Fullstack Assessment – Candidate Instructions

Expectations

- Demonstrate a solid and robust architecture.
- Highly maintainable and legible code.
- Use and knowledge of design patterns.
- Follow software development best practices.
- Effective and consistent naming conventions.
- Code crafted with modern development methodologies.
- If you choose to simplify your code in the name of speed (i.e. "prototype" style development) be able to explain the shortcuts you took and why
- Smart UI patterns are not allowed.
- Code snippets from old projects and/or opening any old projects are strictly forbidden.
- Prioritize good code design and architecture over development speed.
- Read the instructions carefully as only complete and working solutions will be reviewed.
- Your code should compile and run with us doing a simple package restore and nothing more.

Submitting your code

- Go to git.number8.com and register or login if you already have an account.
    - Please create your account following this naming convention: firstName.lastName
    - You may also add your second name and/or second last name if the user already exists.
- Create a repository to work on and do constant commits as you would normally do. Please use the assessment name in the repo name. Ex: fullstack-assessment
    - You may create your solution on a mono repo or having a repo for frontend and another for backend. You just need to identify your repositories accordingly.
- You have no time limit to complete the assessment.
- Once you are ready, add a brief explanation of your solution and architecture to the README file and share your git repository URL with your recruiter.

Simple Employee Maintenance web app

First Task (Database):
Create a database to store the following information (you may use any database of your preference, unless otherwise specified by your recruiter):
1. First Name
2. Last Name
3. Hire Date
4. Department
5. Phone
6. Address

Create all needed tables and relationships. Field data types are expected to be correct.

Create the structure using migrations provided by the ORM of choice. If you prefer to avoid migrations you should use a Database Tool IDE like DBeaver, Phpmyadmin, Toad DBM or Robomongo, and include the scripts on the repo to create the database.

As soon as you finish creating the database, generate or create a diagram for the database structure.

Second Task (Programming Exercise):
Create a Web Service that exposes the following endpoints – You may use the technology of your preference, unless otherwise specified by your recruiter:
1. For Employee Entity: * Should include the employee department in text form.
   a. CreateEmployee
   b. GetAllEmployees
   c. GetEmployeeById
   d. UpdateEmployee
   e. DeleteEmployee
2. For Department Entity
   a. GetAllDepartments

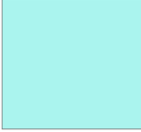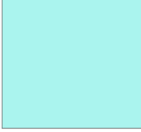**Third Task (Employee List page):**

Create a new page to show a list of employees. You may use the framework/library of your preference unless otherwise specified by your recruiter.

You can use the following mockup as reference or create your own design if you prefer.
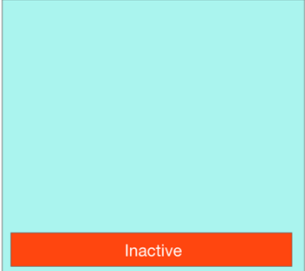
This page should show the following information:

- Employee Name, Department, Hire Date, Employee Avatar
- It should display the date in the following format: May 2, 2021 (2y – 1m – 4d)
  - The date in parenthesis determines how long that employee has been in the company.
- A "View Details" button that will take the user to the following page.
- A "New Employee" button that will show a modal with a form to create a new employee.
- A red "X" button that will delete the selected employee.

New Employee

Employee Name (Department)

Hire Date
May 2, 2021 (2y - 1m - 4d)

View Details   X

Employee Name (Department)

Hire Date
May 2, 2021 (2y - 1m - 4d)

View Details   X

Employee Name (Department)

Hire Date
May 2, 2021 (2y - 1m - 4d)

View Details   X

Employee Name (Department)

Hire Date
May 2, 2021 (2y - 1m - 4d)

View Details   X

Fourth Task (Add employee details view):

Employee Name

Employee ID:
Department:
Telephone:
Address:

Hire Date
May 2, 2021
2y - 1m - 4d

Deactivate

Inactive

Update Department
Department Name ▽   Update

Department History

Date	Department
03/22/2021	Department Name
02/15/2021	Department Name
12/26/2020	Department Name
09/13/2020	Department Name
06/05/2020	Department Name
04/19/2020	Department Name
02/28/2020	Department Name

1. Create an additional page or modal where the user can see the details of the selected employee.
   a. This will be displayed after the user clicks on the "View details" button from the previous task.
2. You can use the mockup above as a guide or create your own design if you prefer.
3. This new section should have a dropdown with all the departments available, a button with text "Update" and the following behavior:
   a. The dropdown should have the employee's current department selected by default.
   b. The button will be green if a change has been made to the dropdown, if no change has been made, the button should be disabled.
   c. When the user clicks on the "Update" button, it should update the employee's department and update the UI to reflect this change.
4. Add a "Deactivate" button that will show an "Inactive" label on top of the employee's avatar. It will have the following behavior:
   a. When the selected employee is ACTIVE:
      1. The employee's avatar won't have any labels.
      2. The button will be RED with text "Deactivate".
   b. When the selected employee is INACTIVE
      1. The employee's avatar will have a label with text "Inactive".
      2. The button will be GREEN with text "Activate".
   c. When the button is clicked it will update the employee's status on the database and reflect the changes on the UI
Bonus:
5. Add a history table to show all the Department changes from the selected employee. This should be updated automatically after the Employee's department is updated.