nexthink

Full-Stack Software Engineer Assignment

Thank you for answering these questions to the best of your competences. The purpose of this assessment is to evaluate your areas of expertise as well as your analytical thought process. Please return your completed assignment preferably within 7 days (max. 10) of reception. Depending on your proficiency level, answering all questions should take you between 6 and 12 hours of work time. The questions are broad, so prioritize your efforts aggressively on key information and simple solutions. If a question is unclear, feel free to rephrase it or lay new assumptions, but don't forget to document them. It is advised to read and understand all questions before starting to answer them.

For more information on the Nexthink product visit our website, blog, and documentation. Before starting the assignment, watch these short videos created by our partners, in order to understand the main components of our product and how it is used:

- https://www.youtube.com/watch?v=VAQMQDJYEZc
- https://www.youtube.com/watch?v=FL8awrXz_z8

Please provide your writeup in `.pdf` format, along with the source code and other relevant files packaged in a single zip archive named like `NXDIAG_firstname_lastname.zip`. If the package is larger than 10Mb please provide a link to download it.

If you have **any** feedback on this assignment, we're listening.

Good luck!

Assignment 1: Device Performance Index

IT Directors, CIOs, and many high-level managers of big companies are some of the main users of our products. These are also the same people that use dashboards and simple KPIs regularly to assess the health of the company's IT infrastructure, as well as to monitor its evolution. In this exercise our goal is to provide them with a global Device Performance Index (DPI), which will be computed for every device. The DPI will mainly be used by our customers to:

- compare the overall device performance across various offices (internal to a customer)
- benchmark overall performance against other customers
- retrieve bad and well performing device IDs to further investigate what makes them special

For this assignment, you will:

1. Implement a service that calculates the DPI based on a static dataset, according to the specifications below.
2. Create an API using that service to allow for device data retrieval based on the device, customer and office IDs as well as DPI values. Selection logic should cover: equals, less than a value, higher than a value, value range.
3. Build a front end in React using that API showing a list of devices with the following features:
    a. Display on each line the device, customer, office IDs and DPI value
    b. A system to allow the user to filter the list on those properties by leveraging the API you previously built.
    c. A small widget showing the average DPI for all devices selected in the list

When implementing please write your code as if it were going into production. Please take note of any vital work to be done before the end-user release that you don't have the time to implement.

DPI specifications

The DPI must have values in $[0, 10]$, 0 representing a terrible device and 10 being a top performer.

The static dataset for computing the DPI is attached in the `device_performance_index.csv` file, where each line contains information about a single device, totaling 9 different customers and 742 offices. Description of the data and units are given in the accompanying `readme` file.

For the score computation, we first normalize each metric value according to the minimum and maximum values calculated from historical data:

$$X_i^{norm} = \min\left(1, \max\left(0, \frac{X_i - X_i^{min}}{X_i^{max} - X_i^{min}}\right)\right)$$

Because all the metrics excepted the `System_Free_Space` indicate worse performance with higher values, we flip them over:

$$X_i^{norm} = 1 - X_i^{norm}, \text{for } i \in [1,6]$$

Then we simply add them together to obtain the DPI:

$$DPI = \frac{10}{7}\sum_{i=1}^{7} X_i^{norm}$$

Your service needs to include a function to determine the X_i^{min} and X_i^{max} configuration parameters from historical data. Because we want to ignore outliers, X_i^{min} and X_i^{max} will be calculated as the 2nd and 98th percentile of each metric. For the assignment, you will calculate those values from the provided sample dataset, but for production, we will regularly compute it with a dataset containing hundreds of data points for millions of devices.

Assignment 2: Alerting Design

In addition to the previous DPI use-cases, many of our clients export the DPI scores and other metrics at regular intervals into their own business intelligence data platform, where they can monitor the evolution of the DPI and act when changes or degradations appear in their dashboards.

We want to expand our product features to cover this monitoring use-case by providing a cross-client cloud solution with:

- integrated DPI time-series data storage, update and retrieval
- automated alert creation through DPI anomaly detection
- real-time notification of users when alerts occur

Your objective is to sketch the high-level design and architecture of a monitoring and alerting system aimed to detect sudden drops of individual DPI scores and changes in around 20 other metrics. Since alerting on a single device can be sensitive to noise, we want to have both alerting at device level as well as alerting at office level, with different sensitivity. The alerts should be accessible through a web front-end, but they should also generate notifications (e-mail, mobile, etc.).

The product and technical requirements for the monitoring and alerting system are the following:

- an alert should be triggered if a significant drop of one of the DPI scores and metrics is detected during the last 10 minutes compared to baselines computed on past data
- alerting should be done at both device and office level
- Solution should scale to 1 billion devices, globally distributed among 1000 clients with an average of 1000 offices/clients (min 10, max 40000)

Along with the design, you should provide us with:

- The assumptions you made
- The existing tools and solutions you would choose and the rationale behind the choice
- The main challenges you foresee
- The tests, or proof-of-concepts, you should build to make sure your solution works in a production environment