Everon Back-End coding assignment

Dear candidate,

Thank you in advance for taking your time to complete our coding assignment.

You will have up to 5 days to complete the test. However, I would appreciate it, if you could complete the test as soon as possible, since we will have to review the result and only thereafter move forward in our recruitment process.

Purpose

Main purpose is to describe the requirements for the coding assignment, and to test practical skills of designing, coding, testing and deploying java applications.

The scope of the assignment

The assignment should contain the implementation of several endpoints to work with ChargingSession object, one of the main concepts Everon is working with.

MoSCoW prioritization technique is used for requirements description:

- **M** - MUST: Describes a requirement that must be satisfied in the final solution for the solution to be considered a success;
- **S** - SHOULD: Represents a high-priority item that should be included in the solution if it is possible. This is often a critical requirement but one which can be satisfied in other ways if strictly necessary;
- **C** - COULD: Describes a requirement which is considered desirable but not necessary. This will be included if time and resources permit;
- **W** - WON'T: Represents a requirement that stakeholders have agreed will not be implemented in a given release but may be considered for the future. (note: occasionally the word "Would" is substituted for "Won't" to give a clearer understanding of this choice).

Prerequisites

Basic java knowledge, along with experience in OOP, API design, unit and integration testing, and understanding of basic algorithms and data structured is a necessary prerequisite for taking this test.

Assignment requirements

1. The key concept of the assignment is a ChargingSession, which represents the charging process on the charging station. It **must** have the following fields:
    - *id* - unique identifier of the charging session;
    - *stationId* - the id of the station where charging happened;
    - *startedAt* - timestamp which indicates when the charging session started;
    - *status* - the status of the charging session indicates if it is in progress or finished.
2. The candidate **could** extend the model of ChargingSession and include fields like *endedAt*, *consumption* etc.
3. The following endpoints **must** be implemented and **must** be thread-safe:
    - POST /chargingSession – submit a new charging session for the station. The complexity of storing the charging session **must** be O(1);
    - PUT /chargingSession/<session_id> – stops charging session. The complexity of the operation **must** be O(1);
    - GET /chargingSessions – retrieves a summary of submitted charging sessions including:

      totalCount – total number of charging sessions for the last minute

      startedCount – total number of started charging sessions for the last minute

      stoppedCount – total number of stopped charging sessions for the last minute

      The complexity of retrieval **must** be O(1);

1. The following endpoints **could** be implemented:
    - GET /chargingSessions/stopped – retrieves a summary of stopped charging sessions:

      stoppedCount – total number of stopped charging sessions for the last minute

      The complexity of retrieval **must** be O(1);

      1.
    - GET /chargingSessions/started – retrieves a summary of started charging sessions:

      startedCount – total number of started charging sessions for the last minute

      The complexity of retrieval **must** be O(1);

      1.

    - DELETE /chargingSessions – removes stopped charging sessions. The complexity must be O(1).
2. Application code **must** be covered with tests. The testing frameworks and approach should be chosen by the candidate;
3. The candidate **must** provide the documentation of the implemented functionality and instructions how to run (javadocs and README);
4. The code **could** be built using tools like maven or gradle;
5. As part of this assignment persistence layer **won't** be implemented. Code base **must** only use in-memory data-structures.

We hope you will enjoy this assignment and should you have any questions – please reach out to us 🙂

Thanks again for your interest!