

Rapport de testing

Remplissez ce document avec les informations indiquées :

- Avant d'effectuer vos tests, détaillez vos recommandations sur les scénarios qui méritent d'être testés (Section I) ;
- Complétez le bilan de campagne de test après avoir effectué vos tests à partir de vos recommandations (Section II).

I. Recommandations

A. Tests à automatiser

Détaillez ici les deux tests que vous avez désignés comme étant les plus critiques. Expliquez pourquoi et comment vous les avez choisis.

Pour cette campagne de test, j'ai désigné deux tests comme étant les plus critiques à automatiser : la **connexion** et le **panier**.

J'ai analysé les fonctionnalités principales de l'application et identifié celles qui ont le plus grand impact sur l'expérience utilisateur et la sécurité.

La **connexion** a été priorisée en raison de sa nature fondamentale et de son rôle critique dans l'accès aux autres fonctionnalités de l'application. La connexion implique des aspects critiques de sécurité, comme l'authentification des utilisateurs et la protection des données sensibles. Si la fonctionnalité de connexion présente des erreurs ou échoue, les utilisateurs ne pourront pas accéder à leur compte ou pourraient rencontrer des délais de connexion, des échecs intermittents, ou des messages d'erreur incorrects, ce qui affectera l'expérience utilisateur et pourrait entraîner une perte de clients. La connexion est une fonctionnalité de base du processus d'achat. Si la connexion échoue, l'utilisateur ne peut pas effectuer le processus d'achat.

Le processus de gestion du **panier** a été sélectionné en raison de son importance critique dans le cycle d'achat, où d'éventuelles erreurs pourraient rendre le processus d'achat difficile ou impossible.

La gestion du panier implique plusieurs interactions complexes, comme l'ajout, la suppression et la mise à jour de produits, ainsi que l'intégration avec les systèmes de paiement.

Toute anomalie dans le processus de gestion du panier peut directement entraîner une perte de ventes.

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 :

- Qu'est-ce que vous conseillez pour les tests restants ?

Je conseille d'automatiser les tests pour l'affichage des produits et de considérer également les tests de performance, de compatibilité et de sécurité..

- Valent-ils la peine d'être automatisés ou pas ?

Oui, ces tests valent la peine d'être automatisés.

- Pour quelles raisons ?

Affichage des produits: Assurer que les utilisateurs peuvent voir et interagir correctement avec les produits est essentiel pour les ventes. Des erreurs ici peuvent directement impacter le revenu, notamment en cas de manque d'images, d'erreurs typographiques, de problèmes d'alignement ou d'autres défauts d'affichage qui peuvent nuire à l'attrait visuel, à la clarté des informations fournies aux clients, et à l'impression générale laissée sur eux.

Tests de performance: Un site performant améliore l'expérience utilisateur et peut empêcher les abandons de panier dus à des temps de chargement lents. Il est essentiel de vérifier comment le site gère les charges élevées et le nombre important d'utilisateurs simultanés. En cas de trafic élevé ou de charge importante sur les serveurs, le site doit rester stable et réactif, évitant ainsi les blocages ou les ralentissements significatifs pendant la navigation. Ces tests de performance permettent d'identifier les points faibles et d'optimiser le site afin d'assurer une expérience utilisateur fluide même dans des conditions d'utilisation intensives.

Tests de compatibilité : Les tests de compatibilité visent à vérifier que l'application fonctionne correctement sur divers dispositifs, navigateurs et systèmes d'exploitation. Cela inclut la validation du design responsive pour assurer une bonne adaptation aux différentes tailles d'écran, la compatibilité uniforme sur différents navigateurs populaires comme Chrome, Firefox, Safari, Edge, ainsi que l'évaluation des performances et de l'accessibilité pour garantir une expérience utilisateur fluide et accessible.

Tests de sécurité : Les tests de sécurité sont cruciaux pour protéger les données des utilisateurs et assurer l'intégrité du site. Il est important de

vérifier que les utilisateurs ne peuvent pas accéder à des informations sensibles ou effectuer des actions non autorisées.

Tests de sécurité recommandés :

Test de l'authentification et de l'autorisation :

Vérifier que les utilisateurs non authentifiés ne peuvent pas accéder aux pages protégées par des contrôles d'accès.

Test de la confidentialité des données :

Vérifier que les informations de panier ne sont pas accessibles en utilisant un lien ou un identifiant d'utilisateur non valide.

Test de l'unicité du panier :

Vérifier que chaque panier est propre à l'utilisateur connecté et ne peut pas être modifié par un autre utilisateur.

Test de la sécurité des API :

Vérifier que les appels API retournent des erreurs appropriées pour des requêtes non authentifiées et que les données sensibles sont correctement protégées.

II. Bilan de campagne de validation : tests automatisés

À remplir après avoir effectué vos tests

Remplissez ce document avec les informations indiquées. Vous pouvez vous référer au [bilan de campagne de test de Marie](#).

Document revu par :

Nom	Fonction	Version	Signature
Ruxandra Strat	<i>QA Engineer</i>	<i>1.0.0</i>	

Contexte :

Rappelez le contexte de votre projet. Voici quelques questions pour vous guider dans la rédaction d'un résumé court :

- Cette campagne a été effectuée en vue de quel projet commercial ?
Cette campagne de test a été effectuée dans le cadre du lancement de notre site de vente en ligne pour Eco Bliss Bath, spécialisé dans la vente de produits de beauté écoresponsables dont le produit principal est un savon solide.
- Combien de personnes travaillent sur cette campagne ? Quels sont leurs rôles au sein de l'entreprise/de l'équipe ?
Est-ce des tests automatiques ou des tests manuels ?
Trois personnes ont travaillé sur cette campagne :

Marie : Testeuse manuelle, responsable des tests manuels de la première version du site.

Fabio : CTO, responsable de la supervision technique et de la validation des tests.

Ruxandra Strat : QA Engineer, responsable des tests automatiques et de l'analyse des résultats des tests manuels.

Les tests effectués ont été principalement manuels, mais des tests automatiques ont également été réalisés, incluant des tests d'API, des tests fonctionnels et des smoke tests.

- Combien de temps avez-vous passé dessus si c'était une des contraintes ?
2 semaines
- Si c'est web, sur quel navigateur et quelle version ?
Chrome V126
- Si c'est système, sur quel système et quelle version ?
Windows 10 Pro Version 22H2

Objectif(s) :

Listez ici l'objectif principal de votre campagne de test, ainsi que des objectifs mineurs (voir template du bilan de campagne du cours COURSE NAME pour un exemple).

Objectif principal :

L'objectif principal de cette campagne de test est de s'assurer que la première version du site de vente en ligne est fonctionnelle et prête pour le lancement public.

Objectifs mineurs :

Tester le processus complet d'achat en ligne, depuis la sélection des produits jusqu'à la confirmation de la commande.

Assurer la performance du site sous différentes charges de trafic.

S'assurer que l'expérience utilisateur est fluide et intuitive.

S'assurer de la sécurité des données des utilisateurs.

Tests effectués :

Complétez cette section avec le plus d'informations possible sur les tests que vous avez effectués. Vous pouvez vous référer au bilan de campagne de Marie comme exemple. L'idée est de détailler vos tests de façon à ce que n'importe quelle personne puisse les reproduire.

Test 1 : Les Tests d'API

Objectif : Vérifier la bonne fonctionnalité des endpoints de l'API en testant des scénarios typiques de demandes GET et POST.

GET Requests:

- **Requête pour obtenir les données confidentielles d'un utilisateur avant connexion**

URL : <http://localhost:8081/orders>

Objectif : Ce test vérifie que l'accès aux données de l'utilisateur (le panier, en l'occurrence) nécessite une authentification. En accédant à cet endpoint sans être connecté, l'API doit renvoyer une erreur 403 Forbidden, indiquant que l'utilisateur n'a pas les droits nécessaires pour accéder aux informations.

Avant le test : Assurez-vous de ne pas être connecté.

Ce qu' on vérifie : Le code de statut devrait être 403 (Forbidden), indiquant que l'utilisateur n'a pas les bons droits. Cependant, le test retourne une erreur 401 (Unauthorized), ce qui signifie que l'utilisateur n'est pas authentifié.

- **Requête pour obtenir la liste des produits dans le panier**

URL : <http://localhost:8081/orders>

Objectif : Ce test s'assure que on peut récupérer les détails du panier de l'utilisateur connecté.

Avant le test : Connectez-vous avec des informations valides.

Ce qu' on vérifie : La réponse doit contenir les informations des produits présents dans le panier, telles que le nom du produit, la quantité, etc. On doit obtenir un code de statut 200 OK.

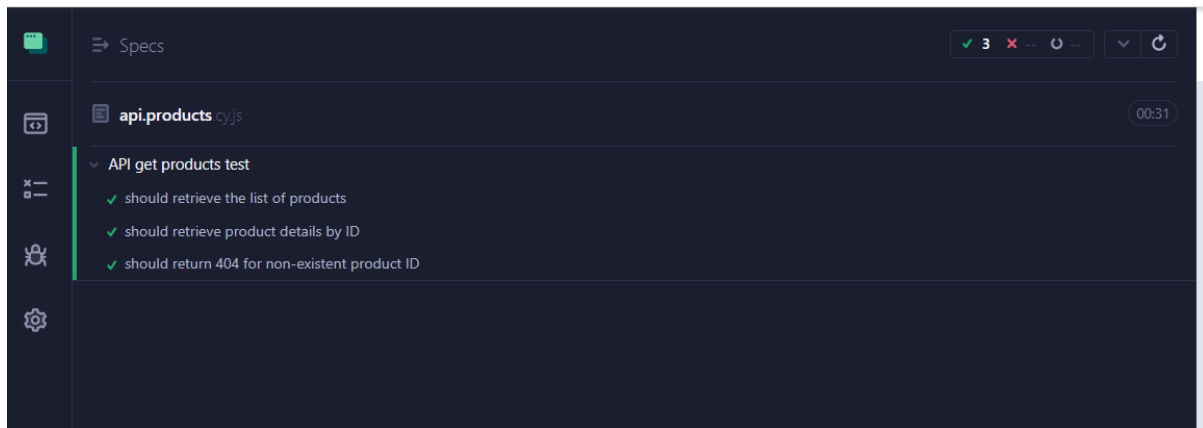
- **Requête pour obtenir les détails d'un produit spécifique**

URL : <http://localhost:8081/products/{id}>

Objectif : Ce test vérifie que vous pouvez obtenir les informations détaillées d'un produit en utilisant son identifiant.

Avant le test : Ayez l'identifiant d'un produit valide.

Ce qu' on vérifie : La réponse doit inclure les informations du produit, comme le nom, la description, le prix, stock et une image du produit. On doit obtenir un code de statut 200 OK.



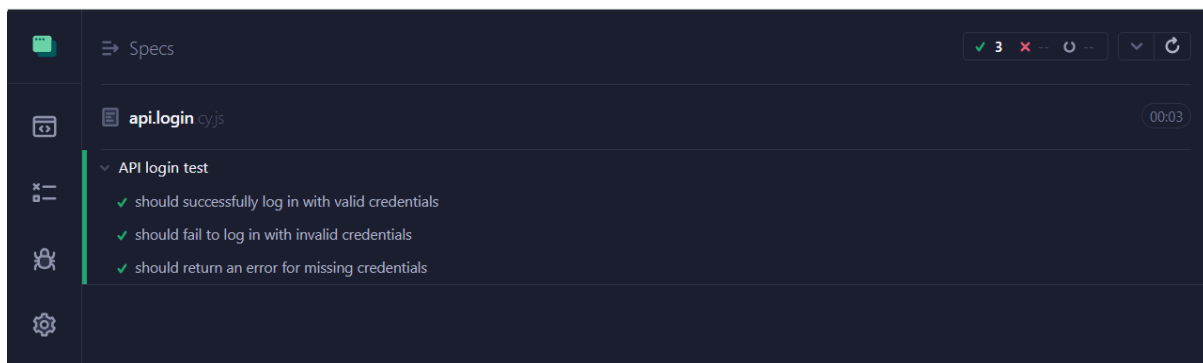
- **Authentication (Login)**

URL : <http://localhost:8081/login>

Objectif : Vérifier que vous pouvez vous connecter avec des informations d'utilisateur valides et recevoir un token d'authentification.

Avant le test : Essayez de vous connecter avec un utilisateur connu et un utilisateur inconnu.

Ce qu' on vérifie : Pour un utilisateur connu, vous devriez obtenir un code 200 OK et un token d'authentification. Pour un utilisateur inconnu, vous devriez obtenir un code 401 Unauthorized.



- **Ajouter un produit disponible au panier**

URL : <http://localhost:8081/orders/add>

Objectif : Tester que vous pouvez ajouter un produit disponible au panier en spécifiant l'identifiant du produit et la quantité.

Avant le test : Assurez-vous que le produit est en stock.

Ce qu' on vérifie : La réponse doit confirmer l'ajout du produit au panier avec un code de statut 200 OK.

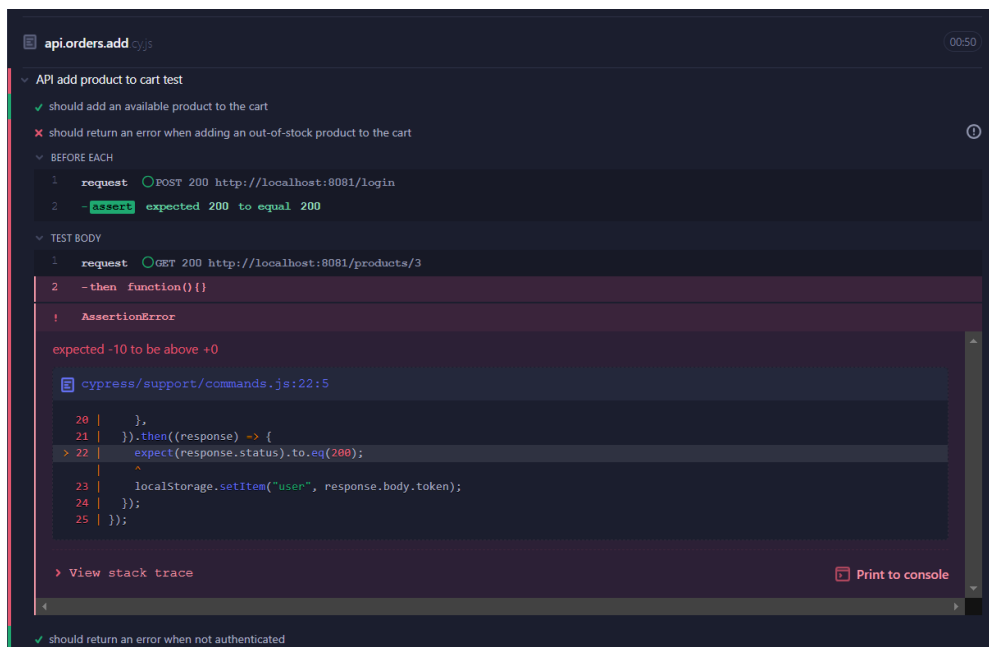
- **Ajouter un produit en rupture de stock au panier**

URL : <http://localhost:8081/orders/add>

Objectif : Vérifier que vous ne pouvez pas ajouter un produit qui est en rupture de stock au panier.

Avant le test : Assurez-vous que le produit est en rupture de stock.

Ce qu' on vérifie : Vous devriez recevoir une erreur, généralement un code 400 Bad Request, ce qui indique que le produit ne peut pas être ajouté au panier en raison du stock insuffisant.



```
api.orders.add.cys
00:50

API add product to cart test
  ✓ should add an available product to the cart
  ✗ should return an error when adding an out-of-stock product to the cart

  BEFORE EACH
    1 request POST 200 http://localhost:8081/login
    2 - assert expected 200 to equal 200

  TEST BODY
    1 request GET 200 http://localhost:8081/products/3
    2 - then function() {}

    ! AssertionError
    expected -10 to be above +0

    cypress/support/commands.js:22:5

    20 |   },
    21 |   }).then((response) => {
    22 |     expect(response.status).to.eq(200);
    23 |     localStorage.setItem("user", response.body.token);
    24 |   });
    25 | });

  > View stack trace
  Print to console

  ✓ should return an error when not authenticated
```

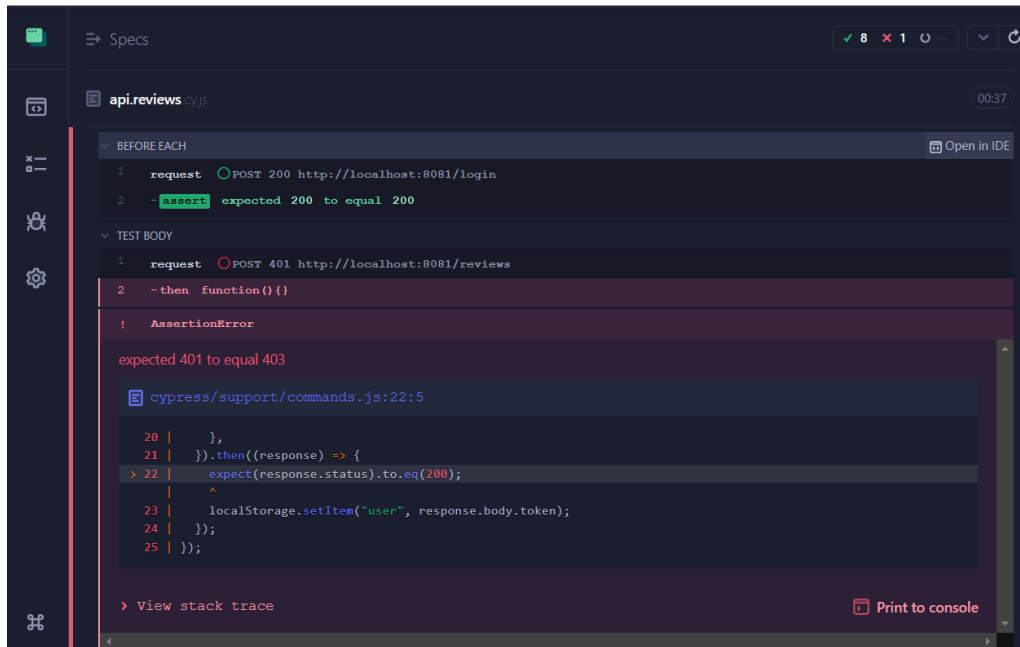
- **Ajouter un avis sur un produit**

URL : <http://localhost:8081/reviews>

Objectif : Tester que vous pouvez soumettre un avis pour un produit spécifique.

Avant le test : Assurez-vous que vous êtes connecté

Ce qu' on vérifie : La réponse devrait confirmer que l'avis a été ajouté avec un code 200 OK.



Test 2 : Connexion

Objectif : Vérifier le processus de connexion et l'accès aux fonctionnalités protégées.

Tester la connexion via le Frontend

Étapes :

Cliquer sur le bouton de connexion : Accéder à la page de connexion via le frontend.

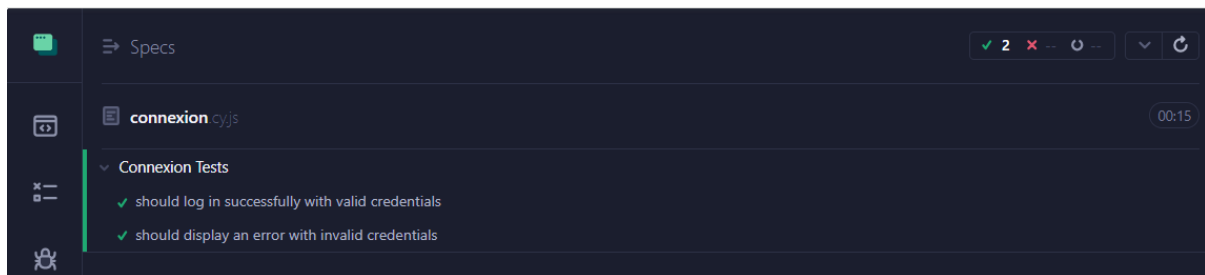
Remplir le formulaire :

- Email : Entrer test2@test.fr.
- Mot de passe : Entrer testtest.

Vérifier l'état de connexion : Vous devez être redirigé vers la page d'accueil et le bouton « Mon panier » est visible dans le Menu de navigation, ce qui indique que vous êtes connecté avec succès.

Avant le test : Assurez-vous que vous avez les bonnes informations de connexion.

Ce qu'on vérifie : La connexion doit réussir et vous devez voir le bouton « Mon panier » sur la page, ce qui confirme que vous êtes bien connecté.



Test 3 : Panier

Objectif : Vérifier les fonctionnalités liées au panier de l'utilisateur, y compris l'ajout de produits et les vérifications de stock.

- Ajouter un produit disponible au panier

Étapes :

Connectez-vous : Assurez-vous que vous êtes connecté avec les informations données précédemment.

Cliquer sur un produit : Accéder à la page de détails du produit.

Vérifier le stock : Assurez-vous que le stock est supérieur à 0.

Ajouter au panier : Cliquer sur le bouton pour ajouter le produit au panier.

Vérifier le panier : Accéder à la page du panier et assurez-vous que le produit a été ajouté.

Retourner sur la page du produit et vérifier que le stock a enlevé le nombre de produits qui sont dans le panier.

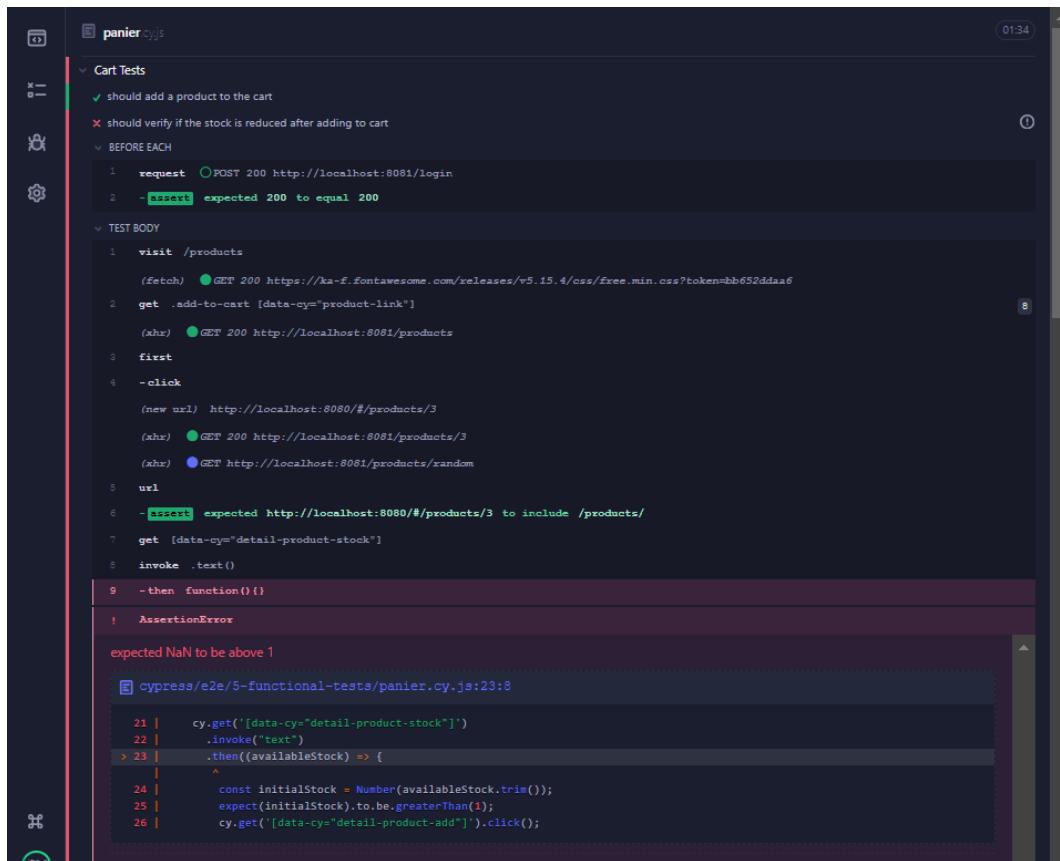
Avant le test : Assurez-vous que le produit a un stock suffisant.

Ce qu' on vérifie :

Le produit doit apparaître dans le panier.

La quantité du produit doit être correctement ajustée.

Le stock du produit doit être mis à jour pour refléter l'ajout au panier.



```
panier cy.js 01:34
✓ Cart Tests
  ✓ should add a product to the cart
  ✗ should verify if the stock is reduced after adding to cart
    BEFORE EACH
      1 request POST 200 http://localhost:8081/login
      2 -assert expected 200 to equal 200
    TEST BODY
      1 visit /products
      (fetch) GET 200 https://ka-f.fontawesome.com/releases/v5.15.4/css/free.min.css?token=bb652dda6
      2 get .add-to-cart [data-cy="product-link"]
      (xhr) GET 200 http://localhost:8081/products
      3 first
      4 -click
      (new url) http://localhost:8080/#/products/3
      (xhr) GET 200 http://localhost:8081/products/3
      (xhr) GET http://localhost:8081/products/random
      5 url
      6 -assert expected http://localhost:8080/#/products/3 to include /products/
      7 get [data-cy="detail-product-stock"]
      8 invoke .text()
      9 -then function() {}
      ! AssertionError
      expected NaN to be above 1
      cypress/e2e/5-functional-tests/panier.cy.js:23:8
      21 |   cy.get('[data-cy="detail-product-stock"]')
      22 |     .invoke("text")
      > 23 |     .then((availableStock) => {
          |       ^
      24 |       const initialStock = Number(availableStock.trim());
      25 |       expect(initialStock).to.be.greaterThan(1);
      26 |       cy.get('[data-cy="detail-product-add"]').click();
```

➤ Tester les limites de quantité

Étapes :

Essayez d'ajouter une quantité négative : Testez si le système accepte des quantités invalides.

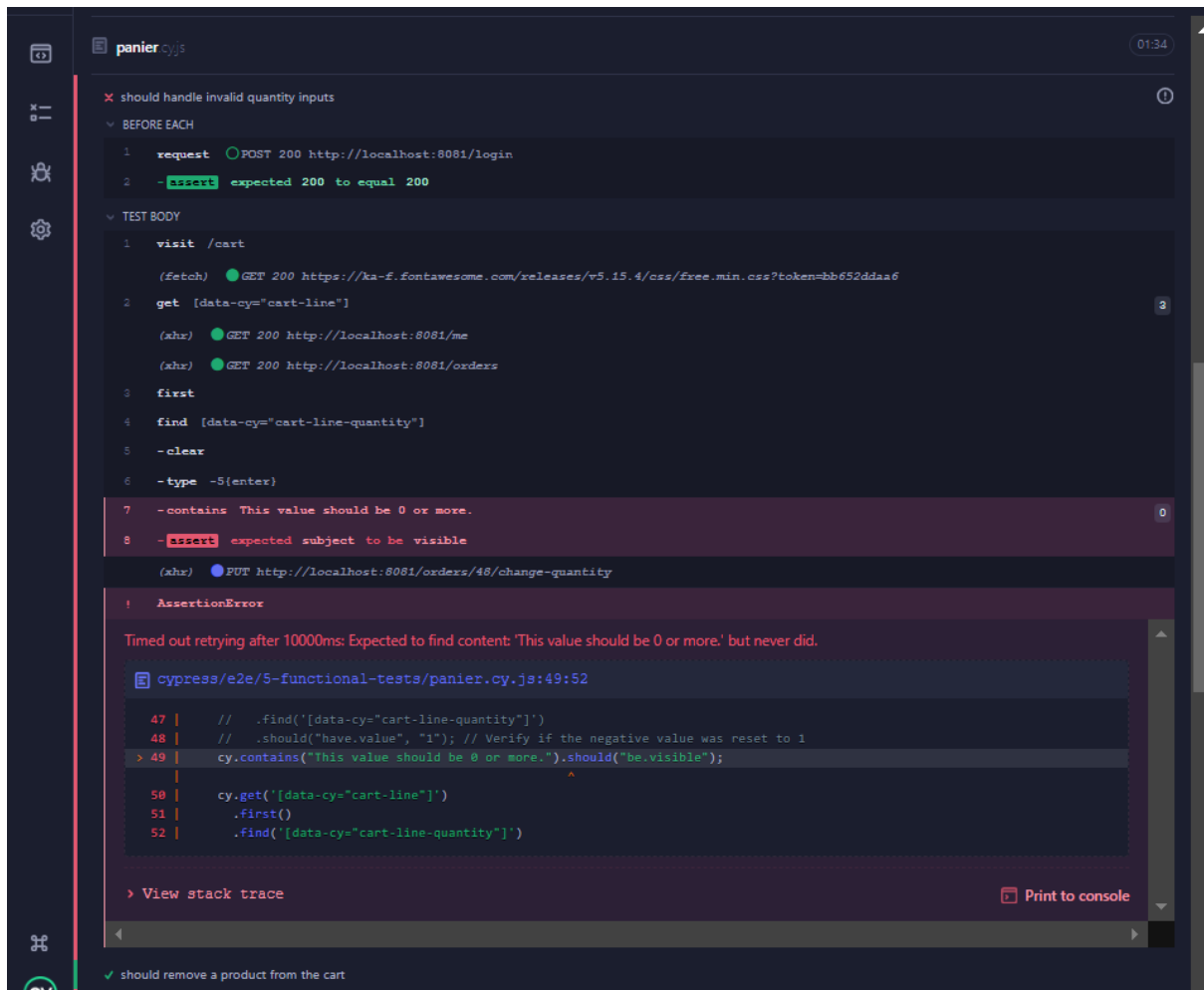
Essayez d'ajouter une quantité supérieure au stock : Testez si le système empêche l'ajout de plus de produits que ce qui est disponible en stock.

Avant le test : Testez avec une quantité négative et une quantité qui dépasse le stock.

Ce qu' on vérifie :

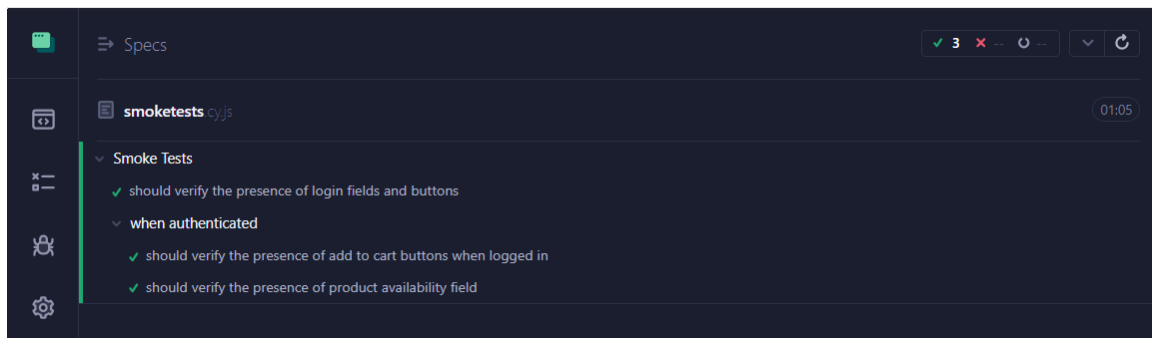
Le système doit refuser les quantités invalides avec un message d'erreur approprié.

Le produit ne doit pas être ajouté au panier si la quantité dépasse le stock disponible.



Test 4 : Smoke tests

- vérifier la présence des champs et boutons de connexion ;
- vérifier la présence des boutons d'ajout au panier quand vous êtes connecté ;
- vérifier la présence du champ de disponibilité du produit.



Résultats de tests

Notez ici tous les résultats obtenus

Comparez le résultat obtenu avec le résultat attendu : est-ce qu'il y a des tests qui sont en échec ? Si oui, pourquoi ?

N'hésitez pas à détailler les étapes nécessaires pour reproduire le défaut ou faites référence aux rapports de bugs que vous avez créés avec les informations nécessaires pour les reproduire.

Soyez objectif et précis.

Résultats de tests

Cas de test	Étapes	Attendu	Résultat	Commentaire
Connexion	Accéder à la page d'accueil du site	La page d'accueil s'affiche	OK	
	Cliquez sur le bouton de connexion	La page de connexion avec le formulaire s'affiche	OK	
	Entrer "test2@test.fr" dans le champ de l'email	On est connecté et on voit le bouton panier	OK	
	Entrer "testtest" comme mot de passe			
	Appuyer sur « Se connecter »			
Ajout de produit disponible au panier	Verifier si on est connectés avec les infos données précédemment	On est connecté et on voit le bouton « Mon panier »	OK	
	Accéder à la page Produits du menu	La page de Produits s'affiche	OK	
	Cliquez sur un des produits disponibles	La page avec les details de ce produit s'affiche	OK	
	Cliquez sur le bouton ajouter pour ajouter le produit disponible au panier	Le produit a été ajouté au panier	OK	
	Retournez sur la page du produit et vérifiez que le stock a enlevé le nombre de produits qui sont dans le panier	Le stock a enlevé le nombre de produits qui sont dans le panier	KO	Le stock est negatif
	Entrez un chiffre négatif dans le champ quantité	Pour un nombre négatif entré dans le champ de quantité,	KO	Aucun message d'erreur n'apparaît.

		un message d'erreur doit apparaître.		
Ajout de produit indisponible au panier (stock insuffisant)	Vérifier si on est connectés avec les infos données précédemment	On est connecté et on voit le bouton « Mon panier »	OK	
	Accéder à la page Produits du menu	La page de Produits s'affiche	OK	
	Cliquez sur un des produits indisponibles et cliquez sur le bouton ajouter pour ajouter au panier	Le stock doit être supérieur à 0 pour pouvoir être ajouté	KO	Le produit a été ajouté au panier même lorsque le stock est inférieur à 0 / insuffisant
	Retourner à la page Produits du menu	La page de Produits s'affiche	OK	
	Cliquez sur un des produits disponibles et entrez un chiffre supérieur à 20 dans le champ quantité (1000 par exemple)	Un message d'erreur doit apparaître La requête doit échouer avec un status 400 (Bad request).	KO	Aucun message d'erreur n'apparaît, la quantité reste celle entrée, même s'il n'y a pas assez de stock Status 200, le produit est ajouté au panier malgré le stock insuffisant.
Supprimer un produit du panier	Accéder à la page Mon panier	La page de Mon panier s'affiche	OK	
	Vérifier si nous avons des produits dans le panier	Nous avons des produits dans le panier	OK	
	Vérifier si l'image d'une poubelle est visible	L'image d'une poubelle est visible	OK	
	Cliquer sur cette image de poubelle	Au clic, l'élément est supprimé	OK	

Champs et boutons de connexion	Accéder à la page de connexion du site	La page de connexion avec le formulaire s'affiche	OK	
	Vérifier la présence des champs et boutons de connexion	Le champs Email et Mot de passe existent Le bouton Se connecter existe Le lien S'inscrire existe	OK	
Boutons d'ajout au panier	Accéder à la page de connexion du site	La page de connexion avec le formulaire s'affiche	OK	
	Entrer "test2@test.fr" dans le champ de l'email	On est connecté	OK	
	Entrer "testtest" comme mot de passe			
	Accéder à la page de Produits du site	La page de Produits s'affiche	OK	
	Choisir un produit et appuyer sur Consulter	On voit la page de produit en appuyant sur Consulter	OK	
	Vérifier la présence del bouton « Ajouter au panier »	Le bouton « Ajouter au panier » est présent.	OK	
Champ de disponibilité du produit	Accéder à la page de Produits du site	La page de Produits s'affiche	OK	
	Choisir un produit et appuyer sur Consulter	On voit la page de produit en appuyant sur Consulter	OK	
	Vérifier la présence du champ de disponibilité du produit (stock)	Le champ de disponibilité du produit (stock) est présent.	OK	
Ajouter un avis	Accéder à la page Avis du site	La page Avis s'affiche	OK	
	Vérifier la présence des autres avis	On voit autres avis	OK	

	Entrer " Super " dans le champ de titre	Le champ de titre contient "Super"	OK	
	Entrer " Je l'aime bien, j'achèterai à nouveau ! " dans le champ commentaire	Le champ de commentaire contient " Je l'aime bien, j'achèterai à nouveau ! "	OK	
	Sélectionner une note (ex: 5 étoiles)	La note est sélectionnée (5 étoiles)	OK	
	Cliquer sur le bouton "Publier"	L'avis est soumis et apparaît dans la liste des avis	OK	
Ajouter un avis avec script tag dans le champ description	Accéder à la page Avis du site	La page Avis s'affiche	OK	
	Entrer "Test XSS" dans le champ de titre	Le champ de titre contient "Test XSS"	OK	
	Entrer "<script>alert('XSS Attack');</script>" dans le commentaire	Le champ de commentaire contient "<script>alert('XSS Attack');</script>"	OK	
	Sélectionner une note (ex: 3 étoiles)	La note est sélectionnée (3 étoiles)	OK	
	Cliquer sur le bouton " Publier "	Un message d'erreur doit apparaître La requête doit échouer avec un status 403 (Forbidden).	KO	L'avis est envoyé avec le statut 200 et aucun message d'erreur n'apparaît

[Si anomalie, rapport d'incident]

Si vous avez trouvé des anomalies, vous devez écrire le rapport d'incident ici.

- **Rapport d'Incident :** [Anomalie dans l'API pour l'Ajout d'un Produit au Panier](#)

Description de l'Incident

Lors des tests d'API pour l'ajout d'un produit au panier, il a été constaté que la méthode `PUT` est utilisée au lieu de `POST`, contrairement aux pratiques RESTful courantes et aux attentes documentées.

Impact

Cette anomalie pourrait entraîner des erreurs dans la mise en œuvre des tests et une mauvaise compréhension des méthodes HTTP par les développeurs.

Détail du Problème

- Méthode HTTP documentée : `POST` est la méthode attendue pour l'ajout d'un produit au panier.
- Méthode HTTP observée : `PUT` est utilisée, ce qui est atypique pour l'ajout de ressources dans les pratiques RESTful.

Recommandations

- Réviser la documentation Swagger pour s'assurer que les méthodes HTTP sont correctes.
- Vérifier l'implémentation API pour s'assurer que `PUT` est bien justifié ou modifier l'API pour utiliser `POST` si cela est plus conforme aux pratiques RESTful.

Conclusion

L'anomalie nécessite une révision des spécifications API pour corriger la méthode HTTP utilisée pour l'ajout de produits au panier.

- **Rapport d'Incident :** [Erreur 401 au lieu de 403 pour l'Accès aux Données Confidentielles d'un Utilisateur](#)

Description de l'Incident

Lors des tests de l'API pour l'endpoint `/orders`, il a été constaté que le code de statut HTTP retourné est 401 (`Unauthorized`) au lieu de 403

(`Forbidden`). Le code de statut approprié pour une tentative d'accès non autorisée sans authentification devrait être 403, indiquant que l'utilisateur n'a pas les permissions nécessaires.

Impact

Cette anomalie peut entraîner :

- Confusion dans les tests de sécurité et de permission.
- Problèmes de Sécurité : Une gestion incorrecte des erreurs peut mener à des failles de sécurité.
- Incohérence entre la documentation API et l'implémentation.

Détail du Problème

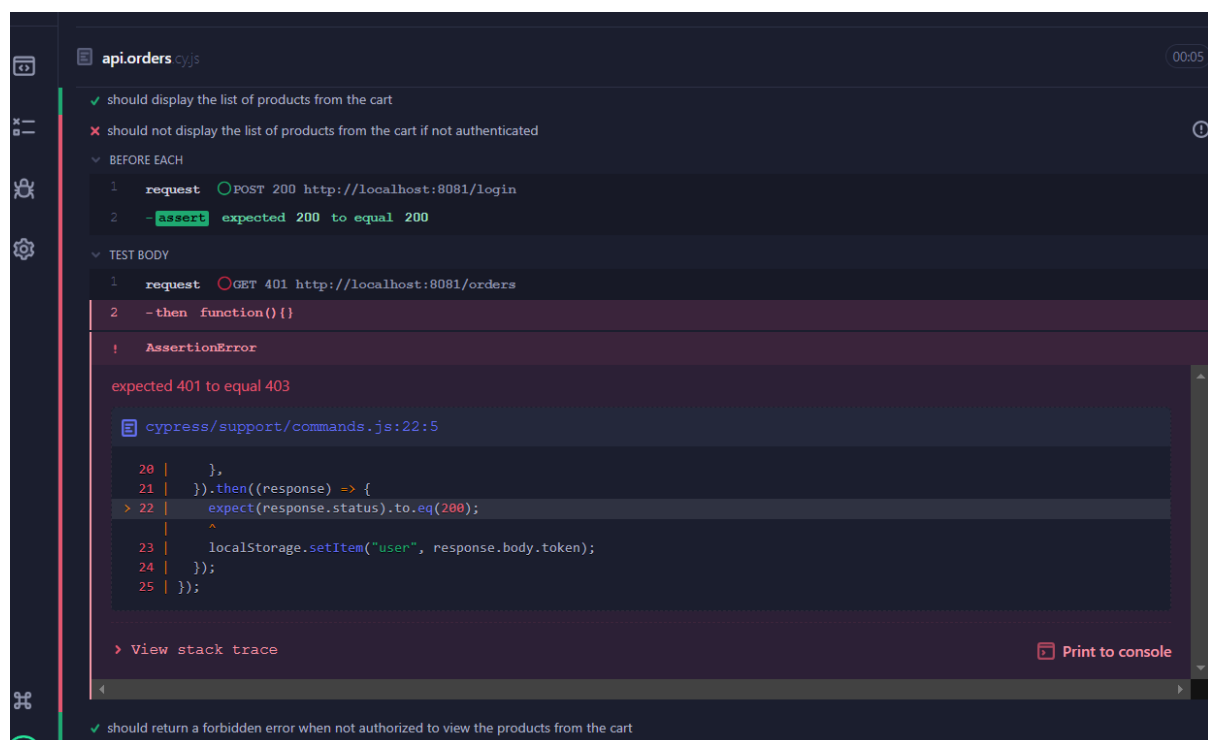
- Endpoint : `/orders`
- Code de Statut Attendu pour Accès Non Autorisé : 403 (Forbidden)
- Code de Statut Observé pour Accès Non Autorisé : 401 (`Unauthorized`)

Recommandations

Revérifier les spécifications Swagger pour confirmer les codes de statut appropriés pour les différents scénarios

Mise à jour de l'API pour s'assurer qu'elle renvoie le code de statut 403 pour les tentatives d'accès non autorisées sans authentification.

S'assurer que les tests sont alignés avec les spécifications pour garantir une gestion correcte des erreurs et des permissions.



The screenshot shows the Cypress test runner interface. The test file is named `api.orders.cypress`. The test suite includes two tests: "should display the list of products from the cart" (passed) and "should not display the list of products from the cart if not authenticated" (failed). The failed test is expanded, showing the test body. The test body includes a "BEFORE EACH" hook that performs a POST request to `http://localhost:8081/login` and an assertion that the response status is 200. The main test body performs a GET request to `http://localhost:8081/orders` and then a function that asserts the response status is 403. The assertion fails because the response status is 401. The error message is "AssertionError: expected 401 to equal 403". The stack trace shows the error occurred in `cypress/support/commands.js:22:5`. The code snippet for the stack trace is as follows:

```
20 | },
21 | }).then((response) => {
22 |   expect(response.status).to.eq(200);
   |   ^
23 |   localStorage.setItem("user", response.body.token);
24 | });
25 | });
```

At the bottom of the test runner, there is a "View stack trace" link and a "Print to console" button. The test runner also shows a "00:05" timer in the top right corner.

- **Rapport d'Incident : Ajout de Produit Indisponible au Panier (Stock Insuffisant)**

Description de l'Incident

Lors des tests de l'ajout de produits au panier via l'interface utilisateur, il a été constaté que des produits avec un stock insuffisant peuvent être ajoutés au panier. De plus, lorsqu'une grande quantité est entrée pour un produit (1000), le produit est toujours ajouté au panier, même si le stock est insuffisant.

Impact

Cette anomalie pourrait permettre aux utilisateurs de commander des quantités de produits au-delà du stock disponible, ce qui pourrait entraîner des problèmes de gestion des stocks et de traitement des commandes. De plus, cela pourrait également entraîner des plaintes et des réclamations des clients, qui seraient frustrés de ne pas recevoir les produits qu'ils ont vus sur le site et qu'ils ont pu ajouter à leur panier et commander.

Détail du Problème

Ajout de produits avec un stock insuffisant :

- Comportement attendu : Si une commande dépasse le stock disponible, l'ajout au panier doit échouer
- Comportement observé : Le produit est ajouté au panier même lorsque la quantité demandée dépasse le stock disponible.

Recommandations

L'anomalie identifiée nécessite une révision et une correction de l'implémentation de l'API et de l'interface utilisateur pour s'assurer qu'un produit ne peut pas être ajouté au panier si le stock est insuffisant.

Mettre à jour la logique côté client si la quantité entrée dépasse le stock disponible.

S'assurer que les requêtes échouent avec un status approprié (400) si le stock est insuffisant pour la quantité demandée.

Ajouter des tests automatisés pour vérifier que les produits ne peuvent pas être ajoutés au panier lorsque le stock est insuffisant.

- **Rapport d'Incident : Stoc Négatif**

Description de l'Incident

Lors des tests, il a été constaté que le stock des produits peut devenir négatif après l'ajout de quantités au panier qui dépassent le stock disponible. De plus, le stock des produits apparaît déjà avec un chiffre négatif directement sur la page du produit, avant même que l'utilisateur connecté n'ajoute les produits au panier.

Impact

Cette anomalie pourrait entraîner une mauvaise gestion des stocks et des erreurs dans le traitement des commandes, ainsi que des plaintes et des réclamations des clients.

Détail du Problème

Comportement attendu : Le stock doit être réduit correctement après l'ajout d'un produit au panier. Le stock ne doit jamais être négatif (il doit être supérieur à 0 pour qu'un produit puisse être ajouté au panier).

Comportement observé : Le stock de certains produits apparaît avec un chiffre négatif sur la page du produit. Le stock devient négatif après l'ajout de produits au panier.

Recommandations

- Correction du Code de Gestion des Stocks : Vérifier et corriger le code qui gère la mise à jour des stocks pour s'assurer que les quantités ne deviennent jamais négatives.
- Validation des Entrées : Ajouter des validations pour empêcher l'ajout de quantités supérieures au stock disponible.
- Mise à Jour des Tests : Modifier les tests pour s'assurer qu'ils couvrent tous les scénarios de gestion des stocks
- Affichage des Messages d'Erreur : S'assurer que des messages d'erreur appropriés sont affichés à l'utilisateur si la quantité demandée dépasse le stock disponible.

```
Specs
api.orders.add cy.js 00:22
  ✓ API add product to cart test
    ✓ should add an available product to the cart
    ✗ should return an error when adding an out-of-stock product to the cart
      BEFORE EACH
        1 request POST 200 http://localhost:8081/login
        2 -assert expected 200 to equal 200
      TEST BODY
        1 request GET 200 http://localhost:8081/products/3
        2 -then function() {}
        ! AssertionError
        expected -16 to be above +0
        Cypress/support/commands.js:22:5
        20 | },
        21 | }).then((response) => {
        > 22 | expect(response.status).to.eq(200);
           | ^
        23 |     localStorage.setItem("user", response.body.token);
        24 |   });
        25 | });
        > View stack trace
        Print to console
      ✓ should return an error when not authenticated
```

```
panier cy.js 01:04
  ✗ should verify if the stock is reduced after adding to cart
    BEFORE EACH
      1 request POST 200 http://localhost:8081/login
      2 -assert expected 200 to equal 200
    TEST BODY
      1 visit /products
      (fetch) GET 200 https://ka-f.fontawesome.com/releases/v5.15.4/css/free.min.css?token=bb652dda6
      2 get .add-to-cart [data-cy="product-link"]
      (xhr) GET 200 http://localhost:8081/products
      3 first
      4 ~click
      (new url) http://localhost:8080/#/products/3
      (xhr) GET http://localhost:8081/products/3
      (xhr) GET http://localhost:8081/products/random
      5 url
      6 -assert expected http://localhost:8080/#/products/3 to include /products/
      7 get [data-cy="detail-product-stock"]
      8 invoke .text()
      9 -then function() {}
      ! AssertionError
      expected NaN to be above +0
      Cypress/e2e/5-functional-tests/panier.cy.js:23:8
      21 | cy.get('[data-cy="detail-product-stock"]')
      22 | .invoke("text")
      > 23 | .then((availablestock) => {
         | ^
      24 |     const initialStock = Number(availablestock.trim());
      25 |     expect(initialStock).toBeGreaterThan(0);
      26 |     cy.get('[data-cy="detail-product-add"]').click();
```

- **Rapport d'Incident :** Aucun Message d'Erreur pour une Quantité Négative Saisie par l'Utilisateur dans le Champ de Quantité

Description de l'Incident:

Lorsqu'un utilisateur entre une quantité négative dans le champ de quantité lors de l'ajout d'un produit au panier, aucun message d'erreur ne s'affiche pour signaler que la quantité est invalide.

Impact :

Problèmes de Validation des Données
Erreurs de Traitement des Commandes
Mauvaise Expérience Utilisateur

Détail du Problème:

Comportement Attendu : Affichage d'un message d'erreur pour une quantité négative

Comportement Observé : Aucun message d'erreur n'apparaît pour une quantité négative.

Recommandations:

- Ajout de la Validation des Quantités : Implémentez une validation pour rejeter les valeurs négatives dans le champ de quantité et afficher un message d'erreur approprié.
 - Amélioration de l'Interface Utilisateur : Assurez-vous que le message d'erreur est visible et clair pour l'utilisateur lorsqu'une quantité invalide est saisie.
-
- **Rapport d'Incident :** Le test pour un ID de produit inexistant retourne le code 404 dans la console, mais aucun message d'erreur n'apparaît sur la page.

Description de l'Incident :

Le test API pour un ID de produit inexistant renvoie le code d'état 404 dans la console, ce qui est correct. Cependant, aucun message d'erreur approprié n'est affiché sur la page pour informer l'utilisateur que le produit n'existe pas.

Impact :

Cette anomalie peut entraîner une mauvaise expérience utilisateur, car les utilisateurs ne reçoivent pas d'informations claires lorsqu'ils essaient d'accéder à un produit qui n'existe pas. Cela pourrait également mener à des plaintes de la part des utilisateurs et des confusions quant à l'existence des produits.

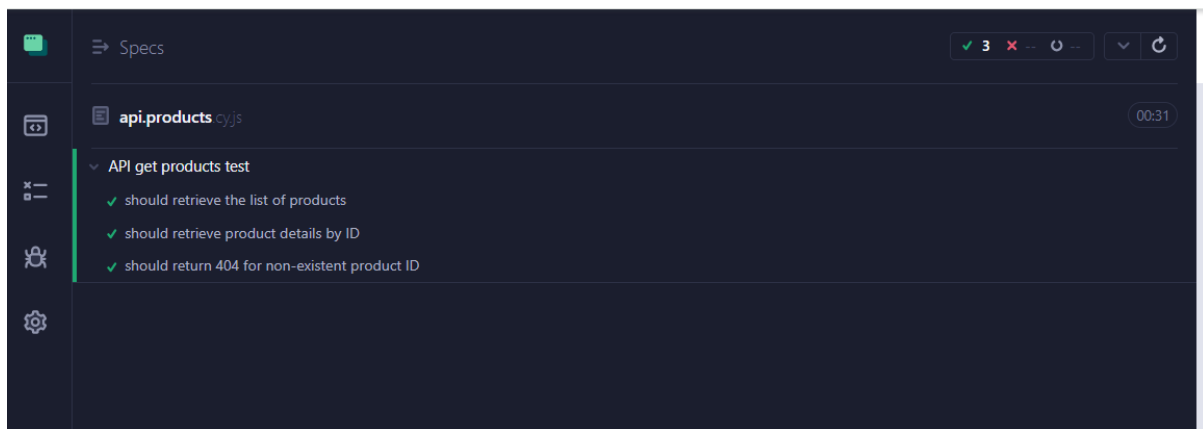
Détail du Problème :

Comportement attendu : Lorsqu'un utilisateur essaie d'accéder à un produit avec un ID inexistant, un message d'erreur clair devrait être affiché sur la page pour informer l'utilisateur que le produit n'existe pas.

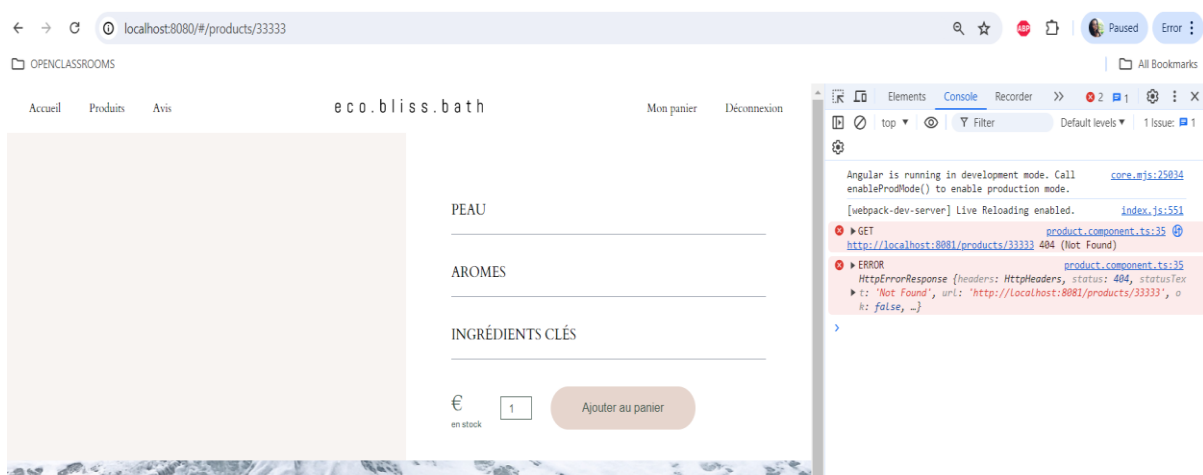
Comportement observé : Le test API vérifie que l'ID de produit inexistant renvoie un code d'état 404, mais aucun message d'erreur n'est affiché sur la page pour informer l'utilisateur que le produit n'existe pas.

Recommandations :

- Ajouter un Message d'Erreur : Implémentez un message d'erreur approprié sur la page produit pour informer les utilisateurs lorsque l'ID du produit est invalide ou inexistant.
- Améliorer la Gestion des Erreurs : Assurez-vous que les erreurs 404 retournées par l'API sont correctement traitées par le frontend pour afficher un message d'erreur clair et compréhensible.
- Tester la Visibilité du Message : Ajoutez des tests fonctionnels pour vérifier que le message d'erreur est visible sur la page lorsque l'utilisateur accède à un produit avec un ID inexistant.



Capture d'écran – Résultat du test Cypress – api.products.cy.js



Capture d'écran – URL du site – ID inexistant - http://localhost:8080/#/products/33333

- **Rapport d'Incident :** Ajouter un avis avec un script tag dans le champ commentaire

Description de l'Incident

Lors de l'ajout d'un avis contenant un script tag dans le champ de commentaire, l'application ne bloque pas l'entrée et permet l'envoi de l'avis avec succès, sans afficher de message d'erreur.

Impact

Cette anomalie pourrait entraîner des vulnérabilités de type XSS (Cross-Site Scripting), ce qui pourrait permettre à des attaquants d'exécuter des scripts malveillants sur les navigateurs des utilisateurs finaux. Cela peut compromettre la sécurité et l'intégrité des données des utilisateurs.

Détail du Problème

Comportement attendu : Lorsqu'un utilisateur tente de soumettre un avis contenant un script tag dans le champ de commentaire, l'application doit rejeter la requête avec un statut 403 (Forbidden) et afficher un message d'erreur approprié.

Comportement observé : L'avis contenant un script tag dans le champ de commentaire est envoyé avec succès avec un statut 200 et aucun message d'erreur n'est affiché.

Recommandations

- Validation côté serveur : Implémenter des validations côté serveur pour détecter et rejeter les entrées contenant des balises de script ou tout autre contenu potentiellement dangereux.
- Validation côté client : Ajouter des validations côté client pour empêcher les utilisateurs de soumettre des entrées contenant des balises de script.
- Échappement des entrées utilisateur : S'assurer que toutes les entrées utilisateur sont correctement échappées avant d'être affichées sur la page.
- Tests de sécurité : Effectuer régulièrement des tests de sécurité pour identifier et corriger les vulnérabilités potentielles, y compris les attaques XSS.

Curl

```
curl -X "POST" \
'http://localhost:8081/reviews' \
-H 'accept: application/json' \
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IHYXZio9ImFkbWwucmVudCkiOnR5cyBjbGFjaXI6ImNvbmEgYXZhdGEyLm9kaSkiOjoiMDEzMjcycDdAdGVzdSc3Sm3r.aG-i7rZOspf6QzgJm' \
-H 'Content-Type: application/json' \
-d '{
    "title": "super",
    "comment": "<script>alert('\\"XSS Attack'\")</script>",
    "rating": 5
}'
```

<http://localhost:8081/reviews>

Server response

Code	Details
------	---------

200

Response body

```
{
  "id": 27,
  "date": "2024-07-04T07:19:53:00:00",
  "title": "Super",
  "comment": "<script>alert('XSS Attack')</script>",
  "rating": 5,
  "author": {
    "id": 10,
    "email": "test2@test.fr",
    "userId": "test2@test.fr",
    "username": "test2@test.fr",
    "roles": [
      "ROLE_USER"
    ]
  },
  "password": "$2y$13$6jvii1adi.xsB9wK6TEkaQ3u/ec/g1qWao91AoGwHnN3BmZnETea",
  "salt": null,
  "firstName": "test",
  "lastName": "test"
}
```

Avez-vous confiance en la version qui va sortir ? Pourquoi ?

On a émis des réserves concernant la version qui va sortir. Bien que cette version ait réussi plusieurs tests critiques, tels que l'ajout de produits au panier et la gestion des tokens d'authentification, il y a eu plusieurs échecs dans les tests. Ces échecs ont révélé des points critiques qui nécessitent une attention particulière (par exemple, le stock des produits apparaît négatif, et nous ne sommes pas empêchés d'ajouter des quantités supérieures au stock dans le panier). Pour garantir une sécurité et une fonctionnalité optimales, il est impératif de corriger ces anomalies avant la mise en production.

1. Anomalie dans l'API pour l'Ajout d'un Produit au Panier (PUT au lieu de POST)

Description : Lors des tests d'API pour l'ajout d'un produit au panier, il a été constaté que la méthode PUT est utilisée au lieu de POST, contrairement aux pratiques RESTful courantes et aux attentes documentées.

Impact : Cette anomalie pourrait entraîner des erreurs dans la mise en œuvre des tests et une mauvaise compréhension des méthodes HTTP par les développeurs, ce qui peut affecter l'intégrité des opérations de l'API et entraîner des erreurs dans les fonctionnalités liées au panier.

Recommandation : Pour garantir le bon fonctionnement de l'API et maintenir une conformité aux pratiques RESTful, il est impératif de corriger cette anomalie.

2. Erreur 401 au lieu de 403 pour l'Accès aux Données Confidentielles

Criticité : Moyenne

Description : Retourne une erreur 401 (Unauthorized) au lieu de 403 (Forbidden) pour des utilisateurs avec des permissions insuffisantes.

Impact : Problèmes de sécurité et incohérence de l'API.

Recommandation : Cette anomalie doit être corrigée immédiatement pour assurer une gestion correcte des permissions et éviter des failles de sécurité potentielles.

3. Ajout de Produits avec une Quantité Dépasant le Stock

Criticité : Haute

Description : L'application permet actuellement l'ajout de produits en quantité supérieure au stock disponible, même si le test est correctement implémenté pour vérifier cette condition.

Impact : Problèmes de gestion des stocks, erreurs de commande, et mauvaise expérience utilisateur.

Recommandation : Il est crucial de corriger la logique backend pour s'assurer que la quantité ajoutée ne dépasse pas le stock disponible. Cette correction est nécessaire pour garantir une gestion précise des stocks et éviter des problèmes opérationnels.

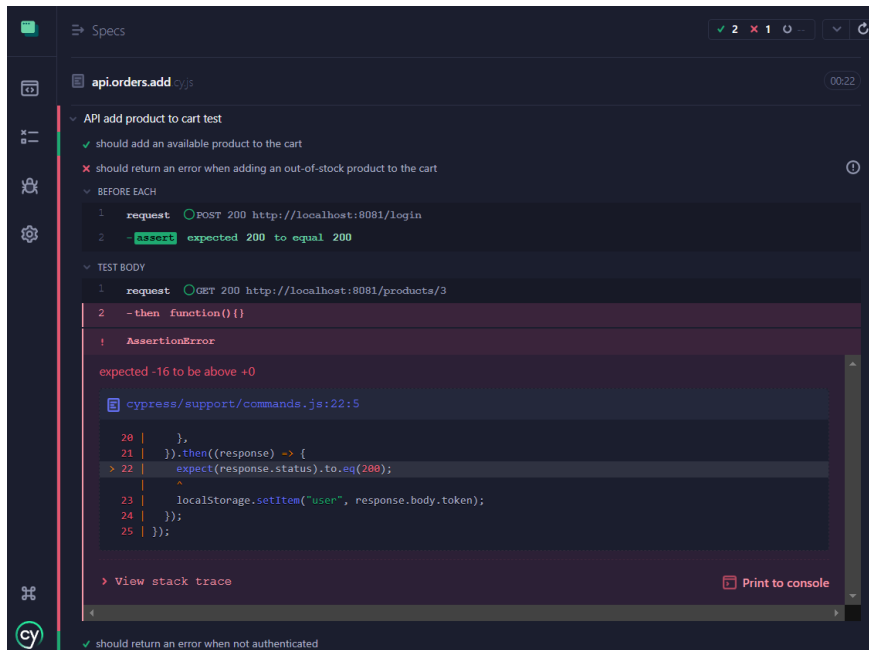
4. Stock Négatif

Criticité : Haute

Description : Le stock d'un produit peut apparaître en négatif, ce qui indique une erreur grave dans la gestion des stocks.

Impact : Problèmes de gestion des stocks et risques opérationnels importants.

Recommandation : Cette anomalie doit être corrigée immédiatement pour éviter des problèmes de gestion des stocks qui pourraient entraîner des erreurs de commande et de facturation.



5. Aucun Message d'Erreur pour une Quantité Négative Saisie par l'Utilisateur dans le Champ de Quantité

Criticité : Haute

Description : Lorsqu'un utilisateur entre une quantité négative dans le champ de quantité lors de l'ajout d'un produit au panier, aucun message d'erreur ne s'affiche pour signaler que la quantité est invalide.

Impact : Cette anomalie entraîne des problèmes de validation des données, des erreurs potentielles dans le traitement des commandes, et une mauvaise expérience utilisateur.

Recommandation : Il est essentiel de corriger cette anomalie et de mettre en place une validation stricte pour les quantités dans le champ de quantité afin de garantir que seules les valeurs valides sont acceptées, et de fournir un message d'erreur clair pour informer les utilisateurs des erreurs de saisie.

6. Le test pour un ID de produit inexistant retourne le code 404 dans la console, mais aucun message d'erreur n'apparaît sur la page.

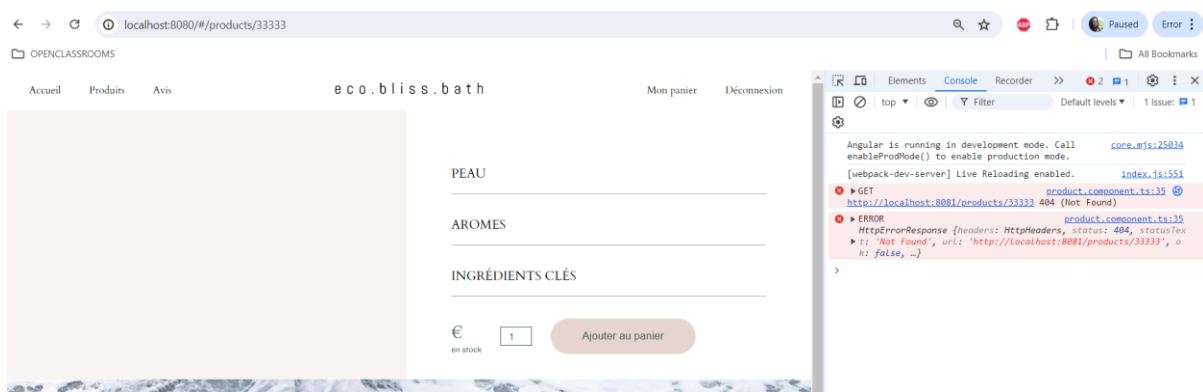
Criticité : Moyenne

Description : Le test API pour un ID de produit inexistant renvoie le code d'état 404 dans la console, ce qui est correct, mais aucun message d'erreur

approprié n'est affiché sur la page pour informer l'utilisateur que le produit n'existe pas.

Impact : Mauvaise expérience utilisateur, confusion

Recommandation : Cette anomalie doit être corrigée pour améliorer l'expérience utilisateur et éviter toute confusion. Il est essentiel que le frontend gère correctement les erreurs retournées par l'API, en affichant des messages d'erreur clairs et informatifs sur la page lorsqu'un produit n'existe pas. Cela garantit que les utilisateurs comprennent pourquoi une opération a échoué et renforce la transparence et la fiabilité du site.



7. Ajout d'avis avec un script tag dans le champ commentaire

Criticité : Haute

Description : L'application permet l'ajout d'avis contenant des balises de script dans le champ de commentaire, ce qui peut potentiellement conduire à des vulnérabilités de type XSS (Cross-Site Scripting).

Impact : Cette vulnérabilité expose les utilisateurs à des attaques XSS, permettant à des attaquants d'exécuter des scripts malveillants sur les navigateurs des utilisateurs finaux. Cela peut compromettre la sécurité et l'intégrité des données des utilisateurs, entraîner une perte de confiance des clients et des problèmes juridiques pour l'entreprise.

Recommandation : Doit être corrigée de toute urgence pour éviter des vulnérabilités de type XSS (Cross-Site Scripting) qui peuvent compromettre la sécurité des utilisateurs et l'intégrité des données. Assurer la validation et l'échappement des entrées utilisateur est essentiel pour prévenir ces attaques et maintenir la confiance des utilisateurs dans le site.

Effectuer des tests de sécurité réguliers pour identifier et corriger les vulnérabilités potentielles, y compris les attaques XSS.

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8081/reviews' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOiJlZjMwMzcxMTIsImV4cCI6MTcyMDA4MDkxMiwicm9sZXkiOiJlUk9MRV9VU0V5IiwiaWF0IjoiZGVzdDQAdGVzdC5mc139.ag-1rZ90spF602gJw' \
  -H 'Content-Type: application/json' \
  -d '{
    "title": "super",
    "comment": "<script>alert('\''XSS Attack'\''</script>",
    "rating": 5
  }'
```

Request URL

http://localhost:8081/reviews

Server response

Code Details

200

Response body

```
{
  "id": 27,
  "date": "2024-07-04T07:19:53+00:00",
  "title": "super",
  "comment": "<script>alert('\''XSS Attack'\''</script>",
  "rating": 5,
  "author": {
    "id": 10,
    "email": "test2@test.fr",
    "userIdentifier": "test2@test.fr",
    "username": "test2@test.fr",
    "roles": [
      "ROLE_USER"
    ],
    "password": "$2y$13$6jvi1adLxsB9wk6TekaQ3u/ec/gIqVWao91AoGx9wNjBmTznETea",
    "salt": null,
    "firstname": "test",
    "lastname": "test"
  }
}
```

Conclusion

[NO-GO]

Pour garantir une version fiable et sécurisée, il est impératif de corriger les anomalies mentionnées ci-dessus avant la sortie. Cela renforcera la sécurité de l'application, évitera les problèmes opérationnels et assurera une expérience utilisateur plus cohérente et satisfaisante. De plus, une gestion rigoureuse des stocks dans le panier est cruciale. Une gestion cohérente des stocks en fonction de la disponibilité des produits est essentielle pour éviter les situations de rupture de stock, améliorer la fiabilité du système et garantir que les utilisateurs puissent acheter des produits sans rencontrer de problèmes liés aux quantités disponibles. Ce point doit être traité en priorité pour maintenir la confiance des clients et assurer le bon fonctionnement de la boutique en ligne.