Plan de test

1 Présentation

Le plan de test est conçu pour prescrire la portée, l'approche, les ressources et le calendrier de toutes les activités de test du projet INF352.

Le plan identifie les éléments à tester, les fonctionnalités à tester, les types de tests à effectuer, le personnel responsable des tests, les ressources et le calendrier requis pour réaliser les tests, ainsi que les risques associés au plan.

1.1 Portée

1.1.1 Dans la portée

Toutes les fonctionnalités du dossier js qui ont été définies dans le dossier doivent être testés.

Nom du module	**Rôles applicables**	**Description**
Basket	- calculateTotal(basketItems, discount = null) - showAdverts(user) - searchBasket(basketItems, searchQuery) - getBasketItem(basketItems, event) - createBasketItem(basketItems, event, requiredTickets) - serializeBasketItemsToJson(basketItems)	Ils contiennent les fonctions qui fournissent des outils essentiels pour gérer les éléments d'un panier d'achat dans une application, incluant le calcul du total, la gestion des publicités, la recherche dans le panier, la gestion des éléments du panier et la sérialisation des données du panier
Error-handling	- InvalidEventNameError - InvalidEventPriceError - InvalidUsernameError - InvalidReferralCodeError - UserHasAccountError	L'objectif principal de l'application est de permettre aux utilisateurs de créer un panier de produits et d'obtenir des promotions. Le plan de test détaillé couvre les différents scénarios et cas de tests pour assurer la qualité et la fiabilité de l'application, en mettant un accent particulier sur la gestion des erreurs via le module.
Events	- isSoldOut() - getTagLine(event, minimumTicketCount, isPopular) - createEvent(name, price, availableTickets) - today(event) - next7Days(event) - next30Days(event) - getEvents(events, searchPredicate)	Ils contiennent les fonctions permettant de gérer des évènements et d'obtenir des informations relatives à ces évènements comme le nombre le nombre de tickets disponibles pour l'évènement, la date à laquelle se tiendra lieu l'évènement
Promotions	- calculatePercentageDiscount(percentage, minimumSpend, currentPrice) - calculateMoneyOff(discount, minimumSpend, currentPrice) - generateReferralCode(userId) - applyDiscount(discountCode, currentTotal) - getExchangeRate(currencyCode, callback) - getDiscount(code) - -	Ce module est pour la gestion des promotions ainsi que toutes les fonctionnalités liées à celle-ci. Nous avons par exemple la gestion des rabais (discount), les taux d'échanges des devises, ainsi que génération des codes promo.
users	- userExists(username) - createUserId() - isValidUserName(userName) - createAccount(username) - getPastPurchases(userId) - parsePurchaseResponse(purchaseData) - getPurchaseHistory(userId)	Ce module contient les fonctions relative a la gestion des utilisateurs des comptes et des fonctions que les utilisateurs peuvent effectuer concernants leur achats , tel que consulter l'historique des produit achetees … etc

1.1.2 Hors de portée

Ces fonctionnalités ne sont pas testées car elles ne sont pas incluses dans les spécifications logicielles requises :

Les fonctions de tous les modules sauf ceux du module js

1.2 Objectif qualité

Les objectifs du test sont de **vérifier** la fonctionnalité du projet inf352, le projet devrait se concentrer sur le test des différents fichiers se trouvent dans le dossier js tel que basket, erro-handling, events, promotion, user.

1.3 Rôles et responsabilités

Numero	**Membre**	**Taches**
1	TCHAMI Sorelle	Basket
2	KOUAM NOUBISSI BRICE	Error-handling
3	Stéphane Roylex	Events
4	TSAFACK Brunel	Promotions
5	SOKOUDJOU Manuel	Users

2 Méthodologie des tests

2.1 Niveaux de tests

Dans le projet INF352, 1 type de test doit être effectués.

Test unitaire.

2.2 Complétude des tests

Pour qu'une phase de test soit considérée comme complète et réussie :

- **Couverture des tests (Coverage Rate) :**

    - Tous les segments de code doivent être testés à 100%, sauf en cas de justification valable.
- **Taux de réussite des tests (Pass Rate) :**
    - Au moins 80% des tests doivent réussir.

2.3 Tâche, estimation et calendrier du projet

Taches	Membres	Estimer l'effort
Basket	TCHAMI S.	50 heures-Homme
Error-handling	KOUAM NOUBISSI BRICE	50 heures-Homme
Events	Stéphane Roylex	50 heures-Homme
Promotions	TSAFACK Brunel	50 heures-Homme
users	SOKOUDJOU	50 heures-Homme

3 Besoins en ressources et en environnement

3.1 Outils de test

Numéro	Ressources	Descriptions
1	Ordinateurs	Nous avons eu besoin de 5 ordinateurs.
2	Logiciel	Nous utilisons Node.js, qui utilise la bibliothèque npm (Node Package Manager) pour la gestion des dépendances.
3	Bibliothèque de test	Nous utilisons la bibliothèque de test Vitest pour écrire et exécuter nos tests.

3.2 Environnement de test

Il mentionne la configuration matérielle et logicielle minimale requise qui sera utilisée pour tester l'Application

Windows 11 et Windows 10.

4 Description des cas de tests

Basket file

Function calculateTotal

id	Test type	Description	Expected results	Actual result	Tested by
1	Test unit	Doit retourner 0 si taille basketItems = 0 et discount = null0	0	0	TCHAMI
2	Test unit	Doit retourner 10 si taille de basketItems = 1 avec discount = null	10	10	TCHAMI
3	Test unit	Doit retourner 10 si basketItems est egal 1 et discount egal a 5	10	10	TCHAMI
4	Test unit	Doit retourner 30 si basketItems > 1 et discount = null	30	30	TCHAMI
5	Test unit	Doit retourner 20 si basketItems > 1 et discount = 10	20	20	TCHAMI
6	Test unit	Doit retourner -5 si basketItems est egal a 0 et discount = 5	-5	-5	TCHAMI

Function showAdverts

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test unit	Doit retourner faux si user est premium	false	false	TCHAMI
2	Test unit	Doit retourner vrai si user n'est pas premium	true	True	TCHAMI

3	Test unit	Doit retourner vrai si user est vide	true	true	TCHAMI

Function searchBasket

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test unit	Doit retourner basketItems si searchQuery est vide	[{ event: { name: "Concert" } }, { event: { name: "Festival" } }]	[{ event: { name: "Concert" } }, { event: { name: "Festival" } }]	TCHAMI
2	Test unit	Doit retourner le vide si shearchQuery ne correspond a aucun élément	[]	[]	TCHAMI
3	Test unit	Doit retourner enregistrement si searchQuery correspond est partiellement a un enregistrement de basketItems (casse ignorée)'	[{ event: { name: "Concert" } }]	[{ event: { name: "Concert" } }]	TCHAMI
4	Test unit	Doit retourner enregistrement si searchQuery correspond exactement a un enregistrement de basketItems (casse ignorée)	[{ event: { name: "Concert" } }]	[{ event: { name: "Concert" } }]	TCHAMI
5	Test unit	Doit retourner les enregistrements qui contiennent element de searchQuery	[{ event: { name: "Concert" } } { event: { name: "Concert Hall" } }]	[{ event: { name: "Concert" } } { event: { name: "Concert Hall" } }]	TCHAMI
6	Test unit	Doit retourner une liste vide si basketItems est vide	[]	[]	TCHAMI

Function getBasketItem

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test unit	Doit retourner enregistrement correspondant s\il a ete trouver dans basketItems	{ event: { id: 1, name: "Concert" } }	{ event: { id: 1, name: "Concert" } }	TCHAMI
2	Test unit	Doit retourner null si aucun objet ne correspond aux objets de basketItems	NULL	NULL	TCHAMI
3	Test unit	Doit retourner null si la basketItems est vide	NULL	NULL	TCHAMI
4	Test unit	Doit retourner enregistrement meme si evénement est avec une propriétés supplémentaires	{ event: { id: 1, name: "Concert", date: "2024" } }	{ event: { id: 1, name: "Concert", date: "2024" } }	TCHAMI

Function createBasketItem

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test unit	Doit retourner le nouvel enregistrement creer du panier pour un événement non présent dans le panier	{ id: 2, name: "Festival" }	{ id: 2, name: "Festival" }	TCHAMI
2	Test unit	Doit retourner null si evénement est déjà présent dans le panier	NULL	NULL	TCHAMI

3	Test unit	Doit retourner le nouvel enregistrement creer si le panier est vide	{ id: 1, name: "Concert" }	{ id: 1, name: "Concert" }	TCHAMI

Function serializeBasketItemsToJson

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test unit	Doit retourner les enregistrements entrer	[{ event: { id: 1, name: "Concert" }, requiredTickets: 2 }, { event: { id: 2, name: "Festival" }, requiredTickets: 3 }]	[{ event: { id: 1, name: "Concert" }, requiredTickets: 2 }, { event: { id: 2, name: "Festival" }, requiredTickets: 3 }]	TCHAMI
2	Test unit	Doit retourner un tableau vide si les basktItems est vide	[]	[]	TCHAMI
	Test unit	Doit retourner les enregistrements entrer meme s\il contient plus de parametre	[{ event: { id: 1, name: "Concert" }, requiredTickets: 2, type: 'Numerique' }]	[{ event: { id: 1, name: "Concert" }, requiredTickets: 2, type: 'Numerique' }]	TCHAMI

Events.test.js File

- **Fonction isSoldOut**

ID	Test Type	Description	Expected Results	Actual Results	Tested by

1	Test Unitaire	Vérifier que la fonction isSoldOut renvoie vrai lorsqu'il n'y a plus de tickets disponibles	Vrai	Vrai	Stéphane Roylex
2	Test Unitaire	Vérifier que la fonction isSoldOut renvoie faux lorsqu'il y'a encore des tickets disponibles	Faux	Faux	Stéphane Roylex

- **Fonction getTagLine**

ID	**Test Type**	**Description**	**Expected Results**	**Actual Results**	**Tested by**
1	Test Unitaire	Vérifier que la fonction getTagLine renvoie un message indiquant qu'il n'y a plus de tickets disponibles lorsqu'il n'y en a plus	Event Sold Out!	Event Sold Out!	Stéphane Roylex

2	Test Unitaire	Vérifier que la fonction getTagLine renvoie un message indiquant qu'il n'y a presque plus de tickets lorsque le nombre de tickets restant est inférieur à un certain seuil	Hurry only x ticket(s) left	Hurry only x ticket(s) left	Stéphane Roylex
3	Test Unitaire	Vérifier que la fonction getTagLine renvoie un message	This Event is getting a lot of interest. Don't miss	This Event is getting a lot of interest. Don't miss	Stéphane Roylex
		indiquant qu'un évènement est populaire et qu'il y'a encore suffisamment de tickets disponibles	out, purchase your ticket now!	out, purchase your ticket now!	
4	Test Unitaire	Vérifier que la fonction getTagLine renvoie un message indiquant qu'un évènement n'est pas très populaire et qu'il y'a encore suffisamment de tickets disponibles	Don't miss out, purchase your ticket now!	Don't miss out, purchase your ticket now!	Stéphane Roylex

- **Fonction createEvent**

ID	Test Type	Description	Expected Results	Actual Results	Tested by
1	Test Unitaire	Vérifier que le nom d'un évènement est une chaine d'au plus 200 caractères	Event name cannot exceed 200 characters	Event name cannot exceed 200 characters	Stéphane Roylex
2	Test Unitaire	Vérifier que le coût de l'évènement est un nombre positif	Event price must be more or equal to 0	Event price must be more or equal to 0	Stéphane Roylex
3	Test Unitaire	Vérifier que le nombre de tickets disponibles est un nombre	Event tickets must be more than 0	Event tickets must be more than 0	Stéphane Roylex
		strictement positif			
4	Test Unitaire	Vérifier que la fonction createEvent retourne un object de type Event si tout ses paramètres sont valides	Event (null, name, price, availableTick)	Event (null, name, price, availableTick)	Stéphane Roylex

Filters.test.js :

- **Fonction today**

ID	Test Type	Description	Expected Results	Actual Results	Tested by

1	Test Unitaire	Vérifier que la fonction today retourne vrai lorsque la date d'aujourd'hui correspond à la date de l'évènement	Vrai	Vrai	Stéphane Roylex
2	Test Unitaire	Vérifier que la fonction today retourne faux lorsque la date d'aujourd'hui ne correspond pas à la date de l'évènement	Faux	Faux	Stéphane Roylex
3	Test Unitaire	Vérifier que la fonction next7Days retourne vrai lorsque la date de l'évènement est au plus à	Faux	Faux	Stéphane Roylex
		7jours de la date d'aujourd'hui			

- **Fonction next7Days**

ID	Test Type	Description	Expected Results	Actual Results	Tested by
1	Test Unitaire	Vérifier que la fonction next7Days retourne vrai lorsque la date de l'évènement est au plus à 7jours de la date d'aujourd'hui	Vrai	Vrai	Stéphane Roylex
2	Test Unitaire	Vérifier que la fonction next7Days retourne faux lorsque la date de l'évènement est soit déjà passé	Faux	Faux	Stéphane Roylex
3	Test Unitaire	Vérifier que la fonction next7Days retourne faux lorsque la date de l'évènement est à plus de 7jours de la date d'aujourd'hui	Faux	Faux	Stéphane Roylex

- **Fonction next30Days**

ID	Test Type	Description	Expected Results	Actual Results	Tested by

1	Test Unitaire	Vérifier que la fonction next30Days retourne vrai lorsque la date de l'évènement est au plus à 30jours de la date d'aujourd'hui	Vrai	Vrai	Stéphane Roylex
2	Test Unitaire	Vérifier que la fonction next30Days retourne faux lorsque la date de l'évènement est soit déjà passé	Faux	Faux	Stéphane Roylex
3	Test Unitaire	Vérifier que la fonction next30Days retourne faux lorsque la date de l'évènement est à plus de 30jours de la date d'aujourd'hui	Faux	Faux	Stéphane Roylex

PROMOTIONS.JS

calculatePercentageDiscount function

ID	Test Type	Description	Expected Results	Actual Results	Tested by
1	Test Unitaire	should return current price if current price is less than minimum amount spent.	499	499	Tsafack Brunel

2	Test Unitaire	should return current price with percentage discount if current price greater than or equal to minimum amount spent	400.8	400.8	Tsafack Brunel

calculateMoneyOff function

ID	Test Type	Description	Expected Results	Actual Results	Tested by
1	Test Unitaire	should return current price if current price is less than minimum amount spent.	499	499	Tsafack Brunel
2	Test Unitaire	should return the difference between current price and discount, if current price greater than or equal to minimum amount spent.	481	481	Tsafack Brunel

applyDiscount function

ID	Test Type	Description	Expected Results	Actual Results	Tested by
1	Test Unitaire	should return current total if discount is invalid	499	499	Tsafack Brunel

2	Test Unitaire	should apply MONEYOFF when discount is valid and discount type is MONEYOFF	450	450	Tsafack Brunel
3	Test Unitaire	should apply PERCENTAGEOFF when discount is valid and discount type is PERCENTAGEOFF	400	400	Tsafack Brunel

discount.js file

getDiscount function

ID	Test Type	Description	Expected Results	Actual Results	Tested by
1	Test Unitaire	fetches discount data for a valid code	{data: { isValid: true, type: 'MONEYOFF', value: 10 }}	{data: {data: { isValid: true, type: 'MONEYOFF', value: 10 }}, status: 200}	Tsafack Brunel
2	Test Unitaire	handles errors from the server	Error	Error	Tsafack Brunel

exchange.js

getExchangeRate function

ID	Test Type	Description	Expected Results	Actual Results	Tested by
1	Test Unitaire	should return correct data for supported currencies	{ originalCurrency: 'GBP', newCurrency: 'USD', exchangeRate: 1.25 }	{ originalCurrency: 'GBP', newCurrency: 'USD', exchangeRate: 1.25 }	Tsafack Brunel
2	Test Unitaire	should throw error for	{ originalCurrency: 'GBP',	actual results to be an error	Tsafack Brunel

		unsupported currencies	newCurrency: 'USD', exchangeRate: 1.25 }		

exchangeRateProvider.js file

exchangeRateProvider.calExchangeRateProvider function

ID	Test Type	Description	Expected Results	Actual Results	Tested by
1	Test Unitaire	should return correct exchange rate for USD	1.25	1.25	Tsafack Brunel
2	Test Unitaire	should return correct exchange rate for EUR	1.18	1.18	Tsafack Brunel
3	Test Unitaire	should return correct exchange rate for NZD	1.93	1.93	Tsafack Brunel
4	Test Unitaire	should throw error for unsupported currency	Error	Error	Tsafack Brunel

Account file

test isValidUserName function

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test Unitaire	should return false if username is empty	**false**	**false**	SOKOUDJOU Manuel

2	Test Unitaire	Should return false if username doesnot contains @ .	**false**	**false**	SOKOUDJOU Manuel
3	Test Unitaire	Should return true if username is valid	**true**	**true**	SOKOUDJOU Manuel

test createAccount function

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test Unitaire	Should throw exception if username is invalid or user exist.	**reject("User already exists")**	**reject("User already exists")**	SOKOUDJOU Manuel
2	Test Unitaire	Should return false if username doesnot contains @ .	**false**	**false**	SOKOUDJOU Manuel
3	Test Unitaire	Should return user object if user exists	**{ "userId": 2, "username": username, }**	**{ "userId": 2, "username": username, }**	SOKOUDJOU Manuel

User file

test userExists function

ID	Test type	Description	Expected results	Actual result	Tested by

1	Test Unitaire	Should return False if user doesnot exist	**false**	**false**	SOKOUDJOU Manuel
2	Test Unitaire	Should return True if user exist	**true**	**true**	SOKOUDJOU Manuel

test createUserId function

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test Unitaire	Should return 2 when called	**2**	**2**	SOKOUDJOU Manuel

Test fonction getPurchaseHistory

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test Unitaire	Test fonction getPurchaseHistory	an instance of Axios Request	an instance of Axios Request	SOKOUDJOU Manuel
2	Test Unitaire	Should return an instance of Axios Request	Promise<void>	Promise<void>	SOKOUDJOU Manuel
3	Test Unitaire	Should return an array of purchase	[{ name: "Punk Goes Pop - 90s", tickets: 2, price: 40.00, }, {	Array of Purchase of length 3	SOKOUDJOU Manuel

			name: "Adventures Live!",
			
			tickets: 5,
			price: 120.00,
			},
			{
			
			name: "Folk dance party!",
			
			tickets: 3,
			price: 75.00,
			}
			]		

Test fonction parsePurchaseResponse

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test Unitaire	Should return an array of purchase	an array of purchase	an array of purchase	SOKOUDJOU Manuel

Test fonction getPastPurchase

ID	Test type	Description	Expected results	Actual result	Tested by
1	Test Unitaire	Should throw error if no purchase history	Error(/history/)	Error(/history/)	SOKOUDJOU Manuel

error_handling.js file

ID	Test type	Description	Expected results	Actual result	Tested by

1	Test Unitaire	Verification de l'instance InvalidEventNameError	L'instance est correctement créée avec le message approprié	L'instance est correctement créée avec le message approprié	KOUAM NOUBISSI BRICE
2	Test Unitaire	Vérification des messages pour toutes les classes d'erreurs	Chaque instance est correctement créée avec le bon message	Chaque instance est correctement créée avec le bon message	KOUAM NOUBISSI BRICE
3	Test Unitaire	Vérification de la trace de la pile pour InvalidEventNameError	La trace de la pile contient le nom correct de la classe d'erreur	La trace de la pile contient le nom correct de la classe d'erreur	KOUAM NOUBISSI BRICE
4	Test Unitaire	Vérification des cas limites pour InvalidUsernameError	Les instances sont correctement créées et les messages sont gérés correctement	Les instances sont correctement créées et les messages sont gérés correctement	KOUAM NOUBISSI BRICE

5 Risques

5.1 Identifier les Risques

Catégories de Risques

Pour identifier les risques pertinents, il est utile de les catégoriser en fonction de leur nature:

1. Risques Fonctionnels:

- **Dysfonctionnements:** L'application ne se comporte pas comme prévu, entraînant des erreurs ou des bugs.
- **Incomplétude:** Des fonctionnalités attendues sont manquantes ou non implémentées correctement.
- **Intégrité des données:** Les données stockées ou traitées par l'application sont incorrectes, incomplètes ou non sécurisées.

2. Risques Non Fonctionnels:

- **Performance:** L'application est lente, peu réactive ou ne parvient pas à gérer les pics de charge.
- **Sécurité:** L'application est vulnérable aux intrusions, aux fuites de données ou aux cyberattaques.
- **Usabilité:** L'application est difficile à utiliser, peu intuitive ou non accessible à tous les utilisateurs.

- **Compatibilité:** L'application ne fonctionne pas correctement sur tous les navigateurs, appareils ou systèmes d'exploitation.

5.2 Évaluer la Criticité et la Probabilité des Risques

Évaluation des Risques

Une fois les risques identifiés, il est nécessaire de les évaluer en fonction de leur criticité et de leur probabilité d'occurrence:

Criticité:

- **Élevée:** Le risque peut causer des dommages importants à l'entreprise, aux utilisateurs ou à la réputation.
- **Moyenne:** Le risque peut causer des perturbations modérées ou des désagréments aux utilisateurs.
- **Faible:** Le risque a un impact minime sur les opérations ou la satisfaction des utilisateurs.

Probabilité:

- **Élevée:** Le risque est susceptible de se produire fréquemment ou dans un avenir proche.
- **Moyenne:** Le risque peut se produire occasionnellement ou dans des circonstances particulières.
- **Faible:** Le risque est peu probable de se produire.
- **Identification des Risques**

Catégorie	Risque	Description
Fonctionnel	Dysfonctionnement de la page de paiement	L'utilisateur ne peut pas finaliser sa commande à cause d'une erreur sur la page de paiement.
Fonctionnel	Erreur de calcul des taxes	Le montant total de la commande n'inclut pas les taxes correctes.
Fonctionnel	Intégrité des données des produits	Les descriptions, prix ou images des produits sont incorrects ou incomplets.
Non Fonctionnel	Performance lente du site	Le site web est lent à charger et à répondre aux requêtes des utilisateurs.
Non Fonctionnel	Faille de sécurité XSS	L'application est vulnérable aux attaques XSS permettant à des scripts malveillants d'être injectés.
Non Fonctionnel	Mauvaise ergonomie sur mobile	L'application est difficile à utiliser sur les appareils mobiles en raison d'une mise en page inadaptée ou de contrôles trop petits.

Non Fonctionnel	Incompatibilité avec Internet Explorer	L'application ne fonctionne pas correctement sur le navigateur Internet Explorer.

2. Évaluation des Risques

Risque	Criticité	Probabilité	Priorité
Dysfonctionnement de la page de paiement	Élevée	Moyenne	1
Erreur de calcul des taxes	Moyenne	Élevée	2
Intégrité des données des produits	Moyenne	Moyenne	3
Performance lente du site	Moyenne	Moyenne	4
Faille de sécurité XSS	Élevée	Faible	5
Mauvaise ergonomie sur mobile	Moyenne	Moyenne	6
Incompatibilité avec Internet Explorer	Faible	Faible	7