MVP - Functional and Technical Specifications for the Kanap Website

CONTENTS

General Architecture

The online app will be made up of 4 pages:

- A homepage which will (dynamically) display all of the articles that are available for purchase
- A “product” page which will (dynamically) display the details of the product that the user clicked on when on the homepage. From the product page users will be able to select the required quantity and colour and then add the product to the cart
- A “cart” page. This page will contain several parts:
    - A summary of the products in the cart and the total price. Users should be able to change the quantity of product or to completely remove a product from the cart
    - A form to be submitted in order to confirm the order. The data entered into this form should be correct and properly formatted before it is sent to the back end, for example there should be no numbers in the name field.
- A “confirmation” page:
    - An order confirmation message thanking the user for their order and displaying the order number sent by the API

Test plan

A plan for a series of acceptance tests to cover the entire range of features listed in this document (the functional and technical specifications for Kanap).

Here is a model for you to base your plan on: model test plan

Further information

The homepage

This page presents the entire range of products identified by the API.
For each product we need to see an image of the product, its name and the start of the product description.
When users click on the product they will be redirected to the product page so that they can have access to more information about the item.

The product page

Each product page will present just one product, the drop-down menu will allow users to choose from various options for customizing the item as well as allowing them to choose the quantity required. All selected items should then be displayed in the cart.

The cart page

On this page users will be able to modify the quantity of product in their cart and when they do so the total cost identified on the page should also be updated.
Users will also be able to completely remove a product from their cart and when they do so the product should disappear from the page.

The data entered by users should be verified and approved. You should check the format and the data type before sending this data to the API. It would not be acceptable to have a name containing numbers or email addresses without the “@” symbol, for example. If there are any issues with the data entered, error messages should be displayed beneath the field(s) in question.

The confirmation page

On this page the user’s order number should be displayed. You must ensure that this number is not stored anywhere.

The source code

The code should be indented and should use comments at the start of each function to describe their roles. The code should also be divided into several reusable (named) functions. Each function should be short and should meet a specific need. You should avoid long functions which meet several needs all at once. For example: you should not have just one function for collecting, processing and sending data.

API

As regards the API you should use promises to avoid callbacks. You may also use alternative solutions, such as fetch, as long as these include promises. The current version of the API is a preliminary version. The post request that you will have to formulate in order to confirm orders does not yet take into account the quantity or colour of the products bought.

How the cart operates

In the cart you should always display the products grouped by model and by colour.
If a product, in the same colour, is added to the cart several times then this product should appear just once but with the quantity adjusted.
If a product is added to the cart several times but in different colours then these products should appear on different rows with the colour and quantities clearly identified for each of the products.

Data types

All products have the following characteristics:

Field	Type
colors	array of string
id	string
name	string
price	number
imageUrl	string
description	string
altTxt	string

Technology used

HTML, CSS, JavaScript.

URLs of the APIs

- The sofa catalogue: http://localhost:3000/api/products

Parameters of the APIs

Each API contains 3 parameters:

Verb	Parameter	The Request to be Made	Response
GET	/	-	Sends back a table with all the elements
GET	/{product-ID} {product-ID} should be replaced by the actual ID of the product	-	Sends back the element for {product-ID}, the ID of a specific product
POST	/order	A JSON request containing a contact object and a product table	Sends back a contact object, a product table and an orderId (string)

Approving the data

For POST requests the contact object sent to the server must contain the following fields: firstName, lastName, address, city and email. The product table sent to the back end should be an array of product ID strings. The types of these fields and the presence of data in each of them must be approved before the data is sent to the server.