# TEST PLAN DOCUMENT

Présenté par : **Groupe 9**

| NOMS | MATRICULES | DOSSIERS/ FICHIERS TESTES |
|---|---|---|
| **WANGA KOTTE GIOVANNI CHRISTELLE** | 21T2594 | EVENT (event.js, search.js, filter.js) |
| **AMBOMO TIGA GEDEON** | 21T2496 | USER (user.js, account.js, purchaseHistory.js) |
| **KAMDEM RAOUL IVAN** | 21T2284 | PROMOTION (promotion.js) |
| **NYOBE LUCRESSE GEORGINA** | 20U2704 | BASKET (basket.js, basketitem.js) |
| **TOUDJI YEMEFACK FERRINO** | 20U2870 | PROMOTION (exchange.js, exchangeRateProvider.js, discount.js) |

# Table des matières


1. INTRODUCTION ..... 3
2. OBJECTIFS DES TESTS ..... 3
3. APPROCHE DE TEST ..... 3
4. CALENDRIER DES TESTS ..... 4
5. ENVIRONNEMENT DE TEST ..... 4
6. DONNEES DE TEST ..... 4
   - Cas de test ..... 5
     - basket.js ..... 5
     - event.js ..... 6
     - Filter.js ..... 7
     - Search.js ..... 8
     - Promotion.js ..... 8
     - Disount.js ..... 9
     - Exchange.js ..... 9
     - exchangeRateProvider.js ..... 10
     - Acount.js ..... 10
     - purcharseHistory.js ..... 11
     - User.js ..... 11
7. AUTOMATISATION DES TESTS ..... 12
   - Risques et problèmes ..... 12
   - Rapports et communication ..... 13

Conclusion ..... 13

La création d'un document de plan de test est une partie essentielle de la planification des processus de test, et il fournit une feuille de route détaillée pour le processus de test.

# 1. INTRODUCTION

Il nous été demandé de réaliser le test logiciel (en particulier une série de test unitaire) sur un dossier appelé « JS », comportant un ensemble de fichiers java-script. L'objectif ici est de parcourir chaque fichier, d'analyser et de produire des cas de test qui permettront de valider les sorties de chaque fichier

# 2. OBJECTIFS DES TESTS

Cette section doit indiquer clairement les objectifs de test et les buts que l'équipe de test vise à atteindre. Les objectifs des tests doivent être aligné sur les exigences de l'entreprise et doit être mesurable et réalisable.

# 3. APPROCHE DE TEST

Cette section doit décrire approche de test global que l'équipe de test suivra suivre. Il doit inclure les méthodes de test et techniques qui seront utilisées et comment les tests seront effectués menée. Cette section devrait également inclure les rôles et les responsabilités de l'équipe de test.

## 4. CALENDRIER DES TESTS

Cette section doit fournir un calendrier du processus de test. Il devrait inclure les dates de début et de fin de chaque phase de test, y compris planification, conception, exécution et reporting.

## 5.ENVIRONNEMENT DE TEST

Cette section doit décrire l'environnement de test requis pour exécuter les cas de test. Il doit inclure des informations sur le matériel, les logiciels, et les configurations réseau requises pour les tests.

## 6. DONNEES DE TEST

Cette section doit décrire les données de test requis pour exécuter les cas de test. Il devrait inclure la source des données de test, le type de données requis et toute contrainte ou limitation sur les données de test.

- ## Cas de test

Cette section doit fournir une liste détaillée des cas de tests qui seront exécutés pendant les tests processus. Il doit inclure des informations sur le cas de test ID, la description du scénario de test, le résultat attendu et le statut de chaque cas de test.

- ### basket.js

✓ tests/basket/basket.test.js (22)

✓ calculateTotal (5)

✓ should calculate the correct total for a single item
✓ should calculate the correct total for multiple items
✓ should handle an empty basket
✓ should apply the discount correctly
✓ should handle negative discounts

✓ showAdverts (2)

✓ should return true for non-premium users

✓ should return false for premium users

✓ searchBasket (4)

✓ should return the correct items for the search query ""

✓ should return the correct items for the search query "non-existent"

✓ should return the correct items for the search query "event"

✓ should return the correct items for the search query "special"

✓ getBasketItem (4)

✓ should return null when the basket is empty
✓ should return the basket item for the given event
✓ should return null if the event is not found in the basket
✓ should return null if the event does not have an id property

✓ createBasketItem (1)

✓ should return null when the event is already in the basket

✓ serializeBasketItemsToJson (4)

✓ with empty basket
✓ with single item
✓ with multiple items
✓ does not modify the original basket items

✓ class BasketItem (2)

✓ doit creer une instance de BasketItem
✓ doit prix total des tickets

- **event.js**

✓ tests/events/event.test.js (9)

✓ isSoldOut function (2)

✓ should return true if all tickets are sold (ticketsRemaining === 0)

✓ should return false if tickets are remaining (ticketsRemaining > 0)

✓ createEvent (2)

✓ should return a new event with valid propieties ie peoprieties that macthes the type's constraint

✓ should return an error message if the caracteristics does not match the type and the lenght constraint

✓ getTagLine function (5)

✓ should return 'Event Sold Out!' for a sold-out event

✓ should return a hurry message for low ticket availability

✓ should return a generic tagline for a non-popular event with low availability

✓ should return a generic tagline for a non-popular event with enough tickets

✓ should return a popular event tagline for a popular event with enough tickets

- **Filter.js**

✓ tests/events/filter.test.js (6)

✓ today function (2)

✓ should return true if today's date matches the event's date

✓ should return false if today's date does not match the event's date

✓ next7Days function (2)

✓ should return false if the event's date is after the next 7 days

✓ should return false if the event's date is in the past

✓ next30Days function (2)

✓ should return true if the event's date is within the next 30 days

✓ should return false if the event's date is after the next 30 days

- **Search.js**

✓ tests/events/search.test.js (1)

✓ getEvents (1)

✓ doit retourner la list des objet ayant la propriete passe en parametre

- **Promotion.js**

✓ tests/promotions/promotion.test.js (5)

✓ test calculatePercentageDiscount (2)

✓ doit retourner un reel valide si currentPRice >= minimumSpend

✓ doit retourner prixCourant si prixCourant  <depenseMinimale

✓ test calculateMoneyOff (2)

✓ doit retourner la difference entre le prixCourant et la reduction si prixCourant >= depenseMinimale

✓ doit retourner le prixCourant si la prixCourant < depenseMinimale

✓ test generateReferralCode (1)

✓ doit retourner un mot ayant userId

- **Disount.js**

✓ tests/promotions/discount/discount.test.js (1)

✓ getDiscount (1)

✓ doit retourner une liste objet

- **Exchange.js**

✓ tests/promotions/exchange/exchange.test.js (3)

✓ getExchangeRate (3)

✓ Doit appeler exchangeRateProvider.callExchangeRateProvider avec un code correct

✓ Doit appeler le callback avec un object correct associé comme reponse

✓ Dois gerer correctement les erreurs liées à exchangeRateProvider.callExchangeRateProvider

- ## exchangeRateProvider.js

✓ tests/promotions/exchange/exchangeRateProvider.test.js (4)

✓ callExchangeRateProvider (4)

✓ Doit me retourner le taux d echange correct pour USD

✓ Doit me retourner le taux d echange correct pour EUR

✓ Doit me retourner le taux d echange correct pour NZD

- ## Acount.js

✓ tests/users/account.test.js (8)

✓ Purchase (8)

✓ constructor (1)

✓ doit creer une instance de la classe Purchase

✓ isValidUserName (3)

✓ doit retourner true si 'le username est valide'

✓ doit retourner false si 'le username est invalide'

✓ doit retourner false si 'le username est invalide'

✓ createAccount (3)

✓ doit renvoyer une exception si le nom d'utilisateur est invalid

✓ doit créer un utilisateur si son username est nouveau

✓ doit renvoyer un string si le username est déjà existant

✓ getPastPurcharses (1)

✓ doit retourner un objet si la réponse du serveur est prête

- **purcharseHistory.js**

✓ tests/users/purchaseHistory.test.js (2)

✓ getPurcharseHistory (2)

✓ doit retourner un objet si la réponse du serveur est prête

✓ doit me retourner un tableau d'objet

- **User.js**

✓ tests/users/users.test.js (4)

✓ User (4)

✓ constructor (1)

✓ doit creer une instance de la classe User

✓ userExists (2)

✓ doit retourner true si 'le nom d\'utilisateur est deja exista...'

✓ doit retourner false si 'le nom d\'utilisateur est inexistant'

✓ createUserId (1)

✓ doit retourner une valeur entière

# 7. AUTOMATISATION DES TESTS

Cette section doit décrire tout test l'automatisation qui sera utilisée pendant le processus de test. Il devrait inclure des informations sur les outils qui seront utilisés, les scripts de test qui seront créés et les types de tests qui seront automatisés.

- **Risques et problèmes**

Cette section doit fournir une liste des risques et problèmes potentiels qui pourraient avoir un impact sur les tests processus. Il doit inclure une description de chaque risque ou problème, l'impact que cela pourrait avoir sur le processus de test, et toute stratégie d'atténuation qui sera utilisée pour y remédier.

- ## Rapports et communication

Cette section doit décrire comment les résultats des tests seront rapportés et communiquée aux parties prenantes. Il devrait inclure des informations sur la fréquence des rapports, le format des rapports et les parties prenantes qui recevront les rapports.

# Conclusion

Cette section doit résumer les principaux points du document du plan de test et fournir tout informations complémentaires ou recommandations.