Test Technique

Context :

Dans le cadre de ce test technique, un environnement de test vous sera partagé. L'objectif est d'évaluer vos compétences en analyse, test manuel et automatisation.

Voici le lien de serveur de test : https://recrutement.arvea-test.ovh/

Nom d'utilisateur: qa-test
Mot de passe: mdw@@2025

Compte:
Login : TN25000000
Mot de passe: maisonduweb123

🏢 **Définition générale du projet :**

ARVEA Nature est une marque tunisienne fondée en 2013, spécialisée dans les produits naturels de beauté, bien-être et maquillage. Elle fonctionne avec un modèle de **vente directe (MLM)**, offrant à chacun la possibilité de devenir distributeur indépendant, de vendre les produits, et de développer un réseau tout en gagnant des commissions et des bonus.

Phase 1 : Test manuel

1. **Tâches :**
    - Effectuer un test exploratoire ayant pour but de **valider le fonctionnement du système**, de la sélection du produit jusqu'à sa confirmation.
    - Rédiger au moins trois scénarios de test
    - Tester la fonctionnalité de passage de commande
    - Remonter les anomalies détectées
    - Assigner une criticité à chaque anomalie identifiée

Phase 2 : Automatisation d’un scénario

Étapes 1: Scénario à automatiser :

1. **Étapes :**
    - Se connecter à l’application.
    - Accéder à la rubrique **Passage de commande**.
    - Ajouter un produit dans le panier.
    - Ajuster la quantité à **5**.
    - Sélectionner le mode de livraison **“Siège”**.
    - Choisir le siège **“Agence Tunis”**.
    - Sélectionner le mode de paiement **“À la livraison”**.
    - Cliquer sur **“Commander”**.
2. **Attentes :**
    - Vérifier que la commande est créée avec succès.

Exigences :

- Utiliser **Selenium** avec **Python** et l’architecture **Page Object Model (POM)**.
- Utiliser **Selenium WebDriver** pour interagir avec les éléments du formulaire.
- Implémenter des assertions pour vérifier les résultats attendus (par exemple, messages de succès ou d’erreur).
- Simuler plusieurs combinaisons d’entrées à l’aide de tests paramétrés.

Étapes 2 : Dockerisation du projet

Tâches :

1. Ajouter un **Dockerfile** pour containeriser l’environnement de test.
2. Créer un fichier **docker-compose.yml** avec les configurations suivantes :
    - Mettre en place un service **Selenium Grid**.
    - Configurer un conteneur pour exécuter les tests.

Exigences :

- Fournir des instructions claires dans un fichier **README.md** pour :
    - Construire et exécuter le conteneur Docker.
    - Lancer les tests avec une commande simple via Docker.
- S’assurer que le projet est prêt pour une exécution automatisée dans l’environnement containerisé.

Étapes 3 : Hébergement Git

- Héberger le projet dans un dépôt Git privé.
- Partager le lien du dépôt pour évaluation.

Phase 3 : Test d'API

Vous pouvez utiliser l'outil de votre choix parmi :

Karate, Postman (tests en JavaScript), REST Assured (Java), Python (requests + unittest/pytest), etc.

Tâches :

1. Écrire un **scénario de test** permettant de s'assurer qu'un produit est disponible ou non dans un dépôt spécifique, en interrogeant un endpoint REST.

➕ **Détails de l'API :**

- Méthode : GET
- Endpoint : /stock/getQuantity
- Paramètres requis (en query string) :
    - product_config_id : identifiant du produit configuré
    - stock_type : valeur à fixer à 'depot' pour ce cas
    - depot_id : identifiant du dépôt à vérifier

Phase 4 : Requête SQL

📌 **Tâche :**

Écrire une requête SQL permettant de **retrouver la disponibilité d'un produit par son nom, sa variante dans un dépôt donné**, en retournant une réponse sous la forme suivante :

Product_name	variant	depot_id	quantity	is_available
Shampooing	250ML	2	10	true
Crème visage	50ML	3	0	false

📚 **Tables utilisées :**

- **Table product**

id	name
1	Shampooing
2	Crème visage

- Table product_configurations

id	product_id	variant
101	1	250ml
102	1	500ml
201	2	50ml

- Table stock

id	product_config_id	depot_id	quantity
1	101	1	20
2	101	2	0

3	102	1	5
4	201	1	0

Livrables attendus :

- **Phase1:**
  - Un document contenant les scénarios de test rédigés.
  - Une liste des anomalies détectées avec leur criticité.
- **Phase2 :**
  - Un script automatisé du scénario demandé.
  - Un environnement Docker fonctionnel avec fichiers Docker associés.
  - Lien vers le dépôt Git hébergeant le travail de l'automatisation
- **Phase 3 :**
  Lien vers le dépôt Git hébergeant le travail de test d'API
- **Phase 4 :**
  Un fichier contenant la requête sql.

Bonne chance !