

Inventory Search – Code Test

Generate an Angular App to create a cleaner looking version of this.

Inventory Search

Criteria

AB

Search By

Part Number

Branches (comma separated)

e.g. SEA,PDX

☐ Only Available

Search

Prev

Page 2 / 4

Next

Code Test Brief

Basic outlines are given for both to help as a starting point

Looking to get a well designed solution.

Looking to see what your design choices are for the UI with regards to look and feel. The example given above is an ugly UI looking for something much cleaner with a professional look to it.

You are allowed to use any tools to help you complete this task expectation is to be completely open about what you used and why and how you used it.

Requirements

- Use Angular 18 preferred
 - HttpClient for API calls

- Strong typing via TypeScript interfaces
 - Reactive forms
- Basic Structure to the code is supplied in the repository with lots on hint comments in the code
- Behavior:
 - Show a loading indicator while searching
 - Cancel in-flight requests if a new search is fired
 - Display errors appropriately
 - Cache the most recent 5 unique searches for 60 seconds (avoid duplicate API calls within TTL)
- Acceptance criteria:
 - Functional form with validation, loading state, cancel-on-new-search
 - Results with sorting, pagination, and expandable rows
 - Caching prevents duplicate API calls within 60 seconds for last 5 distinct queries
- Deliverables:
 - Source code for the feature (Angular TypeScript, HTML, SCSS) so that it can be built then run against the mock inventory server
 - C# files required to build the Server with Service layer unit tests for the C# project
 - Quick start instructions (README)

Inventory-mock-api is the server supplied to allow you to test your angular code. Note this code along with the json returned should be sufficient allow you to complete the C# (Note there may or may not be some missing fields to the supplied C# models)

Inventory-Search contains a starting point for the angular code

InventoryServer contains a starting point for the C# .NET project. The repository layer is an area where you can get creative, if you so desire, as to how you want to generate the data. The mock javascript code randomizes the data you can follow that mechanism if you want or you can do something completely different.

Please be creative what I want to see is well structured easily readable, understandable code.

You are free to modify the API if you deem it appropriate. If doing so please give reasons as to why this is a better solution.

There is no limit on how you choose to develop the solution, I would like a reasoning as to the why and how for using any tools. As software developers our day to day tasks often require the use of these tools this code test should be treated no differently.