C# Development Interview Project

1 Introduction

Ah, welcome to the world of business. You've just struck it out on your own and clients are filing in everywhere! This is great news! However, you've started to notice that all your clients have a single thing in common; They all need an inventory management system.

In the past, you have built these systems individually tailored to each of your clients. Unfortunately, you don't have the time to continuously be re-building and refactoring code for all the new business you have coming in. You now must make a decision…

1. Continue building customized solutions for each client, which may lead to turning customers down.
2. Rethink the architecture of your System and build a re-usable inventory system that meets the needs of most of your new clients and old.

Being the savvy developer-businessman you are, it only seems reasonable to build a more robust underlying architecture from the start. So you decide #2 is the way to go. Now it's only a matter of getting it built.

2 Goals

After a thorough analysis of your customers' needs, you have determined that the following items are what they need most:

1. The ability to add products to the System with arbitrary metadata. In some cases, customers need SKU(s), in others, they need Color, Brand, and other details.
2. The ability to search for products utilizing the metadata
3. The ability to categorize and create hierarchies of products for simple sorting on various UI frontends.
4. The ability to add and remove products from inventory, with the ability to "undo" transactions easily.

With these goals in mind, it is time to get the requirements started!

3 Requirements

After some thought, you determine the following, simple requirements for the System.

1. Products can be added to the System but never deleted
2. Products must allow arbitrary amounts of metadata and categories
3. Products should be searchable by the metadata and 1 or more categories
4. Adding and Removing inventory should happen on an individual product level or multiple products at once.
5. Inventory Counts for a specific product, or subset of metadata on a product, must be retrievable
6. Individual transactions should be able to be removed

7. The System should be API-driven

4 Assignment

4.1 Basics

A set of libraries, a sample MVC ASP.NET Core C# Project, and a database design have been provided. Although not all tables may be necessary, they should be utilized to meet the goals and requirements. The same applies to the libraries.

The basic set of functions should be accessible from the API:

1. Adding a Product with all necessary details: Categories, Metadata, and General Details
2. Searching for Products by Category, Metadata, General Details, or any combination thereof
3. Adding products into Inventory
4. Removing products from Inventory
5. Retrieving a Count of Product Inventory based on a unique product identifier or metadata on the products

There should be no need to add new tables into the database to achieve this set of goals. However, if a better solution is possible, please create said solution but include transformation scripts, as necessary.

4.2 Download the Project

You can download or Fork the project from GitHub: https://github.com/Jcouls29/Development-Project-CSharp

4.3 Submission

You can submit the project in following way:

> Submit a GitHub Pull Request into the master branch (https://github.com/Jcouls29/Development-Project-CSharp)

Note that any PR(s) submitted will ultimately be rejected.

4.4 Grading

It is NOT expected that all requirements be fulfilled. If only 1 and 2 can be achieved within a 2-hour period, but the code meets the general grading metrics below, the submission will be considered successful.

Below are some points that will determine the quality of the submission:

- Readability of the Code
- Use of Interfaces and Dependency Injection
- Organization of abstractions, implementations, models, etc into separate folders, assemblies (libraries).
- The ease of locating aspects of the solution within the Visual Studio interface (i.e. Solution Explorer)
- The ability to generalize and re-use aspects of the coding framework
- The use of naming conventions, configurability, and extension methods to easily implement solution into multiple front-end API(s)

- How configurable is the System?
- Does it build and run without issue?
- Is the System tested via automated testing (unit or integration)?
- The readability and organization of SQL within the System
- The robustness of the code to allow extensions via the open-closed principle
- How well the code is commented for uncommon syntax or clever use cases.
- The use of asynchronous methods and proper library implementation for use by synchronous code
- Which, if any, 3rd-party libraries are used and for what reasons.

4.5 Recommendations

Here are some recommendations:

- Write comments specifically for evaluation of the project and prefix the comment with “EVAL:”
- Take into consideration any edge cases where data may need to be validated first (i.e. too large of integers or string overruns)
- Consider how you might extend the System or alter the API(s) for new customer requirements, that may have not been foreseen, without impacting older customers.
- Metadata on products and categories can be pre-determined, and do not need to be “dynamically” added to the System. Instead, come up with a handful of extra attributes for products and/or categories to be consistent (i.e. SKU(s), Color, Length, Package Unit, etc)
- Focus only on what you consider the highest priority items with the highest return on investment (ROI) first. This will help showcase your skills while also staying within the 2-hour window.
- Re-use as much of the code already within the System. For instance, ISqlExecutor should be utilized for SQL. Consider using Dapper, or a similar library, for easier execution.
- Utilize well-vetted 3rd-party libraries rather than obscure ones. If they are used, understand the advantages to the System more than just whether they take work off yourself.