Efficy RnD fullstack coding assignment

Description

The use-case is a company-wide steps leaderboard application for teams of employees: picture that all employees are grouped into teams and issued step counters. This application needs to receive and store step count increments for each team, and calculate sums.

Focus on showing craftsmanship and code quality over full functionality.

User Stories

The API should expose methods that supports following User Stories:

1.

> As a User
> I want to be able to create a new counter
> So that steps can be accumulated for a team of one or multiple employees

2.

> As a User
> I want to be able to increment the value of a stored counter
> So that I can get steps counted towards my team's score

3.

> As a User
> I want to get the current total steps taken by a team
> So that I can see how much that team have walked in total

4.

> As a User
> I want to list all teams and see their step counts
> So that I can compare my team with the others

5.

> As a User
> I want to list all counters in a team
> So that I can see how much each team member have walked

6.

> As a User
> I want to be able to add/delete teams
> So that I can manage teams

7.

> As a User
> I want to be able to add/delete counters
> So that I can manage team member's counters

Backend

Your app should be implemented as a REST API designed to be used by multiple clients in parallel. For backend part we ask you to usu .NET (C#).

Remarks

- Make the API nice to use for a hypothetical client developer.
- Counter states only have to be stored during the application lifetime and can be forgotten on shutdown (you don't need to implement persistent storage layer).
- Bonus points:
  - nice README and Open API/Swagger specifications
  - if you can host a running app somewhere (AWS/Azure) where we can play around with it

Additional questions

You do not need to actually implement support for the below items, just have an idea for how the app would be changed to support each one. We will discuss them during a subsequent code review session.

Persistence

- How would you add a persistent storage layer such that the app could be restarted without losing counter states?
- What storage technology would you suggest?

Fault tolerance

- How would you design the app in order to make the functionality be available even if some parts of the underlying hardware systems were to fail?

Scalability

- How would you design the app in order to ensure that it wouldn't slow down or fail if usage increased by many orders of magnitude? what if this turned into a global contest with 10x, 100x, 1000x the number of teams and traffic?
- Does your choice of persistence layer change in this scenario?

Authentication

- How would you ensure that only authorised users can submit and/or retrieve data?
- How would you then add support to allow different users to only update specific counters? Or perform only specific operations?

Good luck and have fun!