert. As an example, if the current sensor reading is 30C and the threshold is 20C, it would be considered as an alert condition. But the alert generation logic can be more complex for other sensors such as Pressure sensors and Vibration sensors. **<u>The initial version of the system only supports temperature sensors.</u>**

When there is an alert, the system should notify the user via **Email, SMS or phone calls**. The user should be able to select the notification channels that they want.

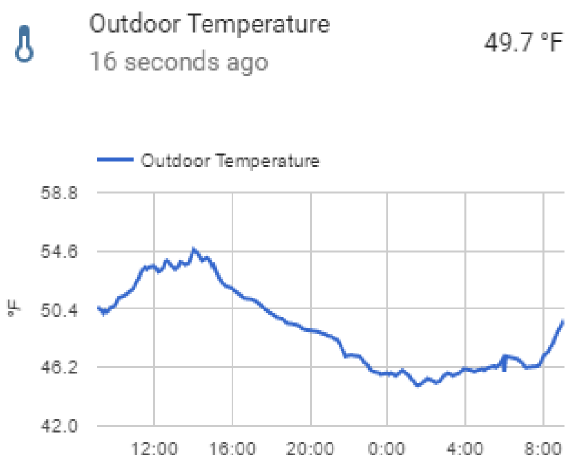
The system should visualize the sensor readings for each sensor and the status of the sensor to the user using a web portal. The sensor readings should be visualized as graphs for each sensor and the user should be able to see the past alerts that occurred in each sensor via the user interface.

Minimum requirement to implement

- **Data ingestion:**
  - Should be able to capture sensor readings from an external sensor reading provider. You can use Postman to demonstrate this.
  - Should be able to format the data and save the data in the database that you have chosen.

- **Backend**:
  - Should have an endpoint to provide the sensor readings of a particular sensor.
  - The alert generation logic for temperature sensors should be implemented. The threshold value must be unique per sensor.
  - Should have the logic to send notifications. You should send actual emails using a third party system like Mailgun, but it is not required to send actual SMS or phone calls. You can use print statements for that.
  - Should have an endpoint to provide the historical events of a particular sensor.
  - Authentication of users should be implemented.

- **Front-end**:
  - Should have a screen to show the past sensor readings in the form of a chart. A dropdown should be available to select the sensor.
    Ex:



  - Should have a screen to show the past alerts of a sensor in the form of a table. A dropdown should be available to select the sensor.