Senior Software Engineer Assignment

The purpose of this assignment is to give us a better sense of your skill-set and fit for the role. We place a lot of value on the assignment and use it to make our decision. Please select and work on only 1 (one) of the listed assignments below.

Your responses should be submitted via google drive and shared with email and email with the subject *'[Name] - Senior Software Engineer Assignment: [Name of selected assignment]'*.

Do not hesitate to reach out with any questions.

Good luck!

Assignment 1 - API rate limiter

I. Context 🛖

Corporation X,Y,Z is a tech company that has launched a notification service for sending SMS and E-mail notifications. They are selling this service to different clients and each client has specific limits on the number of requests they can send in a month. Because they are a startup they have a limited amount of infrastructure to serve all clients at peak capacity because their solution has been very successful.

Each client has the ability to pay for more requests per second. Corporation X,Y,Z is seeing performance issues on their api, because they haven't implemented the limits that have been set out in the software

II. Requirements 🎨

1. Goal

The goal is to design and implement an API rate limiter system for. The solution must use Angular for Frontend and Java Spring Boot / NodeJS for the Backend. *NB : Java Spring Boot is the preferred stack.* On top of these, feel free to use other technologies that may help solve the problem.

2. Use cases

As mandatory use cases,

1. The company can define the number of requests within a windows time from a specific client
2. The company can define the number of requests from a specific client on a per month basis
3. The system will limit number of requests within a windows time from a specific client
4. The system will limit number of requests from a specific client on a per month basis
5. The system will limit number of requests across the entire system

3. Key concerns to address

Points to look for in the candidate's solution:

- How should you try to solve these three issues:

1. Too many requests within the same time window from a client
2. Too many requests from a specific client on a per month basis
3. Too many requests across the entire system

- The rate limiting should work for a distributed setup, as the APIs are accessible through a cluster of servers.
- How would you handle throttling (soft and hard throttling etc.).

III. Delivery Expectations 🚀

1. The Design of the Solution

A Google Slide presenting the architecture of the system and the design choices.
i) For the architecture, we prefer diagrams vs text.
ii) Please ensure that your design choices clearly address the key concerns highlighted above.

2. A working prototype of the solution

👉🏽 Github link for the source code.
👉🏽 A ReadMe file with instructions for how to run / test solution (root of the code base).
👉🏽 Link of the demo solution.

3. Presentation of the solution to Irembo technical lead

This session will be done after the first 2 deliverables are submitted.

👉🏽 Walkthrough of the solution - the demo, the architecture, and the code.

👉🏽 Presentation of how testing has been implemented and the test coverage report.

IV. Good to know 💡

- Provide a clear and comprehensive system architecture (with diagrams).
- Be ready to communicate and defend your design decisions.
- Scale is essential as the company is very successful.
- Clearly show how testing has been implemented.
- Be ready to present the demo of the solution

Assignment 2 - Certificate Management

I. Context 🛖

The startup “Sec CERTIFICATE” is a tech company that wants to launch a service helping organizations to define and generate certificates in PDF through an easy-to-use and secure user experience.

II. Requirements 🎨

1. Goal

The goal is to design and implement a Certificate Management system.
The solution must use Angular for the frontend, Java Spring Boot for the Backend, and PostgreSQL for the database. On top of these, feel free to use other technologies that may help solve the problem.

2. Use cases

As mandatory use cases,

1. The company (“Sec CERTIFICATE”) can onboard a customer to the system.
2. A customer can add a Certificate template to the system. A certificate template describes how a PDF certificate will look and indicate placeholders where actual information will be inserted.
3. Using the system UI, a customer can simulate the generation of a certificate from a template. A certificate is a document generated with the actual values in a template instead of placeholders.
4. Using an API of the system, a customer can generate a certificate (in PDF format) from a template.
5. Using an API of the system, a customer can download a PDF certificate.

3. Key concerns to address

Your design should clearly address the questions below:

- How to ensure that one customer cannot use the template of another customer to generate a certificate (UI and API perspective)?
- How to secure access to certificates (UI and API perspective)?
- How to ensure that the system can generate more than 1000 certificates per minute?
- To avoid fraud, how to ensure that a third party cannot reproduce a certificate generated from the system?

III. Delivery Expectations 🚀

1. The Design of the Solution

A Google Slide presenting the architecture of the system and the design choices.

i) For the architecture, we prefer diagrams vs text.

ii) Please ensure that your design choices clearly address the key concerns highlighted above.

2. A working prototype of the solution

👉🏽 Github link for the source code.

👉🏽 A ReadMe file with instructions for how to run / test solution (root of the code base).

👉🏽 Link of the demo solution.

3. Presentation of the solution to Irembo technical lead

This session will be done after the first 2 deliverables are submitted.

👉🏽 Walkthrough of the solution - the demo, the architecture, and the code.

👉🏽 Presentation of how testing has been implemented and the test coverage report.

IV. Good to know 💡

- Provide a clear and comprehensive system architecture (with diagrams).
- Be ready to communicate and defend your design decisions.
- Scale is essential as the company is very successful.
- Clearly show how testing has been implemented.
- Be ready to present the demo of the solution

Assignment 3 - Ride Sharing System

I. Context 🛖

Corporation “The RIDE” is a tech company want to launch a ride sharing service.

II. Requirements 🎨

1. Goal

The goal is to design and implement a ride sharing service.

The solution must use Angular for Frontend and Java Spring Boot / NodeJS for the Backend. NB : Java Spring Boot is the preferred stack. On top of these, feel free to use other technologies that may help solve the problem.

2. Use cases

As mandatory use cases,

1. The company can onboard drivers
2. Customer can request a ride, and will be match with the nearby drivers
3. Driver can accept a request of a customer

3. Key concerns to address

Points to look for in the candidate's solution:

- The most critical use case—when a customer requests a ride and how to efficiently match them with the nearby drivers?
- How to store millions of geographical locations for drivers and riders who are always moving.
- How to handle updates to driver/rider locations (millions of updates every second)?

III. Delivery Expectations 🚀

1. The Design of the Solution

A Google Slide presenting the architecture of the system and the design choices.

(i) For the architecture, we prefer diagrams vs text.

(ii) Please ensure that your design choices clearly address the key concerns highlighted above.

2. A working prototype of the solution

👉🏽 Github link for the source code.

👉🏽 A ReadMe file with instructions for how to run / test solution (root of the code base).

👉🏽 Link of the demo solution.

3. Presentation of the solution to Irembo technical lead

This session will be done after the first 2 deliverables are submitted.

👉🏽 Walkthrough of the solution - the demo, the architecture, and the code.

👉🏽 Presentation of how testing has been implemented and the test coverage report.

IV. Good to know 💡

- Provide a clear and comprehensive system architecture (with diagrams).
- Be ready to communicate and defend your design decisions.
- Scale is essential as the company is very successful.
- Clearly show how testing has been implemented.
- Be ready to present the demo of the solution