TP INF 362 : JavaScript Unit Testing With Vi-test

PLAN DE TEST

EXAMINATEUR : DR KIMBI XAVIERA

GROUPE 8

MEMBRES	**MATRICULES**
KWEM PEK SAMUEL ROSTAND	**19M2094**
MBO'O ATENA SIDONIE ORNELLA	**18T2746**

SOMMAIRE :

1. Introduction

Ce document présente le plan de test pour l'application mobile d'un restaurant local. L'application a pour objectif de permettre aux utilisateurs de commander des plats et d'obtenir des promotions. Le plan de test définit les étapes à suivre pour garantir le bon fonctionnement de l'application et identifier les éventuels problèmes avant son lancement.

2. Portée

Dépendances fonctionelles :

module	Role applicable	description
Afficher les publicites	Le systeme	Les publicités ne sont pas diffuses au comptes premuim
Creation d'un compte	L'utilisateur	Une adresse email identifie un seul utilisateur
La date d'un evenement	Le systeme	Une date doit etre etre plus grande que la date actuel et plus petite que la date future
Recherche d'un evenements	utilisateur	L'evenement recherché doit faire partir de la liste d'evenement

Creation d’un evenements	Utilisateur	-Le nom d’un evenement ne doit pas depasser 200 carracteres - le prix d’un evenement doit etre superieur ou egal a 0 -les tickets disponible doivent etre superieur ou egal a 0
Verification de l’exchange rate		La monnaie utiliséé doit etre soit EUR , USD , NZD
Verification du username		Le username doit contenir @

Portée non fonctionelles :

Interfaces utilisateur

Interfaces matérielles

Interfaces logicielles

Base de données logique

Interfaces de communication

Sécurité et performances de l’app

3. Objectifs

Les objectifs sont de vérifier les fonctionnalités présent dans les 5 sous dossiers, le projet devrait se consacrer sur le test qui :

- gère des articles de panier et des événements associés,

-gère des erreurs spécifiques à chaque partie du code,

-gère des évènements et leurs disponibilités,

-gère l'application des codes promos en fonction de la monnaie utilisée,

- gère la connexion et l'authentification

4. Approche

Les tests seront effectués à l'aide de tests unitaires écrits avec le framework Vitest. Les tests couvriront les différentes fonctionnalités mentionnées dans la portéé .

5. Calendrier

Taches	**membres**	**Effort**
Réunion de lancement du projet et définition des objectifs de test. -Identifier les fonctionnalités clés de l'application et les cas de test à couvrir. -Prioriser les cas de test en fonction de la criticité et du risque.	Tous les membres	Semaine 1 : Jour 1, 2, 3

-Mettre en place l'environnement de test et configurer Vitest.		
Développer les cas de test de base pour les fonctionnalités essentielles. Se concentrer sur les fonctionnalités critiques pour l'utilisateur, telles que la création de paniers, l'application de promotions et le processus de paiement. Utiliser Vitest pour écrire des tests unitaires	Tous les membres	Semaine 1 : jour 4 et 5
Exécuter les cas de test de base et corriger les bugs identifiés. • Automatiser l'exécution des tests à l'aide d'un outil d'intégration continue (CI). • Documenter les bugs et les problèmes rencontrés.	Tous les membres	Semaine 1 : jour 6-7

6. Environnement

- L'environnement utilisé est Visual Studio Code
- Le framework utilisé est vitest

- Le SE utilisé est Windows 10
- Node JS 20.11.0

7. Données de test

-**source** : les données à tester se trouve dans

le git [GitHub - atemengue/software_testing_labs](#) ,

-**type de donnés** : constitué de plusieurs fichier Java script

-**contraintes** : compatible avec nodeJS 20.11.0

8.Cas de test

Test case	Description	Resultat attendu
L'evenement est aujourd'hui	aujourd'hui= eventDate	today devrait retourner true
L'evenement est dans 7 jours	next7Days devrait retourner un tableau d'événements dans les 7 prochains jours	Le test devrait passer (retourner true). Le tableau retourné par next7Days devrait contenir uniquement les événements avec des dates comprises entre le jour actuel et 7 jours après le jour actuel (incluant les deux). Dans ce cas précis, seuls les événements avec

		les identifiants 1 et 2 devraient être retournés.
L'événement n'a plus de billets restants	(ticketsRemaining est égal à 0)	Le test doit passer et retourner true.
L'événement a encore des billets restants	(ticketsRemaining est supérieur à 0)	Le test doit passer et retourner false
L'événement est complet	L'événement est complet (	isSoldOut retourne true)
Le nom de l'événement est une chaîne de caractères vide	Const eventname=' '	Le test doit échouer avec une erreur InvalidEventNameError indiquant que le nom de l'événement ne peut pas être vide.
Le nom de l'événement est une chaîne de caractères de plus de 200 caractères	Le nom de l'événement est une chaîne de caractères de plus de 200 caractères	Le test doit échouer avec une erreur InvalidEventNameError indiquant que le nom de l'événement ne peut pas dépasser 200 caractères.
Le prix de l'événement n'est pas un nombre	Le prix de l'événement n'est pas un nombre	Le test doit échouer avec une erreur InvalidEventPriceError indiquant que le prix de l'événement doit être un nombre.
L'événement est dans les 30 prochains jours.	L'événement est dans les 30 prochains jours.	La fonction next30Days devrait retourner true.
	L'événement est dans le futur, mais pas dans les 30 prochains jours.	La fonction next30Days devrait retourner false.

Le panier contient un seul article	const basketItems = [{ getPrice: () => 10 }];	La fonction calculateTotal devrait retourner le prix de l'article qui est 10
Le panier contient plusieurs articles	Calcul le total correct pour plusieurs article { getPrice: () => 10 }, { getPrice: () => 20 }, { getPrice: () => 30 },	La fonction calculateTotal devrait retourner la somme des prix de tous les articles du panier. Donc 60
Le panier est vide	const basketItems = [];	La fonction calculateTotal devrait retourner 0.
Le panier contient un article avec un prix réduit	getPrice: () => 10 }, { getPrice: () => 20 }, { getPrice: () => 30 },]; const discount = 5;	La fonction calculateTotal devrait prendre en compte le prix réduit lors du calcul du total.
L'utilisateur est premium	const user = { isPremium: true }	La fonction showAdverts devrait retourner false
L'utilisateur n’est pas premium	const user = { isPremium: false}	La fonction showAdverts devrait retourner true
L'utilisateur recherche "concert"	L'utilisateur recherche le terme "concert" ,La fonction searchEvents devrait retourner un tableau d'objets Event contenant les informations sur les événements pertinents, y compris les descriptions.	Le tableau d'objets Event retourné devrait contenir les informations suivantes pour chaque événement pertinent qui sont l’id , le nom , date , venue , description de l’evenement
L'utilisateur recherche une chaîne de caractères qui n’existe pas	L'utilisateur recherche une qui n’existe, La fonction	Le tableau d'objets Event retourné devrait être vide

	searchEvents devrait retourner un tableau vide	
Le prix actuel est moins que le minumum depensé	Le test simule un rabais de 20% (percentage est défini à 20). Le test définit un montant minimum d'achat requis pour bénéficier de la remise (minimumSpend est défini à 100). Le prix actuel de l'article est de 120 (currentPrice est défini à 120).	Le test attend un prix final de 96 (discountedPrice est attendu à 96).
Le prix actuel est moins que le prix depensé	Le test simule un rabais de 20% (percentage est défini à 20). Le test définit un montant minimum d'achat requis pour bénéficier de la remise (minimumSpend est défini à 100). Le prix actuel de l'article est de 80(currentPrice est défini à 80).	Le test attend un prix final de 80(discountedPrice est attendu à 80).
Le prix est plus grand ou egal au minimum depensé	Un montant de remise de 10$ est défini (discount est défini à 10). Un montant d'achat minimum de 50$ est requis pour bénéficier de la remise (minimumSpend est défini à 50). Le prix actuel du produit est de 60$ (currentPrice est défini à 60).	Le test attend un prix final de 50$ (discountedPrice est attendu à 50).
Le prix actuel est inferieur au prix minimal depense	Un montant de remise de 10$ est défini (discount est défini à 10). Un montant d'achat minimum de 50$ est requis pour bénéficier de la remise (minimumSpend est défini à 50).	Le test attend un prix final de 50$ (discountedPrice est attendu à 40).

	Le prix actuel du produit est de 40$ (currentPrice est défini à 40).	
Le code de reference est au bon format	Le user id = 123	Referal code= 123
Le code promo est valide et est de type moneyoff	const discountCode = 'MONEYOFF10'; const currentTotal = 100;	Discountedtotal = 80
Le code promo est valide et est de type percentageoff	const discountCode = 'PERCENTAGEOFF20'; const currentTotal = 100;	Discountedtotal=80
Le code promo est invalid	const discountCode = 'INVALIDCODE'; const currentTotal = 100;	Discountedtotal=100

9. Risques

les risques possibles sont :

risques	**solutions**
Incompatibilité entre le framework de test et l'application	Sélectionner un framework de test compatible avec l'application.
Couverture de test insuffisante	Définir une stratégie de test complète qui couvre toutes les fonctionnalités, les modules et les composants de l'application. Utiliser des

	outils de test automatisés pour augmenter la couverture des tests
L'application peut ne pas fonctionner correctement sur tous les appareils ou systèmes d'exploitation, ce qui peut limiter son audience	Tester l'application sur une large gamme d'appareils et de systèmes d'exploitation pour identifier et corriger les problèmes de compatibilité

12. test automation

Les tests seront automatisés à l'aide du framework Vitest. Les scripts de test seront écrits dans un format JavaScript clair et concis, et ils seront exécutés à l'aide du runner de test Vitest.

11. Reporting and communication

le plan doit etre soumis dans le repertoire sur le lien
https://github.com/atemengue/software_testing_labs
le format : est le suivant Groupe: 1 Tp: 1 exemple de branche: tp1 groupe1 info_362 exemple de branche: tp1_groupe1_ict_304

Conclusion

Ce plan de test fournit un cadre pour assurer la qualité de l'application mobile de restaurant. En suivant ce plan, nous pouvons garantir que l'application répond aux exigences spécifiées, offre

une expérience utilisateur fluide et est exempte de bogues majeurs. Toute fois les test unitaires ne suffisent pas toujours a dire qu’une application fonctionne bien