

Projet ISTQB

Tests logiciels automatisés d'une application SaaS

Auteurs :

Amine Abbes

Med Bechir Torki

Application testée :

Plateforme de réservation en ligne **Trafft**

Table des matières

1	Introduction	7
2	Présentation de l'application Trafft	8
2.1	Fonctionnalités principales	8
2.2	Parcours utilisateur	8
2.3	Intérêt pour le test logiciel	8
3	Objectifs et périmètre des tests	9
3.1	Objectifs	9
3.2	Périmètre	9
4	Méthodologie de test ISTQB	10
4.1	Analyse des exigences	10
4.2	Conception des tests	10
4.3	Implémentation et exécution	10
5	Techniques de test utilisées	11
5.1	Tests fonctionnels	11
5.2	Tests négatifs	11
5.3	Analyse des valeurs limites	11
5.4	Tests basés sur les états	11
5.5	Tests de performance et de stress	11
6	Architecture du projet de test	12
6.1	Page Object Model	12
6.2	Organisation des suites de tests	12
7	Présentation détaillée des cas de test	13
8	Résultats d'exécution et analyse	14
9	Gestion des anomalies	15
9.1	Bug TC04 – Email en majuscule	15

9.2	Bug TC10 – Téléphone invalide	16
9.3	Bug TC12 – Confusion d’annulation	16
9.4	Bug TC21 – Concurrence	17
10	Limites et perspectives	19
11	Conclusion	20

Liste des tableaux

1	Classification des types de tests	3
2	Environnement matériel et logiciel de test	3
3	Résultats globaux des tests	3

TABLE 1 – Classification des types de tests

Type de test	Description
Tests fonctionnels	Validation des fonctionnalités principales
Tests négatifs	Vérification du rejet des entrées invalides
Tests BVA	Analyse des valeurs limites
Tests basés sur les états	Vérification des transitions de réservation
Tests de performance	Mesure des temps de réponse
Tests de stress	Comportement sous charge élevée
Tests cross-browser	Compatibilité multi-navigateurs

TABLE 2 – Environnement matériel et logiciel de test

Élément	Valeur
Système d'exploitation	Windows 10 64 bits
Navigateur principal	Google Chrome
Navigateurs secondaires	Firefox, Microsoft Edge
Langage	Python 3.11
Framework de test	PyTest
Outil d'automatisation	Selenium WebDriver
IDE	PyCharm

TABLE 3 – Résultats globaux des tests

Indicateur	Valeur
Nombre total de cas de test	22
Cas de test réussis	18
Cas de test échoués	4
Taux de réussite	81.8%
Durée totale d'exécution	50 minutes

Table des figures

1	Architecture générale du projet de test automatisé	5
2	Cycle de vie du test selon l'ISTQB	6
3	Exécution réussie d'un cas de test automatisé	6
9.1	*	15
9.2	*	15
9.3	Bug TC04 – Acceptation incorrecte des emails en majuscules	15
9.4	*	16
9.5	*	16
9.6	Bug TC10 – Validation incorrecte du numéro de téléphone	16
9.7	*	17
9.8	*	17
9.9	Bug TC12 – Confusion des états de réservation	17
9.10	Bug TC21 – Acceptation incorrecte de réservations concurrentes	18

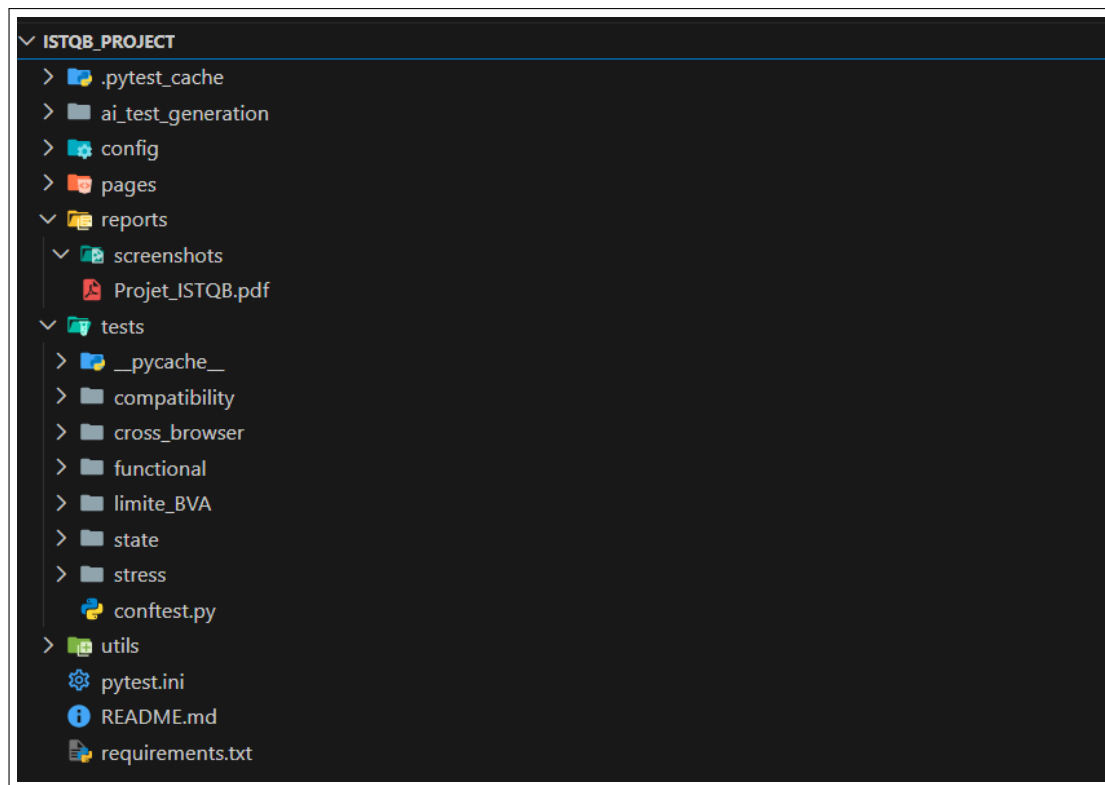


FIGURE 1 – Architecture générale du projet de test automatisé

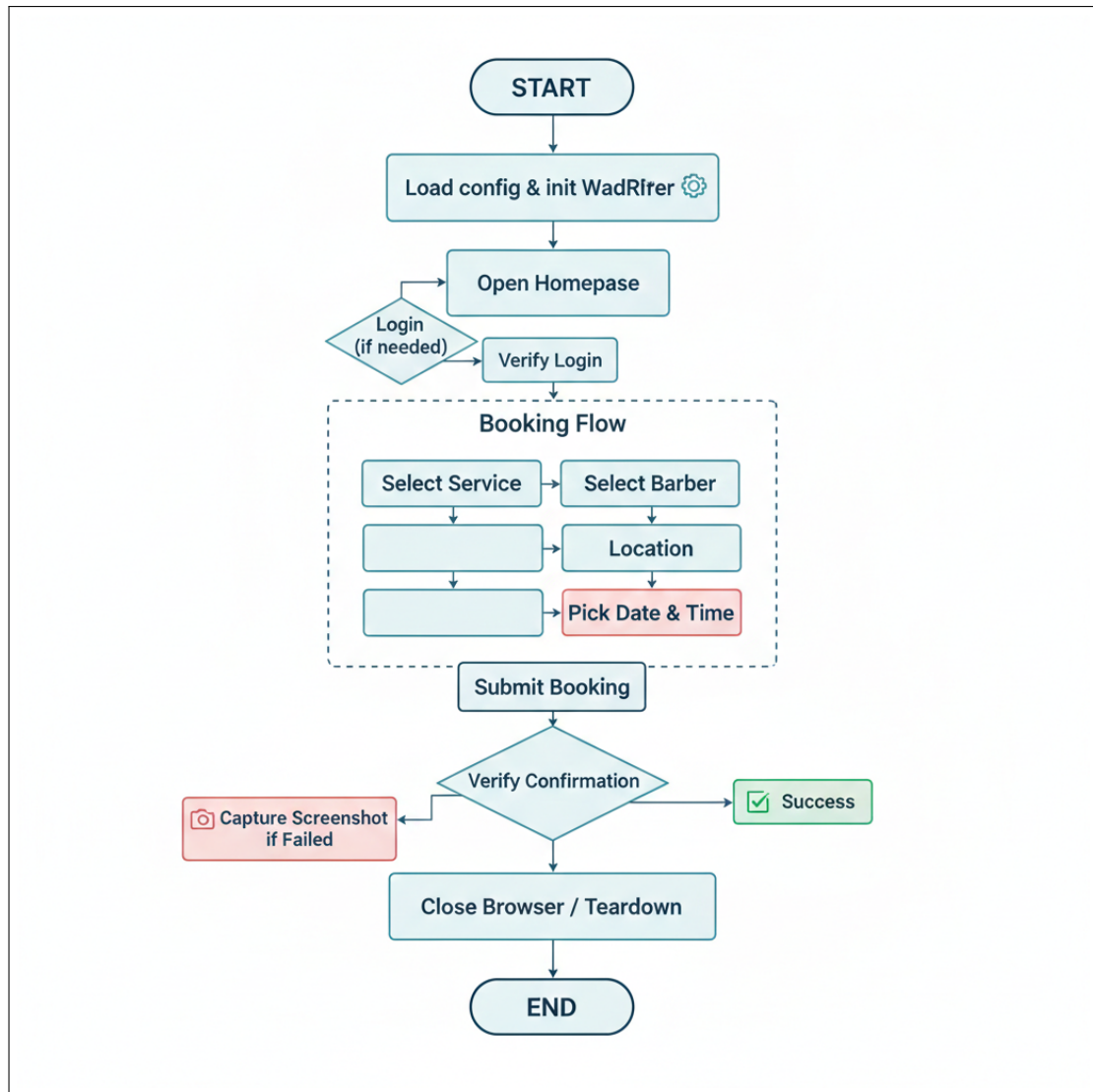


FIGURE 2 – Cycle de vie du test selon l'ISTQB

```

===== test session starts =====
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0
rootdir: C:\Users\dell\OneDrive\Desktop\ISTQB_Project
configfile: pytest.ini
collected 1 item

tests\functional\login_test.py
[INFO] Tentative avec l'email: abbesamine01@gmail.com et le pass: aaabbbccc
Clic Javascript effectué sur le bouton Log In
RÉSULTAT : TC02 réussi avec le mot de passe aaabbbccc

1 passed in 36.81s
  
```

FIGURE 3 – Exécution réussie d'un cas de test automatisé

1 Introduction

Le test logiciel constitue une activité essentielle dans le cycle de vie du développement des systèmes informatiques modernes. Avec la généralisation des applications web et des plateformes SaaS, la qualité logicielle est devenue un facteur déterminant de la satisfaction utilisateur et de la pérennité des entreprises numériques.

Dans ce contexte, la certification ISTQB fournit un cadre méthodologique internationalement reconnu pour structurer les activités de test, depuis l'analyse des exigences jusqu'à la gestion des anomalies. Le présent projet s'inscrit dans cette démarche et vise à appliquer concrètement les concepts théoriques à travers un cas réel.

Ce travail porte sur l'automatisation de tests logiciels pour une application web de réservation de services. L'automatisation a été choisie afin de garantir la répétabilité des tests, d'améliorer la couverture fonctionnelle et de réduire les erreurs humaines.

Ce rapport présente successivement l'application testée, la méthodologie adoptée, les techniques de test utilisées, les résultats obtenus ainsi que les anomalies détectées, avant de conclure sur les limites et les perspectives d'amélioration.

2 Présentation de l'application Trafft

Trafft est une plateforme SaaS dédiée à la réservation de services en ligne. Elle vise à digitaliser l'ensemble du parcours client, depuis la découverte des services jusqu'à la confirmation et la gestion des rendez-vous.

2.1 Fonctionnalités principales

L'application permet aux utilisateurs de consulter un catalogue de services, de sélectionner un prestataire (barber), de choisir un lieu (location) et de réserver un créneau horaire via un calendrier interactif. Les utilisateurs peuvent effectuer ces actions en tant qu'invités ou via un compte authentifié.

2.2 Parcours utilisateur

Le parcours utilisateur repose sur une succession d'étapes critiques : sélection du service, choix du professionnel, sélection de la date et de l'heure, saisie des informations personnelles et confirmation finale. Toute erreur à l'une de ces étapes peut entraîner une perte de réservation ou une insatisfaction client.

2.3 Intérêt pour le test logiciel

Du point de vue de l'ingénieur QA, Trafft présente plusieurs défis :

- Gestion de formulaires complexes,
- Transitions d'états sensibles (annulation, reprogrammation),
- Problèmes de concurrence lors de réservations simultanées,
- Validation des données utilisateur.

3 Objectifs et périmètre des tests

L'objectif principal de ce projet est de vérifier la conformité fonctionnelle et la robustesse de l'application Trafft face à différents scénarios d'utilisation.

3.1 Objectifs

- Détecter les défauts fonctionnels critiques,
- Valider les règles métier liées aux réservations,
- Évaluer le comportement de l'application sous charge,
- Vérifier la compatibilité multi-navigateurs.

3.2 Périmètre

Les tests couvrent les fonctionnalités accessibles côté client. Les aspects internes tels que les API backend ou la base de données ne sont pas directement testés dans ce projet.

4 Méthodologie de test ISTQB

La méthodologie adoptée suit le cycle de vie des tests défini par l'ISTQB.

4.1 Analyse des exigences

Les exigences ont été déduites à partir du comportement observable de l'application et des règles métier implicites.

4.2 Conception des tests

Les cas de test ont été conçus en utilisant différentes techniques ISTQB afin d'assurer une couverture maximale.

4.3 Implémentation et exécution

Les tests ont été automatisés à l'aide de Selenium WebDriver et du framework PyTest, en appliquant le modèle Page Object Model pour améliorer la maintenabilité.

5 Techniques de test utilisées

5.1 Tests fonctionnels

Ils permettent de vérifier que les fonctionnalités principales répondent aux exigences.

5.2 Tests négatifs

Ces tests visent à s'assurer que l'application réagit correctement aux entrées invalides.

5.3 Analyse des valeurs limites

La technique BVA a été utilisée pour tester les champs sensibles tels que le nom, le téléphone et l'email.

5.4 Tests basés sur les états

Les transitions entre les différents états des réservations ont été vérifiées afin d'éviter les incohérences.

5.5 Tests de performance et de stress

Ces tests évaluent la capacité du système à gérer plusieurs requêtes simultanées et à maintenir des temps de réponse acceptables.

6 Architecture du projet de test

Le projet est structuré selon une architecture modulaire respectant les bonnes pratiques de l'automatisation des tests.

6.1 Page Object Model

Chaque page de l'application est représentée par une classe dédiée, ce qui permet de centraliser les localisateurs et les actions.

6.2 Organisation des suites de tests

Les tests sont regroupés par catégorie : fonctionnels, performance, cross-browser et stress.

7 Présentation détaillée des cas de test

Les cas de test suivants ont été conçus et exécutés :

ID	Catégorie	Scénario	Statut attendu
TC01	Fonctionnel	Création d'un compte client	Succès
TC02	Fonctionnel	Connexion valide	Succès
TC03	Fonctionnel	Mot de passe erroné	Échec
TC04	Fonctionnel	Email en majuscule	Échec
TC05	Fonctionnel	Booking utilisateur connecté	Succès
TC06	Fonctionnel	Booking invité	Succès
TC07	BVA	Nom 255 caractères	Succès
TC08	BVA	Nom 256 caractères	Échec
TC09	BVA	Téléphone invalide	Échec
TC10	BVA	Téléphone alphabétique	Échec
TC11	BVA	Email invalide	Échec
TC12	États	Annulation multiple	Succès
TC13	États	Reprogrammation	Succès
TC14	États	Déconnexion en cours	Succès
TC15	Données	Vérification du prix	Succès
TC16	Cross-browser	Chrome	Succès
TC17	Cross-browser	Firefox	Succès
TC18	Cross-browser	Edge	Succès
TC19	Compatibilité	Mobile	Succès
TC20	Compatibilité	Tablette	Succès
TC21	Stress	Réservations simultanées	Échec
TC22	Performance	Temps de chargement	Succès

8 Résultats d'exécution et analyse

L'exécution des tests a permis d'identifier plusieurs anomalies critiques, notamment liées à la validation des données et à la gestion de la concurrence.

Un taux global de réussite élevé a été observé, mais les échecs détectés présentent un impact métier important.

9 Gestion des anomalies

9.1 Bug TC04 – Email en majuscule

Description : Le système accepte des emails en majuscules alors qu'un rejet était attendu.

Impact : Risque de duplication de comptes et incohérence des données d'authentification.

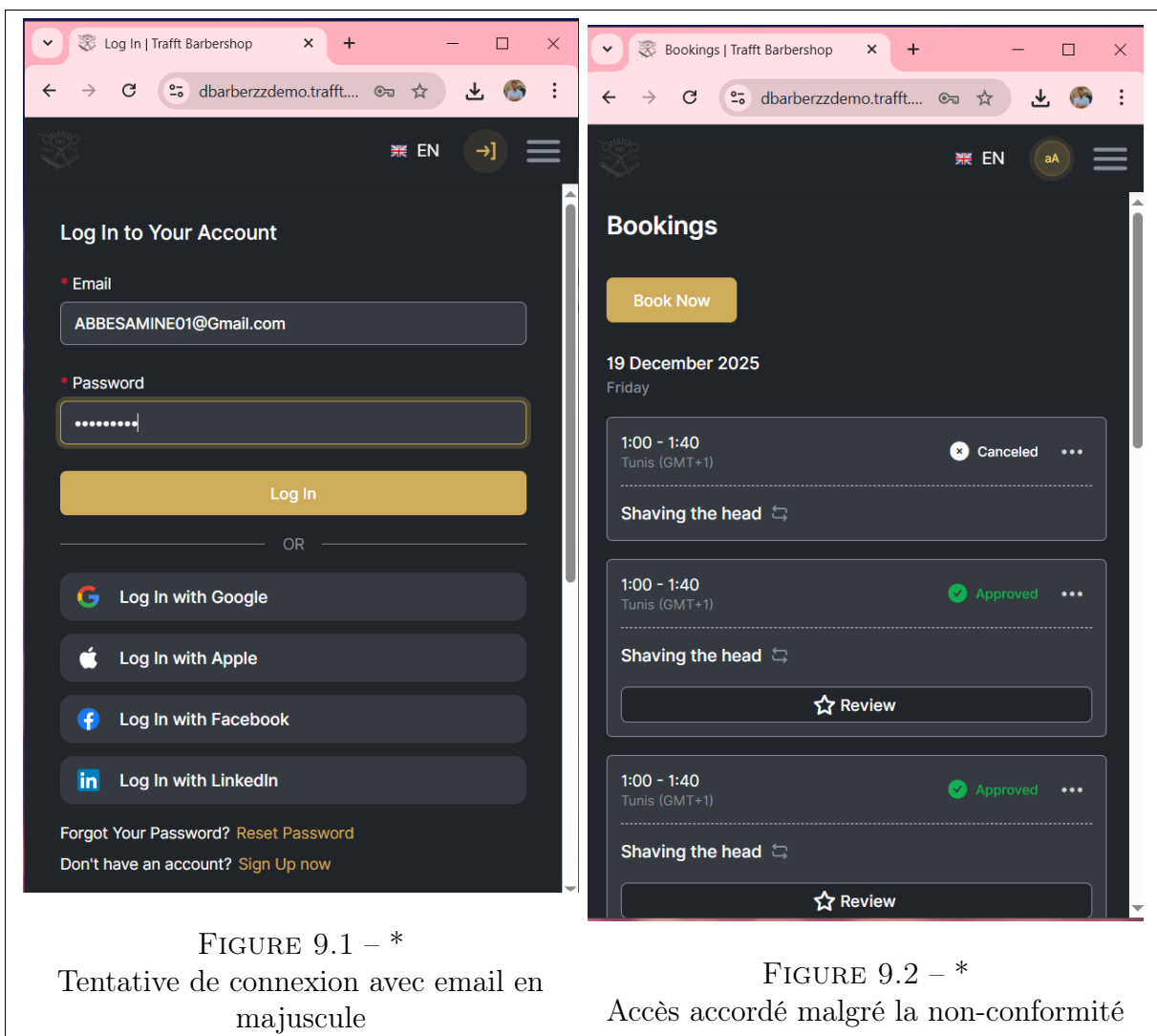


FIGURE 9.3 – Bug TC04 – Acceptation incorrecte des emails en majuscules

9.2 Bug TC10 – Téléphone invalide

Description : Le champ téléphone accepte des chaînes alphabétiques.

Impact : Données de contact non fiables et perte de communication avec le client.

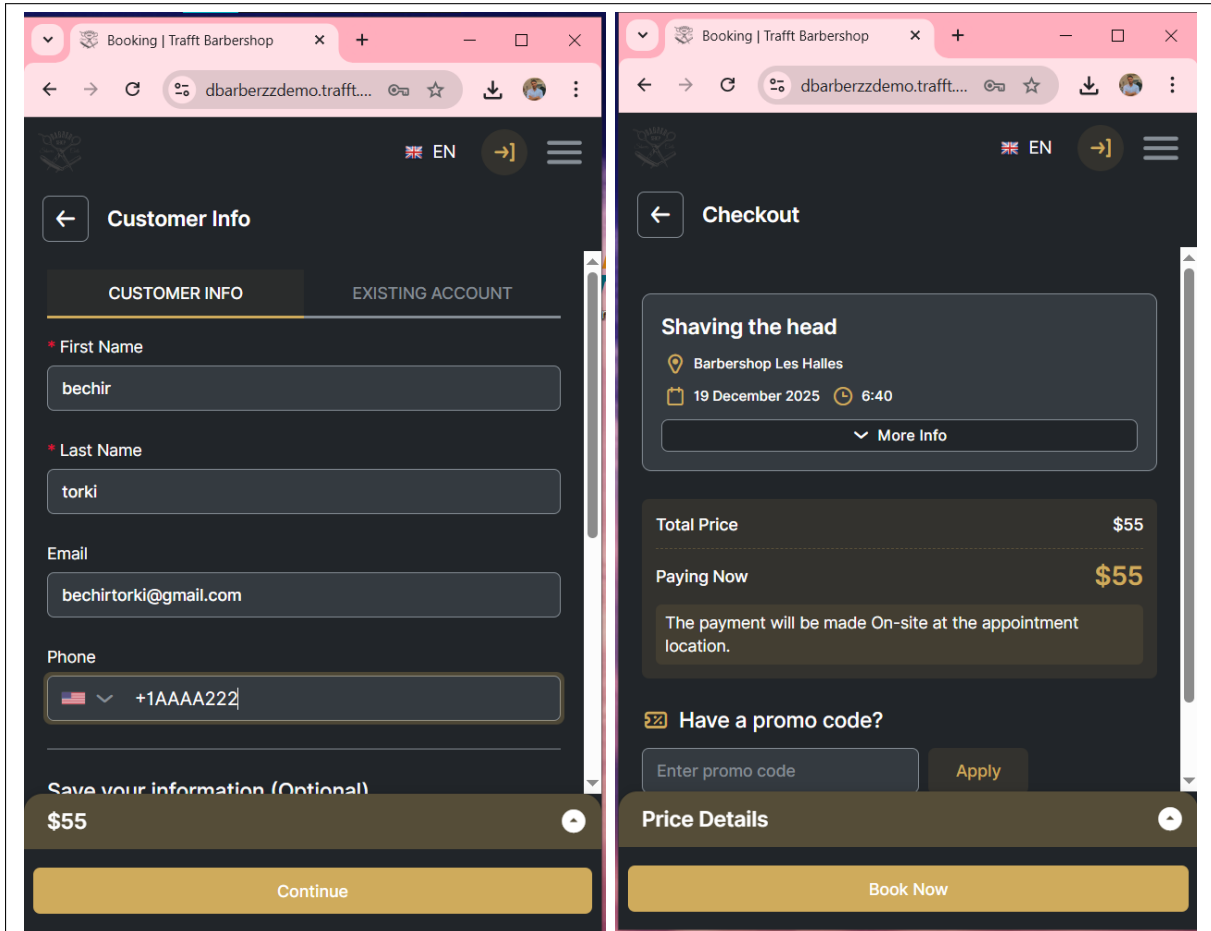


FIGURE 9.4 – *
Saisie d'un numéro alphabétique

FIGURE 9.5 – *
Validation acceptée à tort

FIGURE 9.6 – Bug TC10 – Validation incorrecte du numéro de téléphone

9.3 Bug TC12 – Confusion d'annulation

Description : Le système confond les identifiants lors de multiples annulations d'un même rendez-vous.

Impact : Incohérence des états de réservation et erreurs métier.

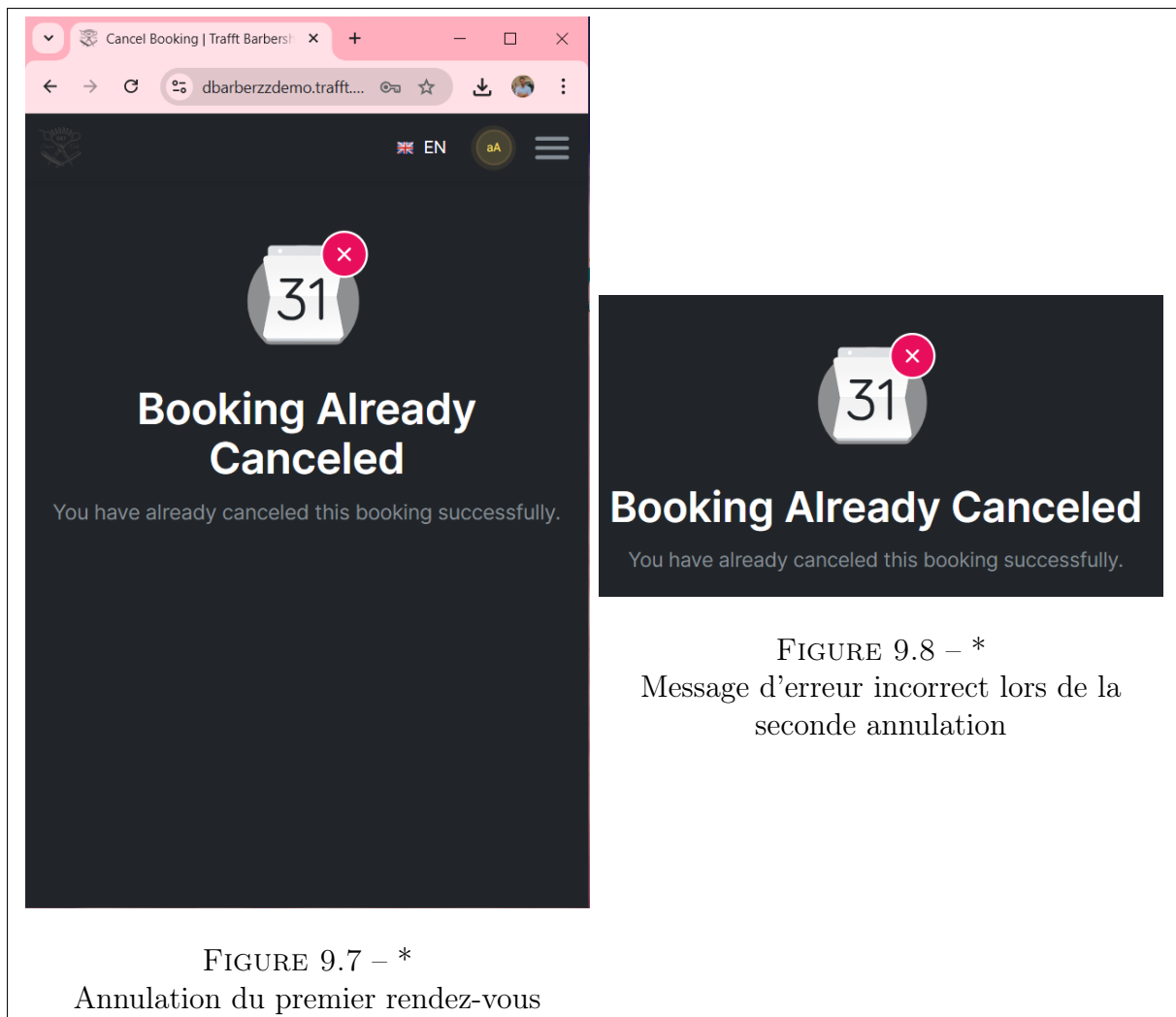


FIGURE 9.9 – Bug TC12 – Confusion des états de réservation

9.4 Bug TC21 – Concurrency

Description : Le système accepte plusieurs réservations simultanées pour le même créneau horaire.

Impact : Double booking critique impactant directement l'activité métier.

```
[ACTION] Clic sur le bouton final Book Now.
[ACTION] Clic sur le bouton final Book Now.
[ACTION] Clic sur le bouton final Book Now.
[ACTION] Clic sur le bouton final Book Now.
[ACTION] Clic sur le bouton final Book Now.

=====
RÉSULTATS DU TEST DE CONCURRENCE
=====
Utilisateur: abbesamine05@gmail.com | Résultat: SUCCESS | Message: Réservation réussie (Gagnant)
Utilisateur: abbesamine02@gmail.com | Résultat: SUCCESS | Message: Réservation réussie (Gagnant)
Utilisateur: abbesamine04@gmail.com | Résultat: SUCCESS | Message: Réservation réussie (Gagnant)
Utilisateur: abbesamine03@gmail.com | Résultat: SUCCESS | Message: Réservation réussie (Gagnant)
Utilisateur: abbesamine01@gmail.com | Résultat: SUCCESS | Message: Réservation réussie (Gagnant)
F
```

FIGURE 9.10 – Bug TC21 – Acceptation incorrecte de réservations concurrentes

10 Limites et perspectives

Les tests se limitent à la couche frontend. Des améliorations futures pourraient inclure des tests API, une intégration CI/CD et l'utilisation de l'IA générative pour la conception automatique des tests.

11 Conclusion

Ce projet a permis d'appliquer concrètement les concepts ISTQB à un cas réel. L'automatisation a démontré son efficacité dans la détection d'anomalies critiques et constitue un levier essentiel pour la qualité des applications SaaS modernes.