Tempo One
Backend Technical Exercise

Thank you for your interest in Tempo, and for investing your time by doing this exercise. It is an important tool for understanding your level as a developer.

There is no expected time for you to complete and submit this. You'll have the opportunity to explain the choices you've made and discuss what you would have done if you had more time.

Please follow the instructions below. They are vague to allow for your interpretation. You may contact us via the contact info provided to you if you want any clarification.

Assignment

Introduction

Take a look at the following endpoints for the Teams REST service:

- Users API https://cgjresszgg.execute-api.eu-west-1.amazonaws.com/users/
- Teams API: Uhttps://cgjresszgg.execute-api.eu-west-1.amazonaws.com/teams/

For further information about each entity, append the ID of a user or team, e.g.

- https://cgjresszgg.execute-api.eu-west-1.amazonaws.com/users/fd282131-d8aa-4819-b0c8-d9e0bfb1b75c

- https://cgjresszgg.execute-api.eu-west-1.amazonaws.com/teams/7676a4bf-adfe-415c-941b-1739af07039b

This service provides a list of users. Each user can be part of zero or more teams. Each team has one user as a team lead.

Main task

You cannot modify the Users or Teams APIs.

Enrich this service by implementing Team Roles, and the ability to associate team members with them.

At minimum three roles should be pre-defined: "Developer", "Product Owner", and "Tester".

"Developer" should be the default role.

Must have

- README.md :
  - Describe how you approach the problem and/or solution
  - Provide information on how to run the code
- Unit Tests
- Kotlin code, delivered via a private github, url or archive file
- If you could improve/modify the Users or Teams APIs, what would you ask their developers for to aid you with your task? Provide your requests to the developers.

Recommended

- docker and docker-compose, the assignment should run with no manual setup

- Validation, edge cases (including if data you use is deleted)
- REST API tests
- Think about architecture considerations, as if the application was big. Avoid logic where it does not belong

Extra points

Extra points if the user of this service is able to do the following through REST:

- Look up a role for a membership, where membership is defined by a user id and a team id
- Look up memberships for a role
- Create a new role
- Assign a role to a member
- A React web app written in TypeScript that uses the APIs implemented