Group Project
Principles of Software Architecture
Deadline: 07/03/2021

Instructions:

- This is a group assignment. Each group should not exceed **three(3)** members.
- This group project is related to the user requirement stated in assignment-1.
- **All the group members should participate in the demonstration and viva session.**
- **The source codes should be committed to a public git repository.** Commit history will be reviewed to evaluate the individual contribution. Make sure to add meaningful messages as commit messages.
- Divide the work among group members before starting the work. **The marks will be given individually**. Even if some part of the system (assigned to somebody else) is not completed, you will get the full mark if your part is completed.
- A pdf document should be submitted on or before the deadline. No need to upload any source codes. The document should contain the following information.
    - Index numbers and names of the group members
    - Git repository link to the project
    - A screenshot of the nifi template.
    - Assumptions that you made during the implementation if any.
    - Individual contribution of each group member.
- Make sure to write extensible and quality codes. Using design patterns and unit tests would be an added advantage.
- **Copied assignments will be given zero marks.**

Task:

- Implement the given user requirement using **a REST API** (For backend)**, a frontend app** (For user interface) and **Data-flow architecture** (For data ingestion).
- Use **Apache NiFi** for data ingestion.
- 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. But it should be extended to support other sensor types such as humidity and pressure in future.
- Assume that the sensor data providers (gateways) can send **POST** requests to your NiFi endpoint in order to send the sensor data periodically. You can use **ListenHTTP** nifi processor to listen to external HTTP requests. 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 nifi 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. In order to send Emails, you can use **Mailgun***(https://www.mailgun.com/)* or a similar free service. It is not required to send the actual phone calls or texts. You can just use print statements for that.
- Show the data reading graphs and alerts 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. **The initial version of the system only supports temperature sensors.**

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.

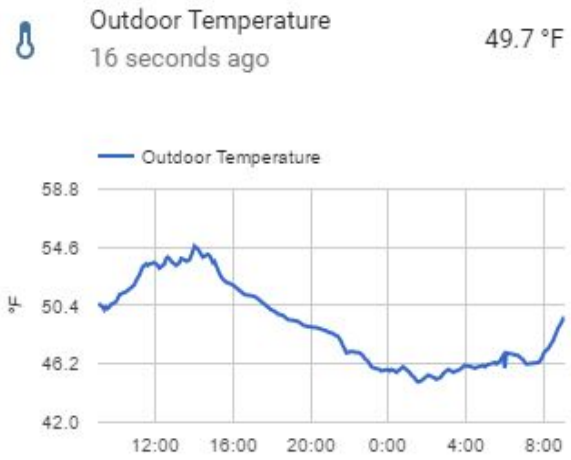
The system should visualize the sensor readings for each sensor and the status of the sensor to the user using a web portal. The sensor readings should be visualized as graphs for each sensor and the user should be able to see the past alerts that occurred in each sensor via the user interface.

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. The real notifications are required only for emails. Having print statements would be enough for SMS and Voice notifications.
    - Should have an endpoint to provide the historical events of a particular sensor.
    - Authentication is nice to have. But not mandatory.

- **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. The date range filter should be available as well.
      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.