Take Home Test – Employee Hierarchy

Set 2 - Test duration: 2 weeks

Assignment

Build an application storing employees with their tree hierarchy. An employee may have a manager to report. An employee may have direct report(s). An employee having manager may not have any direct report. An employee not having manager need to have direct report(s). An employee not having any direct report, need to have a manager.

This application should let user to search for an employee and display his/her managers up to the root of the tree.

Example

Please refers to the attached `correct-employees.json` file for employees hierarchy example. The application should be able to store this employees hierarchy. The end user should be able to search for an employee and display his/her managers up to the root of the tree. E.g.,

1. When the user searches for “Evelina”, then the application should be able to show “Evelina”, and display her managers up to the root of the tree, which are: "Eveleen", "Kacie", and "Raelynn".
2. When the user searches for “Martin”, the application should be able to handle it. For example, displaying a not found message. (Find other edge cases for bonus points)

Please also refers to the attached `faulty-employees.json` file for another example. In this hierarchy, “Keane” and “Kylee” don’t have any hierarchy. The application should be able to handle it and display an error message such as “Unable to process employeee hierarchy. Keane, Kylee not having hierarchy”. (Find other edge cases for bonus points)

Another example can be seen from `another-faulty-employees.json` file. In this hierarchy, “Linton” has multiple managers. The application should be able to handle it and display an error message such as “Unable to process employee tree. Linton has multiple managers: Fletcher, Tabitha”.

Mandatory Specifications

1. The application can be written in any OOP based language such as, but not limited to: C#, Java, JavaScript, Typescript, C++, python.
2. The application can be either frontend or backend, based on your expertise. User interaction is also up to you. E.g. It can be a REST API application, or it can be a CLI, or it can be a web app, etc. Surprise us.
3. The employee hierarchy structure needs to be **in a Tree structure**.
4. The application needs to store employees in memory (**no database**).
5. The aim of this test is to create the code as elegant, readable, modular, maintainable, and testable as possible.
6. Another aim for this test is to check the candidate’s defensive programming skill. The more edge cases handled, even if not specified in this document, the better.
7. The submitted assignment should include:
    a. The code itself,
    b. Instruction how to build and run the application,

    c. Instruction how to run the unit tests (if any),
    d. Providing Demo video / screenshots of the application and unit tests coverage are a plus (optional, helpful). The easier it is for the reviewer to see the application running, the more the submission will stand out.
8. It is up to the candidate how to send this assignment. Sending via email might be blocked by the email provider due to security reasons regarding uploading a code file or executable. Candidate can share the zip file via google drive (shared link), or the code itself via github **private repo**, or gitlab **private repo**, etc.
9. The deadline is in 2 weeks, but the sooner you can submit the take home test, the better. If you can submit the take home test way before the deadline and fulfilled all of the mandatory specifications, then we can have a feedback session, which can boost your take home test to stand out more.

Optional Specifications- Bonus Points

1. Providing unit tests is a plus. More coverage is the better.
2. Adding github coverage is a plus (optional).
3. Using a docker container is a plus. This can also make it easier for the reviewer to run the application. Which can make the submission stand out more.

Assessment

- Code originality
- OOP implementation and structure
- Tree structure
- Code readability, modularity, maintainability, and testability
- Unit tests coverage (If any)
- Defensive programming on unspecified edge cases
- Clarity of documentation. The easier it is for the reviewer to run and test the application, the more the submission will stand out.
- Creativity. Surprise us. If the application can be made better, then feel free to upgrade it. As long as the application intention is still the same as described above; which is to search for an employee and display his/her managers up to the root of the tree.