Tech-Lead – Full stack engineer (mid) assignment

Time – 5 Days.

Deliverables

1. You should submit efficient, clean, and sufficiently commented code. Your submission should be hosted on a **private git repository** and invite the reviewers (they will be reported through an e-mail) to your personal repository. Please include a Read.me file with any necessary documentation
2. You should submit a presentation in PowerPoint where you present at a high-level your solution's design. This presentation will be used to present your solution during the interview following the assignment. You should also submit a **report** (in pdf - 500 words) describing the process followed and key results.
   Note: The presentation will be used to present your solution during the interview following the assignment.

Requirements

Part A

In this part, you will need to **design** a solution for a meteorological web application. In this application there are several data sources:

- Every 10 seconds, IoT devices installed in meteorological stations in each city across the county, measure properties (such as temperature, humidity, and wind) and stream them to the backend.
- Every morning, each meteorological station in each city collects all the data from the last 24 hours and sends them to the backend as a batch
- A user uses daily the UI and submits a form with the weather forecast for tomorrow for each city regarding temperature, humidity, and wind

The backend should be able to receive all data and store them. It should also be able to provide the data to the UI for visualization upon request.

What you need to do:
*Please **design** (not implement) and showcase with a diagram the solution of this application*

Part B

In this part, you will need to implement part of the backend solution of part A, given that:

1. The meteorological stations have the following properties:
   - Code
   - City
   - Latitude
   - Longitude
   - Date of installation

2. Sensors have the following properties:
    a. Sensor Id
    b. Code of the Meteorological Station they belong to
    c. Property of measurement (temperature, humidity or wind)

3. The form submitted by the user through the UI contains the following fields:
    a. Forecast date
    b. City
    c. Temperature
    d. Humidity
    e. Wind

4. The measurements from the IoT devices follow the same JSON structure. A sample of the measurements can be seen here:

```
{
  "identifier": "88d58a78-95d3-4b9c-b0a4-aab2e2ed73d5",
  "sensor": "HUM-001",
  "date ": "2024-05-28T15:22:18+02:00",
  "city": "London",
  "info": [
    {
      "category": "Humidity",
      "measurement": 56,
      "unit": "Percentage"
    }
  ]
}

{
  "identifier": "435k65l345-95d3-4b9c-b0a4-34fsd234fs",
  "sensor ": "TEM-456",
  "date ": "2024-05-28T15:22:18+02:00",
  "city": "London",
  "info": [
    {
      "category": "Temperature",
      "measurement": 56,
      "unit": "Celsius"
    }
  ]
}

{
  "identifier": "9343h43-95d3-4b9c-b0a4-4543gs546gd",
  "sensor ": "WND-245",
  "date": "2024-05-28T15:22:18+02:00",
  "city": "London",
  "info": [
    {
      "category": "WIND",
      "measurement": 11,
      "unit": "m/s"
    }
  ]
}
```


What you need to do:

a) Develop the following UI components interacting with the backend's APIs:
   1. Table with forecast inputs. Inputs can be created, edited and deleted
   2. Weather widget that will display current weather and forecast that is predicted by user
   3. Visualized city temperature (current and predicted) over the last days. This can be achieved in the UI or generated on the backend and served as downloadable file

b) Develop the backend solution that will be able
   1. to receive and store data from the IoT sensors asynchronously and from the UI synchronously
   2. serve any data required by the UI

Note: Please ignore any data coming as a batch.

c) Dockerize the services to be runnable independently of the platform

Guidelines

- Backend should be written in Python (Flask, Fast or Django frameworks are accepted)
- Frontend should be written in TypeScript/React. Optionally WebSocket can be used for real-time UI updates.
- You can use any tool or library you wish
- API documentation is required
- Any other addition is optional (i.e. Testing, websockets, etc.)

Good luck!