

Rapport de testing EcoBliss

I. Recommandations

A. Tests à automatiser:

Tests à prioriser pour l'automatisation en plus des tests demandés :

Connexion :

Vérification d'une connexion réussie.

Vérification des cas invalides (mots de passe incorrects, champs vides).

Panier :

Ajout de produits en stock au panier.

Ajout de produits indisponibles au panier.

Gestion des stocks après ajout au panier.

Limites de quantités (chiffres négatifs, au-dessus de 20).

Validation de l'intégration API (contenu du panier).

Pourquoi est-ce crucial ?

- **La connexion** est la porte d'entrée vers l'application. Sans elle, les utilisateurs ne peuvent pas accéder à leurs comptes ni aux fonctionnalités clés comme le panier ou les commandes.
Une défaillance ici pourrait entraîner une perte de confiance et d'utilisateurs dès leur première interaction.

Risques si non testé :

Perte d'utilisateurs : Si la connexion échoue ou fonctionne mal, cela peut entraîner une perte de clients dès les premières interactions avec l'application.

Problèmes de sécurité : Un formulaire mal géré peut être vulnérable aux attaques comme l'injection SQL ou le contournement des authentifications.

Justification pour l'automatisation :

Les scénarios de test sont répétitifs (par exemple, saisie de différentes combinaisons d'identifiants) et donc idéaux pour l'automatisation.

- **Le panier** est une fonctionnalité centrale pour toute application e-commerce. C'est là où les utilisateurs interagissent pour acheter des produits.

Des erreurs comme l'ajout impossible d'un produit ou une mauvaise gestion de stock impactent directement la satisfaction des utilisateurs et les revenus.

Risques si non testé :

Frustration des utilisateurs : Si des produits ne peuvent pas être ajoutés, si les stocks ne se mettent pas à jour correctement ou si des limites comme les quantités négatives ne sont pas respectées, cela provoquera un désengagement.

Impact direct sur les revenus : Les problèmes de panier peuvent entraîner une perte immédiate de ventes.

Justification pour l'automatisation :

Tester le panier implique plusieurs vérifications complexes: vérifier les stocks, les limites, l'interaction avec l'API qui seraient sujettes à erreur si réalisées manuellement. En automatisant ces tests le calcul s'effectue manuellement, que la première fois et le test automatisé peut être lancé plusieurs fois avant chaque mise en production et autant de fois que ça le nécessite.

B. Préconisations pour la suite

Donnez votre avis sur les actions que l'équipe QA devrait entreprendre à l'avenir, quand ils auront le budget nécessaire :

a. actions à envisager avec un budget plus important:

1. Augmenter le nombre de cas de test:

Diversifier les cas de test en utilisant différentes situations et divers jeux de données permet de couvrir un plus large éventail de scénarios, ce qui contribue à réduire le risque de bugs.

2. Étendre les efforts d'automatisation:

L'automatisation permet de réduire les efforts manuels et garantir la fiabilité de l'application.

3. formation et collaboration:

- Former l'équipe QA aux nouveaux outils et technologies.
- Collaborer étroitement avec les développeurs pour identifier les cas limites et anticiper les problèmes

b. conseils pour les tests restants:

Il est préférable d'automatiser les tests restants pour les raisons suivantes:

1. Gain de temps sur les tests manuels :

Ces tests ciblent des fonctionnalités clés (parcours utilisateur et affichage des produits), fréquemment impactées par les mises à jour.

2. effectuer les tests de non régression :

L'automatisation facilite la détection des régressions en lançant l'ensemble des tests automatisés avec une seule commande.

3.Importance des informations produit :

Les données critiques telles que le prix, le stock et les descriptions jouent un rôle central dans l'expérience utilisateur et les transactions. leurs automatisation permet une vérification plus précise.

II. Bilan de campagne de validation : tests automatisés

Document revu par :

Nom	Fonction	Version	Signature
Marie	Testeur manuel	1.0.0	
Meriam	Testeur en automatisation	1.0.0	

Contexte du projet :

Le projet Eco Bliss Bath concerne la mise en ligne de la première version d'un site e-commerce spécialisé dans la vente de produits de beauté éco-responsables, avec un focus principal sur un savon solide.

Objectifs de la campagne de test :

Cette campagne a pour but de valider la stabilité, la performance et la fiabilité des fonctionnalités critiques du site avant son lancement public. Les tests se sont concentrés sur l'API, les fonctionnalités principales, et les smoke tests.

Nombre de membres de l'équipe impliqués :

Marie a conduit une série de tests manuels et produit un rapport détaillé.
Fabio, CTO, supervise les activités de testing et le processus d'automatisation.
Meriam, responsable de l'automatisation des tests.

Type de tests :



Les tests initiaux étaient manuels.

La nouvelle phase de tests sera automatisée avec Cypress.

Le temps passé est d'une demi-journée pour chaque fonctionnalité.

L'application web a été testée sur le navigateur Chrome.

Tests effectués et résultat :

Tests effectués	Résultat attendu	Résultat obtenu	Analyses de résultat	Recommandations
Requête sur les données confidentielles d'un utilisateur avant connexion Méthode : GET URL : http://localhost:8081/orders Précondition : L'utilisateur n'est pas connecté.	Retourner une erreur 403	KO La requête retourne le code 401 Unauthorized, au lieu du code 403 attendu.	Le comportement de l'API n'est pas conforme aux attentes. Et que l'utilisateur doit s'authentifier et avoir le droit nécessaire pour accéder à la ressource demandée.	Revoir la gestion des statuts HTTP dans l'API
Requête de la liste des produits du panier Méthode : GET sur /orders URL : http://localhost:8081/orders Précondition : L'utilisateur doit être authentifié avec un jeton valide (Bearer Token). Requête d'une fiche produit spécifique Méthode : GET sur /products/{id}	Retourner un code 200 OK. Fournir un tableau des produits présents dans le panier. Retourner un code 200 OK. Fournir les détails du produit correspondant à l'ID	OK Le Code 200 est reçu avec succès. La liste des produits dans le panier est retournée comme attendu. Code 200 est reçu avec succès. Les informations détaillées du produit sont retournées comme attendu.	 Requête 1 : Conforme aux attentes. L'API retourne avec succès la liste des produits présents dans le panier pour un utilisateur authentifié.  Requête 2 : Conforme aux attentes. L'API retourne les détails précis d'un produit en fonction de son ID.	

<p>Ajouter un produit disponible au panier authentication avec un utilisateur connu</p> <p>Méthode : POST sur /login Objectif : Authentifier l'utilisateur avec des identifiants valides et récupérer un jeton pour les requêtes suivantes. Méthode : PUT sur /orders/add</p>	<p>Retourner un code 200 OK si l'ajout est réussi</p>	<p>OK Le Code 200 est retourné. Le produit avec l'ID 5 et la quantité 6 a été ajouté au panier avec succès.</p>	<p>Conforme aux attentes. Le produit est ajouté avec succès, et le code de réponse est correct.</p>	
<p>Ajouter les informations personnelles, validation d'achat. Test de sécurité</p> <p>Méthode : POST sur /orders en ajoutant le script de la faille XSS en body de la requête</p>	<p>La réponse de la requête ne doit pas contenir la balise <script></p>	<p>KO la réponse contient l'adresse tel qu'elle a été envoyé, avec le script injecté intact</p>	<p>Le test montre que le serveur ne nettoie pas les entrées utilisateur, pour empêcher l'injection de code malveillant.</p>	<p>Implémenter une validation stricte des entrées utilisateur</p>
<p>Ajouter un produit disponible au panier authentication avec un utilisateur inconnu Objectif : Authentifier l'utilisateur avec des identifiants invalides et récupérer un jeton pour les requêtes suivantes.</p> <p>Méthode : PUT sur /orders/add</p>	<p>Retourner un code 401 Unauthorized pour signaler que l'utilisateur est inconnu. Aucun jeton d'accès (Bearer Token) ne doit être généré Retourner un code 401 Unauthorized pour signaler que l'accès est refusé</p>	<p>OK Le Code 401 est retourné. L'accès a été refusé comme attendu.</p>	<p>Conforme aux attentes. L'API refuse l'accès à l'opération et retourne le code d'erreur approprié.</p>	
<p>Ajouter un produit en rupture de stock</p> <p>Méthode : PUT sur /orders/add</p>	<p>Valider que l'API retourne une erreur 406 pour signaler l'indisponibilité du produit</p>	<p>KO L'API a retourné un code 200 OK au lieu du 406 attendu</p>	<p>L'API doit vérifier la disponibilité en stock avant d'ajouter un produit au panier et retourner une</p>	<p>Implémenter une vérification stricte de la disponibilité en stock avant de permettre</p>

			erreur appropriée en cas de rupture de stock.	l'ajout d'un produit. Si le produit est indisponible , l'API devrait retourner un code 406 Not Acceptable
Ajouter un avis Méthode : POST sur /reviews	Retourner un code 200 OK pour signaler que l'ajout de l'avis a été réalisé avec succès	OK code 200 retourné avis ajouté avec succès.	L'API permet à un utilisateur authentifié de soumettre un avis pour un produit. L'avis est correctement enregistré dans le système.	
Ajouter un avis test de sécurité Méthode : POST sur /reviews en ajoutant le script de la faille XSS en body de la requête	La réponse de la requête ne doit pas contenir la balise <script>	KO la réponse contient le commentaire tel qu'il a été envoyé, avec le script injecté intact	Le test montre que le serveur ne nettoie pas les entrées utilisateur, pour empêcher l'injection de code malveillant.	Implémenter une validation stricte des entrées utilisateur
Test de connexion-inscription Objectif : Vérifier que l'inscription fonctionne correctement avec des données valides générées dynamiquement.	L'utilisateur doit réussir l'inscription avec des données valide	OK L'utilisateur est enregistré avec succès.	L'inscription fonctionne comme prévu : Les données dynamiques générées sont correctement acceptées par le formulaire. L'utilisateur est redirigé vers l'accueil après l'inscription.	

Connexion avec un utilisateur existant et données valides. Objectif : Vérifier que la connexion fonctionne correctement avec le jeu de données fourni.	L'utilisateur doit réussir la connexion avec des données valide	OK L'utilisateur est enregistré avec succès.	Le bouton "Panier" est visible, indiquant que l'utilisateur est connecté.	
Connexion avec des identifiants incorrects	Les données incorrectes doivent renvoyer un message d'erreur	OK Le message d'erreur est bien affiché	conforme aux attentes l'utilisateur ne peut pas se connecter	
Inscription avec des identifiants invalides	Les données invalides doivent renvoyer un message d'erreur	OK Le système affiche bien une erreur de validation	conforme aux attentes l'utilisateur ne peut pas s'enregistrer	
Test de stock suffisant de produit	Le stock initial est supérieur à 1. Après l'ajout du produit au panier, le stock du produit sélectionné doit être réduit d'une unité.	OK Stock initial récupéré avec succès (supérieur à 1). Ajout au panier effectué correctement. Stock mis à jour correctement après l'ajout (stock réduit d'une unité).	Le test a réussi à vérifier que le système gère correctement la disponibilité des stocks après l'ajout au panier. La logique de décrémentation est bien implémentée et fonctionne comme attendu.	
Test de mise à jours de stock API	L'ajout du produit au panier se fait sans erreur.	OK L'API a retourné un statut 200	Les données de l'API sont cohérentes avec les	

	La requête API retourne un statut 200 et la ligne de commande dans le panier contient le produit ajouté. Le stock du produit devrait être mis à jour après l'ajout au panier.	avec les bonnes données pour le produit ajouté au panier. La ligne de commande récupérée contient bien le produit attendu.	actions effectuées sur l'interface, ce qui montre que l'intégration entre l'interface utilisateur et l'API est fonctionnelle.	
<p>Test des nombres limites d'ajouts de produit au panier.</p> <p>Objectif de test</p> <p>Vérifier que l'application empêche d'ajouter une quantité négative ou une quantité excédant le stock d'un produit au panier et que ces tentatives n'affectent pas l'état du panier</p>	Le produit ne doit pas être ajouté au panier pour une quantité invalide (négative ou excédentaire). Le produit ne doit pas être visible dans la liste des articles du panier.	<p>KO</p> <p>Le produit est toujours visible dans le panier malgré la quantité invalide.</p>	L'erreur indique que l'application n'a pas correctement validé les entrées de quantité dans les champs du formulaire. Cela signifie que les mécanismes de validation côté client ou serveur ne fonctionnent pas comme prévu. Le produit a été ajouté au panier alors qu'il n'aurait pas dû l'être.	Implémenter des restrictions pour empêcher les utilisateurs de saisir des quantités invalides directement dans le champ de saisie. Par exemple, utiliser un min="1" et un max égal au stock disponible ou bien du nombre de produits autorisé par commande.
<p>Smoke test</p> <p>Objectif de test</p> <p>Valider le bon fonctionnement des flux critiques de l'application, notamment la connexion, la navigation vers les produits, et l'accès aux</p>	Les champs et les boutons doivent être accessibles et visibles	<p>OK</p> <p>Les éléments sont correctement visibles.</p>	Le flux critique de l'application fonctionne comme prévu. Aucun problème n'a été détecté lors de ce test.	

fonctionnalités de détail d'un produit.				
--	--	--	--	--

Spec	Tests	Passing	Failing	Pending	Skipped	
✗ api-tests/add-product-existant-user.cy.js	00:02	2	1	1	-	-
✓ api-tests/add-product-inexistant-user.cy.js	254ms	1	1	-	-	-
✗ api-tests/add-review.cy.js	00:01	3	2	1	-	-
✗ api-tests/add-unavailable-product.cy.js	00:01	1	-	1	-	-
✗ api-tests/private-data.cy.js	367ms	1	-	1	-	-
✓ api-tests/product-list.cy.js	958ms	2	2	-	-	-
✓ basket-tests/available-stock-test.cy.js	00:07	1	1	-	-	-
✓ basket-tests/basket-update-api-test.cy.js	00:06	2	2	-	-	-
✗ basket-tests/limit-number-of-product-test.cy.js	00:10	1	-	1	-	-
✓ connexion-tests/registration.cy.js	00:07	1	1	-	-	-
✓ connexion-tests/unvalid-input-registration.cy.js	00:06	1	1	-	-	-
✓ connexion-tests/unvalid-login-access.cy.js	00:04	1	1	-	-	-
✓ connexion-tests/valid-connexion.cy.js	00:03	1	1	-	-	-
✓ smoke-tests/inputs.cy.js	00:03	1	1	-	-	-
✗ 5 of 14 failed (36%)	00:56	19	14	5	-	-

```
Specs 2 x -- 0:01

add-review cyjs 00:01

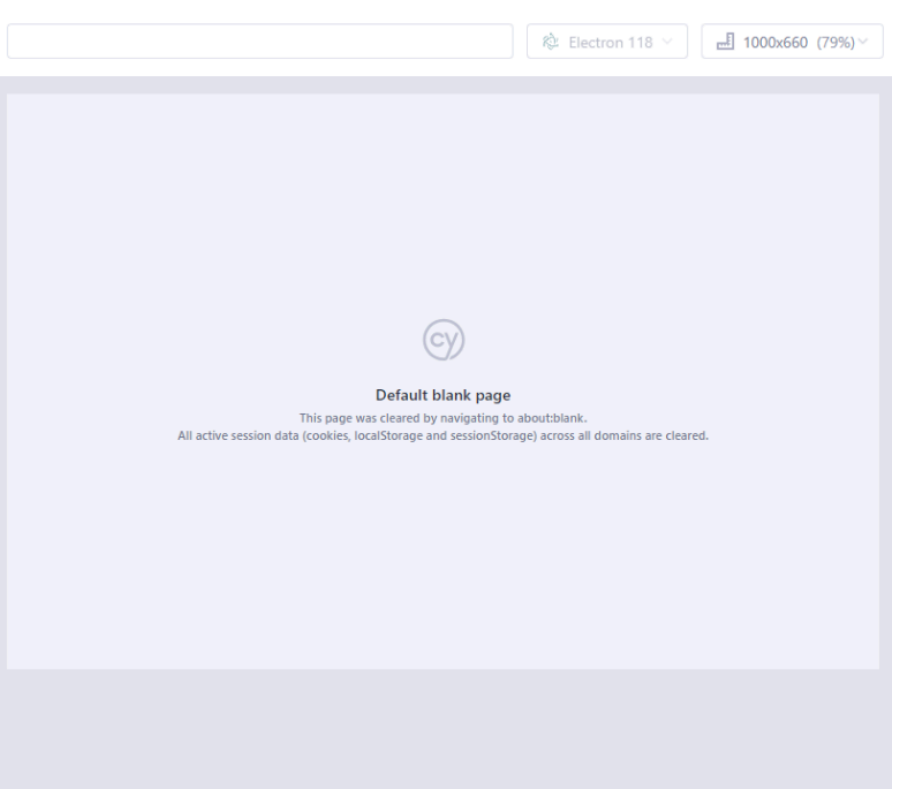
dC5mciJ9.GiFVC4ZPtASGpV3vEqPpKdHTLSyAcw1mtiQ1LKu
yXnKbEra4VJea10dKm0IunFX1ABuneLByAV_x1-
qWruRtJrUlaQhL9Q8xHz07u_91zYRtDXdWgGGrwB7FI60Vp1
4m-3MiWki3i1B1Kn-
p8TULYTGT9I0PZ0Wpx0NdWE1wGOUFILfrPkphcmymvMvMj
LJxWquo-
GM61iJQ175dIXBGgvcLFHhTcXSbudhYZNwI2HyLNLKxg5cGZ
43IaxnWCBW6LZUI-
FR2JMaGjo7xU2p5gCc0GisFi1ZS0eEa_3LutYhJW6hFFAQK_p
wKI3RcEr-hDPoCNLHKKTyDydw

2 - assert expected
eyJ0eXAI0iJKV1Q1LCJhbGciOiJSUzI1NiJ9.eyJpYXQiOiE3
Mzc5MDYwODQsImV4cCI6MTczNzkwOTY3OCwicm9sZXMiOiIsiU
k9MRV9VUBVS10sInVzZXJvYm11IjoiaGVzdD03AdGVzdC6mci
J9.GiFVC4ZPtASGpV3vEqPpKdHTLSyAcw1mtiQ1LKuyXnKbE
ra4VJea10dKm0IunFX1ABuneLByAV_x1-
qWruRtJrUlaQhL9Q8xHz07u_91zYRtDXdWgGGrwB7FI60Vp1
4m-3MiWki3i1B1Kn-
p8TULYTGT9I0PZ0Wpx0NdWE1wGOUFILfrPkphcmymvMvMj
LJxWquo-
GM61iJQ175dIXBGgvcLFHhTcXSbudhYZNwI2HyLNLKxg5cGZ
43IaxnWCBW6LZUI-
FR2JMaGjo7xU2p5gCc0GisFi1ZS0eEa_3LutYhJW6hFFAQK_p
wKI3RcEr-hDPoCNLHKKTyDydw not to be undefined

3 request POST 200 http://localhost:8081/reviews

4 - assert expected 200 to equal 200

5 - assert expected <script>alert('XSS')</script>
to not include <script>
```



```
Specs 1 x -- 0:01

add-product-existant-user cyjs 00:01

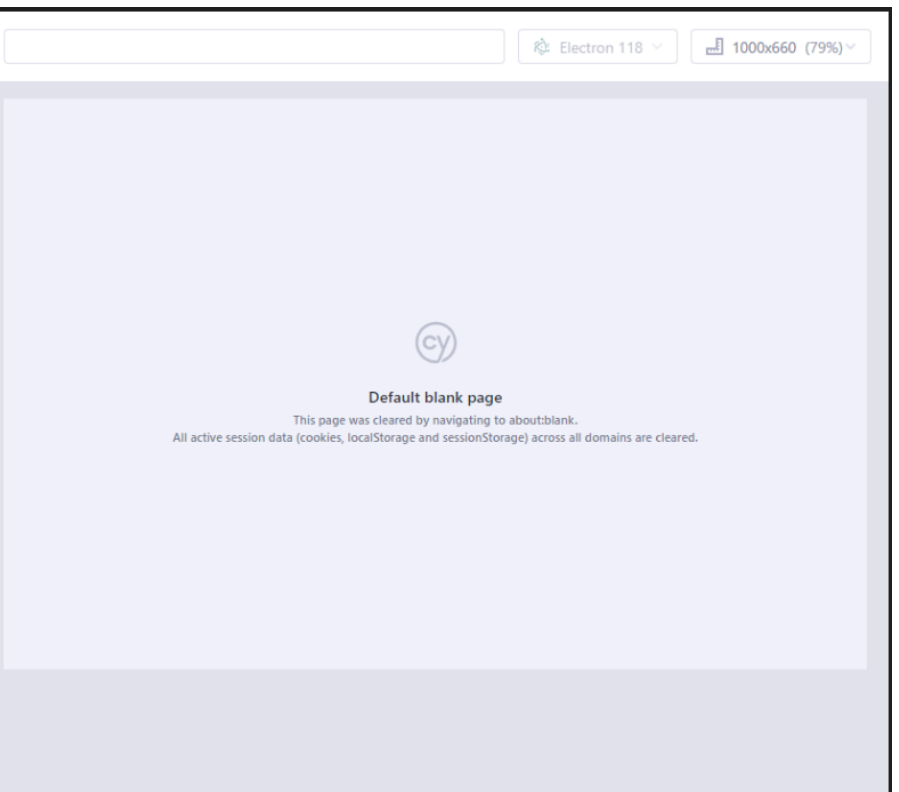
Q10jE3Mzc5MDYwODQsImV4cCI6MTczNzkwOTY3OCwicm9sZXM
i0Ls1Uk9MRV9VUBVS10sInVzZXJvYm11IjoiaGVzdD03AdGVz
dC5mciJ9.UPL60WkjrUmpynx4rR5OYnsfCTCqDpvl_m-
XCWlj736LwbRwLkktIiTpntxs6mqExeIA9vt2mm3mz6TJ8B_
R05BhdYlq9cTWi_RDb8j13R118JmcEHYX6SPju3SUyWCF0ECp
ZQ4PXVnpTdV99kxq73421Nrg37p6xBQnHBB8m8cBLV5ecL33B
-
WeIFY822HLJZB2TID4ceSbyZXOHZz4v57eBCKOUMXrVZkZQ4P
_ZWgLDaNCyompJWIH-EcsyIIE0vR3M-
5snfZvYtnB3ZvVAQ2AF_E4n45LE2Fv1p3_x6MPw2DZ3dnQryp
rycYRdC6yLNI-JGpfb1x3NL9J4LA

2 - assert expected
eyJ0eXAI0iJKV1Q1LCJhbGciOiJSUzI1NiJ9.eyJpYXQiOiE3
Mzc5MDYwODQsImV4cCI6MTczNzkwOTY3OCwicm9sZXMiOiIsiU
k9MRV9VUBVS10sInVzZXJvYm11IjoiaGVzdD03AdGVzdC6mci
J9.UPL60WkjrUmpynx4rR5OYnsfCTCqDpvl_m-
XCWlj736LwbRwLkktIiTpntxs6mqExeIA9vt2mm3mz6TJ8B_
R05BhdYlq9cTWi_RDb8j13R118JmcEHYX6SPju3SUyWCF0ECp
ZQ4PXVnpTdV99kxq73421Nrg37p6xBQnHBB8m8cBLV5ecL33B
-
WeIFY822HLJZB2TID4ceSbyZXOHZz4v57eBCKOUMXrVZkZQ4P
_ZWgLDaNCyompJWIH-EcsyIIE0vR3M-
5snfZvYtnB3ZvVAQ2AF_E4n45LE2Fv1p3_x6MPw2DZ3dnQryp
rycYRdC6yLNI-JGpfb1x3NL9J4LA not to be undefined

3 request POST 200 http://localhost:8081/orders

4 - assert expected 200 to equal 200

5 - assert expected <script>alert('XSS')</script>
to not include <script>
```



⇒ Specs

✓ -- ✗ -- ↺ --

▼

■

add-unavailable-product cy.js 908ms

```
Q1Ue3hZvZnJnSMDQS1mV4CLi6m1CZnJmZnZUWwCw1cm9sZXN1
i0IsiUk9MRV9VU8VSI0sInVzZXJlIjoidGVzdDAdGVzdC6mciJ9.XCi5FTAAfgS8gRxF_ioGxWpDJeurpJGT9I4yK4V8
RR7kjtM7ZeDLcAYzSC6f-fua6QtyugQtHYp7RB_LQVKp9xPp0BYrHPs7vSLcXrXxZA40X1
VXlQaY1Y9o-Ao9tKmZ4FGQlWlWw0G60pZi92m6xc4U2V31Jw2H0-
eZ9mwvHrEVqZwACksVsrJLz0ByBfyncoKITonGLZzDAoLR9hE
TPJWeb7zzLYGpJP-dCykL0c6_cckL7o0r0Vj_cM92fhpCFKY4j0D4p9jdtS_0hiXG
DTznfLi0ApxEsj7skhhyRdXyutCT_bxYxVPM2Q4hm71az9nI_
0Jt6TGkiIdTju7Zg
```

3 - assert expected
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpYXQiOiE3MzYzNjM8MDQsImV4cCI6MTczNjM2NzUwNWw1cm9sZXN1OlsiUk9MRV9VU8VSI0sInVzZXJlIjoidGVzdDAdGVzdC6mciJ9.XCi5FTAAfgS8gRxF_ioGxWpDJeurpJGT9I4yK4V8RR7kjtM7ZeDLcAYzSC6f-fua6QtyugQtHYp7RB_LQVKp9xPp0BYrHPs7vSLcXrXxZA40X1VXlQaY1Y9o-Ao9tKmZ4FGQlWlWw0G60pZi92m6xc4U2V31Jw2H0-eZ9mwvHrEVqZwACksVsrJLz0ByBfyncoKITonGLZzDAoLR9hETPJWeb7zzLYGpJP-dCykL0c6_cckL7o0r0Vj_cM92fhpCFKY4j0D4p9jdtS_0hiXGDTznfLi0ApxEsj7skhhyRdXyutCT_bxYxVPM2Q4hm71az9nI_0Jt6TGkiIdTju7Zg not to be undefined

4 request ○ PUT 200
http://localhost:8081/orders/add

5 - assert expected 200 to equal 500

Electron 118

1000x660 (79%)

⇒ Specs

✓ -- ✗ -- ↺ --

▼

■

private-data cy.js 219ms

GET /orders

○ gets a list of products

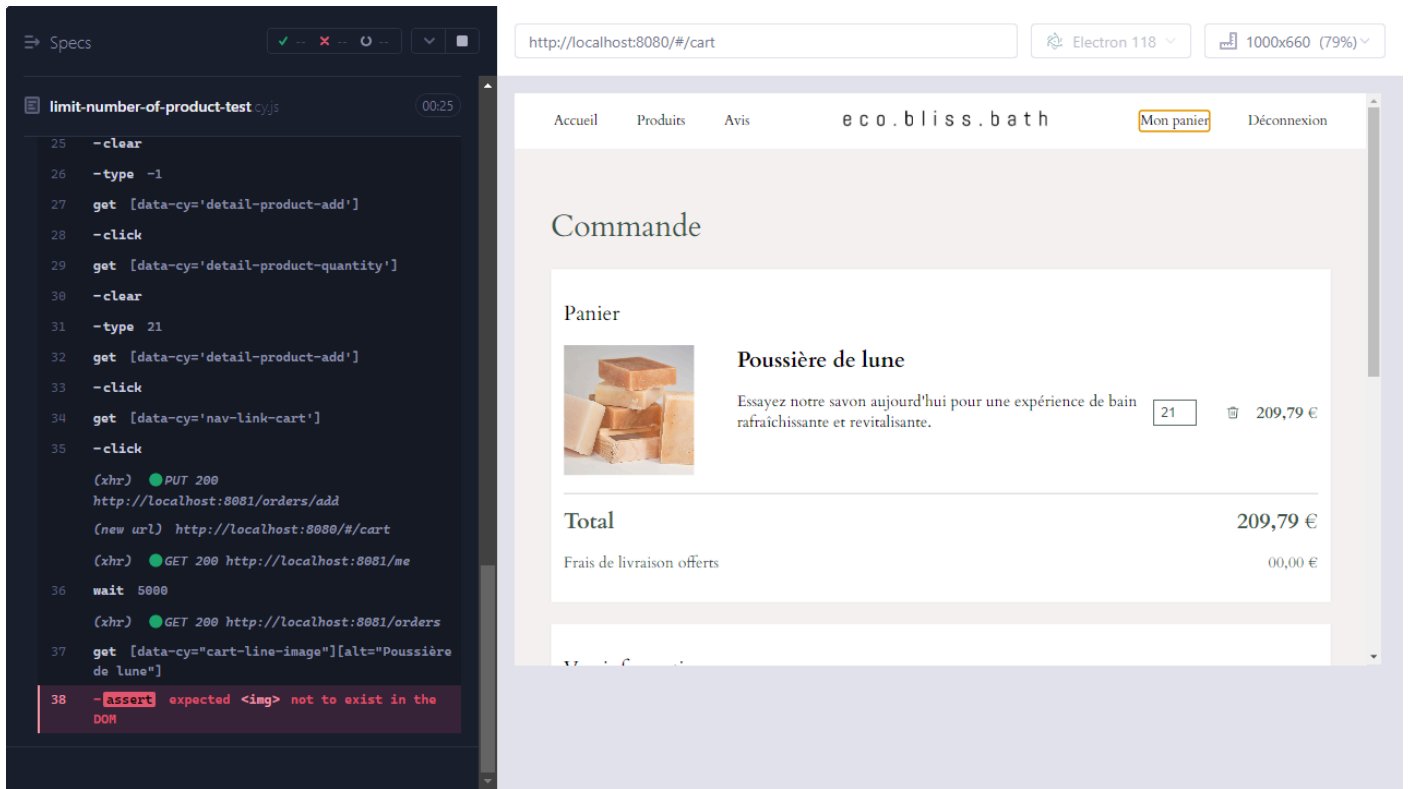
TEST BODY

1 request ○ GET 401 http://localhost:8081/orders

2 - assert expected 401 to equal 403

Electron 118

1000x660 (79%)



Rapport d'incident:

ID de l'incident : API-001

Titre: Erreur HTTP incorrecte retournée par `/orders` (401 au lieu de 403).

Description: Lorsque l'utilisateur non connecté effectue une requête GET sur `/orders`, l'API retourne une erreur 401 Unauthorized alors qu'une erreur 403 Forbidden était attendue.

Étapes de reproduction

1. Assurez-vous qu'aucun jeton d'authentification n'est fourni dans la requête.

2. Définissez l'URL de l'API comme suit :

```
const apiProduct = `${Cypress.env("apiUrl")}/orders`;
```

3. Exécutez le test suivant :

```
context("GET /orders", () => {
```

```
  it("gets a list of products", () => {
```

```
    cy.request({
```

```
      method: "GET",
```

```
      url: apiProduct,
```

```
      failOnStatusCode: false // Permet de capturer les erreurs HTTP
```

```
    }).then((response) => {
```

```
      expect(response.status).to.eq(403); // Vérification du code HTTP attendu
```

```
    });
```

```
  });
```

```
});
```

4. Observez le code HTTP retourné par l'API dans la réponse.

Résultat attendu : La requête *GET /orders* sans jeton d'authentification devrait retourner un code 403 Forbidden, indiquant un problème d'autorisation.

Résultat obtenu: La requête retourne un code 401 Unauthorized, indiquant un problème d'authentification au lieu d'une absence de permissions.

Impact: Cette incohérence peut induire en erreur les clients ou développeurs sur les permissions nécessaires pour accéder à la ressource. Cela complique la distinction entre les cas d'authentification et d'autorisation manquantes, rendant le débogage et l'utilisation de l'API plus difficiles.

ID de l'incident: API-002

Titre: Erreur HTTP incorrecte retournée par */orders/add* (200 au lieu de 406 pour produit indisponible).

Description: Lorsque l'utilisateur effectue une requête *PUT* sur */orders/add* pour ajouter un produit avec une quantité supérieure à la disponibilité en stock, l'API retourne un code 200 OK, alors qu'un code 406 Not Acceptable était attendu pour indiquer l'indisponibilité du produit.

Étapes de reproduction:

1. Effectuez un appel *POST* pour obtenir un jeton valide :

```
cy.request("POST", `${apiUrl}/login`, {
  username: "mer.ben@gmail.com",
  password: "Test1234",
}).then((response) => {
  Token = response.body.token;
});
```

2. Envoyez une requête *PUT* à */orders/add* avec une quantité supérieure à la disponibilité :

```
cy.request({
  method: "PUT",
  url: `${apiUrl}/orders/add`,
  headers: {
    Authorization: `Bearer ${Token}`,
  },
  body: {
    product: 3,
    quantity: 2,
```

```
},  
}).then((response) => {  
    expect(response.status).to.eq(406); // Vérifiez que l'erreur attendue est levée  
});
```

3.Observez le code HTTP retourné par l'API dans la réponse.

Résultat attendu: La requête doit retourner un code 406 Not Acceptable, indiquant que la quantité demandée dépasse la disponibilité en stock.

Résultat obtenu: La requête retourne un code 200 OK, suggérant que l'opération a été réussie, même si les stocks sont insuffisants.

Impact

Ce comportement peut induire les clients en erreur, leur laissant croire que le produit a été ajouté correctement, même si les stocks ne permettent pas d'honorer la demande. Cela pourrait entraîner des problèmes de gestion des commandes et de satisfaction client.

ID de l'incident: UI-003

Titre: Le produit reste visible dans le panier malgré une quantité invalide (quantité supérieure à la limite).

Description: Lorsqu'un utilisateur tente d'ajouter une quantité invalide (une quantité négative, ou une quantité supérieur à 20) d'un produit au panier, le produit est toujours visible dans le panier après la tentative d'ajout. Ce comportement est incorrect et peut prêter à confusion pour les utilisateurs.

Étapes de reproduction

1. Connectez-vous avec un utilisateur valide :
 - Identifiant : **test2@test.fr**
 - Mot de passe : **testtest**
2. Accédez à la page du produit spécifique.
3. Modifiez la quantité du produit en saisissant une valeur négative (-1).
4. Cliquez sur le bouton pour ajouter le produit au panier.
5. Accédez au panier pour vérifier si le produit est présent.
6. Revenir à la page du produit.
7. Modifiez ensuite la quantité à une valeur supérieure à 20 (21).
8. Cliquez à nouveau sur le bouton pour ajouter le produit au panier.
9. Accédez au panier pour vérifier si le produit est présent.
10. Le produit reste visible dans le panier, malgré la tentative d'ajouter une quantité invalide.

Résultat attendu

- Si une quantité invalide (négative) est saisie, et si la quantité dépasse la limite de nombre de produit autorisé qui est de 20, le produit ne doit pas être ajouté au panier et il ne doit pas apparaître dans la liste des produits du panier.
- L'interface doit afficher un message d'erreur pour indiquer que la quantité est invalide.

Résultat obtenu: Le produit reste visible dans le panier même après l'ajout avec une quantité invalide, ce qui ne correspond pas au comportement attendu.

Impact: Les utilisateurs peuvent ajouter un produit au panier même si la quantité saisie est invalide. Cela peut entraîner une mauvaise gestion des commandes et affecter l'expérience utilisateur.

ID de l'incident: API-004

Titre : Faille XSS (Injection de Script) détectée dans le champ "address" lors de la création d'une commande.

Étapes de reproduction :

Effectuer une requête **POST** vers l'**API /orders** avec les informations suivantes dans le corps de la requête :

```
{  
  "firstname": "meriam",  
  "lastname": "Ben",  
  "address": "<script>alert('XSS')</script>",  
  "zipCode": "95870",  
  "city": "Bezons"  
}
```

Comparer la valeur de **response.body.address** avec le script injecté (**<script>**).

Résultat attendu :

La réponse de l'API ne doit pas inclure le script injecté.

```
{ "firstname": "meriam",  
  "lastname": "Ben",  
  "address": "alert('XSS')",  
  "zipCode": "95870",  
  "city": "Bezons" }
```

Impact :

Vulnérabilité critique de type XSS (Cross-Site Scripting)

ID de l'incident: API-005

Titre : Faille XSS (Injection de Script) détectée dans le champ "Comment" lors de l'ajout d'un avis.

Étapes de reproduction :

Effectuer une requête **POST** vers **l'API /reviews** avec les informations suivantes dans le corps de la requête :

```
{  
  "title": "test avis",  
  "comment": "<script>alert('XSS')</script>",  
  "rating": 5  
}
```

Comparer la valeur de **response.body.comment** avec le script injecté (**<script>**).

Résultat attendu :

La réponse de l'API ne doit pas inclure le script injecté.

```
{  
  "title": "test avis",  
  "comment": "alert('XSS')",  
  "rating": 5  
}
```

Impact :

Vulnérabilité critique de type XSS (Cross-Site Scripting)

Niveau de criticité des incidents:

API-001/API-004/API-005: Haute

API-002: Moyenne à haute

UI-003: Moyenne

Avis sur les anomalies les plus critiques à corriger:

Les erreurs de failles XSS(API-004/API-005):

Le test révèle une faille critique de type XSS dans l'application. Une correction immédiate est nécessaire pour protéger l'application et ses utilisateurs contre les attaques potentielles.

Erreur HTTP 401 au lieu de 403 pour les requêtes non autorisées (API-001) :

Cette anomalie doit être corrigée immédiatement, car elle affecte la gestion de la sécurité de l'application et peut compromettre les autorisations d'accès aux API. Une

réponse erronée sur les permissions pourrait causer des failles de sécurité ou des erreurs de comportement qui peuvent être difficiles à détecter.

Produit visible dans le panier malgré son indisponibilité (API-002) :

L'incohérence dans le panier a un impact direct sur l'expérience utilisateur et peut causer des erreurs dans les commandes, ce qui est inacceptable dans un contexte commercial. Cette anomalie devrait être corrigée en priorité pour garantir une expérience d'achat fluide et sans erreur.

Confiance en version.

Bien que la version semble généralement fonctionnelle, les anomalies liées à la sécurité, la gestion des erreurs (authentification et permissions), ainsi qu'à la gestion du panier, sont critiques et doivent être corrigées avant tout déploiement en production. Le niveau de confiance est donc faible à modéré, Pour les raisons suivantes :

- Anomalies critiques identifiées.
- Impact direct sur l'expérience utilisateur

Conclusion:

Ces corrections sont essentielles pour garantir une application fiable, sécurisée et offrant une bonne expérience utilisateur. Une fois ces points adressés, la confiance dans la version pourrait être revue à la hausse.