medchart

ASP.NET & React Take-home Task

Thank you for your interest in joining the Medchart team. In this assignment we will evaluate your capabilities and potential to perform as a core member of the Medchart development team. Foremost, this assignment is designed to test your ability to learn quickly, as well as your problem solving skills and architectural decision making process. You are required to build a fully functional single page application backed by REST API to allow users to view, edit, and add blood work entries on their dashboard profile. You are encouraged to use this document as a guideline but also suggest and/or implement functionalities and features based upon best practices while demonstrating your logic and reasoning behind specific architectural choices.

Objectives

- Evaluate the capabilities of building a functional REST API
- Evaluate the capabilities of building a functional Single Page Application
- Evaluate the capabilities of developing a real piece of software using the Medchart stack

Requirements

1. The Back-end (REST API) must be built using C# and .NET Framework. You may consider additional frameworks such as MVC or WebAPI.
2. The REST API must follow most of the principles of the Representational State Transfer (https://en.wikipedia.org/wiki/Representational_state_transfer).
3. The REST API requests and responses must be in JSON format.
4. The API can store the data on any kind of database or file.
5. The Single Page Application's HTML, CSS and JS must be served from the same server as the API.
6. No authentication or "multi-user" is required although desirable.
7. The Single Page App must be responsive (use bootstrap).
8. All React code must use TypeScript

Use cases

List all blood works (see Appendix 1)

1. The user sees a list off all blood works
2. The list must show created date, exam data and description
3. The list must have a details button which opens all the details of the selected blood work
4. The user can search information on the list

Create new blood work entry

1. The user fills the data (Appendix 1.) in a form
2. The fields are validated when the user leaves the field
3. If no error on the form the information is saved
    A. The user is redirected the list page

Edit blood work

1. The user sees all the data filled and can change this data
2. The fields are validated when the user leaves the field
3. If no error on the form the information is updated
    A. The user is redirected the list page

medchart

Blood work report

1. The user should have a one-line graph for each information (Appendix 1.)
2. The X axis of the Graph is the date
3. The Y axis of the Graph is the value for that information x date

View blood work

1. The user must be able to see all blood work details (Appendix 1.)
2. The user can click on edit to edit the blood work data

Resources

All resources bellow might be required depending on the platform and software you have installed.

- Visual Studio Community Edition: https://visualstudio.com/vs/community/
- SQL Server Express: https://www.microsoft.com/en-ca/sql-server/sql-server-editions-express/
- Windows (Virtual machine file): https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/
- TypeScript: https://www.typescriptlang.org/index.html#download-links
- Node.js: https://nodejs.org/en/download/

Deliverables

ZIP file with source code.

Database scripts (if required).

Setup instructions (if required).

Evaluation criteria

1. Readability
2. Logic clarity
3. Code smell analysis
4. Static code analysis
5. Bugs found
6. Performance
7. Requirements fulfilled
8. Use cases fulfilled
9. Overall completion of assigned tasks

Appendix 1.

The blood work contains the following information:

1. Date Created
2. Exam Date
3. Results Date
4. Description
5. Hemoglobin (Number)
6. Hematocrit (Number)
7. White Blood Cell Count (Number)
8. Red Blood Cell Count (Number)