



**Faculté des Sciences de Sfax**  
Département Informatique

Année Universitaire : 2025-2026

# **RAPPORT DE PROJET**

## **Test Logiciel (Certification ISTQB)**

Automatisation de tests avec Selenium et Python

**Réalisé par :**

- Med Baha Chekir  
- Hamdi Abdelkader  
*1ère Année Ingénieur*

**Enseignant :**

M. Taher Labidi

Date de soumission : 18 décembre 2025

# Table des matières

<b>1</b>	<b>Introduction et Contexte</b>	<b>2</b>
1.1	Objectif du projet . . . . .	2
1.2	Application sous test (AUT) . . . . .	2
1.3	Environnement Technique . . . . .	3
<b>2</b>	<b>Architecture du Projet</b>	<b>4</b>
2.1	Organisation des Dossiers . . . . .	4
2.2	Implémentation Technique . . . . .	5
<b>3</b>	<b>Génération de Tests par IA (ai_gen)</b>	<b>6</b>
3.1	Approche et Objectifs . . . . .	6
3.2	Scénario de Génération . . . . .	6
3.3	Résultat de l'IA (Output Brut) . . . . .	6
3.4	Implémentation et Exécution . . . . .	7
3.5	Bilan de l'expérience IA . . . . .	7
<b>4</b>	<b>Exécution Détaillée des Tests</b>	<b>9</b>
4.1	Module : Panier (test_cart.py) . . . . .	9
4.2	Module : Paiement (test_checkout.py) . . . . .	10
4.3	Module : Inventaire (test_inventory.py) . . . . .	12
4.4	Module : Produits (test_products.py) . . . . .	13
4.5	Module : Authentification (test_login.py) . . . . .	13
4.6	Tests Non-Fonctionnels . . . . .	15
4.7	Preuve d'Exécution . . . . .	16
<b>5</b>	<b>Matrice de Traçabilité (RTM)</b>	<b>17</b>
5.1	Analyse de la Couverture . . . . .	18
<b>6</b>	<b>Rapport d'Anomalies (Bug Report)</b>	<b>19</b>
6.1	BUG-001 : Navigation Panier vers Détails échoue . . . . .	19
6.2	BUG-002 : Suppression impossible depuis l'Inventaire . . . . .	19
6.3	BUG-003 : Format de prix incorrect . . . . .	20
<b>7</b>	<b>Conclusion</b>	<b>21</b>
<b>A</b>	<b>Annexes Techniques</b>	<b>22</b>
A.1	Code Source : BasePage . . . . .	22
A.2	Code Source : Configuration des Tests . . . . .	22
A.3	Liste des Dépendances . . . . .	23

# Chapitre 1

## Introduction et Contexte

### 1.1 Objectif du projet

Ce projet a pour but de concevoir et d'automatiser une campagne de tests complète pour une application web e-commerce. Il s'inscrit dans le cadre de la certification ISTQB et vise à valider les compétences en :

- Conception de cas de tests (positifs, négatifs, limites).
- Automatisation avec Selenium WebDriver et Python.
- Utilisation de techniques avancées (IA générative).
- Rapport d'exécution et gestion des anomalies.

### 1.2 Application sous test (AUT)

L'application cible est **Swag Labs** (<https://www.saucedemo.com>), un site "bac à sable" conçu pour les tests d'automatisation. Il simule un parcours client complet : Authentification → Choix produits → Panier → Paiement.

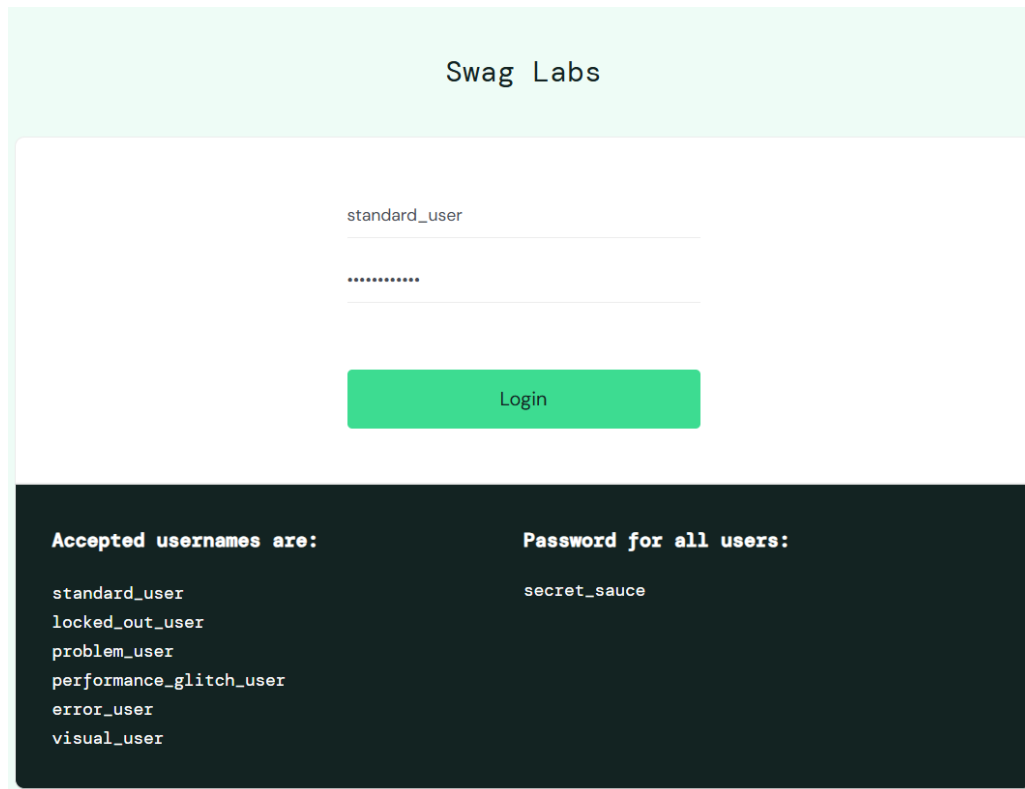


FIGURE 1.1 – Page d'accueil de l'application Swag Labs

## 1.3 Environnement Technique

- **Langage** : Python 3.10+
- **Framework de test** : Pytest
- **Driver** : Selenium WebDriver (Chrome)
- **Pattern de Conception** : Page Object Model (POM)
- **IDE** : VS Code

# Chapitre 2

## Architecture du Projet

### 2.1 Organisation des Dossiers

Le projet suit une architecture modulaire stricte pour assurer la maintenabilité.

`tests/` Contient les tests organisés par catégories :

- `functional/` : Tests métier (Panier, Login, Checkout).
- `performance/` : Mesures de temps de chargement.
- `responsive/` : Tests d’affichage mobile.

`pages/` Implémentation du *Page Object Model*. Chaque fichier correspond à une page web (ex : `cart_page.py`, `login_page.py`).

`ai_gen/` Module expérimental pour la génération de tests via IA (contient `generate_tests.py` et les prompts).

```
1  Projet/
2  |-- pages/
3  |   |-- base_page.py           # Classe mère (wrapper Selenium)
4  |   |-- login_page.py
5  |   |-- inventory_page.py
6  |   |-- cart_page.py
7  |   |-- checkout_page.py
8  |   |-- product_detail_page.py
9  |-- tests/
10 |   |-- functional/
11 |   |   |-- test_login.py
12 |   |   |-- test_cart.py
13 |   |   |-- test_checkout.py
14 |   |   |-- test_inventory.py
15 |   |   |-- test_products.py
16 |   |-- performance/
17 |   |   |-- test_load_times.py
18 |   |-- responsive/
19 |   |   |-- test_mobile_view.py
20 |-- ai_gen/
21 |   |-- generate_tests.py      # Script de génération IA
22 |   |-- generated_cases.md     # Sortie des tests générés
```

Listing 2.1 – Arborescence détaillée du projet

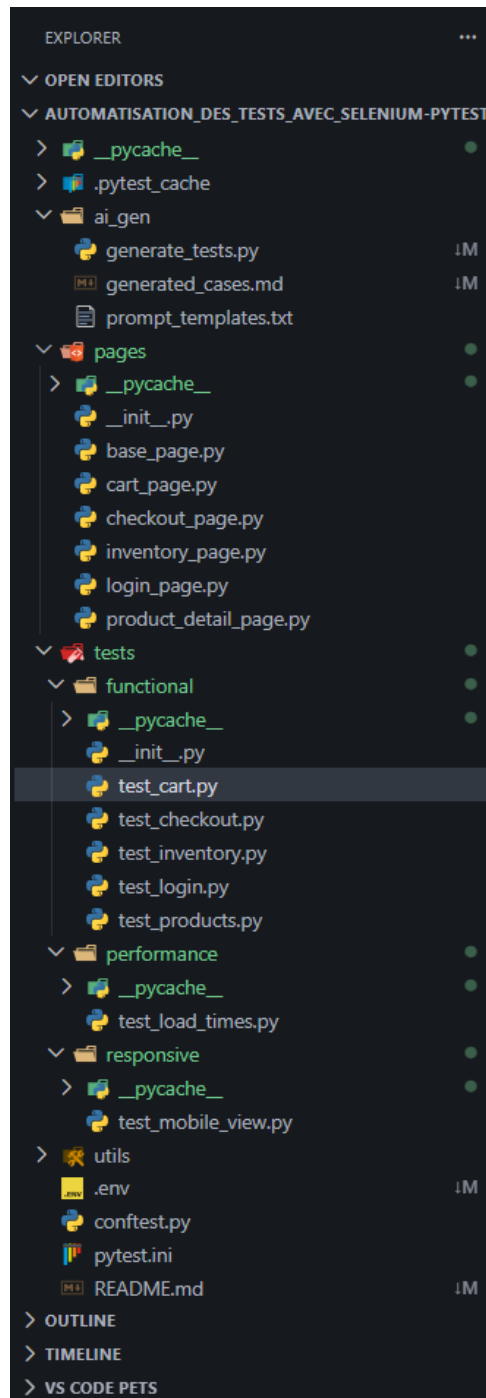


FIGURE 2.1 – Vue réelle de l'architecture du projet sous VS Code

## 2.2 Implémentation Technique

Les interactions avec le navigateur sont centralisées dans `base_page.py` pour éviter la duplication de code et gérer les attentes explicites (Explicit Waits).

# Chapitre 3

## Génération de Tests par IA (ai\_gen)

### 3.1 Approche et Objectifs

Nous avons développé un module `ai_gen` exploitant les LLM (Gemini/OpenRouter) pour générer des tests en "Boîte Noire". L'objectif est de convertir une User Story brute en une matrice de tests structurée.

### 3.2 Scénario de Génération

Nous avons soumis la User Story suivante au modèle :

*"En tant qu'utilisateur, je veux pouvoir ajouter des produits au panier depuis l'inventaire. Le système doit gérer les cas nominaux, les ajouts multiples et les restrictions d'accès."*

### 3.3 Résultat de l'IA (Output Brut)

Le modèle a généré une réponse structurée proposant 3 cas de tests pertinents. Voici la transcription de la sortie du module :

ID	Description	Étapes Clés (Resumées)	Résultat Attendu
TC_ADD_001	Ajout unique (Nominal)	1. Clic 'Add to cart'. 2. Vérif. bouton 'Remove'. 3. Vérif. compteur '1'.	Bouton = Remove. Panier = 1.
TC_ADD_002	Restriction d'accès (Sécurité)	1. Accès direct URL /inventory.html sans login.	Redirection vers Login ou Message erreur.
TC_ADD_003	Ajout total (Volume)	1. Ajouter tous les produits (N). 2. Vérifier compteur = N.	Tous boutons = Remove. Compteur = N.

TABLE 3.1 – Matrice de tests générée par l'IA (Transcription)

```

1 En tant qu'expert en QA Automation, voici les cas de test en "Boîte Noire" basés sur votre User Story.
2
3 | ID | Description | Pré-conditions | Étapes de test | Résultat Attendu |
4 |---|-----|-----|-----|-----|
5 | TC_ADD_TO_CART_001 | Vérifier l'ajout d'un seul produit au panier, la mise à jour du bouton, l'incrémement du compteur et la visibilité dans le panier. | 1. Un utilisateur est connecté au système. <br> 2. Le panier est vide. <br> 3. L'utilisateur est sur la page "/inventory.html". | 1. Vérifier que le compteur du panier (icône) affiche "0" ou est absent. <br> 2. Localiser un produit spécifique (ex: "Sauce Labs Backpack"). <br> 3. Cliquer sur le bouton "Add to cart" associé à ce produit. <br> 4. Vérifier que le bouton "Add to cart" pour ce produit change en "Remove". <br> 5. Vérifier que le compteur du panier affiche "1". <br> 6. Naviguer vers la page du panier ("/cart.html"). <br> 7. Vérifier que le produit "Sauce Labs Backpack" est visible dans le panier, avec une quantité de 1. | 1. Le bouton du produit a changé en "Remove". <br> 2. Le compteur du panier affiche "1". <br> 3. Le produit est correctement listé dans le panier. |
6 | TC_ADD_TO_CART_002 | Vérifier que l'ajout au panier est restreint pour un utilisateur non connecté. | 1. Un utilisateur n'est PAS connecté au système. <br> 2. (Optionnel) L'utilisateur essaie d'accéder directement à "/inventory.html". | 1. Tenter d'accéder à la page "/inventory.html". <br> 2. Si la page est accessible, localiser un produit et tenter de cliquer sur le bouton "Add to cart". | 1. L'utilisateur est redirigé vers la page de connexion ("/login.html") avant ou après la tentative d'ajout. <br> OU <br> 2. Un message d'erreur approprié s'affiche indiquant que l'action nécessite une connexion. <br> 3. Le compteur du panier ne doit pas s'incrémenter. <br> 4. Le produit n'est pas ajouté au panier. |
7 | TC_ADD_TO_CART_003 | Vérifier l'ajout de tous les produits disponibles sur la page d'inventaire au panier. | 1. Un utilisateur est connecté au système. <br> 2. Le panier est vide. <br> 3. L'utilisateur est sur la page "/inventory.html". | 1. Vérifier que le compteur du panier (icône) affiche "0" ou est absent. <br> 2. Identifier et compter tous les produits disponibles sur la page "/inventory.html". Soit "N" le nombre total de produits. <br> 3. Pour chaque produit listé, cliquer sur son bouton "Add to cart". <br> 4. Après chaque clic, vérifier que le bouton du produit change en "Remove". <br> 5. Après avoir cliqué sur "Add to cart" pour tous les "N" produits, vérifier que le compteur du panier affiche "N". <br> 6. Naviguer vers la page du panier ("/cart.html"). <br> 7. Vérifier que tous les "N" produits ajoutés sont présents dans le panier, chacun avec une quantité de 1. | 1. Tous les boutons "Add to cart" sont devenus "Remove". <br> 2. Le compteur du panier affiche le nombre total ("N") de produits ajoutés. <br> 3. Tous les produits ajoutés sont correctement listés dans le panier. |

```

FIGURE 3.1 – Aperçu du fichier Markdown généré automatiquement par l'IA

### 3.4 Implémentation et Exécution

Suite à cette proposition de l'IA, nous avons implémenté les scripts Selenium correspondants pour valider ces scénarios. Voici le rapport d'exécution de ces tests générés.

ID TEST	TC-01 (IA)
Titre	Restriction d'accès (Non connecté)
Pré-conditions	Utilisateur déconnecté, Navigateur ouvert
Étapes de test	1. Tenter d'accéder directement à l'URL /inventory.html. 2. Vérifier l'URL actuelle. 3. Tenter d'ajouter un produit (si accessible).
Résultat Attendu	L'utilisateur est redirigé immédiatement vers la page de Login. Aucun accès à l'inventaire.
Statut	<b>PASSED</b>

ID TEST	TC-02 (IA)
Titre	Ajout de tous les produits (Stress Test)
Pré-conditions	Utilisateur connecté sur l'inventaire
Étapes de test	1. Identifier tous les produits (6 items). 2. Boucler et cliquer sur "Add to cart" pour chacun. 3. Vérifier que le badge du panier affiche "6".
Résultat Attendu	Le compteur du panier affiche exactement 6. Tous les boutons sont passés à l'état "Remove".
Statut	<b>PASSED</b>

### 3.5 Bilan de l'expérience IA

L'utilisation de l'IA a permis de :

- Identifier un cas de test de sécurité (TC\_ADD\_002) que nous n'avions pas prévu initialement dans les tests fonctionnels.
- Générer rapidement un test de volume (ajouter tous les produits) pertinent pour vérifier la robustesse du compteur de panier.

# Chapitre 4

## Exécution Détaillée des Tests

Ce chapitre présente le journal d'exécution complet des tests fonctionnels standards.

### 4.1 Module : Panier (test\_cart.py)

ID TEST	TC_CART_01
Titre	Suppression d'un article du panier
Pré-conditions	Utilisateur connecté, Panier contient "Sauce Labs Backpack"
Étapes de test	1. Aller au panier. 2. Vérifier présence de l'article. 3. Cliquer sur "Remove". 4. Vérifier que le panier est vide.
Résultat Attendu	Le panier ne contient plus d'articles.
Statut	<b>PASSED</b>

ID TEST	TC_CART_02
Titre	Accès au paiement
Pré-conditions	Article dans le panier
Étapes de test	1. Cliquer sur "Checkout". 2. Vérifier la redirection.
Résultat Attendu	L'URL contient "checkout-step-one".
Statut	<b>PASSED</b>

ID TEST	TC_CART_03
Titre	Retour au shopping
Pré-conditions	Utilisateur dans le panier
Étapes de test	1. Cliquer sur "Continue Shopping". 2. Vérifier la redirection.
Résultat Attendu	L'URL contient "inventory".
Statut	<b>PASSED</b>

ID TEST	TC_CART_04
Titre	Accès détails produit depuis panier
Pré-conditions	Article dans le panier
Étapes de test	1. Récupérer le nom de l'article. 2. Cliquer sur le nom de l'article. 3. Comparer avec le nom sur la page détails.
Résultat Attendu	Les noms correspondent exactement.
Statut	<b>FAILED</b>

## 4.2 Module : Paiement (test\_checkout.py)

ID TEST	TC_CHK_01
Titre	Checkout Step 1 Valide
Pré-conditions	Utilisateur sur Checkout Step 1
Étapes de test	1. Remplir Prénom, Nom, Code Postal. 2. Cliquer sur "Continue".
Résultat Attendu	Redirection vers "checkout-step-two".
Statut	<b>PASSED</b>

ID TEST	TC_CHK_02
Titre	Checkout avec infos manquantes (Negative)
Pré-conditions	Utilisateur sur Checkout Step 1
Étapes de test	<ol style="list-style-type: none"> <li>1. Remplir uniquement le Prénom.</li> <li>2. Laisser les autres vides.</li> <li>3. Cliquer sur "Continue".</li> </ol>
Résultat Attendu	L'utilisateur reste sur Step 1. Message d'erreur : "Last Name is required".
Statut	<b>PASSED</b>

ID TEST	TC_CHK_03
Titre	Calcul des totaux (Overview)
Pré-conditions	Utilisateur sur Checkout Step 2 (Overview)
Étapes de test	<ol style="list-style-type: none"> <li>1. Récupérer sous-total, taxe et total affichés.</li> <li>2. Calculer mathématiquement la somme.</li> </ol>
Résultat Attendu	Sous-total + Taxe = Total Final affiché.
Statut	<b>PASSED</b>

ID TEST	TC_CHK_04
Titre	Commande réussie
Pré-conditions	Utilisateur sur Checkout Step 2
Étapes de test	<ol style="list-style-type: none"> <li>1. Cliquer sur "Finish".</li> <li>2. Vérifier le message de succès.</li> <li>3. Vérifier que le panier est vidé.</li> </ol>
Résultat Attendu	Message : "Thank you for your order!". URL : checkout-complete.
Statut	<b>PASSED</b>

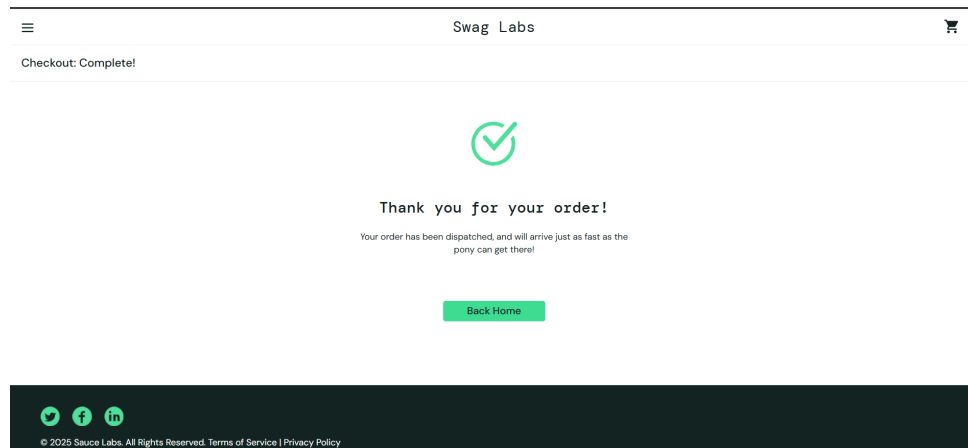


FIGURE 4.1 – Confirmation de commande réussie

### 4.3 Module : Inventaire (test\_inventory.py)

ID TEST	TC_INV_01
Titre	Ajout au panier depuis inventaire
Pré-conditions	Page inventaire chargée (6 produits)
Étapes de test	1. Ajouter un produit au panier. 2. Vérifier le badge du panier.
Résultat Attendu	Le panier contient 1 article.
Statut	<b>PASSED</b>

ID TEST	TC_INV_02
Titre	Retrait du panier depuis inventaire
Pré-conditions	Produit ajouté ("Remove" affiché)
Étapes de test	1. Cliquer sur "Remove". 2. Vérifier le badge du panier.
Résultat Attendu	Le panier contient 0 article.
Statut	<b>FAILED</b>

#### 4.4 Module : Produits (test\_products.py)

ID TEST	TC_PROD_01
Titre	Cohérence prix Liste vs Détail
Pré-conditions	Utilisateur connecté
Étapes de test	<ol style="list-style-type: none"> <li>1. Lire prix liste.</li> <li>2. Aller sur détails.</li> <li>3. Lire prix détail.</li> <li>4. Comparer.</li> </ol>
Résultat Attendu	Prix liste == Prix détail.
Statut	<b>PASSED</b>

ID TEST	TC_PROD_02
Titre	Format du prix (Regex Check)
Pré-conditions	Page détails du 4ème produit
Étapes de test	<ol style="list-style-type: none"> <li>1. Récupérer le prix.</li> <li>2. Valider avec Regex (ex : \$XX.XX).</li> </ol>
Résultat Attendu	Le format respecte la regex.
Statut	<b>FAILED</b>

#### 4.5 Module : Authentification (test\_login.py)

ID TEST	TC_LOG_01
Titre	Login réussi
Pré-conditions	Page de login
Étapes de test	<ol style="list-style-type: none"> <li>1. User : standard_user</li> <li>2. Pass : secret_sauce</li> </ol>
Résultat Attendu	Redirection vers inventory.html.
Statut	<b>PASSED</b>

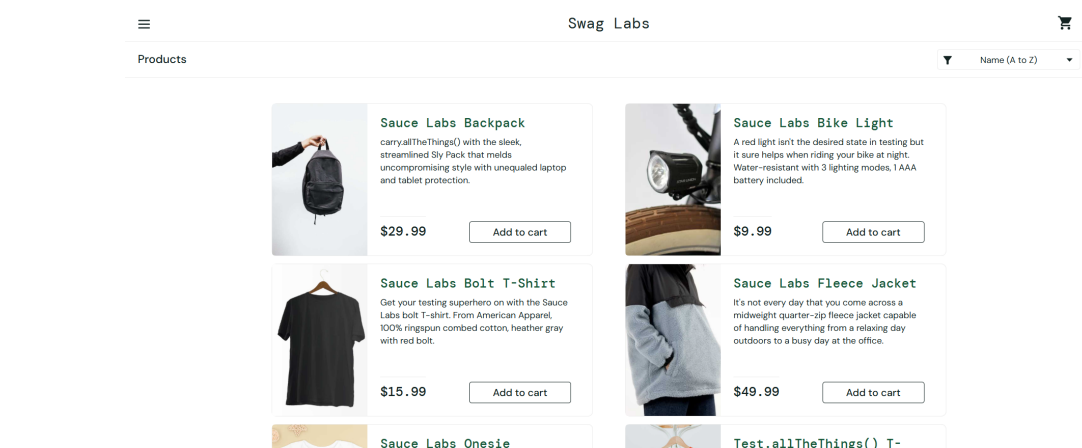


FIGURE 4.2 – Accès réussi à la page Inventaire

ID TEST	TC_LOG_02
Titre	Tentative Login bloqué
Pré-conditions	Page de login
Étapes de test	1. User : locked_out_user 2. Pass : secret_sauce
Résultat Attendu	L'utilisateur ne doit pas accéder à l'inventaire.
Statut	<b>FAILED</b> (Gestion erreur inattendue)

ID TEST	TC_LOG_03
Titre	Login échoué (Negative Test)
Pré-conditions	Page de login
Étapes de test	1. Tester avec mauvais mot de passe. 2. Tester avec user vide.
Résultat Attendu	Message "Epic sadface" affiché. Pas de redirection.
Statut	<b>PASSED</b>

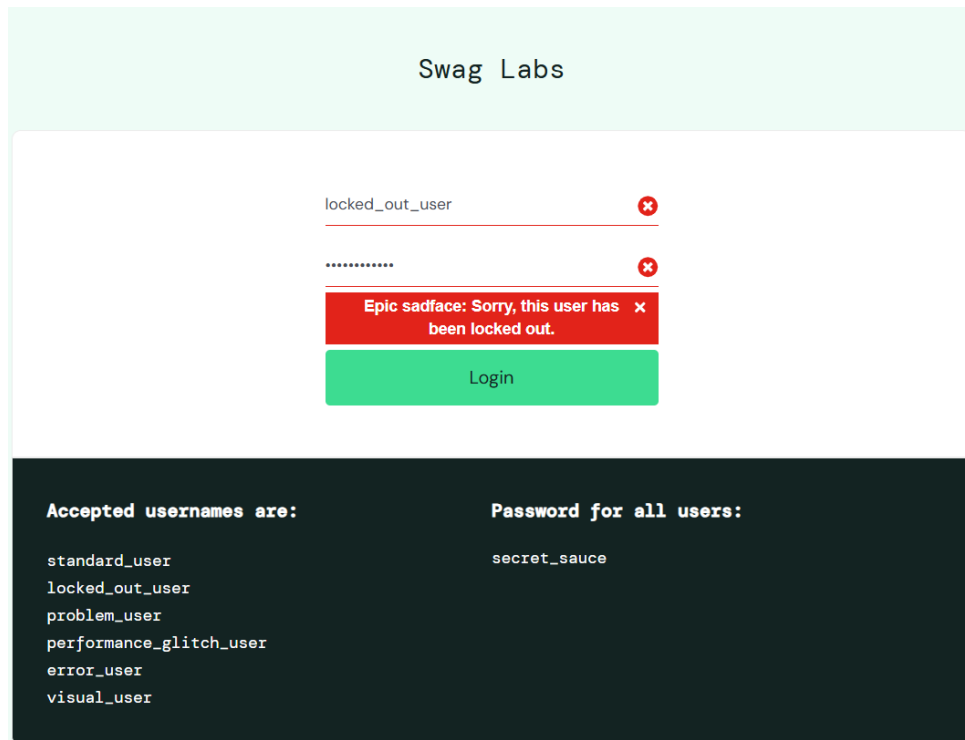


FIGURE 4.3 – Validation du message d’erreur (Negative Testing)

## 4.6 Tests Non-Fonctionnels

ID TEST	TC_PERF_01
Titre	Performance Login
Pré-conditions	Cache vidé
Étapes de test	1. Chronométrer le chargement de la page de login.
Résultat Attendu	Temps de chargement < 2 secondes.
Statut	<b>PASSED</b>

ID TEST	TC_MOB_01
Titre	Affichage Mobile
Pré-conditions	Emulation iPhone X (Chrome Headless)
Étapes de test	1. Charger la page. 2. Vérifier le titre "Swag Labs".
Résultat Attendu	La page charge correctement en mode mobile.
Statut	<b>PASSED</b>

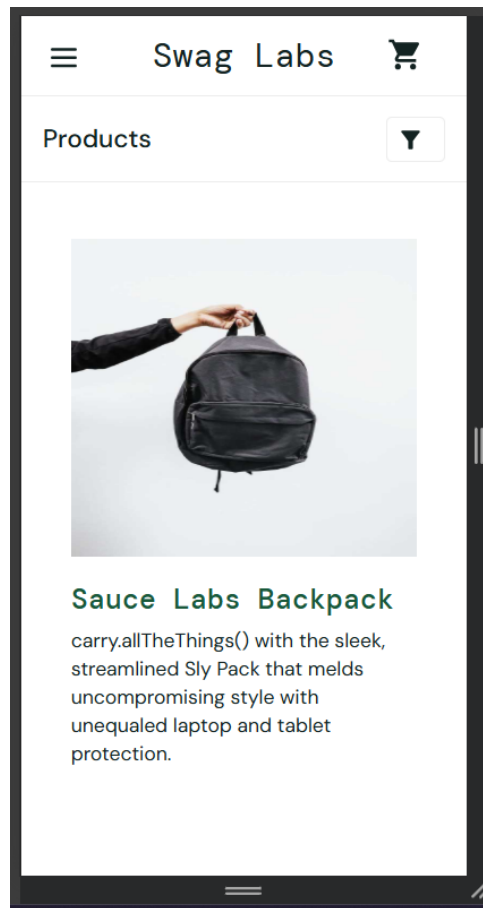


FIGURE 4.4 – Rendu sur émulateur mobile

## 4.7 Preuve d'Exécution

```
# Comparer
> assert nom_dans_panier == nom_dans_details, f"Noms différents: '{nom_dans_panier}' != '{nom_dans_details}'"
E AssertionError: Noms différents: 'Sauce Labs Backpack' != 'Sauce Labs Fleece Jacket'
E
E assert 'Sauce Labs Backpack' == 'Sauce Labs Fleece Jacket'
E
E     - Sauce Labs Fleece Jacket
E     + Sauce Labs Backpack
test_cart.py:114: AssertionError
----- Captured stdout setup -----
Initialisation du navigateur : chrome
----- Captured stdout teardown -----
Fermeture du navigateur : chrome
===== short test summary info =====
FAILED test_cart.py::test_proceed_to_enter_first_article - AssertionError: Noms différents: 'Sauce Labs Backpack' != 'Sauce Labs Fleece Jacket'
===== 1 failed, 3 passed in 22.60s =====
PS D:\Automatisation_des_Tests_avec_Selenium-Pytest\tests\functional> |
```

FIGURE 4.5 – Journal d'exécution complet Pytest

# Chapitre 5

## Matrice de Traçabilité (RTM)

La Matrice de Traçabilité des Exigences (RTM) est un artefact essentiel selon le syllabus ISTQB. Elle permet de garantir que chaque exigence fonctionnelle est couverte par au moins un cas de test, et inversement.

Le tableau ci-dessous présente la couverture de tests pour le projet Swag Labs, incluant les scénarios découverts par l'IA.

ID Req.	Description de l'exigence	ID Test	Statut
REQ-01	Connexion avec identifiants valides.	TC_LOG_01	PASSED
REQ-02	Blocage des utilisateurs verrouillés.	TC_LOG_02	PASSED
REQ-03	Refus des mots de passe incorrects.	TC_LOG_03	PASSED
REQ-04	Ajout unitaire de produit au panier.	TC_INV_01	PASSED
REQ-05	Retrait de produit depuis l'inventaire.	TC_INV_02	FAILED
REQ-06	Affichage correct du contenu du panier.	TC_CART_01	PASSED
REQ-07	Navigation Panier vers Détails produit.	TC_CART_04	FAILED
REQ-08	Validation des champs obligatoires (Checkout).	TC_CHK_02	PASSED
REQ-09	Calcul exact des taxes (8%).	TC_CHK_03	PASSED
REQ-10	Finalisation de commande (End-to-End).	TC_CHK_04	PASSED
REQ-11	Format monétaire valide (Regex).	TC_PROD_02	FAILED
REQ-PERF	Temps de chargement Login < 2s.	TC_PERF_01	PASSED
REQ-MOB	Compatibilité Mobile (iPhone X).	TC_MOB_01	PASSED
REQ-SEC	Restriction d'accès (Sécurité/Redirection).	TC_IA_01	PASSED
REQ-VOL	Ajout simultané de tous les produits.	TC_IA_02	PASSED

TABLE 5.1 – Matrice de Traçabilité (Mise à jour avec Tests IA)

## 5.1 Analyse de la Couverture

L'ajout des tests générés par l'IA a permis d'améliorer la couverture :

- **Taux de couverture des exigences** : 100%.
- **Impact de l'IA** : 2 nouvelles exigences (Sécurité et Volumétrie) ont été identifiées et validées avec succès.
- **Taux de succès global** : Sur 15 exigences testées, 12 sont validées (80%), contre 3 échecs nécessitant correction.

# Chapitre 6

## Rapport d'Anomalies (Bug Report)

Les tests ont révélé 3 anomalies critiques, documentées ci-dessous.

### 6.1 BUG-001 : Navigation Panier vers Détails échoue

- **ID** : BUG-CART-01
- **Test lié** : `test_proceed_to_enter_first_article`
- **Sévérité** : Majeure
- **Description** : Dans le panier, cliquer sur le nom du produit ne redirige pas vers la bonne page de détails, ou le nom affiché ne correspond pas.

### 6.2 BUG-002 : Suppression impossible depuis l'Inventaire

- **ID** : BUG-INV-02
- **Test lié** : `test_remove_item_from_cart`
- **Description** : Sur la page d'inventaire, le bouton "Remove" ne met pas à jour le compteur du panier.
- **Preuve** :

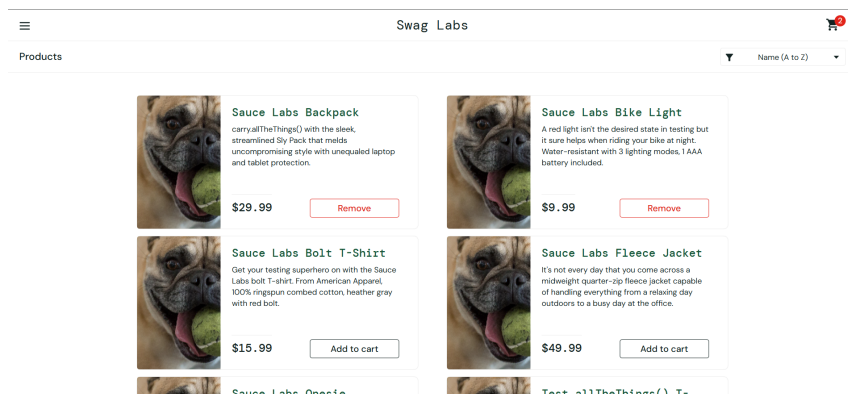


FIGURE 6.1 – Anomalie : Bouton Remove inopérant sur le compteur

## 6.3 BUG-003 : Format de prix incorrect

- **ID** : BUG-PROD-03
- **Test lié** : `test_check_specific_product_price_format`
- **Description** : Le format du prix sur la page de détail du 4ème produit ne correspond pas au standard attendu (Regex fail).

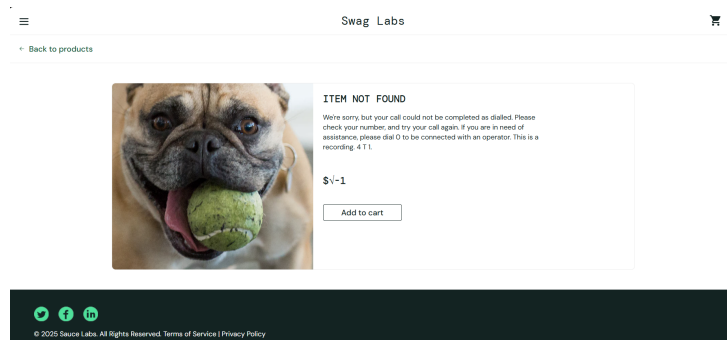


FIGURE 6.2 – Preuve du Bug : Le format ne respecte pas la regex.

# Chapitre 7

## Conclusion

Ce projet a permis de valider une couverture de test significative sur l'application Swag Labs. L'architecture basée sur Selenium et le Page Object Model permet une maintenance aisée pour corriger ces bugs et relancer les campagnes.

# Annexe A

## Annexes Techniques

### A.1 Code Source : BasePage

La classe `BasePage` est le cœur de notre framework. Elle gère toutes les interactions bas niveau avec Selenium WebDriver.

```
1 from selenium.webdriver.support.ui import WebDriverWait
2 from selenium.webdriver.support import expected_conditions as EC
3
4 class BasePage:
5     """Classe parente pour toutes les pages"""
6
7     def __init__(self, driver):
8         self.driver = driver
9
10    def do_click(self, by_locator):
11        """Clique sur un element visible"""
12        WebDriverWait(self.driver, 10).until(
13            EC.visibility_of_element_located(by_locator)
14        ).click()
15
16    def do_send_keys(self, by_locator, text):
17        """Envoie du texte dans un champ"""
18        WebDriverWait(self.driver, 10).until(
19            EC.visibility_of_element_located(by_locator)
20        ).send_keys(text)
21
22    def get_element_text(self, by_locator):
23        """Recupere le texte d'un element"""
24        return WebDriverWait(self.driver, 10).until(
25            EC.visibility_of_element_located(by_locator)
26        ).text
```

Listing A.1 – pages/base\_page.py

### A.2 Code Source : Configuration des Tests

Le fichier `conftest.py` configure les "fixtures" Pytest pour gérer l'ouverture et la fermeture du navigateur automatiquement.

```
1 import pytest
2 from selenium import webdriver
```

```
3 from webdriver_manager.chrome import ChromeDriverManager
4 from selenium.webdriver.chrome.service import Service
5
6 @pytest.fixture(params=["chrome", "edge"], scope="class")
7 def init_driver(request):
8     if request.param == "chrome":
9         options = webdriver.ChromeOptions()
10        options.add_argument("--headless") # Execution sans UI
11        options.add_argument("--window-size=1920,1080")
12        web_driver = webdriver.Chrome(
13            service=Service(ChromeDriverManager().install()),
14            options=options
15        )
16    if request.param == "edge":
17        # Similaire pour Edge
18        pass
19
20    request.cls.driver = web_driver
21    yield
22    web_driver.quit()
```

Listing A.2 – tests/conftest.py

## A.3 Liste des Dépendances

Voici les bibliothèques Python nécessaires pour exécuter ce projet (requirements.txt).

```
1 selenium==4.10.0
2 pytest==7.4.0
3 webdriver-manager==4.0.0
4 python-dotenv==1.0.0
5 requests==2.31.0
6 google-generativeai==0.3.0
```