Bilan de campagne de test

I. Recommandations

A. Tests à automatiser

Les tests portent sur l'affichage des produits et le panier, afin de s'assurer que ces fonctionnalités essentielles offrent une expérience utilisateur correcte et conforme aux attentes.

1) Panier:

Ce test est automatisé dans TestPanier.cy.js.
Ce scénario teste l'ajout d'un produit au panier, la mise à jour du stock, et les limites de quantité autorisées.
C'est un test critique pour la stabilité du parcours d'achat et la fiabilité de la gestion des stocks.
Le test couvre :

- Ajout d'un produit disponible,
- Refus d'ajout d'un produit hors stock,
- Gestion des quantités négatives et trop élevées.
- La mise à jour du stock

2) Affichage des produits :

Ce test est automatisé dans le fichier homeProductsDisplay.cy.js.
Ce test vérifie le bon chargement des produits sur la page d'accueil et des produits, ainsi que la présence et la visibilité de toutes les informations : image, nom, ingrédients, prix, et bouton "Consulter".
Ce test est critique car il touche directement à l'expérience d'achat du client. Un produit mal affiché ou mal renseigné signifie une perte de vente.

B. Préconisations pour la suite

- D'ajouter des tests de compatibilité navigateurs (Safari, Firefox, Edge).

- Ajouter des tests de validation des entrées pour les formulaires (tests de limites, caractères spéciaux, longueur maximale, etc.) afin de garantir la robustesse et la sécurité des formulaires du site.
- Test de suppression de produit, interface utilisateur et API afin de garantir la cohérence des données entre le front-end et le back-end, ainsi qu'une expérience utilisateur fluide et fiable lors de la modification du panier.

II. Tests automatisés

Document revu par :

Nom	Fonction	Version	Signature
Fabio	CTO	1.0.0	
Virgile	Testeur QA	1.0.0	

Contexte

Cette campagne de tests automatisés a été réalisée dans le cadre du projet Eco Bliss Bath, un site e-commerce de produits de beauté éco responsables, dont le produit principal est le savon solide.

- Trois personnes sont assignées à cette campagne : Fabio CTO, Marie QA (manuelle), Virgile QA (automatisation)
- Utilisation de Google Chrome(Version 141.0.7390.123) par défaut sur Cypress
- durée total des tests : 32 secondes

Objectif(s)

Automatiser les tests critiques identifiés par la QA manuelle (Marie) afin d'assurer la stabilité des fonctionnalités de base (affichage produits, panier, connexion, review).
Vérifier l'absence de failles XSS. (vulnérabilité qui permet d'injecter du code malveillant)
Couvrir les endpoints API listés, détecter et documenter les anomalies.

Tests effectués

1. Tests API

a) GET /orders – Accéder au panier (/orders) sans être connecté

fichier : ApiGetOrders.cy.js

Objectif : vérifier que l'accès au panier sans authentification est refusé.
Étapes :

Envoyer une requête GET vers /orders sans jeton.
→ Doit retourner 401
Résultat : conforme aux attentes.

b) POST /login – tester dans différentes situations l'authentification utilisateur

fichier : ApiPostLogin.cy.js

Objectif : tester la connexion avec un utilisateur connu et inconnu.
Étapes :

Essayer de se connecter avec un mauvais identifiant / mauvais mot de passe
→ Doit retourner 401.
Résultat : conforme aux attentes.

Se connecter avec un utilisateur connu.
→ Doit retourner 200 + un token JWT.
Résultat : conforme aux attentes.

c) GET /products/{id} – Vérification des information d'un produit

fichier : ApiGetProduct.cy.js

Objectif : vérifier les informations et la structure de la fiche produit.
Étapes :

Envoyer une requête GET /products/3.

Vérifier que la réponse contient les propriétés :

id, name, availableStock, price, description, ingredients, aromas, picture, varieties.
Résultat : toutes les données sont présentes.

d) POST /orders/add – Test de compatibilité

fichier : ApiPostProduct.cy.js

Objectif : vérifier que l'API accepte POST pour l'ajout d'un produit au panier (Méthode prévu dans cette situation).

Se connecter pour obtenir un token. (POST utilisateur et mot de passe valide)

Envoyer une requête POST à /orders/add
avec l'id d'un produit ainsi que la quantité ajoutée.

Résultat attendu : 200.
Résultat obtenu : échec → seule la méthode PUT fonctionne.
→ Anomalie fonctionnelle (API).

e) PUT /orders/add – Ajout d'un produit disponible au panier

fichier : ApiPostProduct.cy.js

Objectif : tester l'ajout d'un produit en stock au panier.
Étapes :

Se connecter pour obtenir un token. (POST utilisateur et mot de passe valide)

Envoyer une requête PUT à /orders/add avec l'id d'un produit ainsi que la quantité ajoutée.

Vérifier la réponse.
Résultat : code 200, ajout réussi.

f) PUT /orders/add – Ajout d'un produit indisponible au panier

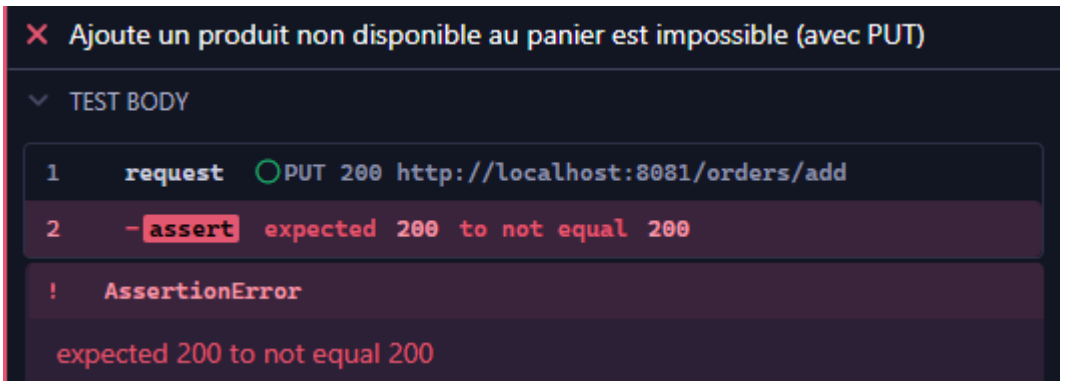
fichier : ApiPostProduct.cy.js

Objectif : tester l'ajout d'un produit en qui n'est pas en stock au panier via API
Étapes :

Se connecter pour obtenir un token. (POST utilisateur et mot de passe valide)

Envoyer une requête PUT à /orders/add avec l'id d'un produit qui est indisponible ainsi que la quantité ajoutée.

Résultat obtenue : Anomalie → permet l'ajout d'un produit indisponible au panier



g) POST /reviews – Ajout d'un avis client

fichier : ApiPostReviews.cy.js

Objectif : tester la création d'un avis (titre, commentaire, note).
Étapes :

Se connecter pour obtenir un token. (POST utilisateur et mot de passe valide)

Envoyer une requête PUT /reviews avec un titre, un commentaire, et une note

Résultat attendu : création d'un objet review avec id, author, title, comment, rating.
Résultat obtenu : conforme.

h) GET /orders – Vérification de la liste du panier (après ajout de produit)

fichier : ApiGetOrdersList.cy.js

Objectif : s'assurer que le panier contient les produits ajoutés.
Étapes :

Se connecter pour obtenir un token. (POST utilisateur et mot de passe valide)

Ajouter au moins deux produits au panier

Envoyer un GET /orders pour obtenir la liste des produits ajoutés

Vérifier que la réponse contient un tableau avec plusieurs éléments
Résultat : panier correctement récupéré avec les produits précédemment ajoutés.

2. Smoke tests

Ces tests visent à vérifier la présence des éléments clés de navigation (connexion, panier).
Ils s'assurent que l'interface principale fonctionne avant tout autre test.

a) Bouton Connexion

fichier : SmokeTestLogin.cy.js
Pages testées : /#/, /#/products, /#/register, /#/reviews
Étapes :

1. Visiter chaque page.

2. Vérifier que le lien "Connexion" est visible, cliquable et redirige bien vers #/login.
   Résultat : conforme sur toutes les pages.

b) Bouton Panier

fichier : SmokeTestAddPanier.cy.js
Pages testées : /#/, /#/products, /#/cart, /#/reviews
Étapes :

1. Se connecter avec un compte valide.

2. Vérifier la présence du bouton “Mon panier”.

3. Cliquer dessus → redirection vers #/cart.
   Résultat : conforme sur toutes les pages.

3. Tests de faille XSS

fichier : xssCom.cy.js
Objectif : vérifier l'absence d'exécution de code JavaScript injecté dans les champs du formulaire d'avis.
Étapes :

Se connecter.
Aller dans la section #/reviews.
Renseigner le champ commentaire avec <script>alert("XSS");</script>.
Faire la même chose pour le champ titre.
Soumettre le formulaire.
Résultat attendu : aucune fenêtre d'alerte ne s'affiche.
Résultat obtenu : conforme → aucune faille XSS détectée.

4. Tests fonctionnels

a) Affichage des produits

homeProductsDisplay.cy.js
Objectif : s'assurer que tous les produits s'affichent correctement sur les pages /products et la page d'accueil
Étapes :

1. Visiter #/products et #

2. Vérifier pour chaque produit :
   image visible,
   nom non vide,
   ingrédients affichés,
   prix visible avec symbole "€",
   bouton "Consulter" présent.
   Résultat : conforme → tous les éléments s'affichent correctement.

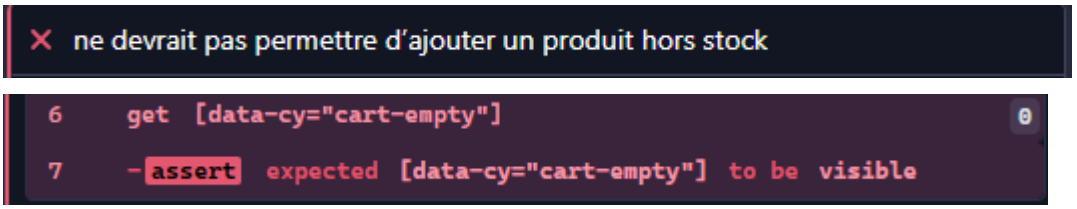
b) Ajout et gestion du stock du panier

fichier : TestPanier.cy.js
Objectif : Vérifier la fonctionnalité du Panier du site (mise à jour du stock, limite, produit indisponible)
Étapes :

1. Se connecter.

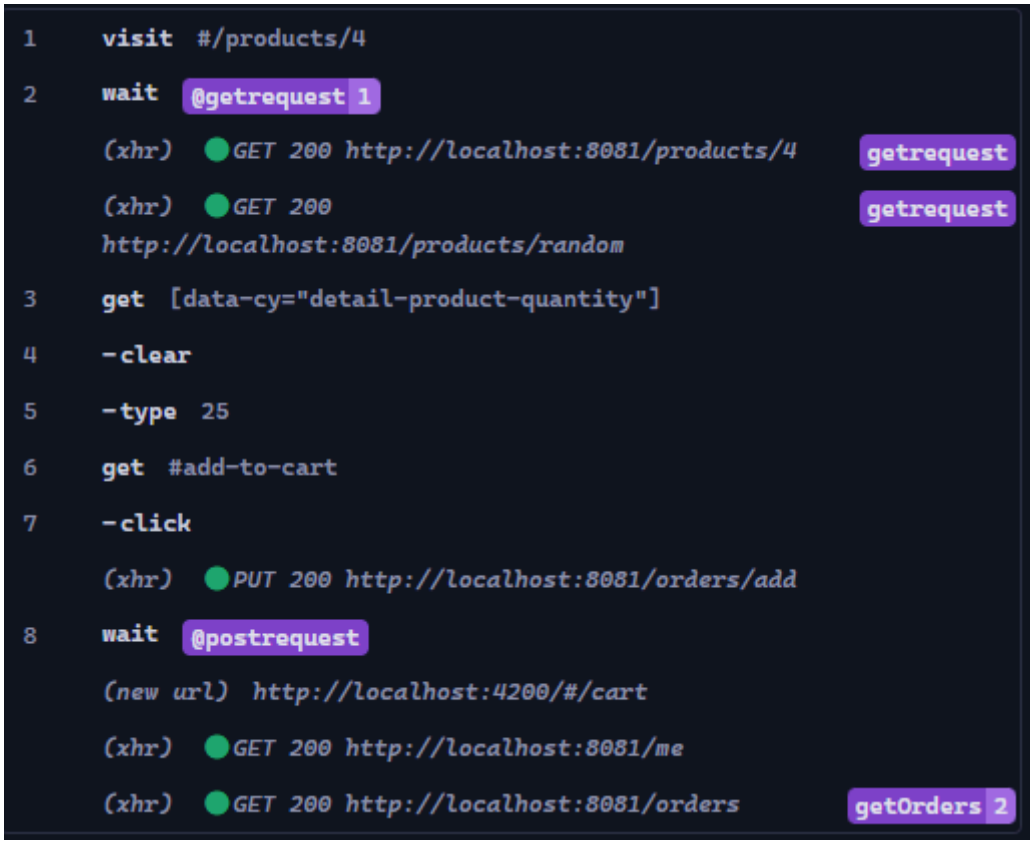
2. Sélectionner un produit et Cliquer sur “Ajouter au panier” → vérifier la mise à jour du stock.
   Résultat : conforme aux attentes.

3. Essayer d’ajouter un produit hors stock (0 / -) → autorisé → Anomalie .



4. Essayer d’ajouter une quantité négative → refusé / rien ne se passe
   Résultat : conforme aux attentes.

5. Essayer d’ajouter une quantité supérieure à 20 → → autorisé → Anomalie .



c) Affichage du détail des produits

fichier : detailsProductsDisplay.cy.js
Objectif : vérifier l'affichage du détails de chaque produit sur leur page respective

Étapes :

1 Visiter chaque page de chaque produit

2 Vérifier pour chaque produit :
image visible,
nom non vide,
détails du produit
prix visible avec symbole "€",
bouton "Ajouter" présent.
Résultat : conforme → tous les éléments s'affichent correctement.

Résultats de tests

Tests conformes aux attentes :

Catégorie	Fichier	Description	Résultat
Authentification	ApiPostLogin.cy.js	Vérifie que le login refuse un utilisateur inconnu (401) et renvoie un token valide pour un utilisateur connu.	Conforme
Produits	ApiGetProduct.cy.js	Vérifie la récupération correcte d'une fiche produit spécifique et la	Conforme

		présence de toutes les propriétés attendues.	
Panier – Stock décrémenté	TestPanier.cy.js	Vérifie que le stock diminue après ajout d'un produit.	Conforme
Panier – Quantité négative	TestPanier.cy.js	Entrée d'un nombre négatif dans le champ quantité : l'application ne valide pas l'action.	Conforme
Reviews	ApiPostReviews.cy.js	Création d'un avis via l'API (200 attendu) et validation du schéma de réponse.	Conforme
Interface – Smoke Tests	SmokeTestLogin.cy.js, SmokeTestAddPanier.cy.js	Vérification de la présence et du fonctionnement des liens de navigation ("Connexion", "Mon panier") sur les principales pages.	Conforme
XSS	xssCom.cy.js	Vérifie l'absence de faille XSS dans les champs "commentaire" et "titre" du module d'avis.	Conforme

Tests en échec ou anomalies constatées

Catégorie	Fichier	Objectif	Résultat attendu	Résultat obtenu
API – POST /orders/add	ApiPostProduct.cy.js	Vérifier que l'API accepte aussi la méthode POST pour ajouter au panier	200 (succès)	405 / échec – seule la méthode PUT fonctionne

API – PUT /orders/add (produit indisponible)	ApiPostProduct.cy.js	Vérifier qu'un produit en rupture de stock ne peut pas être ajouté au panier	Erreur / refus	Produit ajouté malgré stock nu ou négatif
Panier – Produit hors stock	TestPanier.cy.js	Vérifier que l'ajout d'un produit hors stock est bloqué côté UI	Refus de l'ajout	Produit ajouté malgré stock nu ou négatif
Panier – Quantité > 20	TestPanier.cy.js	Vérifier qu'une quantité supérieure à 20 est refusée	Erreur ou refus	Ajout autorisé

Anomalie :

La requête sur l'ajout d'un produit disponible au panier devrait être un POST mais nous avons un PUT.

Il est possible d'ajouter au panier des produits qui ne sont pas en stock, toujours via un PUT. Cela peut entraîner des commandes impossibles à honorer et une mauvaise expérience client.

Aucune restriction n'est appliquée sur la quantité d'un même produit ajouter au panier. Il est ainsi possible d'ajouter plus de 20 unités d'un même produit

Confiance

La version actuelle du site fonctionne globalement, avec des fonctionnalités de base opérationnelles telles que l'authentification, l'affichage des produits et la navigation entre les pages. Les failles XXS sont également prises en charge afin de s'assurer de la sécurité de l'utilisateur.

Cependant certaine anomalie persistent :

Ajout de produit hors stock (critique) : le site permet en effet de commander des produits impossible à livrer. Ce qui nuit gravement à l'expérience de l’utilisateur.

Quantité maximale non respectée (plus de 20). En effet limiter le nombre de produits permet de répartir équitablement, entre tous les utilisateurs, les produits, surtout en cas de stock limité. Éviter les abus (comme la revente). améliorer l’expérience utilisateur.

Méthode POST non supportée pour l’ajout au panier doit être corrigée pour respecter la spécification API et éviter des problèmes pour l’intégration future ou l’automatisation externe.

De plus, je conseille d'intégrer d'autres tests :
tests de compatibilité navigateurs (Safari, Firefox, Edge).
Ajouter des tests de validation des entrées pour les formulaires Ces tests concernent des contraintes d’usage de base pour l’utilisateur.
Test de suppression de produit, interface utilisateur et API.

La version peut être utilisée pour des démonstrations, mais elle ne peut pas être considérée comme totalement fiable pour la mise en production, tant que les anomalies critiques liées au panier et au stock ne sont pas corrigées.
Les correctifs prioritaires sont donc : blocage de l’ajout de produits hors stock.