

## Plan de test de l'API proposé par Michel SADEU NGAKOU.

### 1- Page de vue montrant tous les articles disponibles à la vente.

#### 1.1- Tests unitaires

Ces tests sont effectués sur chaque unité de code. L'on vérifie que chaque unité marche bien, retourne ce qui est attendu et aussi que chaque fonction ait le comportement attendu.

Bout de code (page index.html)	Comportement attendu
<pre>await fetch("http://localhost:3000/api/cameras" ) await response.json().then(result) result.forEach((product)=&gt; {products.innerHTML= ...})  .catch(Error=&gt;console.error('error'))  localStorage.getItem('clé')</pre>	<div>Objets récupérés (et affichés)</div> <div>Afficher l'erreur si ça existe</div> <div>Accès à la valeur dans le localStorage</div>

### 2- Page produit qui affiche l'élément sélectionné par l'utilisateur.

#### 2.1- Tests unitaires

Bout de code (page produit.html)	Comportement attendu
<pre>Id = window.location.search.substring(4) await fetch("http://localhost:3000/api/cameras/" + Id ) await response.json().then(result) ((result)=&gt; {product.innerHTML= ...})</pre>	<div>Objet récupéré(et affiché)</div>

- **Personnalisation du produit**

Bout de code (page produit.html)	Comportement attendu
<pre>for(let i=0; i&lt;result.lenses.length; i++){ select.innerHTML+=&lt;option&gt;\${ result.lenses[i]}&lt;option&gt;  selects.addEventListener('click', ()=&gt;{if(number==0) {alert("veuillez sélectionner une quantité ") }else if (number &lt; 0){alert('choisir un nombre positif')}}})</pre>	<div>Affichage des objets d'option</div> <div>Les alertes sont bien retournées</div>

- **Calculs arithmétiques et stockage dans le panier**

Bout de code (page produit.html)	Comportement attendu
<pre>let cost= localStorage.getItem('clé');  if (cost !=null){ cost = parseInt(cost); localStorage.setItem('clé', cost+ (result.price/100)*quantity) }else { localStorage.setItem('clé', (result.price/100)*quantity);</pre>	<div>Accès à la valeur dans le localStorage</div> <div>Vrai Valeurs enregistrées dans le localStorage</div>

## 2.2- Tests d'intégration

Ce sont les tests pour vérifier que les différents modules de l'application existent, sont présents et que les interactions ne gênent pas mutuellement.

`onclick = 'location.href="produit.html?id=${product._id}"'` → renvoie la page produit avec l'objet

`onclick = 'location.href="panier.html"'` → renvoie la page panier avec tous ses composants

`fetch("http://localhost:3000/api/cameras")` → connexion avec le backend

## 3- Page panier contenant un résumé des produits et un formulaire pour passer la commande.

### 3.1- Tests unitaires

- **Vider le panier**

#### Bout de code (page panier.html)

#### Comportement attendu

```
let btnClear = document.getElementsByClassName('remove')
for(let i=0; i<btnClear.length; i++){ let btn= btnClear[i];
btn.addEventListener('click', function(event){
alert('Vous pouvez retourner vers la page d'accueil choisir des produits)
event.target.parentElement.remove()
localStorageClear()}};
```

Affiche l'alerte de retour à la page d'accueil et vide le panier

- **Vérification de l'existence des données dans le formulaire**

#### Bout de code (page panier.html)

#### Comportement attendu

```
for(let i=0; i<inputs.length; i++){
if(!inputs[i].value){ error= 'veuillez renseigner tous les champs '}
else if(inputs[i].value){ productOrder();
}
```

→ Renvoi l'erreur

→ Exécute la fonction pour commander

- **Passer la commande**

#### Bout de code (page panier.html)

#### Comportement attendu

```
await fetch("http://localhost:3000/api/cameras/order")
.then((data)=>{ let getOrderId= data.orderId
let getCartCost= cost})
.catch((Error)=> console.error('error'));
```

Récupérer Id  
Récupérer le coût total


→ Retourne l'erreur

### 3.2- Tests d'intégration

<code>fetch("http://localhost:3000/api/cameras/order")</code>	→	Connexion avec le backend
<code>window.location="confirmation.html"</code>	→	Retourne la page confirmation.html
<code>window.location.assign("index.html")</code>	→	Retourne la page index.html

## 4- Confirmation de la commande.

### 4.1- Tests unitaires

Bout de code (page confirmation.html)	Comportement attendu
<code>document.getElementById('order-confirmed')</code> <code>).innerHTML+='</code> <code>&lt;div&gt;</code> <code>&lt;p&gt; Numéro de commande : \${ getOrderId }&lt;/p&gt;</code> <code>&lt;p&gt;Prix total : \${getCartCost}&lt;/p&gt;&lt;/div&gt;'</code>	 <ul style="list-style-type: none"><li>Retourne le message de confirmation</li><li>Retourne le message de remerciement</li><li>Retourne le numéro de la commande</li><li>Retourne le prix total de la commande</li></ul>

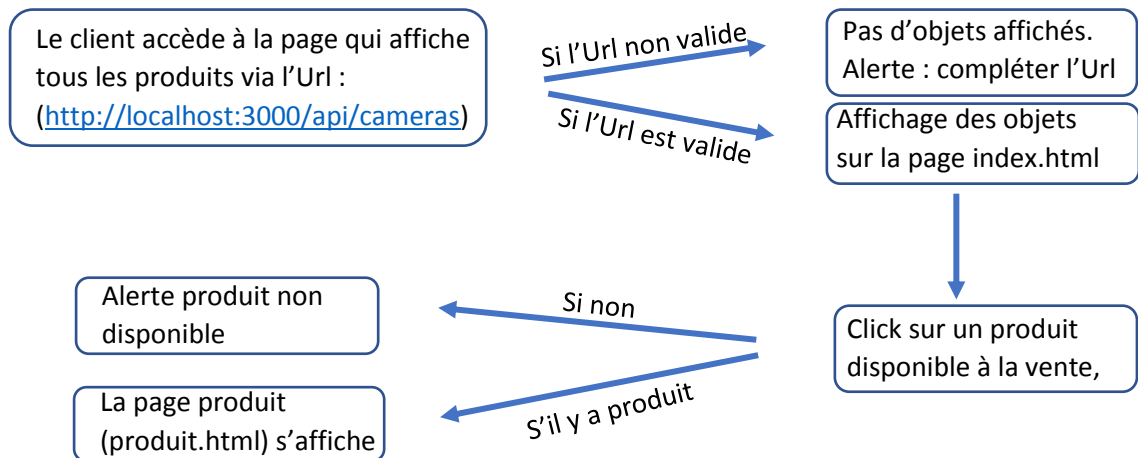
## 5- Tests de validation

Ce sont les tests de recettes qui vérifient les différentes fonctionnalités attendues par le client et qui ont été défini dans les spécifications fonctionnelles. Car l'on peut créer un magnifique site e-commerce mais il faut toujours se poser la question si c'est ce que le client attendait. Ces tests décrivent les actions successives qui mènent à la réalisation d'un objectif. Pour les interactions entre les clients et le serveur nous avons le processus suivant :

Le client arrive sur la page qui affiche tous les articles disponibles à la vente, il sélectionne un produit et une nouvelle page s'ouvre dynamiquement lui permettant de personnaliser son produit et de l'ajouter au panier. Après l'ajout, la page panier s'affiche dynamiquement contenant un résumé des produits, le prix total et un formulaire permettant de passer une commande. Toutefois les données du formulaire doivent être bien formatées avant d'être renvoyées au back-end. Enfin une page de confirmation de commande remerciant l'utilisateur pour sa commande et indiquant le prix total et l'identifiant de commande envoyé par le serveur.

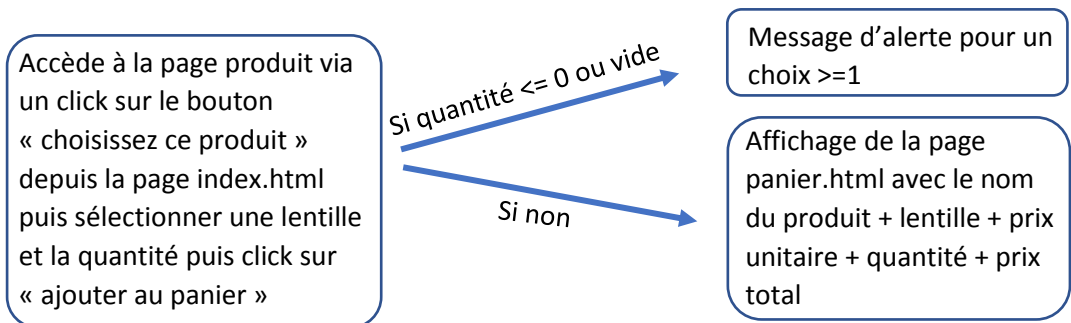
Le récapitulatif des scénarios est représenté comme suite :

➤ **Affichage de tous les articles disponibles à la vente**

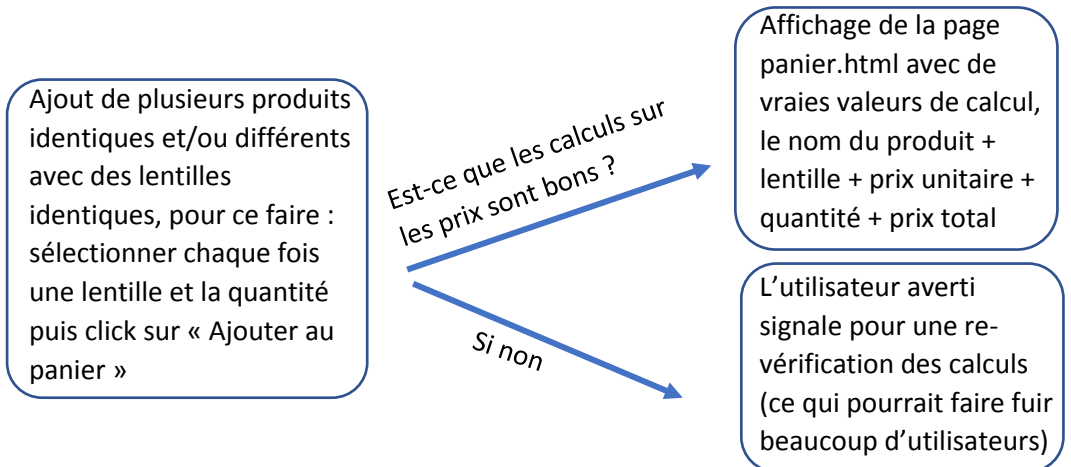


➤ **Personnalisation du produit sélectionné par l'utilisateur**

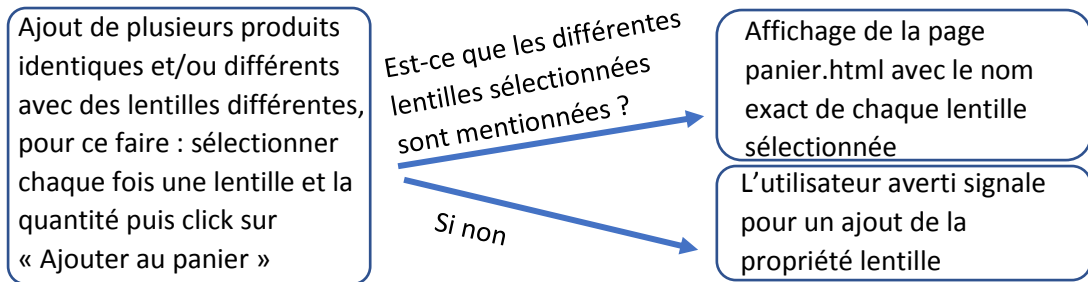
- **Choix de la lentille puis affichage du produit unitaire avec les données du produit**



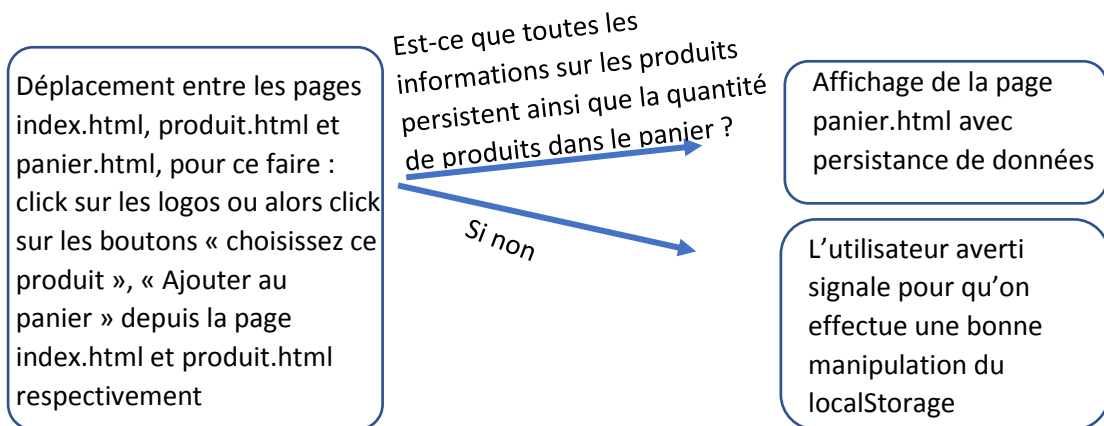
- **Ajout des produits avec des lentilles identiques et vérification des résultats de calculs**



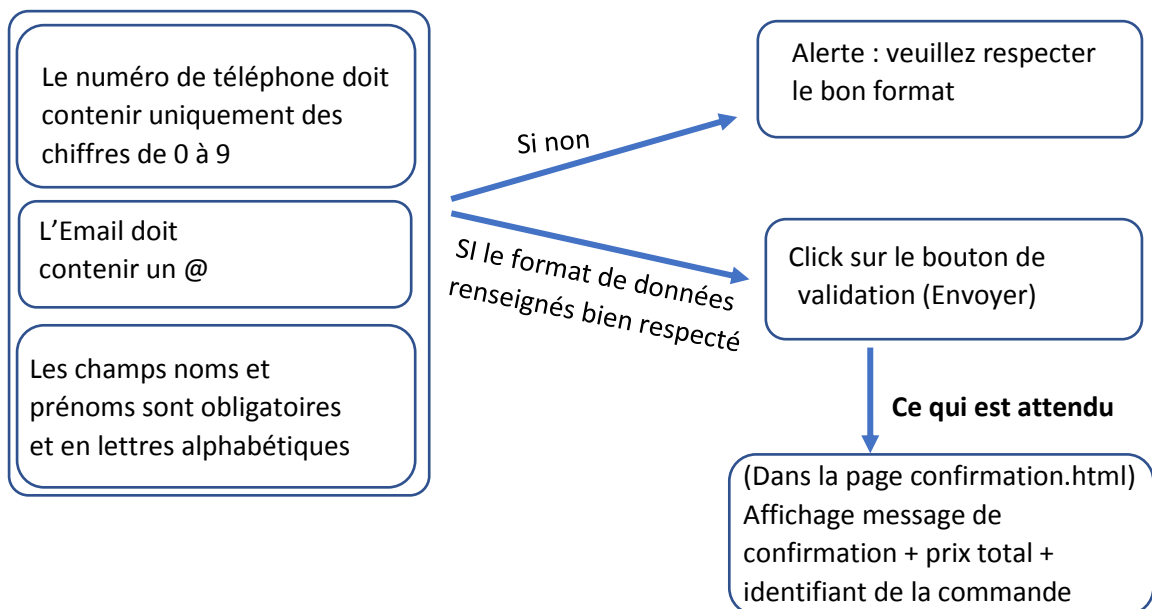
- **Ajout des produits avec lentilles différentes et affichage des noms de ces lentilles**



- **Déplacement entre les pages et persistance des données dans le panier**



➤ **Respect des formats de données et validation du formulaire dans la page panier.html**



- **Panier vidé grâce au déplacement depuis la page panier.html à la page confirmation.html**

