Cahier de Tests - Lab 1 : Tests sur une base de données MongoDB

Informations générales

Projet : Tests automatisés sur MongoDB Atlas avec Robot Framework
Base de données : fakeStoreDB
API de référence : https://fakestoreapi.com/docs
Framework de test : Robot Framework **Bibliothèque**
personnalisée : MongoDBKeywords.py

Entités testées

- **Products** : Gestion des produits •

Users : Gestion des utilisateurs •

Carts : Gestion des paniers

---

1. TESTS CRUD - PRODUITS (Products)

1.1 Opération CREATE

TC_PROD_CREATE_001 - Création d'un produit avec données valides

Objectif : Vérifier qu'un produit peut être créé avec des données valides
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario passant

Données de test :

- Titre : "Test Product Robot Framework" •

Prix : 2900.99

- Description : "Description du produit de test" •

Catégorie : "electronics"

- Quantité : 100
- Image : "https://example.com/image.jpg"

Étapes d'exécution :

1. Se connecter à MongoDB Atlas
2. Appeler la fonction `Insert Product` avec les données valides
3. Vérifier que l'insertion retourne un ID valide

Résultats attendus :

- Un ID de produit est retourné
- Le produit est créé dans la collection "products" •

Message de succès affiché dans les logs

TC_PROD_CREATE_002 - Création d'un produit avec prix invalide

Objectif : Vérifier qu'un produit ne peut pas être créé avec un prix non numérique
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- Titre : None
- Prix : "not_a_number" (chaîne de caractères) •

Description : None

- Catégorie : 123 (nombre au lieu de chaîne) •

Quantité : -1 (valeur négative)

- Image : "invalid_url"

Étapes d'exécution :

1. Se connecter à MongoDB Atlas
2. Appeler la fonction `Insert Product Should Fail` avec des données invalides
3. Vérifier que l'insertion échoue avec le message d'erreur approprié

Résultats attendus :

- Échec de l'insertion avec message d'erreur "Le prix doit être un nombre valide" •

Aucun produit créé dans la base de données

TC_PROD_CREATE_003 - Création d'un produit avec quantité négative

Objectif : Vérifier qu'un produit ne peut pas être créé avec une quantité négative
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- Titre : None
- Prix : 2900.99 (valide) •

Description : None

- Catégorie : "electronics"

- Quantité : -1 (valeur négative) Image : "invalid_url"

Étapes d'exécution :

1. Se connecter à MongoDB Atlas
2. Appeler la fonction `Insert Product Should Fail` avec quantité négative
3. Vérifier que l'insertion échoue

Résultats attendus :

- Échec de l'insertion avec message d'erreur "La quantité doit être un nombre entier"

Aucun produit créé dans la base de données

1.2 Opération READ

TC_PROD_READ_001 - Récupération d'un produit par ID valide

Objectif : Vérifier qu'un produit peut être récupéré avec un ID valide
Prérequis : Un produit existant dans la base de données
Type : Scénario passant

Données de test :

- ID produit : ID généré lors de la création

Étapes d'exécution :

1. Utiliser l'ID d'un produit existant
2. Appeler la fonction `Get Product By ID`
3. Vérifier les données retournées

Résultats attendus :

- Le produit est retourné avec toutes ses propriétés

Les données correspondent à celles insérées

- Aucune erreur générée

TC_PROD_READ_002 - Récupération avec ID invalide

Objectif : Vérifier qu'une erreur est retournée avec un ID invalide
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID produit : "invalid_id_123"

Étapes d'exécution :

1. Appeler la fonction `Get Product By ID` avec un ID invalide
2. Vérifier qu'une exception est levée

Résultats attendus :

- Exception levée
- Message d'erreur approprié •

Aucune donnée retournée

TC_PROD_READ_003 - Récupération avec ID inexistant

Objectif : Vérifier qu'une erreur est retournée avec un ID inexistant
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID produit : "507f1f77bcf86cd799439011" (format ObjectId valide mais inexistant)

Étapes d'exécution :

1. Appeler la fonction `Get Product By ID` avec un ID inexistant
2. Vérifier qu'une exception est levée

Résultats attendus :

- Exception "Produit non trouvé avec ID" levée •

Aucune donnée retournée

1.3 Opération UPDATE

TC_PROD_UPDATE_001 - Mise à jour d'un produit avec données valides

Objectif : Vérifier qu'un produit peut être mis à jour
Prérequis : Un produit existant dans la base de données **Type** :
Scénario passant

Données de test :

- ID produit : ID d'un produit existant
- Nouveau titre : "${VALID_PRODUCT_TITLE} Updated"

Étapes d'exécution :

1. Appeler la fonction `Update Product` avec les nouvelles données
2. Vérifier le nombre de documents modifiés
3. Récupérer le produit pour vérifier la mise à jour

Résultats attendus :

- 1 document modifié
- Le titre du produit est mis à jour
- La date de mise à jour est actualisée

TC_PROD_UPDATE_002 - Mise à jour avec ID invalide

Objectif : Vérifier qu'une erreur est générée avec un ID invalide
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID produit : "invalid_id_123"
- Nouveau titre : "Titre mis à jour"

Étapes d'exécution :

1. Appeler la fonction `Update Product` avec un ID invalide
2. Vérifier qu'une exception est levée

Résultats attendus :

- Exception "ID de produit invalide" levée •

Aucune modification effectuée

TC_PROD_UPDATE_003 - Mise à jour avec ID inexistant

Objectif : Vérifier qu'aucune mise à jour n'est effectuée avec un ID inexistant
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID produit : "507f1f77bcf86cd799439011" (format valide mais inexistant) •

Nouveau titre : "Titre mis à jour"

Étapes d'exécution :

1. Appeler la fonction `Update Product` avec un ID inexistant
2. Vérifier le résultat de la mise à jour

Résultats attendus :

- Exception "Aucun produit mis à jour" levée •

0 document modifié

1.4 Opération DELETE

TC_PROD_DELETE_001 - Suppression d'un produit par ID valide

Objectif : Vérifier qu'un produit peut être supprimé **Prérequis** : Un produit existant dans la base de données **Type** : Scénario passant

Données de test :

- ID produit : ID d'un produit existant

Étapes d'exécution :

1. Appeler la fonction `Delete Product` avec un ID valide
2. Vérifier le nombre de documents supprimés
3. Vérifier que le produit n'existe plus

Résultats attendus :

- 1 document supprimé
- Le produit n'est plus accessible via son ID •

Message de succès dans les logs

TC_PROD_DELETE_002 - Suppression avec ID invalide

Objectif : Vérifier qu'une erreur est générée avec un ID invalide
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID produit : "invalid_id"

Étapes d'exécution :

1. Appeler la fonction `Delete Product` avec un ID invalide
2. Vérifier qu'une exception est levée

Résultats attendus :

- Exception "ID de produit invalide" levée •

Aucune suppression effectuée

TC_PROD_DELETE_003 - Suppression avec ID inexistant

Objectif : Vérifier qu'aucune suppression n'est effectuée avec un ID inexistant
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID produit : "507f1f77bcf86cd799439011" (format valide mais inexistant)

Étapes d'exécution :

1. Appeler la fonction `Delete Product` avec un ID inexistant
2. Vérifier le résultat de la suppression

Résultats attendus :

- Exception "Aucun produit supprimé" levée • 0

document supprimé

---

2. TESTS CRUD - UTILISATEURS (Users)

2.1 Opération CREATE

TC_USER_CREATE_001 - Création d'un utilisateur avec données valides

Objectif : Vérifier qu'un utilisateur peut être créé avec des données valides
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario passant

Données de test :

- Prénom : "ndiankou" •

Nom : "Ndoye"

- Email : "ndiankou.ndoye.test@ example.com" •

Nom d'utilisateur : "ndiankou"

- Mot de passe : "password123" •

Téléphone : "+221771234567"

Étapes d'exécution :

1. Appeler la fonction `Insert User` avec les données valides
2. Vérifier que l'insertion retourne un ID valide
3. Stocker l'ID pour les tests suivants

Résultats attendus :

- Un ID utilisateur est retourné
- L'utilisateur est créé dans la collection "users" •

Message de succès affiché

TC_USER_CREATE_002 - Création d'un utilisateur avec email invalide

Objectif : Vérifier qu'un utilisateur ne peut pas être créé avec un email invalide
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- Prénom : "Test" •

Nom : "User"

- Email : 124 (nombre au lieu de chaîne) •

Nom d'utilisateur : "testuser"

- Mot de passe : "password123" •

Téléphone : "+221771234567"

Étapes d'exécution :

1. Appeler la fonction `Insert User Should Fail` avec un email invalide
2. Vérifier que l'insertion échoue

Résultats attendus :

- Échec avec message "L'email est requis et doit être une chaîne de caractères" •

Aucun utilisateur créé

TC_USER_CREATE_003 - Création d'un utilisateur avec email vide

Objectif : Vérifier qu'un utilisateur ne peut pas être créé sans email
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- Prénom : "Test" •

Nom : "User"

- Email : "" (vide)
- Nom d'utilisateur : "testuser"
- Mot de passe : "password123" •

Téléphone : "+221771234567"

Étapes d'exécution :

1. Appeler la fonction `Insert User Should Fail` avec un email vide
2. Vérifier que l'insertion échoue

Résultats attendus :

- Échec avec message "email est requis" •

Aucun utilisateur créé

2.2 Opération READ

TC_USER_READ_001 - Récupération d'un utilisateur par ID valide

Objectif : Vérifier qu'un utilisateur peut être récupéré par son ID
Prérequis : Un utilisateur existant dans la base de données **Type** :
Scénario passant

Données de test :

- ID utilisateur : ID généré lors de la création

Étapes d'exécution :

1. Appeler la fonction `Get User By ID` avec un ID valide
2. Vérifier les données retournées

Résultats attendus :

- L'utilisateur est retourné avec toutes ses propriétés
- Le nom d'utilisateur correspond aux données insérées •

Aucune erreur générée

TC_USER_READ_002 - Récupération avec ID invalide

Objectif : Vérifier qu'une erreur est retournée avec un ID invalide
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID utilisateur : "invalidid123"

Étapes d'exécution :

1. Appeler la fonction `Get User By Invalid ID` avec un ID invalide
2. Vérifier le résultat

Résultats attendus :

- Résultat "Échec attendu" retourné
- Aucune donnée utilisateur retournée

TC_USER_READ_003 - Récupération avec ID inexistant

Objectif : Vérifier qu'aucun utilisateur n'est retourné avec un ID inexistant
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID utilisateur : "507f1f77bcf86cd799439011" (format ObjectId valide mais inexistant)

Étapes d'exécution :

1. Appeler la fonction `Get User By Invalid ID` avec un ID inexistant
2. Vérifier le résultat

Résultats attendus :

- Message "Aucun utilisateur trouvé comme attendu" •

Aucune donnée utilisateur retournée

2.3 Opération UPDATE

TC_USER_UPDATE_001 - Mise à jour d'un utilisateur avec données valides

Objectif : Vérifier qu'un utilisateur peut être mis à jour **Prérequis**
: Un utilisateur existant dans la base de données **Type** : Scénario
passant

Données de test :

- ID utilisateur : ID d'un utilisateur existant
- Nouveau nom d'utilisateur : "${VALID_USER_USERNAME}_updated"

Étapes d'exécution :

1. Appeler la fonction `Update User` avec les nouvelles données
2. Vérifier le nombre de documents modifiés
3. Récupérer l'utilisateur pour vérifier la mise à jour

Résultats attendus :

- 1 document modifié
- Le nom d'utilisateur est mis à jour
- La date de mise à jour est actualisée

TC_USER_UPDATE_002 - Mise à jour avec ID invalide

Objectif : Vérifier qu'une erreur est générée avec un ID invalide
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID utilisateur : "123"
- Nouveau nom d'utilisateur : "nouveau_nom_utilisateur"

Étapes d'exécution :

1. Appeler la fonction `Update User` avec un ID invalide
2. Vérifier qu'une exception est levée

Résultats attendus :

- Exception "ID d'utilisateur invalide : 123" levée •

Aucune modification effectuée

TC_USER_UPDATE_003 - Mise à jour avec ID inexistant

Objectif : Vérifier qu'une erreur est générée avec un ID inexistant
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID utilisateur : "nonexistent_id"

- Nouveau nom d'utilisateur : "nouveau_nom_utilisateur"

Étapes d'exécution :

1. Appeler la fonction `Update User` avec un ID inexistant
2. Vérifier qu'une exception est levée

Résultats attendus :

- Exception "ID d'utilisateur invalide : nonexistent_id" levée •

Aucune modification effectuée

2.4 Opération DELETE

TC_USER_DELETE_001 - Suppression d'un utilisateur par ID valide

Objectif : Vérifier qu'un utilisateur peut être supprimé **Prérequis** : Un utilisateur existant dans la base de données **Type** : Scénario passant

Données de test :

- Utilisateur temporaire créé pour le test

Étapes d'exécution :

1. Créer un utilisateur temporaire
2. Appeler la fonction `Delete User` avec l'ID de l'utilisateur
3. Vérifier le nombre de documents supprimés
4. Vérifier que l'utilisateur n'existe plus

Résultats attendus :

- 1 document supprimé
- L'utilisateur n'est plus accessible via son ID
- Exception "non trouvé" lors de la tentative de récupération

---

TC_USER_DELETE_002 - Suppression avec ID invalide

Objectif : Vérifier qu'une erreur est générée avec un ID invalide
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID utilisateur : "invalid_id"

Étapes d'exécution :

1. Appeler la fonction `Delete User` avec un ID invalide
2. Vérifier qu'une exception est levée

Résultats attendus :

- Exception "ID d'utilisateur invalide" levée •

Aucune suppression effectuée

TC_USER_DELETE_003 - Suppression avec ID inexistant

Objectif : Vérifier qu'aucune suppression n'est effectuée avec un ID inexistant
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID utilisateur : "507f1f77bcf86cd799439011" (format valide mais inexistant)

Étapes d'exécution :

1. Appeler la fonction `Delete User` avec un ID inexistant
2. Vérifier le résultat de la suppression

Résultats attendus :

- Exception "Aucun utilisateur supprimé" levée • 0

document supprimé

---

3. TESTS CRUD - PANIERS (Carts)

3.1 Opération CREATE

TC_CART_CREATE_001 - Création d'un panier avec données valides

Objectif : Vérifier qu'un panier peut être créé avec des données valides **Prérequis** : Un utilisateur et un produit existants dans la base de données **Type** : Scénario passant

Données de test :

- ID utilisateur : ID d'un utilisateur existant
- Liste produits : [{"product_id": "ID_PRODUIT", "quantity": 5}]

Étapes d'exécution :

1. Créer un utilisateur et un produit de test
2. Préparer la liste des produits du panier
3. Appeler la fonction `Create Cart` avec les données valides
4. Vérifier que la création retourne un ID valide

Résultats attendus :

- Un ID de panier est retourné
- Le panier est créé dans la collection "carts" •

Message de succès affiché

TC_CART_CREATE_002 - Création d'un panier avec quantité négative

Objectif : Vérifier qu'un panier ne peut pas être créé avec une quantité négative
Prérequis : Un utilisateur et un produit existants
Type : Scénario non passant

Données de test :

- ID utilisateur : ID d'un utilisateur existant
- Liste produits : [{"product_id": "ID_PRODUIT", "quantity": -1}]

Étapes d'exécution :

1. Préparer une liste de produits avec quantité négative
2. Appeler la fonction `Create Cart Should Fail`
3. Vérifier que la création échoue

Résultats attendus :

- Échec avec message "La quantité doit être supérieure à 0" •

Aucun panier créé

TC_CART_CREATE_003 - Création d'un panier avec ID produit invalide

Objectif : Vérifier qu'un panier ne peut pas être créé avec un ID de produit invalide
Prérequis : Un utilisateur existant
Type : Scénario non passant

Données de test :

- ID utilisateur : ID d'un utilisateur existant
- Liste produits : [{"product_id": "invalid_product_id", "quantity": 5}]

Étapes d'exécution :

1. Préparer une liste de produits avec ID invalide
2. Appeler la fonction `Create Cart Should Fail`
3. Vérifier que la création échoue

Résultats attendus :

- Échec avec message "ID de produit invalide" •

Aucun panier créé

3.2 Opération READ

TC_CART_READ_001 - Récupération d'un panier par ID valide

Objectif : Vérifier qu'un panier peut être récupéré par son ID
Prérequis : Un panier existant dans la base de données **Type** :
Scénario passant

Données de test :

- ID panier : ID généré lors de la création

Étapes d'exécution :

1. Utiliser l'ID d'un panier existant
2. Appeler la fonction `Get Cart By ID`
3. Vérifier les données retournées

Résultats attendus :

- Le panier est retourné avec toutes ses propriétés •

Les données correspondent à celles insérées

- L'ID utilisateur est converti en string

TC_CART_READ_002 - Récupération avec ID invalide

Objectif : Vérifier qu'une erreur appropriée est retournée avec un ID invalide
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID panier : "invalid_cart_id"

Étapes d'exécution :

1. Appeler la fonction `Get Cart By Invalid ID` avec un ID invalide
2. Vérifier le résultat

Résultats attendus :

- Résultat "NOT_FOUND" retourné
- Aucune donnée de panier retournée

TC_CART_READ_003 - Récupération avec ID inexistant

Objectif : Vérifier qu'une erreur est générée avec un ID inexistant
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID panier : "507f1f77bcf86cd799439011" (format ObjectId valide mais inexistant)

Étapes d'exécution :

1. Appeler la fonction `Get Cart By ID` avec un ID inexistant
2. Vérifier qu'une exception est levée

Résultats attendus :

- Exception "Panier non trouvé" levée •

Aucune donnée retournée

3.3 Opération UPDATE

TC_CART_UPDATE_001 - Mise à jour d'un panier avec données valides

Objectif : Vérifier qu'un panier peut être mis à jour **Prérequis**
: Un panier existant dans la base de données **Type** : Scénario
passant

Données de test :

- ID panier : ID d'un panier existant
- Nouvelle liste produits : [{"product_id": "ID_PRODUIT", "quantity": 10}]

Étapes d'exécution :

1. Appeler la fonction `Update Cart` avec les nouvelles données
2. Vérifier le nombre de documents modifiés
3. Récupérer le panier pour vérifier la mise à jour

Résultats attendus :

- 1 document modifié
- La quantité du produit est mise à jour à 10 •

La date de mise à jour est actualisée

TC_CART_UPDATE_002 - Mise à jour avec quantité négative

Objectif : Vérifier qu'une erreur est générée avec une quantité négative
Prérequis : Un panier existant dans la base de données
Type : Scénario non passant

Données de test :

- ID panier : ID d'un panier existant
- Liste produits : [{"product_id": "ID_PRODUIT", "quantity": -1}]

Étapes d'exécution :

1. Appeler la fonction `Update Cart Should Fail` avec quantité négative
2. Vérifier que la mise à jour échoue

Résultats attendus :

- Échec avec message "La quantité doit être supérieure à 0" •

Aucune modification effectuée

TC_CART_UPDATE_003 - Mise à jour avec ID invalide

Objectif : Vérifier qu'une erreur est générée avec un ID invalide
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID panier : "invalid_cart_id"
- Liste produits : données valides

Étapes d'exécution :

1. Appeler la fonction `Update Cart` avec un ID invalide
2. Vérifier qu'une exception est levée

Résultats attendus :

- Exception "ID de panier invalide" levée •

Aucune modification effectuée

3.4 Opération DELETE

TC_CART_DELETE_001 - Suppression d'un panier par ID valide

Objectif : Vérifier qu'un panier peut être supprimé **Prérequis** : Un panier existant dans la base de données **Type** : Scénario passant

Données de test :

- Panier temporaire créé pour le test

Étapes d'exécution :

1. Créer un panier temporaire
2. Appeler la fonction `Delete Cart` avec l'ID du panier
3. Vérifier le nombre de documents supprimés
4. Vérifier que le panier n'existe plus

Résultats attendus :

- 1 document supprimé
- Le panier n'est plus accessible via son ID
- Exception "Panier non trouvé" lors de la tentative de récupération

---

TC_CART_DELETE_002 - Suppression avec ID invalide

Objectif : Vérifier qu'une erreur est générée avec un ID invalide
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- ID panier : "invalid_cart_id"

Étapes d'exécution :

1. Appeler la fonction `Delete Cart` avec un ID invalide
2. Vérifier qu'une exception est levée

Résultats attendus :

- Exception "ID de panier invalide" levée

- Aucune suppression effectuée

TC_CART_DELETE_003 - Suppression d'un panier actif

Objectif : Vérifier qu'un panier actif ne peut pas être supprimé
Prérequis : Un panier avec statut "active" dans la base de données **Type** : Scénario non passant

Données de test :

- ID panier : ID d'un panier existant
- Statut panier : "active"

Étapes d'exécution :

1. Mettre le panier en statut "active" via `Update Cart`
2. Appeler la fonction `Delete Cart Should Fail`
3. Vérifier que la suppression échoue

Résultats attendus :

- Échec avec message "Impossible de supprimer un panier actif"
- Le panier reste dans la base de données

---

4. TESTS D'AUTHENTIFICATION

4.1 Authentification réussie

TC_AUTH_001 - Authentification avec identifiants valides

Objectif : Vérifier qu'un utilisateur peut s'authentifier avec des identifiants valides
Prérequis : Un utilisateur existant dans la base de données
Type : Scénario passant

Données de test :

- Nom d'utilisateur : "ndiankou"
- Mot de passe : "password123"

Étapes d'exécution :

1. Appeler la fonction `Authenticate User` avec les identifiants valides
2. Vérifier que l'authentification réussit
3. Vérifier que les données utilisateur sont retournées

Résultats attendus :

- Authentification réussie
- Données utilisateur retournées (avec ObjectId converti en string) •

Message de succès dans les logs

4.2 Authentification échouée

TC_AUTH_002 - Authentification avec identifiants invalides

Objectif : Vérifier qu'une erreur est générée avec des identifiants invalides
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- Nom d'utilisateur : "invalid_username!@#" •

Mot de passe : "short"

Étapes d'exécution :

1. Appeler la fonction `Authenticate User` avec des identifiants invalides
2. Vérifier que l'authentification échoue

Résultats attendus :

- Échec avec message "Échec de l'authentification" •

Aucune donnée utilisateur retournée

TC_AUTH_003 - Authentification avec utilisateur inexistant

Objectif : Vérifier qu'une erreur est générée avec un utilisateur inexistant
Prérequis : Connexion à MongoDB Atlas établie
Type : Scénario non passant

Données de test :

- Nom d'utilisateur : "nonexistent_user" •

Mot de passe : "password123"

Étapes d'exécution :

1. Appeler la fonction `Authenticate User` avec un utilisateur inexistant
2. Vérifier que l'authentification échoue

Résultats attendus :

- Échec avec message "Échec de l'authentification"
- Message dans les logs indiquant l'échec pour utilisateur inexistant

---

5. CONFIGURATION DES TESTS

5.1 Prérequis techniques

Environnement :

- Robot Framework installé
- Bibliothèque pymongo installée •

Accès à MongoDB Atlas

- Python 3.x

Variables de configuration :

```
MONGO_URI = "mongodb+srv://username:password@cluster.mongodb.net/database"
DB_NAME = "fakeStoredb"
Collections : products, users, carts
```

5.2 Structure des fichiers

Fichiers de test :

- `product_test.robot` : Tests CRUD pour les produits
- `user_test.robot` : Tests CRUD pour les utilisateurs
- `cart_test.robot` : Tests CRUD pour les paniers

Fichiers de ressources :

- `MongoDBKeywords.py` : Bibliothèque personnalisée avec mots-clés MongoDB
- `variables.py` : Variables de configuration Python
- `variable.robot` : Variables Robot Framework

5.3 Setup et Teardown

Suite Setup :

- Connexion à MongoDB Atlas •

Vérification de la connectivité

- Initialisation des variables de suite

Suite Teardown :

- Nettoyage des données de test
- Fermeture de la connexion MongoDB

Test Cleanup :

- Suppression des données créées pendant les tests ●

Remise à zéro des variables temporaires

---

6. DONNÉES DE TEST

6.1 Données valides

Produit valide :

Titre : "Test Product Robot Framework"
Prix : 2900.99
Description : "Description du produit de test"
Catégorie : "electronics"
Quantité : 100
Image : "https://example.com/image.jpg"

Utilisateur valide :

Prénom : "ndiankou"
Nom : "Ndoye"
Email : "ndiankou.ndoye.test@example.com"
Username : "ndiankou"
Password : "password123"
Téléphone : "+221771234567"

Panier valide :

User ID : ID d'un utilisateur existant
Products : [{"product_id": "ID_PRODUIT", "quantity": 5}]
Status : "active"

6.2 Données invalides

Produit invalide :

Titre : None
Prix : "not_a_number"
Description : None
Catégorie : 123
Quantité : -1
Image : "invalid_url"

Utilisateur invalide :

Email : 124 (nombre)
Email vide : ""
Password court : "short"
Username avec caractères spéciaux : "invalid_username!@#"

Panier invalide :

Quantité négative : -1
Product ID invalide : "invalid_product_id"
User ID manquant : None

---

7. RÉSULTATS ATTENDUS

7.1 Métriques de succès

Tests passants : 12 cas de test

- 3 CREATE (1 par entité)
- 3 READ (1 par entité)
- 3 UPDATE (1 par entité)
- 3 DELETE (1 par entité)

Tests non passants : 24 cas de test •

6 CREATE (2 par entité)

- 6 READ (2 par entité)
- 6 UPDATE (2 par entité)
- 6 DELETE (2 par entité)

Tests d'authentification : 3 cas de test

- 1 passant
- 2 non passants

8. EXÉCUTION DES TESTS

8.1 Commandes d'exécution

Exécution complète :

```
bash

robot tests/
```

Exécution par entité :

```
robot tests/product_test.robot
robot tests/user_test.robot
robot tests/cart_test.robot
```

8.2 Rapports générés

Fichiers de sortie :

- `report.html` : Rapport détaillé des tests
- `log.html` : Log détaillé de l'exécution
- `output.xml` : Résultats au format XML

CONCLUSION

Ce cahier de tests couvre l'ensemble des opérations CRUD sur les trois entités principales de la base de données fakeStoreDB (Products, Users, Carts) avec Robot Framework. Il respecte les exigences du Lab 1 en proposant pour chaque opération CRUD :

- **1 scénario passant** : Validation du comportement attendu
- **2 scénarios non passants** : Validation de la gestion d'erreurs