Indeavor - Software Engineer - Code Challenge

Introduction

Candidates for the Software Engineer role should be able to comprehend, analyze and deliver on the requirements specified in the scenario described below. They should approach this challenge as if they were part of the Indeavor engineering team.

Description

The company wants to release a new Employee Management SaaS product to disrupt the Workforce Management industry, so the engineering teams are asked to design and implement a new solution that is following modern standards and is cloud friendly. The product has to be easily maintainable and should be simple to extend since the revenue stream requires periodic feature updates which aim to increase value for existing customers and attract new ones.

Requirements

The product team has come up with a list of requirements and use cases that the minimum viable product should satisfy. The suggested solution is targeting a company with one or more departments, which needs a way to manage their employees. The company requires employees with specific skills, so the schedulers need a way to manage and administer skills as well.

The application should expose a REST API to allow different kinds of applications to access and modify the company's data.

Companies buying the product are going to be accessing their data through either a web browser or mobile devices. Different engineering teams will be responsible for developing the web and mobile clients, but all engineers will be asked to participate in the development of the REST API.

The team building the Web interface must ensure that modern browsers behave well while rendering the application's pages. The team is free to choose which framework to use to build the web app.

The team building the Mobile app must ensure that the app is offering a friendly and easy to navigate UI where the UX is allowing the user to easily discover what they can do. They are free to choose a cross-platform framework.

The Minimum Viable Product (MVP) must include a REST API and at least one of the desired UI clients.

Skills (MVP must cover this requirement)

Assuming a scheduler has access to the system's resources, they must be able to manage the

skills the company requires employees to have. The product needs to expose pages to:

1. **View skills (Page A)**
    - Show a collection of the available skills required for any job in the company.
    - Navigate to a specific skill's page.
2. **View an existing skill's details (Page B)**
    - View the skill's name and the date of its creation.
    - Update the skill's details.
    - Delete the skill.
3. **Create a new skill (Page C)**
    - Fill a skill form with the skill's name and a description.

Employees (MVP must cover this requirement)

Also, a scheduler must be able to manage the companies' employees, their personal details and their skillsets. The product needs to expose pages to:

1. **View employees (Page D)**
    - Show a collection of the available employees.
    - Navigate to a specific employee's page.
    - Order employees by their surnames with either an ascending or descending order.
    - Order employees by their hiring date with either an ascending or descending order. (optional)
    - Delete multiple employees (optional).
    - Search for employees using their name and/or surname (optional).
2. **View employee details for an employee (Page E)**
    - View the employee's personal details.
    - View the employee's skillset.
    - Update the employee's personal details.
    - Add an skill to the employee's skillset.
    - Delete an assigned skill from the employee's skillset.
    - Delete the employee.
    - View details like when the employee's latest skillset change occurred (optional).
3. **Create a new employee (Page F)**
    - Fill an employee form with personal details and create a new employee.
    - Assign existing skills to the new employee as part of the creation form (optional).
    - Assign non-existing skills as part of the employee creation (optional).

Nice-to-have features and functionality

The product team has also shared optional requirements for extra usability and productivity enhancements that would be nice to have as part of the minimum viable product.

- Since many employees share the same skill(s), it would be nice if a scheduler could filter employees by a skill - or multiple skills - from Page D. The team is free to extend search capabilities as they see fit.
- Schedulers need to audit employee data changes, so access history logs would be a feature that adds value to the product.
- Many schedulers are comfortable with spreadsheets, so it would expedite their work if they could export all skills in a spreadsheet or similar format.
- The Web app's interface should be able to display well in different resolutions and screen sizes for desktop and tablet devices (applies to the team building the Web app).
- Schedulers are using an external system as the record keeping system for employee demographics. It would be a well received feature if the new product supports importing employees from the external system. This can be a scheduled operation or manually triggered by a scheduler. The team is free to decide the integration format. In any case, the process should impose functional limits which make sense and it should never abuse the system's resources.
- Any addition that improves the product's quality based on the engineering teams' experience.

Delivery

Candidates should address all requirements as described in the Requirements section in whatever way they see fit and must include a small report/documentation of their solution. Candidates who choose to deliver optional requirements, should try to tackle those in a way that helps further demonstrate their skills and expertise.

The frameworks below are proposed as recommendations but candidates are free to use the frameworks of their choice.

- `ASP.NET Core MVC` or similar server-side libraries (backend)
- `MS SQL Server` or similar relational database engine (backend)
- `MongoDb` or similar NoSQL engine (backend)
- `ReactJS` or similar client-side libraries (web frontend)
- `Xamarin.Forms` or similar cross-platform frameworks (mobile frontend)

Candidates should use `git`, locally, make well thought structured commits there and must send the working directory, including the `.git` directory, via email.

> The test is confidential, including the submitted solution, and should not be shared publicly in any form.

Time Frame

Candidates should send their solution as late as 5 working days after they accept the challenge.

Epilogue

Answers and deliverables should strive to showcase the candidate's train of thought and their technical skillset to design, architect and implement a solution.

The Indeavor Engineering Team