Position Management System Exercise

Overview

This exercise evaluates a candidate's ability to build a full stack application using the specified technologies. The candidate will create a job positions management system with CRUD operations.

Technical Requirements

- Backend (.NET 8 REST API).
    - CQRS with MediatR: Implement commands and queries for all operations.
    - Logging: write an informative logging message in every command/query. Use the dependency ILogger<T> from Microsoft.Extensions.Logging.
    - Fluent Validations: Validate all incoming requests and produce comprehensive error messages for invalid requests.
    - API Key & Middleware implementation: secure your API by implementing a middleware that rejects any request that does not include an HTTP header 'X-API-KEY' with the value 'da1d2d28-92ed-4a2e-a1a9-10fcf4425f15'.
    - Implement a MediatR behaviour to log the duration of all requests.
    - Entity Framework: Use EF Core with UseInMemoryDatabase for data persistence.
    - Basic unit tests (at least 1 for backend and frontend).
- Frontend (Angular & TypeScript)
    - Reactive forms: For create/update operations
    - RESTful design: implement a RESTFul API with proper endpoints and HTTP methods/responses.

Exercise Details

The application will have to include the following features:

- Create new job positions
- View list of positions
- View position details
- Update existing positions
- Delete positions

Data Model

- Title (required, max 100 chars)
- Description (required, max 1000 chars)
- Location (required)
- Status (draft/open/closed/archived)
- RecruiterId (required)
- DepartmentId (required)
- Budget (required)
- Closing date (optional)
- Status (Open/Closed)

Consider using refence tables for related entities such as Recruiter and Department.

Important: Provide a migration with seed data so that the reviewer can quickly populate data.

API Specification

Endpoints

GET	/api/positions	Lists all positions
GET	/api/positions/{id}	Gets position details
POST	/api/positions	Creates new position
PUT	/api/positions/{id}	Updates a position
DELETE	/api/positions/{id}	Delete position

Error handling

Consider returning the correct HTTP status code in case of errors:

- 400 for invalid requests (e.g. negative budget)
- 404 for non-existing ID.
- 401 for requests without API key
- 403 for requests with invalid API key

Evaluation Criteria

Backend Implementation

- Proper use of CQRS with MediatR
- Effective validation with FluentValidation
- Correct use of EF Core with in-memory provider
- Appropriate middleware implementation
- Usage of MediatR behaviours
- Proper handling of in-memory data lifecycle

Frontend Implementation

- Clean component architecture
- TypeScript best practices
- Effective form handling
- Good API service integration

Overall

- Code organization
- Git commit history

Deliverables

1. Create a new GitHub repository for your project.
2. Ensure the repository includes:
    a. All necessary source code files.
    b. A README.md with clear setup instructions (e.g., dependencies, how to run the project).
    c. Any required configuration files (e.g., .gitignore, appsettings.json).
3. Provide the GitHub repository URL in your submission so the reviewer can:
    a. Clone the repository (git clone <your-repo-url>).
    b. Follow your instructions to run the project locally.

Time frame

- 2 days to complete the assignment.
- Focus on creating the core functionality and demonstrating technical proficiency rather than perfect production polish.