The Treasure Hunter's Arrangements

Treasure Hunters all over the world routinely attempt to conquer the most challenging mazes in search for riches. But modern dungeon crawling has become tougher. What used to be a spontaneous spelunk now places higher demands on both physique and gear.

What is needed is a marvel of technical brilliance to manage and prepare adventurers before departure. If only there was a basic inventory management system, and possibly a lightweight eCommerce solution.

Help, the Treasure Hunters shout; *be calm*, you respond!

Mission

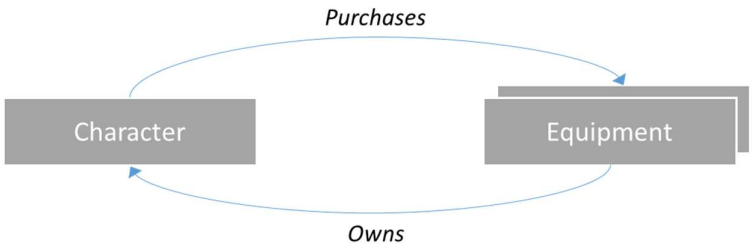
Create a visually appealing, web based equipment store which allows Treasure Hunters to view and purchase gear before heading out for treasure-seeking missions. Occasionally, this will be done on the go, so portability is an important consideration.

Treasure Hunters are many things, but good memory they have not. Therefore, it is ok that the shopping application restarts each time. They will not remember what they bought anyway. Simple creatures as they are, a crisp and clear user experience carries more weight.

Provided for you is a basic boilerplate Vue.js project with TypeScript, along with a simple ASP.NET back-end in C#, which provides most of the endpoints you'll need. These are not meant to constrain you, but to allow you to make spend more time on development and less on plumping and configuration.

The set-up provided for you requires you to run two web-servers, one for ASP.NET and one for the Vue.js project.

Outside of the minimum list of functional requirements below, your creativity is much appreciated! (Product owners can't think of *everything* by themselves, now can they)



Constraints

The Treasure Hunter has had plenty of experience with poorly prepared missions, and was able to quickly whip up some must-haves:

- **Application** is web-based and gets its content updated without page reloads.
- **Application** keeps its state throughout a server session, but does not persist anything.
- **Application** contains two main views:
    - **Character inventory** – showing character *identity*, *attributes* and *equipment*
    - **Equipment shop** – allowing the user to *shop for more equipment*

- **Application** starts by randomizing a new character with a *name* + some *gold coins*, as well as a store of available equipment to purchase.
- **Character attributes** include**:**
    - Hit points
    - Luck
- **Equipment** has a *name* and one of three visually distinct *types*:
    - Armor
    - Weapon
    - Trinket
- Each piece of **equipment** may modify a **character attribute**.
- Once **equipment** is purchased, it is assigned to the character inventory and applies its modifiers to the character.
- **Equipment shop** lists all available equipment, their properties and allows the user to purchase items.
    - If the character cannot afford a purchase, this is in some way indicated
    - An item that has already been purchased cannot be purchased again

Some of the must-have constraints are already provided for you in the server API, to allow you to focus more on front-end development. Again, this is to guide you - you may remove some or all of the code provided.

Technicalities

Develop your solution with the following technical constraints:

- Web application works well on desktop and (modern) mobile browsers
- For server/backend code: .Net Framework or .Net Core implemented with C#
- Frontend: Use the provided .vue.js project

Compiled output is a web application.

Delivery

Deliver the solution in the form of a zipped folder. Do *not* send it directly through e-mail (our spam filter will intercept it), send us a download link instead. A public Git repository is also OK.