Fullstack Developer Assignment

Introduction

Congratulations, you made it to the technical interview for the Fullstack Developer position @Eversports 🥳

Context

You are working in the product team at eversports that is maintaining the SaaS software Eversports Manager. You and your team are working on a bunch of features around customer memberships within the current quarter.

The team also started an initiative to modernize the codebase by refactoring features implemented in an old technology stack to a more modern one.

Assignment

Task 1: Modernisation of membership codebase (backend only)

Your task is to refactor the two endpoints implemented in the **legacy codebase** that can be used to list and create memberships:

GET /legacy/memberships (`src/legacy/routes/membership.routes.js`)
POST /legacy/memberships (`src/legacy/routes/membership.routes.js`)

Your new implementation should be accessible through new endpoints in the **modern codebase** that are already prepared:

GET /memberships (src/modern/routes/membership.routes.ts)
POST /memberships (src/modern/routes/membership.routes.ts)

When refactoring, you should consider the following aspects:

- The response from the endpoints should be exactly the same. Use the same error messages that are used in the legacy implementation.
- You write readable and maintainable code
- You use Typescript instead of Javascript to enable type safety.
- Your code is separated based on concerns.

Assumptions and Decisions

- You are expected to make assumptions and decisions to make progress during the implementation
- Document all assumptions and decisions you think are worth mentioning with co-located code comments or if longer within a README file.

Task 2: Design an architecture to provide a membership export (conception only)

The team discovered that users are interested in **exporting all of their membership** from the system to run their own analysis once a month as a **CSV file**. Because the creation of the export file would take some seconds, the team decided to go for an **asynchronous process** for creating the file and sending it via email. The process will be triggered by an API call of the user.

Your task is to **map out a diagram** that visualizes the asynchronous process from receiving the request to sending the export file to the user. This diagram should include all **software / infrastructure components** that will be needed to make the process as stable and scalable as possible.

Because the team has other things to work on too, the conception of the solution (your task) is timeboxed to **1 hour** and you should share the architecture diagram as a **PDF file**.

Assumptions and Decisions

- You are expected to make assumptions and decisions to plan the best possible solution
- Document all assumptions and decisions you think are worth mentioning alongside the diagram document in the PDF.

Deliverables

- **Code Implementation (Task 1)**
  - Use the provided repository to implement your solutions by using Typescript as the programming language.
  - Ensure clean, readable, and well-documented code.
  - Cover your code changes by automated tests if you see fit.
- **PDF document (Task 2)**
  - Create a diagram of the asynchronous process using the tool of your choice
  - Consider all architecture and infrastructure components necessary and how they communicate with each other.
  - You can include any service of any cloud infrastructure provider (AWS, Google cloud function, Azure,..) in your conception.
- **README File**
  - Provide a README file explaining all decisions you believe are worth mentioning and all assumptions you took during the implementation, as well as a rough estimation of how much time you spent on the assignment.

Good luck! If you have any questions or need further clarification, feel free to reach out.