Mobile front-end task

Your task is to develop mobile app for a monitoring solution.

Task Details

The application is a part of an overall temperature monitoring solution. Various temperature sensors send real time temperature updates to server. There are http mobile clients which are interested in continuously monitoring the temperature and they subscribe for it with the server. Whenever server receives an update, it publishes the updates to all interested clients.

For this exercise, you need to develop the App for this monitoring solution. There is a sample backend running at http://interview.optumsoft.com. You can use this backend to connect and receive real time data. As a mobile app developer, you need to develop mobile app client which should be used to visualize (with graphs) the real time data which is sent from server.

Data detail

In the monitoring system there are multiple temperature sensors which keep sending data to backend.

Sensor Names
You query the number of sensors and their names by this GET request

http://interview.optumsoft.com/sensornames

Here is an example of the response

```
[
  "temperature0",
  "temperature1",
  "temperature2",
  "temperature3"
]
```

Sensor Config
There is a config associated with each temperature sensor. For each sensor, there is a min and max value defined. If a value of sensor reading lies between min and max, then the value is considered to be normal, otherwise the value is considered to be deviation. The mobile app display system has to indicate whether a reading is normal or deviation from the expected range. To get sensor config, you need to make a GET request to:

http://interview.optumsoft.com/config

Here is a sample response:

```
{
  "temperature0": {
    "min": 0,
    "max": 17
  },
  "temperature1": {
    "min": 2,
    "max": 9
  },
  "temperature2": {
    "min": 4,
    "max": 18
  },
  "temperature3": {
    "min": 2,
    "max": 6
  }
}
```

Connecting to backend

The backend exposes sensor data via socket.io. Use the socket.io library for the mobile app platform of your choice (iOS or Android). Your end point of the socket.io library will be *http://interview.optumsoft.com*

Data and time scale

The sensor readings are available in two time scale

- recent - this is raw values of sensor reading. In this scale, you will see data is available at around 2 seconds interval
- minute - the data in minute scale are aggregated from raw value. In this scale, the data is available at around one minute interval.

Subscribe and unsubscribe

Once you are connected to backend, you can subscribe for temperature data. You have to subscribe for every sensor to receive the data of that particular sensor.
To subscribe, use the following code

```
socket.emit("subscribe","temperature0");
```

Please note that the second argument is sensor name.
If you do not want to receive any more data for a sensor, then you need to unsubscribe with following code

```
socket.emit("unsubscribe","temperature0");
```

Events

There are two events that you will have listen in client application

- connection : called when the socket is connected.
- data : called when you receive data from server.

Message received from backend

There are three kinds of messages that you will receive after you subscribe

- Init : This contains current values of sensor reading in both time scale (recent and minute). This message you will receive just after you subscribe.
- Update : Update is received when backend gets a new sensor reading
- Delete: The backend stores only a limited number of data points at a time. That means the backend keeps deleting older data points. When backend deletes the data point, it also sends the update to subscribed client. The client is not supposed to display the data which has been deleted at backend.

Example of Init message

```
{
  type: 'init',
  recent: [
    { key: 1479472143.29, val: 40.780689232309264  },
    { key: 1479472145.297, val: 16.956397675705137}],
  minute: [ { key: 1479472143.29, val: 40.780689232309264  }]
}
```

The init message sends current values of sensor in recent and minute time scale.

Example of Update message

```
{
  type: 'update',
  key: 1479472335.085,
  val: 72.84476149972754,
  sensor: 'temperature0',
  scale: 'recent'
}
```

Whenever a new value is added at backend, then an update message is sent to subscribed clients. The message contains, timestamp as key, sensor reading, name of the sensor and time scale (recent ot minute)

Example of Delete message

```
{
  type: 'delete',
  key: 1479472463.413,
  sensor: 'temperature9',
  scale: 'recent'
}
```

The backend stores only a limited number of data points at a time. That means the backend keeps deleting older data points. When backend deletes the data point, it also sends the update to subscribed client.

The client is not supposed to display the data which has been deleted at backend.

Points to consider

- Use native app development framework to develop the app. For iOS you should use swift. For Android you should use Kotlin or Java.
- You should use the socket.io library to connect to backend
- Use your imagination to create visually appealing graphs for sensor data
- You should give an option to users to choose the sensor whose values will be displayed. Bonus point, if you give option to choose multiple sensors in a single graph
- You should give option to choose the time scale (recent or minute)
- There should be visual indication of deviation values

Questions

Should you have any doubts about this exercise, please feel free to send us your questions to kiran@optumsoft.com (+91 9845015096) or tulsa@optumsoft.com (+91 99100 29644).

We should respond to you within a few hours during work hours, or by next day in case you are burning midnight oil.

Deadline

You have 7 days to send us your response from the moment you receive this email. For example, if you receive this email on 10 am, August 25. Then we expect you to respond by 10 am on September 1.

It will be very helpful if you can send us approximate development time you took to complete the entire exercise breaking it up into individual parts.

Good luck, and we hope you have fun working on this exercise.