Smart AC Proof of Concept

One of our clients wants to create a smart air conditioner. They've already created the physical device and luckily for us, it has full HTTP client functionality.

They want us to do a proof of concept of a backend system which integrates with all of their air conditioning units and provides them with an admin panel to manage their system.

We need to build an HTTP API for the AC units to connect to and a simple web app to serve as an admin panel for this AC company.

The devices API should support:

- Registering a new device:
  - Each device will have a serial number, a registration date, and a firmware version.
- Allowing each device to authenticate and send data from its sensors once per minute. The sensors on each device provide:
  - Temperature in Celsius.
  - Air humidity percentage.
  - Carbon Monoxide level in the air.
  - Device health status. It can be any text, less than 150 characters.
- Allowing a device to send values for its sensors in bulk, up to 500 values for each of its sensors instead of just 1.
  - If the internet goes out in a home, the AC unit will reconnect when it can to the wifi and send all the data to our servers that it couldn't send before. After that, it will continue sending data normally, once per minute.
  - Should our servers go down, the devices may retry a few times. Repeat attempts will pack values together until our server goes back up.

The admin web app should support:

- Logging in and out, with support for recovering/resetting your password.
- A mechanism for inviting others to log in with a private invitation link.
- A way to list all admins and block or re-enabling each of them.
- A way to list all devices.
- A way to search for a device given its serial number.
- A way to see the details of a device, including the values for each of its sensors.
  - Values for each sensor, except health status, should be displayed in a graph.
  - For each sensor, admins should be allowed to look for its values from "today", "this week", "this month" or "this year."

- The web app should notify logged in admins when a device reports a value of Carbon Monoxide of over 9 PPM.
    - The notification should have a way to mark it as resolved, so once an admin marks it as such, the notification doesn't appear again for anyone.
- The web app should also notify logged in admins in the same way when a device reports a health status of "needs_service", "needs_new_filter" or "gas_leak."

Your task consists of creating and deploying both this HTTP API for devices and admin panel for our client, following the requirements outlined above. For this task, you have to follow these requirements:

- By the end of the exercise, you have to provide us with the URL (and a user/pass) for the admin panel, as well as the URL for the API for devices. You can host both things under the same domain.
- You also need to provide us with basic documentation for the API for devices.
- You can use any tools or technologies you want for any part of this task.
- You also need to show us the code you used for accomplishing this. This can be a link to a GitHub repo, a zip file, or any way that works best for you.

We know this seems like a lot. Perhaps it's more work than humanly possible to do in two days. The reason we packed so much into the requirements is that this is what real-life situations are sometimes like. It doesn't mean you need to do everything listed above to pass the exercise though! It just means it is up to you to navigate the situation and figure out how to resolve it.

You need to work with us (the client) to identify what compromises can be made, what things can be put off for later, maybe some functionality could be outsourced or even faked? For us, some requirements are crucial and others are just nice-to-have, and the point of this exercise for you is figuring out which is which, even when we don't tell you outright.

There are no rules, no predefined criteria for success other than if the client is satisfied at the end of the two days, you pass. How you accomplish the goal of client happiness is entirely up to you.

Will you make assumptions? Will you ask important questions? Will you prioritize and organize your work? Will you try to do everything on this list or will you leave out some features? Will you win over the client? If necessary, will you redefine what success is? These are the questions we hope to be able to answer with this exercise. These are more important than the code you write.

For this type of proof of concept, we're not looking to see if the codebase is elegant, polished and well tested. We only care about how you interact with the client and the final software you deliver at the end of the exercise.

We hope you have fun :-) Good luck!