# Front-end developer challenge

## Introduction

Your task would be to extend the existing application with a new feature. You have to study the code, add new parts, and possibly modify some existing parts.

You will need npm and an editor like Visual Studio Code to complete the challenge.

Be aware that some parts of the code are built according to the best practices and some are not – moreover, there are some deliberate bugs. If you find them, try to fix them.

## Running the project

1. Navigate to the src folder
2. Run “npm install”
3. Run “npm start”
4. Navigate to <http://localhost:3000> in the browser

## Pre-built part

There is a web application with a pre authenticated user.

Each user has an amount of gold coins, which is shown somewhere in the top bar.

There is a stock of items to offer, with a price in gold coins and an available amount.

E.g. 3 swords costing 35 each and 2 helmets costing 12 each and so on.

For the simplicity of the setup, pre-built storage is in-memory (i.e. some static collections) and the requests to the “backend” are simulated (see service.js)

## Part to be built

### Functional requirements

Make “Buy” button in the top bar show the dialog which will allow user to buy available items. The dialog shows the following elements:

[image] [description] [-] [amount to buy input] [+] [price]

......

total: [xxx]

[buy] [cancel]

User can see available items and their price, can type in an amount to buy (integer value), or increment/decrement it with buttons + and -. Total cost is calculated in real-time and reflects changes to the amount.

If the total cost exceeds available money, message appears and buy button is disabled.

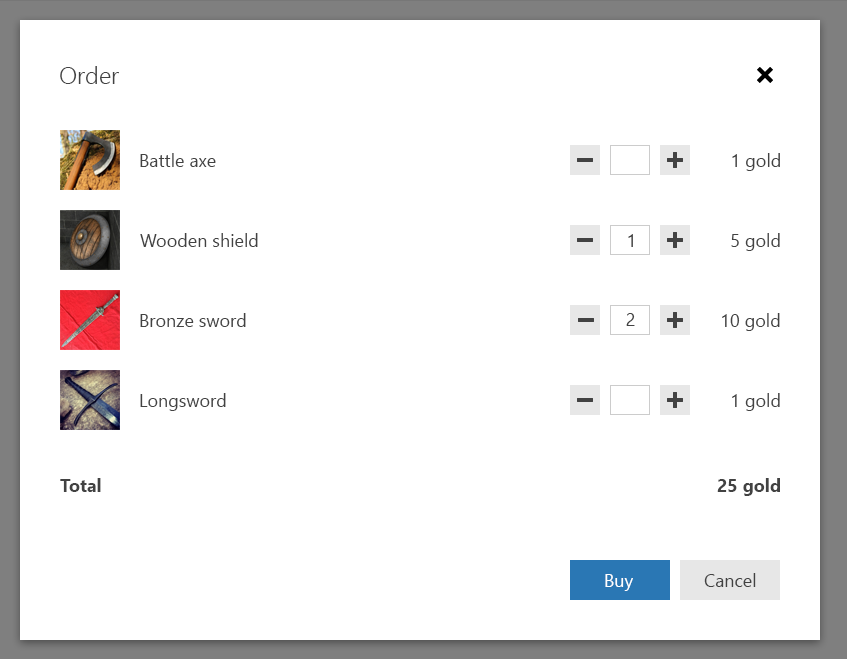
When user clicks "buy", dialog is closed, item stocks are updated and user balance is updated. There is no need to store "bought" items.

User can also close the dialog via cancel or close buttons.

Make sure constraints with regard to number of items available and user balance are not violated. Make sure feedback is available to users if they cannot accomplish the action (e.g. item is no longer available or a request to the backend fails).

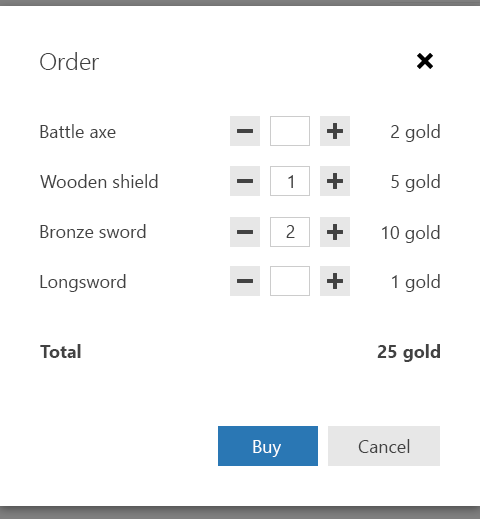
### Design

If screen width is larger than 480 pixels, the following dialog is shown:



The dialog is centered horizontally and vertically on the screen.

If screen width is less or equal to 480 px, the layout changes a bit:



If item name doesn’t fit the available space, use ellipsis.

Check the detailed design guidelines, including sizes and spacings in docs\Design folder. All required images are also present in the solution.

## Implementation requirements

Existing features should continue to function.

The requested design (and styling) for the Buy dialog is implemented (or as close as possible)

The domain logic described for the dialog is implemented (real-time updating, constraints/validation checks, updating of the user/stock on purchase, feedback on failure)

A “random” failure request is implemented for the “Buy” request (e.g. once every 3 requests) and the failure is handled

Browser support:

* Chrome (latest)
* Firefox (latest)

All decisions not described here or in the design mockups are up to you.

Good luck!